Towards Folksonomy-based Personalized Services in Social Media

by

Majdi Rawashdeh

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Ph.D. degree in Computer Science

Ottawa-Carleton Institute for Computer Science
School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Every single day, lots of users actively participate in social media sites (e.g., Facebook, YouTube, Last.fm, Flicker, etc.) upload photos, videos, share bookmarks, write blogs and annotate/comment on content provided by others. With the recent proliferation of social media sites, users are overwhelmed by the huge amount of available content. Therefore, organizing and retrieving appropriate multimedia content is becoming an increasingly important and challenging task. This challenging task led a number of research communities to concentrate on social tagging systems (also known as *folksonomy*) that allow users to freely annotate their media items (e.g., music, images, or video) with any sort of arbitrary words, referred to as tags. Tags assist users to organize their own content, as well as to find relevant content shared by other users. In this thesis, we first analyze how useful a folksonomy is for improving personalized services such as tag recommendation, tag-based search and item annotation. We then propose two new algorithms for social media retrieval and tag recommendation respectively. The first algorithm computes the latent preferences of tags for users from other similar tags, as well as latent annotations of tags for items from other similar items. We then seamlessly map the tags onto items, depending on an individual user's query, to find the most desirable content relevant to the user's needs. The second algorithm improves tag-recommendation and item annotation by adapting the Katz measure, a path-ensemble based proximity measure, for the use in social tagging systems. In this algorithm we model folksonomy as a weighted, undirected tripartite graph. We then apply the Katz measure to this graph, and exploit it to provide personalized tag recommendation for individual users. We evaluate our algorithms on two real-world folksonomies collected

from Last.fm and CiteULike. The experimental results demonstrate that the proposed algorithms improve the search and the recommendation performance, and obtain significant gains in cold start situations where relatively little information is known about a user or an item.

Acknowledgements

(رَبِّ أَوْزِعْنِيٓ أَنْ أَشْكُرَ نِعْمَتَكَ ٱلَّتِيٓ أَنْعَمْتَ عَلَيَّ)

"*My Lord, enable me to be grateful for Your favor which You have bestowed upon me*"

First and foremost, I would like to thank the Almighty *ALLAH*, the compassionate, the Merciful, for HIS immense blessings and guidance. I could have never completed this work without the faith I have in HIM, the Almighty.

I would like to express my deepest gratitude and sincere appreciation to my supervisor, Professor Abedmotaleb El Saddik for his valuable advice, supervision, and financial support throughout my study. He has always been a source of wisdom, encouragement, valuable guidance and motivation. I will always remember his smile, kindness and sense of humor. No words can express my love and appreciation to him.

Special thanks are due to my colleague Dr. Heung Nam Kim who made his in-depth knowledge in recommender system, social media analysis, and user modeling ready for my utilization. His invaluable help, continuous support and careful revisions for the last four years reflected in a successful research achievement. Thank you my best friend Nami.

Also I would like to thank the members of the Multimedia Communications Research Laboratory, for their suggestions, critical remarks, and cooperation during my research work.

Last, but definitely not least, I can't imagine my PhD degree without the love and invaluable help from my family. I should express my boundless gratitude and love to my: wife *Asma*, son *Anas*, *brothers*, *sister*, and *parents* whose consistent encouragement, support and endless love have taken me through all hardships incurred during my studies and research journey.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

FBR	Folksonomy-Boosted Ranking
LASM	Latent Semantic Analysis Model
LDA	Latent Dirichlet Allocation
MAP	Mean Average Precision
MHR	Mean Hit Rate
MRR	Mean Reciprocal Rank
UCTM	User Centric Tag Model

Chapter 1 Introduction

1.1 Background and Motivation

Social media have been changing the way people find information, share knowledge and communicate with each other. This social phenomenon has transformed the masses, who were mere information consumers via the mass media, into information producers. As rich information is shared through social media services, the huge amount of information that has not previously been available is increasing exponentially with daily additions (Markines et al. 2009). For example, according to YouTube announcement, users uploaded 100 hours of video to the site every minute, more than 100 million people take a social action on YouTube (likes, shares, comments, etc) every week, more than 20% of global YouTube views come from mobile devices. With this recent proliferation of social media sites, users are overwhelmed by the huge amount of social media content available. Therefore, organizing and retrieving appropriate multimedia content is becoming an increasingly important and challenging task. This challenging task led a number of research communities to concentrate on social tagging systems also known as folksonomies (Deshpande et al. 2004; Hotho et al 2006; Bao et al. 2007) in order to achieve the full potential offered by the social media sites (Milicevic et al. 2010).

Folksonomy allow users to freely annotate their media items (e.g., music on Last.fm[1], images on Flicker[2], videos on YouTube[3], and bookmarks on Delicious[4]) with any sort of arbitrary words, referred to as tags. Users of social tagging service are flexible and not restricted any more to a rigid hierarchy of content similar to taxonomies or predefined dictionaries. Therefore, social tagging has become a popular way to categorize, share and organize social media content, in turn leading to the sheer amount of user-generated metadata (Bischoff et al. 2008; Hotho et al 2006). Recent studies, such as (Li et al. 2008), analyzed user-generated tags and subsequently observed that a set of the aggregated tags on social media is rich and compact enough to characterize and describe the main concepts of content in the media. These observations indicate that folksonomy aggregated by millions of users contain an important source, as well as make search results more relevant. Accordingly, it is worth examining folksonomy in a way that can be highly beneficial to social media personalization.

In today's social media search, the most popular search paradigm is keyword search. Despite simplicity and efficiency, keyword queries can not accurately describe what the users really want (Lawrenc 2000). In general, users tend to generate short queries when searching, resulting in tremendous ambiguity about their intensions (Ma et al. 2007). People engaged in different areas may have different understandings of the same keywords and thus may differ significantly in the search results they considered to be relevant for the same keywords (Liu et al. 2004; Ma et al. 2007; Brynn et al. 2010).

[1] http://www.last.fm/
[2] http://www.flickr.com/
[3] http://www.youtube.com/
[4] http://www.delicious.com/

Personalized search, which was emerged in response to this problem, generates search results partially based on the personal interests or past search history of the user (Pitkow et al. 2002). It is showing different search results to different users in order to make the search more relevant to the user needs. Popular search engines (e.g., google.com, yahoo.com) are trying to move more toward personalization to search information depending on their users. With the popularity of folksonomies a number of researchers have recently concentrated on tag-based personalized search with social tagging (Carmel et al. 2009; Noll et al. 2007). Because modern social media sites allow users to freely annotate their items with any kind of descriptive tags, the users tend to use the descriptive tags to annotate the contents that they are interested in. Consequently, folksonomy can be an effective and easy way to help identify correct items and make search results more relevant to the user (Golder et al. 2006; Xu et al 2008).

Although social tagging presents promising possibilities for facilitating better search results, personalized searches incorporating social tagging encounter serious limitations regarding quality evaluation. Ideally, personalized search algorithms should discover not only accurate resources suited to user needs, but also a wide range of desirable resources. However, users employ a small proportion of all tags and often use personal and self-referential tags. In addition, some users annotate a particular resource with a generic tag, while other users annotate that same resource with a specific tag; the term for this is basic level variation (Golder et al. 2006). Moreover, such tags may be caused by different users having different ways of describing the resource, according to their personal tagging behavior when organizing resources. Consequently, it is not enough to model an individual user via tags only used by him/her. Similarly, each resource is usually labeled with a small proportion of all tags. Additionally, users make frequent use of ambiguous

and synonymous terms when they annotate and search resources, resulting in tremendous ambiguity about their intentions. In fact, some tags involve partially connected semantics and share some meanings with other tags (De Meo et al. 2009). Accordingly, users may fail to find valuable resources if they retrieve solely those resources that already have tags contained in the query.

To deal with these issues, we introduce in this thesis a new tag-based personalized search algorithm that benefits from folksonomy. Our proposed algorithm named Folksonomy-Boosted Ranking (*FBR*) determines first the similarities among resources and among tags. After that our *FBR* algorithm builds two latent models: i) the latent tag preference model that reflects how a certain user has assigned tags similar to a given tag and, ii) the latent tag annotation model that captures how users have tagged a certain tag to resources similar to a given resource. We then seamlessly map the tags onto items, depending on an individual user's query, to find the most desirable content relevant to the user's needs.

Tags assist users to organize their own content, as well as to find relevant content shared by other users. In social tagging systems, the suggestion of tags when a user is annotating a certain item can help users build more coherent folksonomies (Clements et al. 2010; Krestel et al. 2009). In addition it makes items more browsable, searchable, and sharable by other users. In particular, it is often the cases in folksonomy that little information is known about a user or an item, posing the challenge of recommending/uncovering tags relevant to such user/item. Some studies tackled the issue of tag recommendation from a graph-based perspective, where they viewed a folksonomy as a tripartite hypergraph in which each hyperedge connects a user, a tag and an item (Hotho et al. 2006; Clements et al. 2010). Since folksonomies grow and change

rapidly, the need for an approach to predict accurately the links in the folksonomy graph-based representation becomes an important issue to be addressed. As such, this thesis proposes a new graph-based ranking algorithm named *KatzBm25*, to identify the suitable tags for an item. Our proposed algorithm adapts the Katz measure, a path-ensemble based proximity measure, for the use in social tagging systems. We model a folksonomy as a weighted, undirected tripartite graph. We then apply the Katz measure to this graph, and exploit it to provide personalized tag recommendation for individual users. In addition to that our *KatzBm25* algorithm enriches items with relevant hidden tags that have not previously annotated to the item, in order to make such items more browsable, searchable, and sharable by other users.

1.2 Problem Statement

As social tagging involves rich information useful in characterizing individual users, it is worthwhile examining social tagging as a method to facilitate more accurate personalized searches and personalized recommendations. Accordingly, we effectively analyze social tagging information in ways that may prove highly beneficial in accentuating users' interests and characterizing resources. In this thesis, we focus on personalized information filtering to address the following fundamental question:

Q1 How should social media resources be ranked and ordered that their rank potentially is different for each user and each search?

Q2 How to make social media resources more visible to users by uncovering relevant hidden tags that have not previously annotated to the resource in the past?

Q3 How to identify and suggest suitable tags to users while annotating a certain social media resource?

The principal objective of our study is to model effective algorithms that will be readily applicable to social web sites. Based on our algorithms, we recommend/find the appropriate media contents/tags that are personally tailored toward a user's interests. By leveraging social tagging to recommending/finding items, we alleviate typical problems, such as sparseness of data and cold start users, in personalized recommender systems.

1.3 Contribution

Our thesis makes the following contributions toward personalized recommender systems.

First, Design and development of a tag-based personalized search algorithm (*FBR*) that can discover and rank resources relevant to user needs no matter whether query tags have been annotated in the resources or not.

Second, Design and development of a personalized tag recommendation algorithm (*KatzBm25*) to identify a list of tags that are personally tailored to a user's preferences for a given resource. More precisely, the proposed algorithm estimates proximity between users and tags, and between items and tags based on the Katz measure, and thus discovers new triangle graphs that are likely to appear within a given folksonomy.

Third, Design and development of an item annotation algorithm (*KatzBm25*) to make social media resources more visible to users by uncovering relevant hidden tags that have not previously annotated to the resource.

Forth, Provide in-depth experimental evaluations with two real publically available datasets, namely CiteULike and Last.fm. We achieve some interesting insights regarding the problems inherent to personalized recommendations —Cold Start—, which may ultimately result in the development of better solutions for this problem.

1.4 Scholarly Achievements

The following papers have been published during my PhD work:

Papers at refereed journals:

[1] M. Shamim Hossain, Mehedi Masud, Ghulam Muhammad, Majdi Rawashdeh, Mohammad Mehedi Hassan. Automated and user involved data synchronization in collaborative e-health environments. *Computers in Human Behavior*, 30:485-490, 2014

[2] Majdi Rawashdeh, Heung-Nam Kim, Jihad Mohamad Alja'am, Abdulmotaleb El Saddik. Folksonomy link prediction based on a tripartite graph for tag recommendation. *Journal of Intelligent Information Systems*, 40(2):307-325, 2013

[3] Heung-Nam Kim, Majdi Rawashdeh, Abdullah Alghamdi, Abdulmotaleb El Saddik. Folksonomy-based personalized search and ranking in social media services. *Information Systems*, 37(1): 61-76, 2012

Papers at refereed Conferences:

[1] Majdi Rawashdeh, Mohammed. F. Alhamid, Heung-Nam Kim, Vanessa Maclsaac, Awny alnusair, Abdulmotaleb El Saddik. Graph-Based personalized recommendation in social tagging systems. In: *Proceedings of the 2nd IEEE International workshop on Ambient Multimedia and Sensory Environment (AMUSE)*, China, 2014

[2] Heung-Nam Kim, Majdi Rawashdeh, Abdulmotaleb El Saddik. Tailoring Recommendations to Groups of Users: A Graph Walk-based Approach. In:

Proceedings of the international conference on User Intelligent User Interfaces (IUI), USA, 2013

[3] Mohammed. F. Alhamid, Majdi Rawashdeh, Hussein Al Osman, Abdulmotaleb El Saddik. Leveraging biosignal and collaborative filtering for context-aware recommendation. In: *Proceedings of the 1st ACM international workshop on Multimedia indexing and information retrieval for healthcare (MIIRH)*, Spain, 2013

[4] Mohammed. F. Alhamid, Majdi Rawashdeh, Abdulmotaleb El Saddik. Towards Context-Aware Recommendations of Multimedia in an Ambient Intelligence Environment. In: *Proceedings of The 9th IEEE International Workshop on Multimedia Information Processing and Retrieval (MIPR)*, USA, 2013

[5] Majdi Rawashdeh, Heung-Nam Kim, Abdulmotaleb El Saddik. Social Media Annotation and Tagging Based on Folksonomy Link Prediction in a Tripartite Graph. In: *Proceedings of the 19th international conference on International Conference on Multimedia Modeling (MMM)*, volume 7732 of Lecture Notes in Computer Science, pp. 24 –35. China, 2013

[6] Mark Bloess, Heung-Nam Kim, Majdi Rawashdeh, Abdulmotaleb El Saddik. Knowing Who You Are and Who You Know: Harnessing Social Networks to Identify People via Mobile Devices. In: *Proceedings of the 19th international conference on International Conference on Multimedia Modeling (MMM)*, volume 7732 of Lecture Notes in Computer Science, pp. 130 –140. China, 2013

[7] Aysha Akther, Heung-Nam Kim, Majdi Rawashdeh, Abdulmotaleb El Saddik. Applying Latent Semantic Analysis to Tag-Based Community Recommendations.

In: *Proceedings of the 25th Canadian Conference on Artificial Intelligence (AI)*, volume 7310 of Lecture notes in Computer science, pp. 1-12. Canada, 2012

[8] Majdi Rawashdeh, Heung-Nam Kim, Abdulmotaleb El Saddik. Folksonomy-boosted social media search and ranking. In: *Proceedings of the 1st ACM International Conference on Multimedia Retrieval (ICMR)*, Italy, 2011

[9] Heung-Nam Kim, Majdi Rawashdeh, Abdulmotaleb El Saddik. Leveraging collaborative filtering to Tag-Based personalized search. In: *Proceedings of the 19th international conference on User Modeling, adaptation and Personalization (UMAP)*, volume 6787 of Lecture Notes in Computer Science, chapter 17, pp. 195–206. Spain, 2011

1.5 Thesis Organization

This thesis is organized as follows:

Chapter 1 presents the introduction and motivations of this work, as well as the major contributions.

Chapter 2 covers background information about social tagging which has become a major part in Web 2.0. The chapter discusses narrow and broad folksonomy in addition to the graph representation of folksonomy. Finally it presents literatures that exploit various aspects of social tagging to different personalized services.

Chapter 3 introduces a new tag-based personalized search algorithm (*FBR*) that profit from folksonomy. We aim in this chapter to build a straightforward, accurate model easily applicable to social media services. Our model explores not only personal interests but also social wisdom in the facilitation of more accurate search results

Chapter 4 presents our personalized tag recommendation/item annotation algorithm (*KatzBm25*) and how this algorithm can utilize the social tagging information to provide personalized tag recommendation, and to uncover the relevant hidden tags that have not previously annotated to the item. This chapter shows how to convert folksonomy into tripartite graph, build the adjacency matrix, computing the Katz score and finally how to compute the personalized service ranking score.

Chapter 5 present the proof-of-concept of our proposed model named *SongSter*. The *SongSter* application is a mobile application for the Android platform that has a database of different songs from which users can get song suggestions and find similar users.

Chapter 6 presents the evaluation metrics used to measure the accuracy and the performance of our algorithms, also shows our algorithms comparison results with other baseline algorithms.

Finally, Chapter 7 summarizes our research and presents the future work to be completed.

Chapter 2 Background and Related Work

This chapter provides background information about social tagging which has become a major part in Web 2.0. Narrow and broad folksonomy in addition to the graph representation of folksonomy are also discussed in this chapter. Finally we study literatures that exploit various aspects of social tagging to different personalized services.

2.1 Social Tagging

The rise of Web 2.0 has transformed users from passive consumers to active producers of content. We have seen a proliferation of social websites focusing on information sharing and collaboration. Figure 1 illustrates an example of these social websites.

Figure 1 Example of social websites

An important aspect of these social websites and platforms is social tagging; which allow users to freely annotate their media items (e.g., music, images, videos, blog entries, bookmarks, etc.) with any sort of arbitrary words, referred to as tags. Tags are keywords that describe the characteristics of the media item they are assigned to, and can be made of one or more words. Users of social tagging services are flexible and not restricted any more to a rigid hierarchy of content similar to taxonomies or predefined dictionaries when they choose their own tags. Therefore, social tagging has become a popular way to categorize, share and organize social media content. Many other names have been proposed for social tagging, including social classification, collaborative tagging, social indexing, folk classification, and free tagging (Hammond et al. 2005). In their core social tagging systems are all similar (Hotho et al. 2006), once a user is logged in to the system, he can upload an item to the system; annotate the item with arbitrary tags. The aggregation of a user's annotations is his personomy, and the collection of all personomies constitutes the folksonomy. User can navigate through his personomy as well as the whole folksonomy, once a user clicks on an item; he obtains all tags assigned to this item, all similar items tagged with the same tags, and all users who tagged this item. Folksonomy, a term coined by Thomas Vander Wal in 2005, is a portmanteau of folk and taxonomy, so folksonomy is the aggregation of the tagging efforts of all users, which allow any user to navigate freely between items, tags, and other users.

Tagging is not only an individual process of categorization, but implicitly it is also a social process of indexing, a social process of knowledge construction (Hassan-Montero et al. 2006). Tag clouds are widely used to visualize a set of related tags which best describe an item within a folksonomy, by giving more weight or importance either in color or font size to the most frequently used tags (Zanardi et al. 2008). In tag clouds,

when a user clicks on a tag, a user obtains an ordered list of tag-described items, as well as a list of others related tags (Hassan-Montero et al. 2006). Figure 2 shows an example of a tag cloud.

Figure 2 Bibsonomy tag cloud[5]

2.2 Broad and Narrow Folksonomy

Users share their resources with their tags, generating an aggregated tag-index so-called folksonomy (Hassan-Montero et al. 2006). Depending on how the social tagging is implemented within a social website, folksonomy can be classified into broad or narrow folksonomy. The main difference between narrow and broad is who has the right or privilege to tag a given item. Broad folksonomy has many users tagging the same item and every user can annotate any given item with his own tags in his own vocabulary. Figure 3 illustrates broad folksonomy, where the content creator makes the item available to all groups (**A**, **B**, **C**), and annotates the item with tags (**1**, **2**) only. Group **A** annotate the

[5] http://www.bibsonomy.org/

item with tags (**1**, **2**), and group **B** assigns tags (**2, 3**) to the same item. Although tag **3** was not added by the content creator originally, but he can still retrieve the item using tag **3** as well as tag **1** and tag **2**. Similar to this is group **C**, where each user in this group can retrieve or consume the item by tag **3** even though the users in this group never tagged the item in the past. Any tag can be applied more than once to the item by different users, for example tag **2** was assigned to the item by the content creator, group **A**, and group **B**. A real example of board folksonomy is adding a link to delicious.com. Users of delicious website have the ability to add, tag and browse any link within the delicious community.

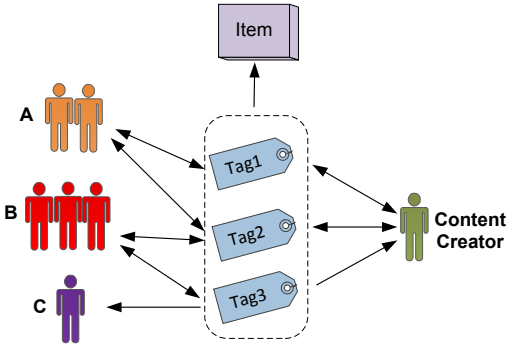

Figure 3 Broad Folksonomy

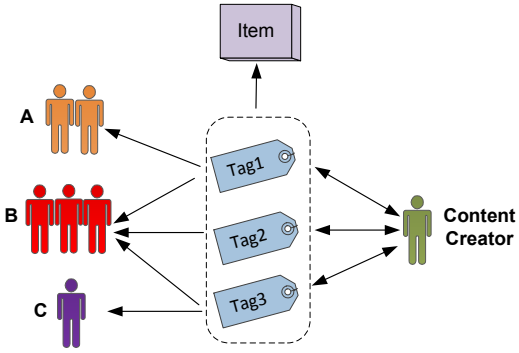

Figure 4 Narrow Folksonomy

On the other hand is narrow folksonomy as depicted in Figure 4, the content creator posts the item and annotates it with one or more tags, and all other users are dependent on the creator's tags. In Figure 4, the content creator assigns tags **1**, **2** and **3** to the item; users may use any of these tags to retrieve the item. For example users of group **C** may use tag **3** to retrieve the item, while users of group **B** may use tags **1**, 2 and **3,** and users of group **A** may use tag **1**. Flicker is an example of narrow folksonomy, where the owner alone has the right to upload any photo, and annotates it with one or more tags. Other users of

flicker can browse their own photos as well other people's photos but they have the right to tag their own photos only.

2.3 Folksonomy Social Graph Representation

Some researchers (Lambiotte et al. 2006) viewed folksonomy as a social graph; a graph composed of three types of nodes: the users, the items—such as a video, image, text, audio, URL, etc— and the tags that are used by the users to describe the items. Users may assign one tag or several tags to the item. Figure 5 illustrates the graph representation of the folksonomy.

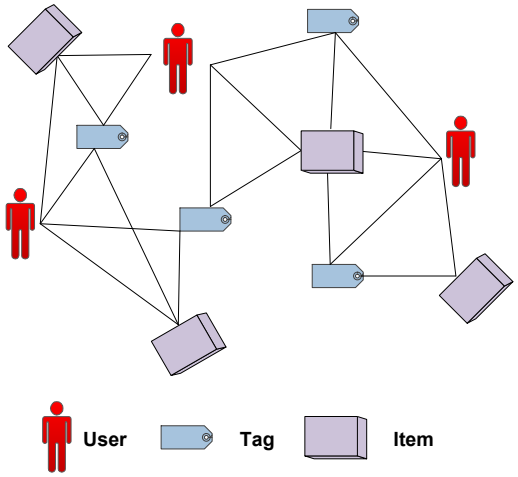

Figure 5 Graph structure of a folksonomy

2.4 Related Work

A number of studies examine various aspects of social tagging in different personalized services. However, this section mainly reviews literature that uses social data, particularly social tagging, to enhance tag recommendation, tag-based search, and item annotation.

2.4.1 Tag Recommendation and Item Annotation

Recent years have seen an increasing number of studies in the area of social tagging services. Some early work in using social tagging is presented by (Hotho et al. 2006). The authors proposed a formal model for folksonomies, called FolkRank. It computes a PageRank vector from the social graph induced by the Folksonomy. FolkRank used a folksonomy structure for tag recommendation within the tagging system. A User-centric Tag Model (*UCTM*) is proposed by (Wetzker et al. 2010). This model uses a 3-order tensor to form the association between users, items, and tags. The model maps individual tag vocabularies, called personomies, on the corresponding folksonomies with the tagged items. The UCTM model can be applied for tag-based search and tag recommendation. A tag ranking scheme is proposed by (Liu et al. 2009). The scheme captures the tag's relevance level to the image content and ranks these tags accordingly. The relevance score of a tag is quantified based on probability density estimation. Thereafter, a random walk over a tag similarity graph is performed to refine the relevance score. This scheme can be applied mainly in tag recommendation, and image group recommendation. A popular tag model is presented by (Jäschke et al 2008); in this model the most-popular tags used for a particular resource will be recommended.

(Ramezani 2011) proposed a weighted directed graph which models the informational channels of a folksonomy and applied the PageRank algorithm to this graph so as to enhance graph-based tag recommendation techniques. A modified form of K-Nearest Neighbor for tag recommendation in folksonomies is presented in (Gemmell et al. 2009), by incorporating user, resource and tag information into the algorithm. (Gemmell et al. 2011) also proposed a linear-weighted hybrid algorithm for tag recommendation. The algorithm was analyzed across six real-world datasets with different characteristics. In

(Budura et al. 2009), a tag recommendation method based on the neighborhood of a document-tag graph is proposed. The tag rank relies on the occurrence of the tag in the neighbors of an active resource, co-occurrence of the tag to a set of tags inferred from the active resource, and the distance between resources and tags. (Guan et al. 2009) studied personalized tag recommendation by considering association between resources and association between resources and tags. The study considers the preference vector of tags that the user used more frequently to generate personalized tag recommendations. Tags are ranked by incorporating document ranking and active users' most frequent tags in a ranking function.

(Lipczak et al. 2010) proposed a system that recommends tags based on merged scores of user profile related tags and resource related tags. The system uses the content of each added post to update all stored information so that new recommendations reflect users' current interests. (Song et al. 2011) tackled the issue of tag recommendation from a machine learning perspective. The authors proposed two document-centered approaches for recommending tags in social networking systems. The first approach is a graph-based approach where the tagged data is represented by two bipartite graphs, and the second one is a prototype-based approach that uses a sparse multiclass Gaussian process classifier for efficient document classification. (Rendle et al. 2010) presented a new factorization model for tag recommendation which is a special case of the Tucker decomposition. The new factorization model extends the Bayesian personalized optimization criterion to the task of tag recommendation and explicitly reflects the pairwise interactions between users, items and tags. (Hamouda et al. 2011) proposed a personalized tag recommendation system for social bookmarking systems using

collaborative filtering. The proposed system recommends tags based on similar users and similar bookmarks.

(Bu et al. 2010) proposed a hypergraph model which combines social media information and music acoustic-based content. Another model was proposed by (Horsburgh et al. 2011). They incorporated content-based representation into a tag-based recommender system. This model constructs a tag-track matrix by including audio content into a tag space and then learning hybrid concepts using latent semantic analysis. In another work, (Levy et al. 2009) built a representation matrix that combines clustered content representation and tags, and then employed probabilistic latent semantic analysis (PLSA) to learn new concepts which generalize both content and tags. (Miotto et al. 2012) presented an approach to music search and discovery based on a graph representation. This approach combines acoustic similarities and tags in a single probabilistic framework. Tags and the acoustic similarity were used together at the same time for ranking music. (Font et al. 2012) utilized an online audio sharing site, Freesound, to evaluate the performance of four variants of algorithms on tag recommendation based on the tag semantic similarity derived from tag co-occurrences in the Freesound folksonomy. (Symeonidis et al. 2008) presented a tensor model to recommend music according to users' multimodal perception of music, by applying latent semantic analysis and dimensionality reduction using the higher order singular value decomposition. (Tatli et al. 2011) proposed a method for creating music recommendation based on the user-supplied tags that are augmented with a hierarchical structure extracted for top level genres from DBpedia. This approach aimed to represent individual tracks (songs) in a lower dimensional space and to use multi-domain information in recommendations.

(Krestel et al. 2009) investigated a method based on the latent topic model. This method applied Latent Dirichlet Allocation (LDA) to tag an item. The proposed method extracts latent topics from a dense folksonomy to be used to recommend additional tags for new items. A similar work is addressed by (Diaz-Aviles et al. 2010) for a new item, the proposed approach creates an ad hoc corpus of similar items, and then applies LDA to extract the latent topics for the item and the associated corpus. The automatic tagging for the item is based on the most likely tags derived from the latent topic identified. (Liben-Nowell et al. 2007) proposed a personalized *PageRank* algorithm based on a random walk with restarts to find the latent relevant tags. The algorithm tackles tag annotation as a link predication where it predicts the edge between two nodes that will be added to the social network at two different times. Table 1 summarizes the related work by describing the difference from our tag recommendation algorithm (*KatzBm25*).

In chapter 6, we will compare our tag-recommendation/item annotation algorithm (*KatzBm25*) against the following baseline algorithms: i) User-Centric Tag Model (*UCTM*) proposed by Wetzker et al. (2010), ii) the *FolkRank* algorithm which is one of the most frequently cited studies among folksonomy-based algorithms (Hotho et al. 2006), iii) the most *Popular Tag* algorithm (Jäschke et al. 2008), iv) the personalized *PageRank* algorithm based on a random walk with restarts (Liben-Nowell et al. 2007), and v) the LDA-based algorithm described in (Krestel at el. 2009). These baseline algorithms are closely relevant to our tag recommendation algorithm, that make use of social tagging without the need to analyze the media content itself, rather it utilizes the social tagging information to suggest relevant tags to the users during their tagging annotation process. There are no source codes available to the baseline algorithms; therefore we implemented these algorithms based on the published papers

Table 1 Tag recommendation related work comparison summary

Paper	Difference
Liu et al. 2009	The presented scheme can be applied to images only, whereas our algorithm (*KatzBm25*) can be applied to any type of media.
Ramezani 2011	The tag score is calculated based on directed weighted tripartite graph, while our algorithm (*KatzBm25*) uses the undirected weighted tripartite graph.
Budura et al. 2009	This algorithm computes the tag score with respect to the neighbors of an active item and neglects the role of the user.
Guan et al. 2009	Can be applied to text documents only.
Lipczak et al. 2010	Uses rule-based approaches which require a significant amount of expert experience to build.
Song et al. 2011	This approach estimates the tag rank based on a weighted directed bipartite graph
Hamouda et al. 2011	Applies the collaborative filtering techniques only, so the algorithm fails to recommend tags to cold start users, especially if the data is sparse.
Bu et al. 2010, Horsburgh et al. 2011, Miotto et al. 2012, Levy et al 2009, Font et al. 2012, Symeonidis et al. 2008, Tatli et al. 2011	These music-related studies differ from our algorithm in that they attempted to analyze music content, e.g., acoustic or sound features. Our algorithm, does not require analysis of the actual media content itself, rather it uses the tripartite relations inherent in a folksonomy and thus suggest relevant tags to the users during their tagging annotation process.

2.4.2 Tag-based Search

(Bao et al. 2007) proposed two algorithms, SocialSimRank and SocialPageRank. The SocialSimRank algorithm exploits the similarity between a user's queries and the corresponding tags; whereas the SocialPageRank captures a page's popularity based on its annotations. (Xu et al. 2008) presented a personalized search framework for item recommendation. This framework has the ability to rank a webpage based on the topic

matching to the user's interests and the input query in the tag space. In another study, (Wu et al. 2006) considered social tags' emergent semantics and applied these semantics to searches for Web bookmarks. They proposed a global semantic model that could help to disambiguate tags and group synonymous tags together in concepts. (Levy et. al 2008) presented the Latent Semantic Analysis Model (*LSAM*) to uncover emergent semantics from tags labeled to music. (Vallet et al. 2010) proposed the *BM25-Based Personalization Model* (*CosBM25*) which is an adaption of the BM25 probabilistic model to folksonomy systems based on the user and item representation. (Wang et al. 2010) presented a collaborative item search model, by assuming conditional independence between users and the tags they had given an item, the model integrated the collaborative nature of recommender systems with smoothing methods from information retrieval.

In another study, (Noll et al. 2007) presented a personalized Web search model based on social bookmarking and tagging. In this model, data collection and user profiling are separated from the information system, whose content and indexed documents are being searched. In addition, this model is search-engine independent, i.e., the users are free to select the search engine they prefer, even if it does not natively support personalization. (Yanbe et al. 2007) exploited data from social bookmarking to enhance Web searches. Similarly, (Heymann et al. 2008) analyzed social tags on the social bookmarking services Delicious to show how social bookmarking services can enhance Web searches. They also observed an overlap between tags found on Delicious and search query terms. (Vallet et al. 2009) utilized users' tagging information within a social tagging service to personalize a retrieval system. They employed the concept of global tag importance to enhance traditional Web search methods. All these studies differ from our work in that they mainly aimed at improving existing Web searches through integration with social

bookmarking and tagging. Moreover, some studies integrated social networks with social searches. (Carmel et al. 2009) tackled personalized social searches based on a user's social relationships in several social networks, re-ranking the search results according to social relationships with individuals in a user's social network. (Schenkel et al. 2008) presented a similar concept, proposing the ContextMerge algorithm to support efficient user-centric searches in social networks, dynamically including related users in its execution. This algorithm relies mainly on the social expansion which considers the strengths of relationships among users.

Some studies focused on query expansions for personalization, by utilizing folksonomies. (Biancalana et al. 2009) presented a model focusing on how to personalize Web searches using query expansion. This model inserted the metadata corresponding to Web resources' assigned tags as a third dimension, extending the two-dimensional matrix of co-occurrence. Similarly, the proposed model of (De Meo et al. 2010) expanded a user's search query tags to enrich the original search query, storing both the expanded and the original search tags in the user profile, to improve the performance of content-based recommender systems via folksonomy. The main difference between these studies and our work is our study does not consider an explicit query expansion. Instead, our model implicitly contains the effect of the expansion.

Similar to our work, (Zanardi et al. 2008) proposed a social ranking algorithm, to answer a user's query. It aimed to transparently improve content searches based on emergent tags' semantics and exploited users' similarities according to tag overlap to improve accuracy, as well as employing tags' similarities based on their association with the content to increase coverage. Although the algorithm exploit tag-tag similarities derived from tag-item relationships, but it explicitly expands the user queries. More

importantly, the algorithm did not consider a user's preferences regarding tags when expanding such tags for a given query. (Wetzker et al. 2010) proposed a user-centric tag model to map individual tag vocabularies, called personomies, onto the corresponding folksonomies using resources as intermediates for recommending tags, as well as search resources. This model can be used for tag recommendation and tag-based search. Table 2 summarizes the related work by describing the difference from our tag-based search recommendation algorithm (*FBR*).

For comparison purposes in chapter 6, we experimented our tag-based search algorithm (*FBR*) with the following algorithms: i) Social Ranking without query expansion (*SRk0*) and with query expansion using the k most similar tags (*SRk5*) described in (Zanardi et al. 2008), ii) the User-Centric Tag Model (*UCTM*) presented in (Wetzker et al. 2010) , iii) the BM25-Based Personalization Model (denoted *CosBM25*) described in (Vallet et al. 2010), and iv) the Latent Semantic Analysis Model (*LSAM*) described in (Levy et al. 2008). Because those algorithms were mostly tailored to our search scenario with social tagging, we implemented them to the best of our knowledge, based on the published papers.

Table 2 Tag-base search related work comparison summary

Paper	Difference
Noll et al. 2007, Yanbe et al. 2007, Heymann et al. 2008, Vallet et al. 2009, Wu et al. 2006, Xu et al. 2008	These approaches are applicable to web searches only through integration with social bookmarking and tagging.
Carmel et al. 2009, Schenkel et al. 2008,	These approaches rely only on the user-user similarity, whereas our tag-based search algorithm (*FBR)* considers both the tag-tag, and the item-item similarity.
Biancalana et al. 2009, De Meo et al. 2010	These algorithms consider an explicit query expansion. Instead, our tag-based search algorithm

	implicitly contains the effect of the expansion.
Zanardi et al. 2008	This algorithm explicitly expands the user queries. More importantly, the algorithm did not consider a user's preferences regarding tags when expanding such tags for a given query.
Wetzker et al. 2010, Wang et al. 2010	These algorithms use collaborative filtering technologies to identify similar tags only, instead our tag-based search algorithm identify not only similar tags but also similar items.

Chapter 3 Tag-based Personalized Search

In this chapter, we propose a new tag-based personalized search algorithm named Folksonomy-Boosted Ranking (*FBR*). Our *FBR* algorithm first determines the similarities among resources and among tags. After that the algorithm builds two latent models: i) the latent tag preference model that reflects how a certain user has assigned tags similar to a given tag, and ii) the latent tag annotation model that captures how users have tagged a certain tag to resources similar to a given resource. Then the algorithm seamlessly map the tags on the items depending on a particular user's query in order to find the most attractive media content relevant to the user needs.

3.1 Definitions and Notations

Before going into further detail, we formalize notations and introduce concepts that will be exploited in this thesis. We henceforth use the term *items* to refer to social media resources. Matrices are denoted using boldface, upper-case letters, such as **A**; whereas the corresponding italicized, lower-case letters with two subscript indices represent entries in the matrices, such as $a_{u,i}$. Table 3 summarizes notations used in the rest of this chapter.

Table 3 The meaning of notations

Notations	Meaning
U	Set of users
T	Set of tags
I	Set of items
A ($\tilde{\mathbf{A}}$)	User-tag matrix (Column normalized matrix of **A**)
C	User-item matrix
N ($\tilde{\mathbf{N}}$)	Tag-item matrix (Column normalized matrix of **N**)
E ($\mathbf{E}^k$)	Tag-Tag similarity matrix (Matrix of **E** in which each row contains only k most similar tags)
H ($\mathbf{H}^{k'}$)	Item-item similarity matrix (Matrix of **H** in which each row contains only k' most similar items)
P	User-tag latent preference matrix
W	Tag-item latent annotation matrix

3.2 Folksonomy Model

For a set of users $U = \{u_1, u_2,...,u_{|U|}\}$, a set of tags $T = \{t_1,t_2,...,t_{|T|}\}$, and a set of items $I = \{i_1, i_2,...,i_{|I|}\}$, (Hotho et al. 2006) formalized a folksonomy as a tuple $F := (U, T, I, Y)$ where $Y \subseteq U \times T \times I$ is a ternary relationship, called a *tag assignment*. Therefore, a folksonomy can be viewed as a three-dimensional space of users, tags, and items; consequently this three-dimensional space can be projected onto three two-dimensional matrices, as shown in Figure 6.

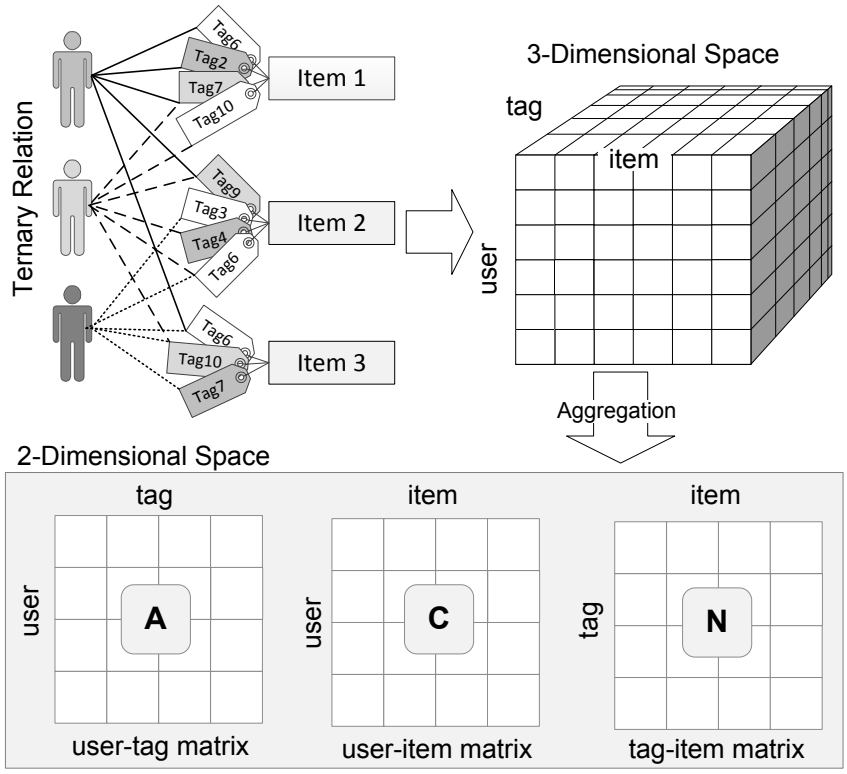

Figure 6 Projecting a 3-dimensional space into three 2-dimensional spaces

We first define the three matrices obtained by aggregating over items, tags, and users, as follows:

- User-Tag matrix $\mathbf{A}=[a_{u,t}]_{|U|\times|T|}$, where $a_{u,t}$ represents the number of items user u has tagged with tag t.
- User-Item matrix $\mathbf{C}=[c_{u,i}]_{|U|\times|I|}$, where $c_{u,i}$ represents the number of tags that user u has assigned to item i.
- Tag-Item matrix $\mathbf{N}=[n_{t,i}]_{|T|\times|I|}$, where $n_{t,i}$ represents the number of users who have tagged item i with tag t.

We normalize both user-tag matrix **A**, and tag-item matrix **N** and discuss them in more detail in the next sections.

3.3 Similarities from Folksonomy

In this thesis, we calculate two types of similarities, the tag-tag similarity and the item-item similarity. For that, we can make use of different matrices, in turn, leading to different similarity values. When computing the similarity between two tags, we can use not only the user-tag matrix **A** but also the tag-item matrix **N**. As pointed out in previous studies (Bischoff et al. 2008; Li et al. 2008), tags labeled to an item can cover the main concepts and topics for that item. If two tags frequently appear together in multiple items, those tags would involve the same or closely related semantics. In this case, we could successfully measure tags' similarities in terms of items. As mentioned earlier, different users often have different personal tagging behaviors, so sometimes tags' similarities in terms of users may not be reliable. In order to personalize the search results according to a searcher's topics of interest, rather than tagging activities of other users similar to the searcher, we make use of **N** for computing the similarity between tags instead of using **A**. As for items' similarities, if two items frequently contain a set of tags in common, those items may share similar topics that emerge organically from the tags. In addition, the statistical information (the number of tags that a user assigned to an item) contained in the user-item matrix **C** would contribute toward the item similarity less than the statistical information (the number of users who tagged an item with the same tag) contained in **N**. For this reason, we use **N** for determining items' similarities instead of using **C**.

In this thesis, we employed the cosine-based similarity as number of studies has successfully used the well-known cosine similarity method for computing the similarity in social tagging (Jäschke et al. 2008; Kim et al. 2010; Xu et al. 2008; Zanardi et al. 2008).

3.3.1 Tag-Tag Similarity

Formally, the tag-item matrix **N** can be decomposed into row vectors:

$$\mathbf{N} = [\mathbf{t}_1, \ldots, \mathbf{t}_{|T|}]^{\mathbf{T}},$$

where $\mathbf{t}_a = [n_{a,1}, \ldots, n_{a,|I|}]$, for $a = 1, \ldots, |T|$

Therefore, we calculate the cosine similarity, which quantifies the similarity of a pair of vectors according to their angle, between given two row vectors ($\mathbf{t}_x$ for tag x and $\mathbf{t}_y$ for tag y) by the following equation:

$$e_{x,y} = \cos(\mathbf{t}_x, \mathbf{t}_y) = \frac{\mathbf{t}_x \cdot \mathbf{t}_y}{\|\mathbf{t}_x\| \times \|\mathbf{t}_y\|} \quad (1)$$

The basic idea behind the similarity calculation is that we first investigate items tagged with two tags and then examine how frequently other users have used these two tags for labeling such items.

Finally, for $|T|$ tags, the similarity of tags can be represented as a tag-tag similarity matrix $\mathbf{E}=[e_{x,y}]_{|T|\times|T|}$, where both rows and columns represent tags. We consider the k most similar tags of each tag. Therefore, we set $e_{x,y}$, which represents the x-th tag for the y-th tag, to the similarity value between a pair of tags x and y if the corresponding similarity value is greater than the k highest similarity value in the y-th column of **E** and 0 otherwise. In other words, each column in the matrix **E** contains at most k non-zero values indicating the similarity values between the corresponding column tag and the k *most similar tags*. Note that all entries on the main diagonal in the matrix equal 1, i.e., for any two tags, x and y, where $x \in T$ and $y \in T$, if x is equal to y, then $e_{x,y}=1$. We denote the resulting matrix as $\mathbf{E}^k$ where k is the number of similar tags. To build a model for each user on the corresponding tag, we use the non-zero entries of each column, called the k

most similar tags. A key objective in replacing less similar tags with zero is to avoid containing unnecessary information for each tag to reduce the computational cost.

3.3.2 Item-Item Similarity

Alternatively, the matrix **N** can also be decomposed into column vectors:

$$\mathbf{N} = [\mathbf{i}_1, \ldots, \mathbf{i}_{|I|}],$$

where $\mathbf{i}_b = [n_{1,b}, \ldots, n_{|T|,b}]^{\mathbf{T}}$, for $b = 1, \ldots, |I|$

With respect to the item-item similarity, two column vectors are used. Formally, the similarity between two items x and y is given by the following equation:

$$h_{x,y} = \cos(\mathbf{i}_x, \mathbf{i}_y) = \frac{\mathbf{i}_x \cdot \mathbf{i}_y}{\|\mathbf{i}_x\| \times \|\mathbf{i}_y\|} \qquad (2)$$

In contrast to the tag-tag similarity, this calculation looks into the tags labeling both items and, then examines how frequently such tags have been tagged to the other items.

Finally, for $|I|$ items, the similarity of items can be represented as an item-item similarity matrix $\mathbf{H} = [h_{x,y}]_{|I| \times |I|}$. Analogous to in the k most similar tags, we prune any unnecessary values for each item. We denote the resulting matrix by $\mathbf{H}^{k'}$, where each column stores k' most similar items to the column's corresponding item; in the matrix $\mathbf{H}^{k'}$, $h_{x,y}$ is set to the similarity value between two items x and y, if the corresponding similarity value is greater than the k' highest similarity value in the y-th column of $\mathbf{H}^{k'}$ and 0 otherwise. Additionally, all entries on the main diagonal in the similarity matrix are also equal to 1. Finally, we use the non-zero entries per each column (the *k' most similar items*) to build a model for each tag on the corresponding item

3.4 Latent Modeling of tags and items annotation

Since every user has different tastes on items, we should provide different search results to different users to make the search more relevant to each user's inquiries. Therefore, to support accurate personalized searches, we build two latent models.

3.4.1 Latent tag preference for users

Our first model, namely, a latent tag preference model, reflects the manner in which a given user tends to assign tags similar to a given tag. To this end, we build a new user-tag matrix, which we derive from the product of two matrices:

$$\mathbf{P} = \tilde{\mathbf{A}}\,\mathbf{E}^k \tag{3}$$

where $\tilde{\mathbf{A}}$ is the normalized matrix of $\mathbf{A}$ and $\mathbf{E}^k$ is a tag-tag similarity matrix that stores the *k most similar tags* per each column. For this matrix normalization $\tilde{\mathbf{A}}$, each column vector of tags in the user-tag matrix $\mathbf{A}$ is normalized as $\| \mathbf{a}_v \| = 1$, for $v = 1, 2, \ldots, |T|$. Therefore, the normalized user-tag matrix $\tilde{\mathbf{A}}$ can be defined as $\tilde{\mathbf{A}} = [\tilde{a}_{u,t}]_{|U|\times|T|}$, where $\tilde{a}_{u,t}$ is obtained as follows:

$$\tilde{a}_{u,t} = a_{u,t} \Big/ \sqrt{\sum_{j=1}^{|U|} (a_{j,t})^2} \tag{4}$$

In the matrix $\mathbf{P}$, each entry represents the corresponding user's preference value for the corresponding tag. More precisely, let $\tilde{\mathbf{a}}_u{}^T$ denote the u-th row vector of $\tilde{\mathbf{A}}$ and $\mathbf{e}_t$ the t-th column vector of $\mathbf{E}^k$. Then, an entry in the $\mathbf{P}$'s u-th row, t-th column, $p_{u,t}$, can be filled by the dot product of the two row vectors:

$$p_{u,t} = \tilde{\mathbf{a}}_u^T \cdot \mathbf{e}_t = \sum_{j=1}^{|T|} \tilde{a}_{u,j} \times e_{j,t} \tag{5}$$

The salient concept behind this computation process is that the more a tag resembles a user's tag the more the tag influences the calculation of the preference value. Normalization of each column has the effect that, in Equation 5, tags labeled by numerous users contribute less than tags labeled by a small number of users. Figure 7 illustrates the process of building the user-tag preference model **P**.

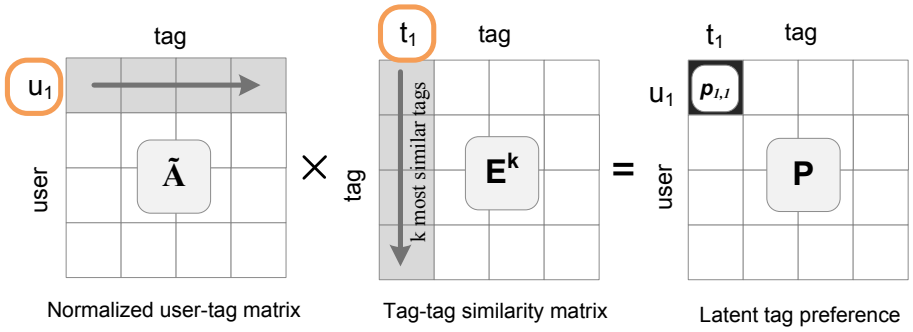

Figure 7 The process of computing the latent tag preference model P

3.4.2 Example: building a latent preference model

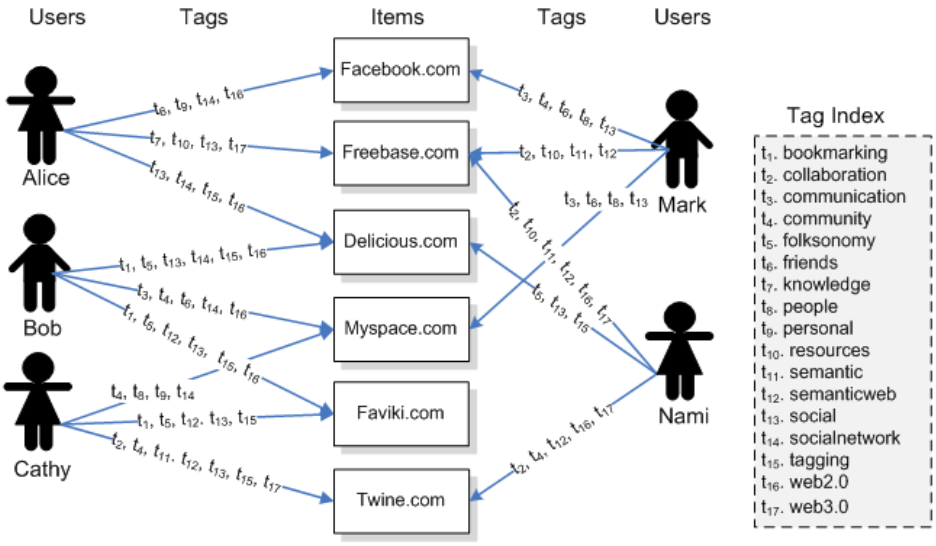

Figure 8 An example of tag assignments of five users

To illustrate a simple example of building the latent tag preference model, consider five users' tag assignments on six items, shown in Figure 8. We utilize this social tagging as reference examples throughout this section. When we aggregate the tag assignments over items, we can obtain the user-tag matrix shown in Table 4.

Table 4 An example of a user-tag matrix, A

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}
Alice	0	0	0	0	0	1	1	0	1	1	0	0	2	2	1	2	1
Bob	2	0	1	1	2	1	0	0	0	0	0	1	2	2	2	3	0
Cathy	1	1	0	2	1	0	0	1	1	0	1	2	2	1	2	0	1
Mark	0	1	2	1	0	2	0	2	0	1	1	1	2	0	0	0	0
Nami	0	2	0	1	1	0	0	0	0	1	1	2	1	0	1	2	2

From the example matrix **A**, we can easily derive the normalized matrix **Ã**. For example, the normalized value of tag t_{13} (social) for Alice is calculated as $\tilde{a}_{Alice,social} = 2/\sqrt{2^2+2^2+2^2+2^2+1^2} = 0.49$, whereas the value of tag t_{14} (socialnetwork) is $\tilde{a}_{Alice,socialnetwork} = 2/\sqrt{2^2+2^2+1^2} = 0.67$. Even though the number of items Alice annotated with tag t_{13} equals that of items she labeled with tag t_{14}, the latter tag, "socialnetwork," possesses more influence than the former tag does with regard to her preferences, because the other users also commonly used the tag "social."

Table 5 Tag similarities between tag "bookmarking" and its five most similar tags

	t_1 (bookmarking)	t_5 (folksonomy)	t_{13} (social)	t_{15} (tagging)	t_{16} (web2.0)
t_1 (bookmarking)	1	0.95	0.76	0.84	0.60

Now, in computing Alice's latent preference for tag t_1 (bookmarking), when we calculate the tag-tag similarity between tag t_1 and every other tag via Equation 1 and consider the five most similar tags, we determine the tags and similarities shown in Table

5. From the normalized and similarity values, Alice's latent preference value for tag t_1, $p_{Alice,t1}$, is calculated as follows: $p_{Alice,bookmarking}$ = (0×1) + (0×0.95) + (0.49×0.76) + (0.32×0.84) + (0.49×0.6) = 0.92. According to the tendency that Alice has used similar tags, the latent value for the tag "bookmarking" can be estimated as 0.92 even though she has not previously used this tag.

In an analogous fashion, all users' tag preferences for all tags can be computed as shown in Table 6.

Table 6 A latent tag preference model, **P**, for the given example of social tagging

	Alice	Bob	Cathy	Mark	Nami
t_1 (bookmarking)	0.922	3*	1.732*	0.368	1.125
t_2 (collaboration)	1.115	0.258	1.881*	1.214*	2.697*
t_3 (communication)	1.555	1.331*	1.366	2.563*	0
t_4 (community)	0.789	1*	1.589*	2.289*	0.377*
t_5 (folksonomy)	1.058	3.193*	1.846*	0.416	1.258*
t_6 (friends)	1.587*	1.571*	1.902	2.781*	0.267
t_7 (knowledge)	1.866*	0	1.094	1.382	2.248
t_8 (people)	1.555	1.331	1.366*	2.563*	0
t_9 (personal)	1.587*	1.304	1.367*	2.514	0
t_{10} (resources)	1.866*	0.183	1.459	1.565*	2.614*
t_{11} (semantic)	1.798	0	1.352*	1.481*	2.643*
t_{12} (semanticweb)	0.667	0.316*	1.746*	1.430*	2.746*
t_{13} (social)	1.243*	3.144*	1.749*	0.485*	1.350*
t_{14} (socialnetwork)	1.397*	1.919*	1.020*	1.686	0.554
t_{15} (tagging)	1.145*	3.176*	1.833*	0.440	1.311*
t_{16} (web2.0)	1.728*	2.801*	1.526	0.471	1.263*
t_{17} (web3.0)	1.115*	0.258	1.881*	1.214	2.697*

(* tags that are previously tagged by the user)

3.4.3 Latent tag annotations for items

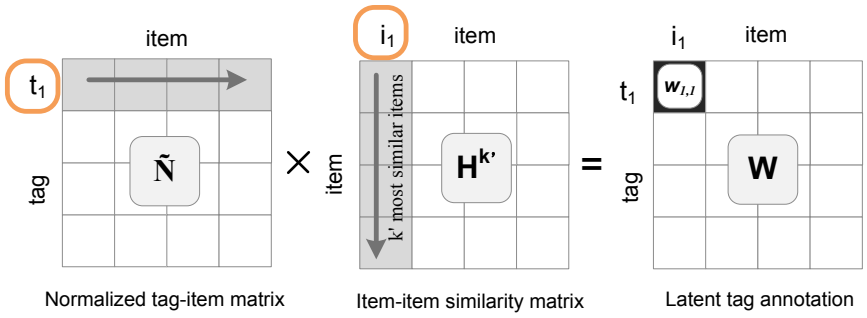

Figure 9 The process of computing the latent tag annotation model **W**

For the second model, we capture how a certain tag has been labeled to items similar to a given item. For this, we build a new tag-item matrix using the product of two matrices:

$$\mathbf{W} = \tilde{\mathbf{N}}\, \mathbf{H}^{k'} \tag{6}$$

where $\tilde{\mathbf{N}}$ is a normalized tag-item matrix of $\mathbf{N}$ and $\mathbf{H}^{k'}$ is an item-item similarity matrix that stores *k' most similar items* per each column. Analogous to the matrix $\tilde{\mathbf{A}}$, for the normalization of the tag-item matrix $\mathbf{N}$, each column vector of items is normalized as $\| \mathbf{i}_b \| = 1$, for $b = 1, 2, \ldots, |I|$. Formally, we can define the normalized tag-item matrix $\tilde{\mathbf{N}}$ as $\tilde{\mathbf{N}} = [\tilde{n}_{t,i}]_{|T|\times|I|}$, where we obtain $\tilde{n}_{t,i}$ by the following equation:

$$\tilde{n}_{t,i} = n_{t,i} \Big/ \sqrt{\sum_{j=1}^{|T|} (n_{j,i})^2} \tag{7}$$

In the matrix $\mathbf{W}$, each entry represents a latent annotation value of the corresponding tag for the corresponding item. More precisely, let $\tilde{\mathbf{n}}_t^T$ denote the t-th row vector of $\tilde{\mathbf{N}}$ and $\mathbf{h}_i$ the i-th column vector of $\mathbf{H}^{k'}$. Then, $\mathbf{W}$'s entry in the t-th row, the i-th column, $w_{t,i}$, can be filled by the dot product of the two vectors:

$$w_{t,i} = \tilde{\mathbf{n}}_t^T \cdot \mathbf{h}_i = \sum_{j=1}^{|I|} \tilde{n}_{t,j} \times h_{j,i} \quad (8)$$

During this computation process, items that are more similar to a particular item contribute more to estimating a weight of that item. By normalizing each column in the matrix **N**, items tagged with fewer tags present greater contributions in Equation 8 than do items tagged with more tags. Figure 9 illustrates the process of building this tag-item weight model, **W**.

3.4.4 Example: building a latent tag annotation model

Table 7 An example of a tag-item matrix **N**, shown with normalized values for each entry

	Delicious	Facebook	Faviki	Freebase	MySpace	Twine
t_1 (bookmarking)	1 (0.18)		2 (0.44)			
t_2 (collaboration)				2 (0.38)		2 (0.45)
t_3 (communication)		1 (0.30)			2 (0.42)	
t_4 (community)		1 (0.30)			2 (0.42)	2 (0.45)
t_5 (folksonomy)	2 (0.36)		2 (0.44)			
t_6 (friends)		2 (0.60)			2 (0.42)	
t_7 (knowledge)				1 (0.19)		
t_8 (people)		1 (0.30)			2 (0.42)	
t_9 (personal)		1 (0.30)			1 (0.21)	
t_{10} (resources)				3 (0.57)		
t_{11} (semantic)				2 (0.38)		1 (0.22)
t_{12} (semanticweb)			2 (0.44)	2 (0.38)		2 (0.45)
t_{13} (social)	3 (0.54)	1 (0.30)	2 (0.44)	1 (0.19)	1 (0.21)	1 (0.22)
t_{14} (socialnetwork)	2 (0.36)	1 (0.30)			2 (0.42)	
t_{15} (tagging)	3 (0.54)		2 (0.44)			1 (0.22)
t_{16} (web2.0)	2 (0.36)	1 (0.30)	1 (0.22)	1 (0.19)	1 (0.21)	1 (0.22)
t_{17} (web3.0)				2 (0.38)		2 (0.45)

We now elaborate on building a latent tag annotation model. We concentrate on the example in Figure 8. However, in this instance we obtain the tag-item matrix **N** by aggregating the tag assignments over users, as shown in Table 7.

Table 8 The item similarities between "Delicious.com" and every other item

	Delicious	Facebook	Faviki	Freebase	MySpace	Twine
Delicious	**1**	**0.38**	**0.78**	0.17	**0.34**	0.32

(Boldface entries correspond to the four most similar items)

When we apply Equation 2 to compute item similarities between "*Delicious.com*" and other items, we can calculate those similarities as shown in Table 8. First, we calculate a latent annotation value for tag t_{13} (social) on the item "*Delicious.com*." From the row labeled "t_{13} (social)" in Table 7 and the second row labeled "Delicious" in Table 8, the latent value is estimated as follows: $w_{social,Delicious} = (0.54 \times 1) + (0.30 \times 0.38) + (0.44 \times 0.78) + (0.19 \times 0.17) + (0.21 \times 0.34) + (0.22 \times 0.32) = 1.17$; however, when we consider only the four most similar items (i.e., Delicious, Facebook, Faviki, and Twine), we compute the value as $w_{social,Delicious} = 1.065$. As for tag t_6 (friends) on *Delicious.com*, we are able to estimate that as $w_{friends,Delicious} = 0.369$. That is, no matter whether the tag "friends" has been used to tag *Delicious.com*, we can predict the tag's latency regarding the item by considering other similar items annotated with "friends." Table 9 shows all tag latency values for any given item using the four most similar items.

Table 9 A latent tag annotation model, **W**, for the given example of social tagging

	Delicious	Facebook	Faviki	Freebase	MySpace	Twine
t_1 (bookmarking)	0.522*	0.068	0.577*	0.156	0.061	0.249
t_2 (collaboration)	0	0.121	0.306	0.680*	0.125	0.703*
t_3 (communication)	0.255	0.695*	0	0	0.701*	0
t_4 (community)	0.255	0.815*	0.196	0.302	0.826*	0.447*
t_5 (folksonomy)	0.701*	0.136	0.718*	0.187	0.121	0.307
t_6 (friends)	0.369	0.996*	0	0	0.986*	0
t_7 (knowledge)	0	0	0.055	0.189*	0	0.128
t_8 (people)	0.255	0.695*	0	0	0.701*	0
t_9 (personal)	0.185	0.498*	0	0	0.493*	0
t_{10} (resources)	0	0	0.164	0.567*	0	0.383
t_{11} (semantic)	0	0.060	0.207	0.529*	0.063	0.479*
t_{12} (semanticweb)	0.342	0.121	0.742*	0.806*	0.125	0.894*
t_{13} (social)	1.065*	0.763*	1.012*	0.558*	0.737*	0.716*
t_{14} (socialnetwork)	0.614*	0.831*	0.282	0.061	0.822*	0.115
t_{15} (tagging)	0.881*	0.265	0.957*	0.369	0.244	0.588*
t_{16} (web2.0)	0.715*	0.695*	0.653*	0.464*	0.676*	0.563*
t_{17} (web3.0)	0	0.121	0.306	0.680*	0.125	0.703*

(* tags that are previously labeled to the item)

3.5 Folksonomy-Boosted Ranking

A user query comprises a set of tags. In general, users tend to generate short queries when searching, resulting in tremendous ambiguity about their intentions (Ma et al. 2007). In addition, people engaged in different areas may have different understandings of the same tags and, thus, may differ significantly in the search results they consider to be relevant for the same query tags (Liu et al. 2004).

To deal with these issues, we use the two latent models — latent tag preference model **P** and Latent tag annotation **W** — to build our personalized algorithm for ranking items

according to a given query. We name our personalized ranking algorithm as *Folksonomy-Boosted Ranking* (*FBR*). Formally, for a specific query $q=\{t_1, t_2, \ldots t_m\}$, $m \leq |T|$, it calculates a ranking score of item i for user u as:

$$FBR_u(i,q) = \sum_{t \in q} p_{u,t} \times w_{t,i} \qquad (9)$$

where $p_{u,t}$ is the value of the t-th column (i.e., the tag) of the u-th row (i.e., the user) in the matrix **P** and $w_{t,i}$ is the value of the i-th column (i.e., the item) of the t-th row (i.e., the tag) in the matrix **W**. As mentioned in the previous section, a preference value of $p_{u,t}$ reflects the query user's own tagging taste for tags similar to tag t. Therefore, this value can give the effect of an implicit query expansion according to a given searcher's interests. In addition, a weight value of $w_{t,i}$ reflects tag t's usages by all users on items similar to item i. Consequently, by utilizing both models, **P** and **W**, items likely to fit a user's needs rank higher in the searched list regardless of whether the items have explicitly labeled query tags. Figure 10 illustrates the process of computing an *FBR* score for certain query tags. For a given query by user u, once our strategy computes a ranking score for items that this user has not previously tagged, it ranks the items in descending order of *FBR* score.

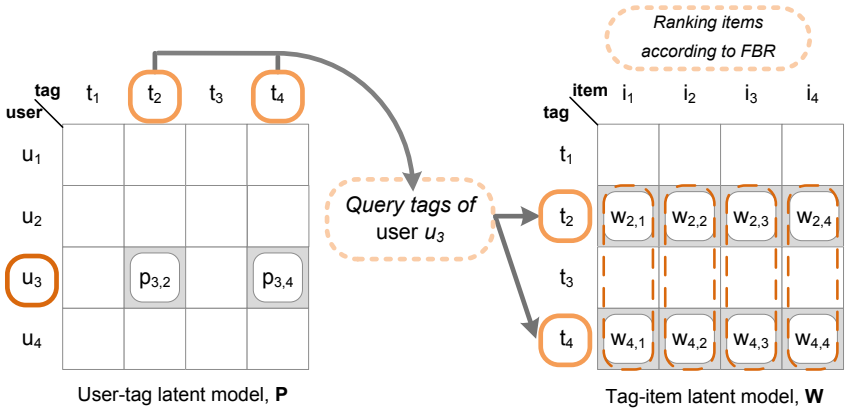

Figure 10 Computing item ranking scores depending on a user and a query

Recommender systems that produce item recommendations for a user without any given queries can easily adapt *FBR* to their use by assuming that a certain user submits all tags as their query. In this sense, user *u*'s *FBR* score for item *i* can be simply computed as the dot product of the row vector of user *u* in the matrix **P** and the column vector of item *i* in the matrix **W**. As we have seen, our *FBR* algorithm seamlessly maps the tags onto items, depending on an individual user's query, to find the most desirable content relevant to the user's needs.

3.5.1 Example: computing *FBR* score

Assume that Alice submits "semantic" and "tagging" as her query tags. Computing the *FBR* scores for the items should tailor her search results. Alice's *FBR* for *Faviki.com* is calculated from $p_{Alice,semantic} = 1.798$ and $p_{Alice,tagging} = 1.145$ in Table 4 and $w_{semantic,Faviki} = 0.207$ and $w_{tagging,Faviki} = 0.957$ in Table 9, resulting in a ranking score for *Faviki.com* in this example query as: $FBR_{Alice}(Faviki, semantic\ tagging) = (1.798 \times 0.207) + (1.145 \times 0.957) = 1.469$.

Table 10 Alice's *FBR* scores for the example queries

query	Delicious*	Facebook*	Faviki	Freebase*	MySpace	Twine
semantic tagging	1.009	0.411	1.469	1.374	0.392	1.535
semantic web3.0	0	0.243	0.714	1.710	0.252	1.645
social collaboration	1.324	1.082	1.598	1.452	1.055	1.674

(* items that are previously tagged by Alice)

Analogously, for another query such as "semantic" and "Web 3.0," the *FBR* score for *Faviki.com* can be computed as 0.714. Although the tags, "semantic" and "web3.0," have not previously annotated in *Faviki.com*, our approach can reveal *Faviki.com*, because

those tags share some common meanings with tags labeled to *Faviki.com*, such as tag "semanticweb." Table 10 shows the *FBR* scores for Alice according to the different queries.

To examine how the *FBR* in the search results differs among users, consider the situation where all users submit the same query, "web3.0" and "community." Table 11 shows that rankings according to *FBR* scores differ indeed, depending on the user. For example, Alice has *Twine.com* ranked in the top-1 position, whereas *MySpace.com* becomes the top-1 for Bob in the searched results. The fact that Alice prefers the tag "Web 3.0" to the tag "community," whereas Bob prefers the latter tag over the former (according to the latent tag preference in Table 6) affects this result. Thus, Alice seems to prefer services which relate closely to Web3.0 and the semantic Web over general community sites, such as *Facebook.com* and *MySpace.com*, based on her tagging behaviors. On the other hand, Bob seems more interested in the community sites.

Table 11 The *FBR* scores for the query "Web 3.0" and "community" depending on users

	Delicious	Facebook	Faviki	Freebase	MySpace	Twine
Alice	0.201*	0.778*	0.496	0.997*	0.791	1.137
Bob	0.255	0.847	0.275*	0.478	0.859*	0.629
Cathy	0.405	1.523	0.887*	1.760	1.549*	2.032*
Mark	0.583	2.013	0.820	1.518*	2.043*	1.877
Nami	0.096*	0.633	0.898	1.949*	0.650	2.064*

(* items that are previously tagged by the user)

3.6 Complexity Analysis

We analyze the computational complexity of our Folksonomy-boosted Ranking *FBR* algorithm according to the number of users $|U|$, the number of items $|I|$, the number of tags $|T|$, the number of similar tags k, and the number of similar items k'. The

computational complexity can be divided between *an offline phase* and *an online phase*; the former can be accomplished offline whereas the latter has to be done in real time. Offline computation closely connects to the time required to compute the preference matrix **P** and the weight matrix **W**. Prior to building the models, the tag-tag similarity matrix **E** and the item-item similarity matrix **H** should be computed. The complexities of these steps are $O(|T|^2|I|)$ and $O(|T||I|^2)$, respectively, in the worst case. For computing **P** and **W**, we additionally require approximately $O(k|U||T|)$ and $O(k'|T||I|)$, respectively. Therefore, the total complexity of building the preference model **P** becomes $O(|T|^2|I| + k|U||T|)$, whereas that of building the weight model **W** becomes $O(|T||I|^2 + k'|T||I|)$. However, as pointed out in previous studies (Wu et al. 2006; Zanardi et al. 2008), because the user-tag matrix **A**, the user-item matrix **C**, and the tag-item matrix **N** show extreme sparsity in practice, the actual computational complexity tends to be significantly lower. For example, the complexities of the matrix **E** and **H** are closer to $O(|T||I|)$, as most items have a very small number of tags, and most tag vectors contain a small number of items in the tag-item matrix **N**. Therefore, the complexities of **P** and **W** are reduced to approximately $O(|T||I| + k|U||T|) \cong O(|T||I| + |U||T|)$ and $O(|T||I| + k'|T||I|) \cong O(|T||I|)$, respectively. Consequently, the computational complexity becomes $O(|T||I| + |U||T|)$ during the offline phase.

The experiments are executed on an Intel Core i7 with 2.80 GHz of CPU, and 8 GB of RAM. For our experimental dataset containing 14,169 tags and 3393 items, the runtime required to compute similarities between all pairs of tags for **E** was 900.11 seconds, and between all pairs of items for **H** was 1544.92 seconds. It took on average 0.064 seconds to compute similarities between a single tag and every other tag, whereas it took on

average 0.455 seconds to compute similarities between a single item and every other item.

In addition, when k and k' were set to 350, it took on average 386.98 seconds to build **P** whereas it took 496.29 seconds to build **W**. On average, we could finish the computation of a single user's preferences for all tags in 0.345 seconds and the computation of tags' annotations for a single item in 0.146 seconds. As k and k' values decreased, the runtime became much shorter for building the both models.

Once the models are built, for a new user joining with new (or existing) tags or a new item labeled with new (or existing) tags, we may need not to renew the whole models in practice, but to update the model for only that user or that item so as to reflect the new information quickly. Also note that each row (or each entry) in **P** and each column (or each entry) in **W** can be updated individually and periodically because each computation process is independent of every other computation. Furthermore, the computation can be easily implemented with parallel computation because it is simply the sparse matrix-vector multiplication. Rebuilding the entire models could usually be accomplished on a regular basis (e.g., weekly, monthly, etc.).

With respect to real-time performance, online is more important than offline complexity. Note that the most time-consuming tasks in building our two latent models **P** and **W** can be conducted offline although the two models may not immediately reflect full up-to-date data. For m tags contained in a user's query, the computational complexity required to compute FBR of a particular item is $O(m)$; accordingly, the total complexity of computing $|I|$ items becomes $O(m|I|) \cong O(|I|)$. In our experiments, when the number of tags contained in a user's query was three, we generated rankings about 4000 items in 0.67 seconds on average.

3.7 Summary

In this chapter, we have presented a new Folksonomy-Boosted Ranking (*FBR)* algorithm in a folksonomy space that aims to search and rank social media relevant to users' inquiry. The *FBR* algorithm computes the cosine similarity between items and tags. Then it builds the two latent models **P** and **W**. The FBR algorithm uses both latent models to seamlessly map the tags onto items, depending on an individual user's query, to find the most desirable content relevant to the user's needs.

Chapter 4 Personalized Tag Recommendation

This chapter presents our new algorithm *KatzBm25*, and how this algorithm can utilize the social tagging information to provide personalized tag recommendation and item annotation. In our algorithm, folksonomy can be represented by a tripartite graph. We formalize the tag recommendation, and item annotation as a link-prediction problems. On this tripartite graph, we exploit the Katz measure—a path-ensemble based proximity measure —to quantify the proximity of two nodes based on weighted sums over collections of possible paths connecting the two nodes. The assumption underlying this approach is that an appropriate node for a given node would be in close proximity to that node from a graph viewpoint. In the case of recommending suitable tags for a given user-item pair, our algorithm speculates as to how a certain node is close not only to that user, but also to that item. Based on the proximity of nodes, we uncover triangle graphs that are likely to appear in the tripartite folksonomy graph. In the item annotation case, our algorithm *KatzBm25* only considers the proximity from a given item to tags.

4.1 Preliminaries

From the folksonomy tag assignments, three matrices can be obtained by aggregating over items, tags, and users, respectively: user-tag matrix **A**, user-item matrix **C**, and tag-item matrix **N**. By using these three matrices, a folksonomy can be converted into an undirected tripartite graph $G = (U \cup T \cup I, E)$ whose nodes can be partitioned into three disjoint sets, U, T, and I, such that every node of each set is adjacent to at least one node in each of the two other sets (Hotho et al. 2006). Each edge (or link) in E connects the corresponding row and column of nonzero entries in the matrices, **A**, **C**, and **N**. In addition, each link has a weight which is equal to an entry's value in the corresponding matrix. Figure 11 shows an example of a weighted undirected tripartite graph that represents a folksonomy.

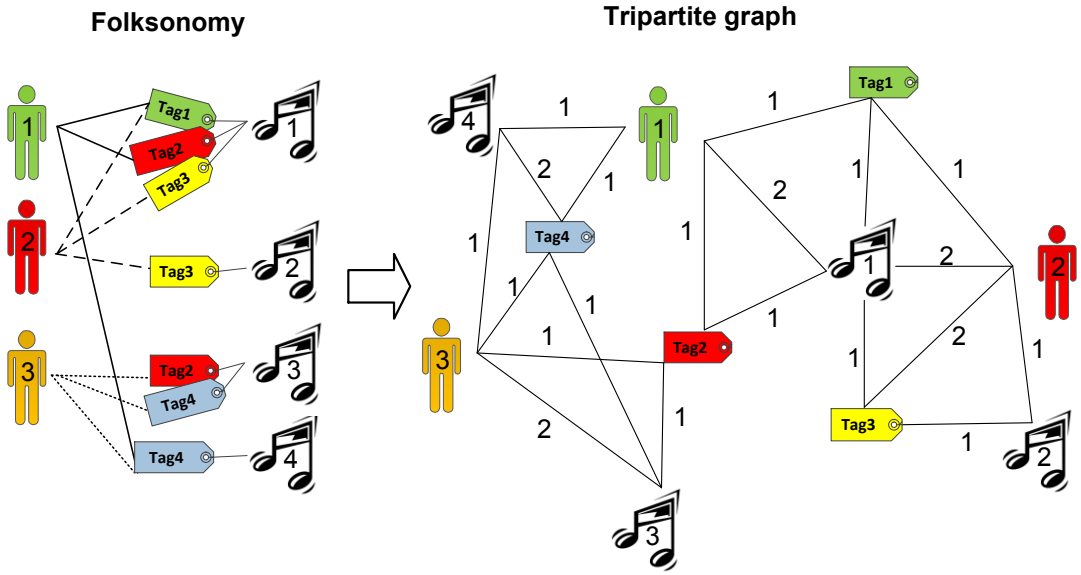

Figure 11 Transforming a folksonomy to a weighted undirected tripartite graph

4.2 Link-Predication Problem

Given the fact that a folksonomy can be converted into the undirected tripartite graph *G*, the problem of recommending tag for non tagged item can be seen as a link-prediction problem.

4.2.1 Tag Recommendation

In general, the goal of personalized tag recommender algorithms is to identify a set of tags suited to a given user-item pair (*u*, *i*). From a tripartite graph point of view as shown in Figure 12, if user *u* is interested in tagging item *i*, then new link (*u*, *i*) between user *u* and item *i* appear in the graph. To recommend tags for a given link (*u*, *i*), our algorithm attempts to predict both the link (*u*, *t*) between user *u* and tag *t*, and the link (*i*, *t*) between item *i* and tag *t*. Thereafter, our algorithm computes a ranking score of tag *t* based on such predicted links and thus generates a list of top *N* ranked tags suited to user *u* for item *i*.

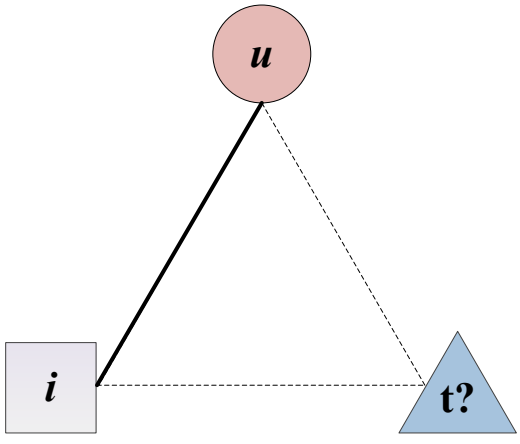

Figure 12 Link-Predication in a tag recommendation scenario

4.2.3 Item Annotation

In the item annotation problem, the objective is to uncover *latent tags* relevant to a given item, even if those tags have not been previously labeled to that item. To identify such tags from a tripartite graph viewpoint, our algorithm attempts to predict the link (i, t) between item i and tag t that is likely to emerge from many user nodes.

4.3 Adjacency matrix for folksonomy graph

We begin by defining the adjacency matrix **D** corresponding to the undirected tripartite graph ***G***, as shown in Figure 13. Each entry in the adjacency matrix is exactly equal to the weight associated with the link between the corresponding two nodes in the graph.

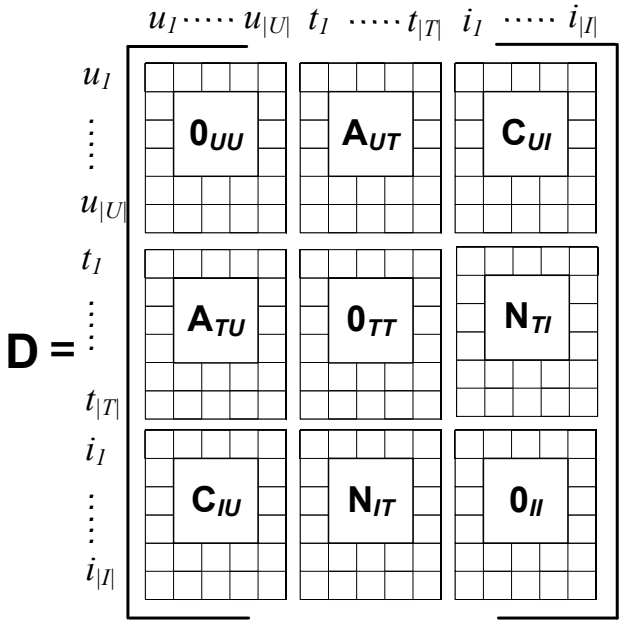

Figure 13 Folksonomy Adjacency Matrix

Where $\mathbf{0}_{UU}$, $\mathbf{0}_{TT}$, and $\mathbf{0}_{II}$ are $|U|$ x $|U|$, $|T|$ x $|T|$, and $|I|$ x $|I|$ are zero matrices respectively. Additionally, $\mathbf{A}_{TU}$, $\mathbf{C}_{IU}$, $\mathbf{N}_{IT}$ are the transpose matrices of $\mathbf{A}_{UT}$, $\mathbf{C}_{UI}$, $\mathbf{N}_{TI}$ respectively.

Different weighting schemes can be used for assigning weights on links in the graph, which result in different adjacency matrices. The weights are:

– *Binary*: the weight is either 0 or 1. For example, if user *u* assigned tag *t* to item *i*, then we set the associated weight to the corresponding links (*u*, *t*), (*u*, *i*), and (*t*, *i*) to 1, and 0 otherwise.

– *Frequency*: the weight is similar to *term frequency* used in information retrieval. Here, the weight associated with the links (*u*, *t*), (*u*, *i*), and (*t*, *i*) denotes the number of items user *u* tagged with tag *t*, the number of tags user *u* assigned to item *i*, and the number of users who tagged item *i* with tag *t*, respectively.

- *Okapi BM25*: this weight is used to increase/decrease the importance of tags within/among users and items (Sparck Jones et al. 2000; Xu et al. 2008; Vallet et al. 2010). Unlike other weighing scheme that assumes the same user/item length. The BM25 consider not only whether a certain tag is common or rare but also how frequently the tag appears with the user or the item. In our case, the length of an item represents the number of unique tags annotated in the item, whereas the length of a user refers to the number of unique tags used by him/her. Formally, a BM25 weight of tag *t* in item *i*, i.e, the link (*t*, *i*) is computed by:

$$BM_{25}(t,i) = \log \frac{|I|}{N(i_t)} \times \frac{tf(t,i) \times (k_1 + 1)}{tf(t,i) + k_1 . \times (1 - b + b. \frac{|i|}{avg(I)})} \qquad (10)$$

Where $|I|$ is the total number of items, $N(i_t)$ is the number of items annotated with tag *t*, $tf(t, i)$ is the number of times item *i* has been tagged with tag *t*, $|i|$ is the length of item *i*,

avg(I) is the average length of items. Parameters k_1 and b are constants normally chosen, $k = 2.0$ and $b = 0.75$. In an analogue fashion, we can compute a BM25 weight $BM_{25}(u, t)$ of tag t with respect to user u by replacing item i with user u.

In the next sections, we will apply the BM25 weight to our tag recommendation algorithms; also in the experimental chapter we will have a detailed section to show the effect of these different weights on our tag recommendation algorithm.

4.4 Computation of the Katz Matrix

After building the adjacency matrix, we compute the Katz score for all pair of nodes in **D**. The Katz measure is one of the most effective path-based measure that has been successfully applied to different applications such as link prediction (Liben-Nowell et al. 2007), graph theory (Foster et al. 2001; Huang e al. 2005). Initially the Katz score measures proximity between a pair of nodes via a weighted sum of the number of paths between the two nodes (Katz 1953). Before introducing how to calculate the Katz score between two nodes in the graph, we first define a *path* and its *path weight* in our folksonomy graph ***G***. A path in a graph generally represents a way to get from a start node to a destination node by traversing links in the graph. Formally, a path of length n, $n \geq 1$, from a node to another node can be represented as a sequence of nodes, $v_1, v_2, \ldots, v_n, v_{n+1}$ so that links are $(v_k, v_{k+1}) \in E$ for each $k = 1, \ldots, n$. For convenience, the weight associated to a link (v_k, v_{k+1}) is denoted using $W(v_k, v_{k+1})$. The length of a path refers to the number of links contained in the path. Therefore, if the length of a certain path is 1, there is only a direct link between two nodes. Let $P^n(x, y)$ be the set of all possible paths

of length n in G from node x and node y. A path weight of a path $p \in P^n(x, y)$ is defined as:

$$PW_p^n(x,y) = \prod_{k=1}^{n} W(v_k, v_{k+1}) \tag{11}$$

where the initial node v_1 is node x and the terminal node v_{n+1} is node y. Then, the Katz score from x and y is calculated as:

$$K_{x,y} = \sum_{n=1}^{\infty} \alpha^n \times \left(\sum_{p \in P^n(x,y)} PW_p^n(x,y) \right) \tag{12}$$

where $\alpha \in (0, 1)$ is an attenuation parameter (usually a small value, e.g. 0.05 or 0.005). The Katz score measures the proximity of two nodes based on weighted sums over collections of possible paths connecting those nodes. The Katz score of all pairs of nodes in the graph can be expressed in matrix form as follows (Liben-Nowell et al. 2007):

$$\mathbf{K} = \alpha\mathbf{D} + \alpha^2\mathbf{D}^2 + \alpha^3\mathbf{D}^3 + \cdots\cdots = (\mathbf{I} - \alpha\mathbf{D})^{-1} - \mathbf{I} \tag{13}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{D}$ is an adjacency matrix for the graph. The series expansion converges if $\alpha < 1/\lambda_{max}(\mathbf{D})$, where $\lambda_{max}(\mathbf{D})$ is the largest absolute value of any eigenvalue of $\mathbf{D}$; thus, this condition determines how large α can be (Katz 1953). Each entry in the Katz matrix $\mathbf{K}$ represents the sum of the path weights of all lengths from the corresponding row node to the corresponding column node in the graph.

Let us utilize the example shown in Figure 14 to illustrate the tag recommendation problem. We have four users' tag assignments on five items.

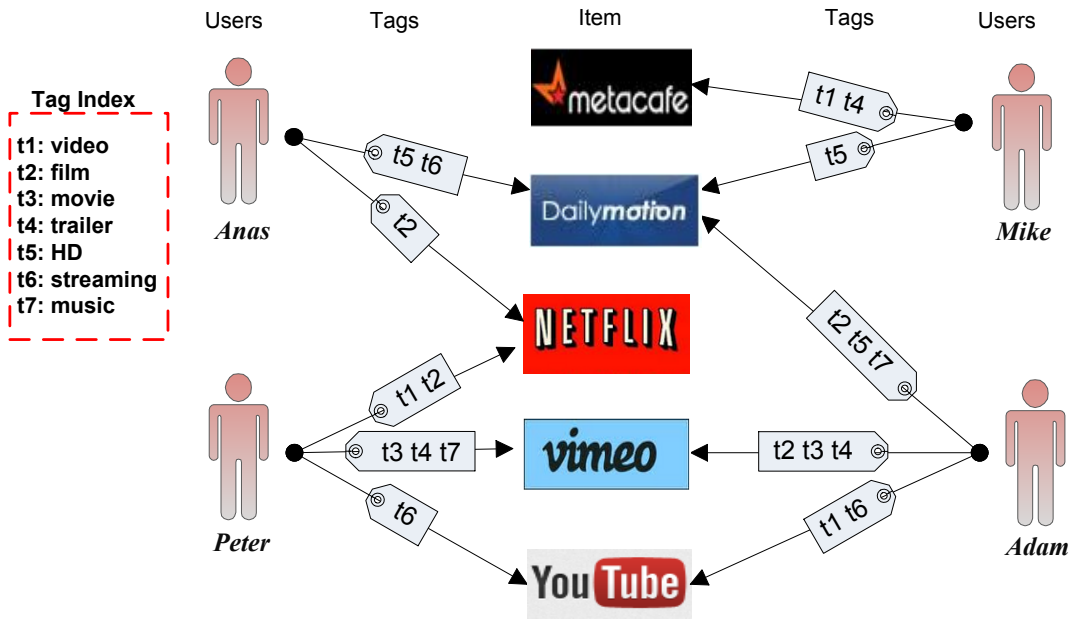

Figure 14 An Example of tag assignments

The users are: u_1(Mike), u_2(Adam), u_3(Anas) and u_4(Peter). Five items are: i_1(metacafe), i_2(dailymotion), i_3(netflex), i_4(vimeo) and i_5(youtube). Seven tags are: t_1(video), t_2(film), t_3(movie), t_4(trailer), t_5(HD), t_6(streaming), and t_7(music).We start by aggregating the tag assignments over items, tags and users respectively. From this aggregation we can obtain three matrices as shown in Table 12, Table 13, and Table 14.

Table 12 user-tag matrix

	t_1(video)	t_2(film)	t_3(movie)	t_4(trailer)	t_5(HD)	t_6(streaming)	t_7(music)
u_1(Mike)	1	0	0	1	1	0	0
u_2(Adam)	1	2	1	1	1	1	1
u_3(Anas)	0	1	0	0	1	1	0
u_4(Peter)	1	1	1	1	0	1	1

Table 13 item-tag matrix

	t_1(video)	t_2(film)	t_3(movie)	t_4(trailer)	t_5(HD)	t_6(streaming)	t_7(music)
i_1(metacafe)	1	0	0	1	0	0	0
i_2(dailymotion)	0	1	0	0	3	1	1
i_3(netflex)	1	2	0	0	0	0	0
i_4(vimeo)	0	1	2	2	0	0	1
i_5(youtube)	1	0	0	0	0	2	0

Table 14 user-item matrix

	i1(metacafe)	i_2(dailymotion)	i_3(netflex)	i_4(vimeo)	i_5(youtube)
u_1(Mike)	1	1	0	0	0
u_2(Adam)	0	1	0	1	1
u_3(Anas)	0	1	1	0	0
u_4(Peter)	0	0	1	1	1

The user-tag matrix and the item-tag matrix have frequency weight. For the user-item matrix, we binarize it as follows: if a certain user has tagged an item, we set the corresponding entry to 1, and we set it to 0 otherwise. For example, u_1(Mike) tagged i_1(metacafe), and i_2(dailymotion) , the corresponding entry for the two items would be 1 and for the other items 0. The three matrices taken together produce a weighted undirected tripartite graph, and the adjacency matrix which represents this graph is shown in Table 15.

Table 15 Adjacency matrix

	u_1	u_2	u_3	u_4	t_1	t_2	t_3	t_4	t_5	t_6	t_7	i_1	i_2	i_3	i_4	i_5
u_1(Mike)	0	0	0	0	1	0	0	1	1	0	0	1	1	0	0	0
u_2(Adam)	0	0	0	0	1	2	1	1	1	1	1	0	1	0	1	1
u_3(Anas)	0	0	0	0	0	1	0	0	1	1	0	0	1	1	0	0
u_4(Peter)	0	0	0	0	1	1	1	1		1	1	0	0	1	1	1
t_1(video)	1	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1
t_2(film)	0	2	1	1	0	0	0	0	0	0	0	0	1	2	1	0
t_3(movie)	0	1	0	1	0	0	0	0	0	0	0	0	0	0	2	0
t_4(trailer)	1	1	0	1	0	0	0	0	0	0	0	1	0	0	2	0
t_5(HD)	1	1	1	0	0	0	0	0	0	0	0	0	3	0	0	0
t_6(streaming)	0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	2
t_7(music)	0	1	0	1	0	0	0	0	0	0	0	0	1	0	1	0
i_1(metacafe)	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
i_2(dailymotion)	1	1	1	0	0	1	0	0	3	1	1	0	0	0	0	0
i_3(netflex)	0	0	1	1	1	2	0	0	0	0	0	0	0	0	0	0
i_4(vimeo)	0	1	0	1	0	1	2	2	0	0	1	0	0	0	0	0
i_5(youtube)	0	1	0	1	1	0	0	0	0	2	0	0	0	0	0	0

The adjacency matrix as shown in Table 15 has frequency weight. To convert the frequency weight into BM25 weight we can apply Equation 10. For example the BM25 weight from t_1(video) to i_3(netflex) can be computed as follows:

$$BM_{25}(video, netflex) = \log\frac{5}{3} \times \frac{1 \times (2+1)}{1 + [2 \times (1 - 0.75 + [0.75 \times \frac{2}{2.8}])]} = 0.259$$

where $|I| = 5$, $N(i_t) = 3$, $tf(t, i) = 1$, $|i| = 2$, $avg(I) = 14/5 = 2.8$, $k = 2.0$ and $b = 0.75$.

Before computing the Katz score between all pair of nodes in the tripartite graph via Equation 12, we will show the steps on how to compute the Katz score for one link only. Let's use Figure 15 which includes only part of the whole tripartite graph to compute the Katz score for the link u_1(Mike) to i_1(metacafe).

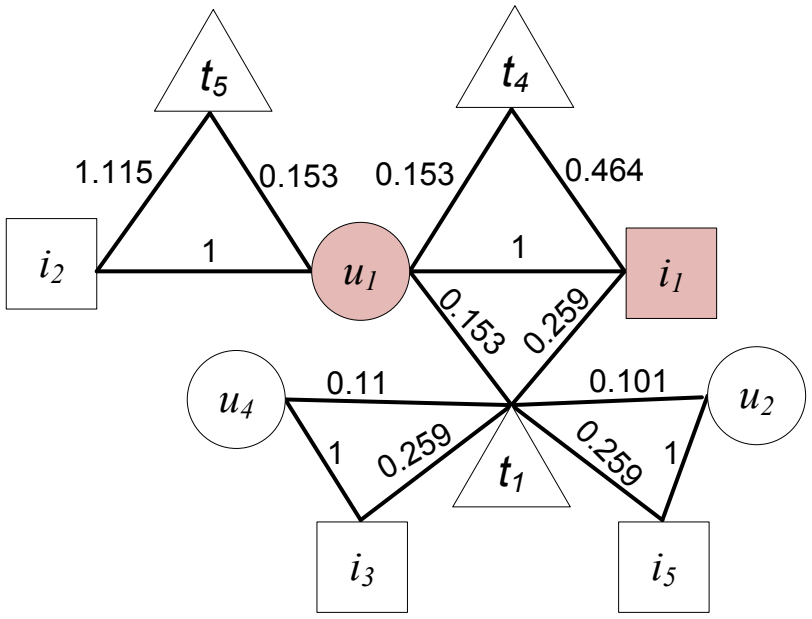

Figure 15 Part of the whole tripartite graph to show all possible paths from u_1 to i_1

Table 16 shows the steps to compute the Katz score between u_1(Mike) to i_1(metacafe). The symbol $\rightarrow$ shows the direction of the path in Table 16. After 10 iterations the series in Equation 12 converges.

Table 16 Steps to calculate the Katz score for u_1(Mike) to i_1(metacafe) at alpha 0.1

Path length *n*	Number of Paths *p*	Actual path	weight of each Path of length *n*	Sum of weights of all paths of length *n*	Score = $\alpha^n \times$ weight
1	1	$u_1 \rightarrow i_1$	1	1	$0.3 \times 1 = 0.3$
2	2	$u_1 \rightarrow t_4 \rightarrow i_1$ $u_1 \rightarrow t_1 \rightarrow i_1$	$0.153 \times 0.464 = 0.071$ $0.153 \times 0.259 = 0.04$	0.071+0.04= 0.111	$(0.3 \times 0.3) \times 0.111 =$ 0.01
3	7	$u_1 \rightarrow t_4 \rightarrow u_1 \rightarrow i_1$ $u_1 \rightarrow t_1 \rightarrow u_1 \rightarrow i_1$ $u_1 \rightarrow i_1 \rightarrow u_1 \rightarrow i_1$ $u_1 \rightarrow t_5 \rightarrow u_1 \rightarrow i_1$ $u_1 \rightarrow i_2 \rightarrow u_1 \rightarrow i_1$ $u_1 \rightarrow i_1 \rightarrow t_4 \rightarrow i_1$ $u_1 \rightarrow i_1 \rightarrow t_1 \rightarrow i_1$	$0.153 \times 0.153 \times 1 = 0.023$ $0.153 \times 0.153 \times 1 = 0.023$ $1 \times 1 \times 1 = 1$ $0.153 \times 0.153 \times 1 = 0.023$ $1 \times 1 \times 1 = 1$ $1 \times 0.464 \times 0.0464 = 0.215$ $1 \times 0.259 \times 0.259 = 0.067$	0.023+0.023+1+ 0.023+1+0.215 +0.067=2.351	$(0.3 \times 0.3 \times 0.3) \times 2.351$ = 0.068
4	42	…….	…….	0.9268	0.0075
5	200	…….	…….	9.5768	0.0233
6	1554	…….	…….	8.0119	0.0058
7	9416	…….	…….	55.6536	0.0122
8	69850	…….	…….	72.9072	0.0048
9	458351	…….	…….	388.7130	0.0077
10	3271068	…….	…….	665.8636	0.0039
					Total score 0.4432

Similar we can calculate the Katz score for all pair of nodes in the tripartite graph. The tables below 17,18 and 19 show all pair of nodes' score. Even though u_1(Mike) only assigned t_1(video), t_4(trailer) and t_5(HD), this approach can estimate the score for all other tags that were not previously used by u_1(Mike). Same thing i_1(metacafe) for example was tagged only with t_1(video) and t_4(trailer), while the score with respect to other tags can also be estimated.

Table 17 user-tag Katz score

	t_1(video)	t_2(film)	t_3(movie)	t_4(trailer)	t_5(HD)	t_6(streaming	t_7(music)
u_1(Mike)	0.147[s]	0.066	0.029	0.151[s]	0.182[s]	0.050	0.042
u_2(Adam)	0.200[s]	0.379[s]	0.209[s]	0.225[s]	0.245[s]	0.237[s]	0.211[s]
u_3(Anas)	0.050	0.206[s]	0.034	0.040	0.197[s]	0.166[s]	0.050
u_4(Peter)	0.190[s]	0.246[s]	0.192[s]	0.203[s]	0.071	0.200[s]	0.178[s]

[s] Tags assigned by the user

Table 18 item-tag Katz score

	t_1(video)	t_2(film)	t_3(movie)	t_4(trailer)	t_5(HD)	t_6(streaming	t_7(music)
i_1(metacafe)	0.133[s]	0.030	0.019	0.137[s]	0.032	0.020	0.018
i_2(dailymotion)	0.085	0.255[s]	0.066	0.089	0.456[s]	0.212[s]	0.182[s]
i_3(netflex)	0.156[s]	0.303[s]	0.050	0.057	0.062	0.071	0.051
i_4(vimeo)	0.084	0.243[s]	0.300[s]	0.311[s]	0.077	0.090	0.190[s]
i_5(youtube)	0.166[s]	0.099	0.058	0.065	0.060	0.280[s]	0.059

[s] Tags assigned previously to the item

Table 19 user-item Katz score

	i_1(metacafe)	i_2(dailymotion)	i_3(netflex)	i_4(vimeo)	i_5(youtube)
u_1(Mike)	0.443[s]	0.194[s]	0.039	0.064	0.041
u_2(Adam)	0.053	0.309[s]	0.130	0.296[s]	0.218[s]
u_3(Anas)	0.014	0.229[s]	0.164[s]	0.062	0.060
u_4(Peter)	0.046	0.119	0.195[s]	0.261[s]	0.199[s]

[s] Items tagged previously by each user

4.5 Computing the Tag Recommendation Score

In this section we present our new tag ranking algorithm for a given user-item pair based on the **K** matrix. Since every user has a different taste on tags, we should recommend different tags to each user based on his personal interests.

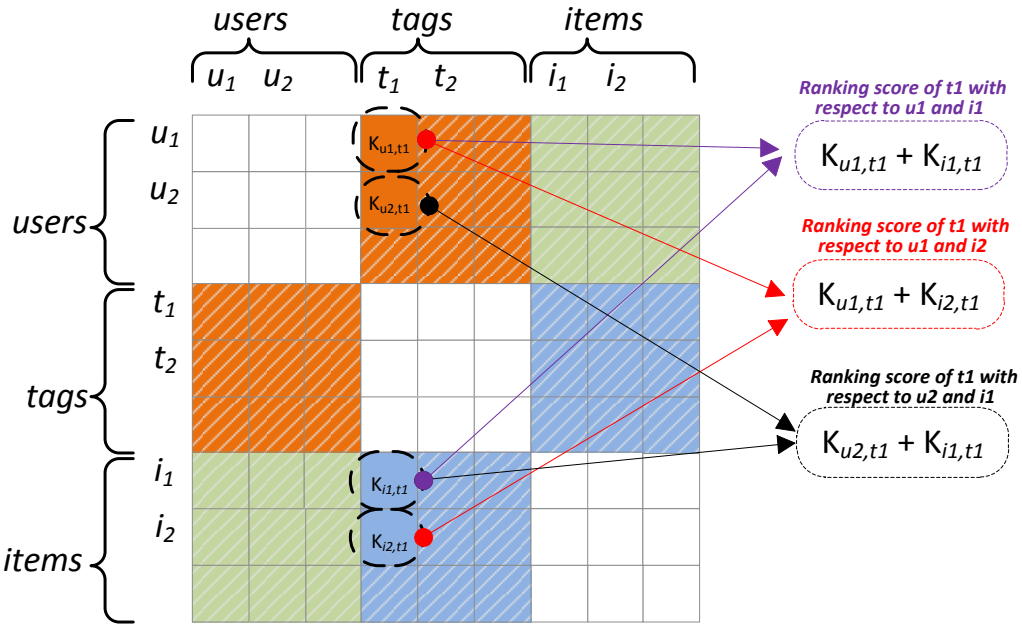

Figure 16 The process of computing tag ranking scores

To identify top-N suitable tags for a given user-item pair, we assess how a certain tag is in close proximity not only to the user, but also to the item. To that end, we compute a ranking score as a sum of Katz scores. Formally, given a user u and an item i, a ranking score for a particular tag t is computed by:

$$KatzBm25_{u,i}(t) = K_{u,t} + K_{i,t} \tag{14}$$

where $K_{u,t}$ and $K_{i,t}$ are the Katz scores from user u to tag t and from item i to tag t, respectively. We can succinctly express all ranking scores of tags for user u and item i in matrix form as:

$$\mathbf{s} = \mathbf{v}^T\mathbf{K} \tag{15}$$

where $\mathbf{v}^T$ is a vector that has value ones at entries corresponding to user u and item i, and zeros everywhere else. According to the combined Katz values contained in the resultant ranking vector $\mathbf{s}$, the set of N ordered tags with the highest values are recommended to

user u in regard to item i. Figure 16 illustrates the overall process of computing tag ranking scores of the same item i_1 for u_1 and u_2, as well as those of two different items i_1 and i_2 for the same user u_1.

4.5.1 Example: Computing the Tag score

Assume u_1(Mike) is interested in tagging i_3(netflex), in order to recommend the appropriate tags with respect to u_1(Mike) and i_3(netflex) , we need first to computer the Katz score for all tags based on Equation 14.

$$KatzBm25_{\text{Mike, netflex}}(\text{video}) = K_{\text{Mike, video}} + K_{\text{netflex, video}}$$

The value of $K_{\text{Mike, video}} = 0.147$ in Table 17, and $K_{\text{netflex, video}} = 0.156$ in Table 18, resulting in a ranking score $KatzBm25_{\text{Mike, netflex}}(\text{video}) = 0.147 + 0.156 = 0.303$ for t_1(video), similar we calculate the ranking score for all tags for i_3(netflex). Analogously, for other items that are not tagged previously by u_1(Mike), such as i_4(vimeo) or i_5(youtube), the ranking score can be computed as 0.230 and 0.313 respectively. Although u_1(Mike) never tagged these items in the past, our algorithm recommends the most appropriate tags for each untagged item. Taking the top 3 tag scores for each item, our algorithm recommends different tags for each item, for example it recommends { t_2(film)=0.369, t_1(video)=0.303, t_5(HD)=0.44 } for i_3(netflex), { t_4(trailer)=0.462, t_3(movie)=0.329, t_2(film)=0.309 } for i_4(vimeo), and { t_6(streaming)=0.329, t_1(video)=0.313, t_5(HD)=0.242 } for i_5(youtube). Table 20 shows all recommended tags ranking scores to the 3 different untagged items.

Table 20 Mike's recommended tags ranking score for 3 different items

	t_1(video)	t_2(film)	t_3(movie)	t_4(trailer)	t_5(HD)	t_6(streaming	t_7(music)
i_3(netflex)	0.303[s]	0.369[s]	0.080	0.208	0.244	0.120	0.093
i_4(vimeo)	0.230	0.309[s]	0.329[s]	0.462[s]	0.259	0.140	0.232[s]
i_5(youtube)	0.313[s]	0.165	0.087	0.216	0.242	0.329[s]	0.102

[s] Tags assigned previously to the item

Not only our algorithm recommends different tags with respect to different items, but also among users, for example u_2(Adam) never tagged i_3(netflex), our approach computes the top 2 tags score as t_2(film)=0.681 and t_1(video)=0.357, similar to u_1(Mike) but the ranking score for other tags are different as shown in Table 21.

Table 21 Recommended tags ranking score for i3(netflex) according to Mike and Adam

	t_1(video)	t_2(film)	t_3(movie)	t_4(trailer)	t_5(HD)	t_6(streaming	t_7(music)
u_1(Mike)	0.303[s]	0.369	0.080	0.208[s]	0.244[s]	0.120	0.093
u_2(Adam)	0.357	0.681[s]	0.260	0.282	0.307[s]	0.308[s]	0.261

[s] Tags that are previously used by the user

4.6 Computing the Item Annotation Score

In the item annotation case, we are interested in determining the relevant tags for an item. These relevant tags are not labeled to the item in the past, but can be of great benefit to users in retrieving more appropriate items. To discover such tags for a given item, we estimate how a specific tag is in close to the item. Given an item, therefore, the ranking score for a tag is equal to the Katz score from the item to a tag, simply defined as:

$$KatzBm25_i(t) = K_{i,t} \qquad (16)$$

where $K_{i,t}$ is the Katz scores from item i to tag t. Tags that obtain the highest scores, but have not previously annotated in the item, are regarded as tags that are most likely to be

relevant to that item. Figure 17 illustrates an intuitive view of computing tag ranking scores.

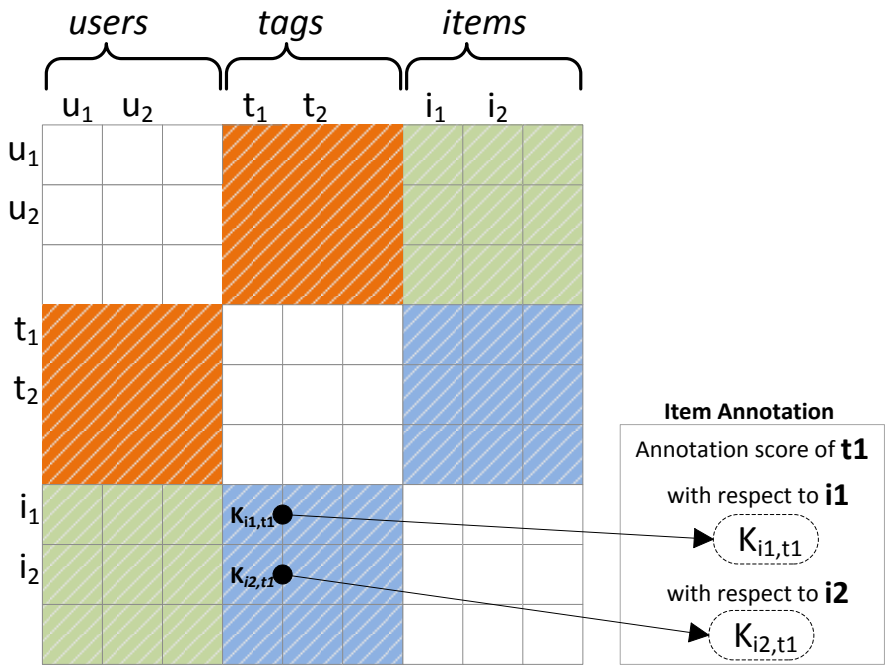

Figure 17 Computing tag ranking scores

4.7 Complexity Analyses

We analyzed the computational complexity of our tag recommendation algorithm *KatzBm25* according to the number of users $|U|$, the number of items $|I|$, and the number of tags $|T|$. The computational complexity can be divided between *an offline phase* and *an online phase*; the former can be accomplished offline whereas the latter has to be done in real time.

Offline computation reflects the time required to compute the adjacency matrix **D** and the Katz matrix **K**. The complexity to compute the adjacency matrix **D** is $O((|U|+|T|+|I|)^2)$ in the worst case, and the complexity to compute the Katz matrix **K** is $O((|U|+|T|+|I|)^3)$

(Cohen et al 2005). Therefore, the total complexity of building the adjacency matrix **D** and Katz matrix **K** becomes $O(\ (|U|+|T|+|I|)^2 + (|U|+|T|+|I|)^3)$. Consequently, the computational complexity becomes $O((|U|+|T|+|I|)^3)$ during the offline phase.

We executed our experiments using an Intel Core i7, with CPU 2.80 GHz, and RAM 8 GB. For our experiment on both CiteULike and Last.fm dataset the runtime required to compute the adjacency matrix **D** was 71 seconds and 95 seconds, respectively. The runtime required to compute the Katz matrix **K** was 1678 seconds on CiteULike and 2452 seconds on Last.fm. The computation can be easily implemented with parallel computation. Rebuilding the entire matrix to reflect new users, items or tags could be accomplished on a regular basis (e.g., weekly, monthly, etc.).

With respect to real-time performance, online is more significant than offline complexity. As we can see that the most time-consuming task in building our model can be done offline although the model may not immediately reflect the up-to-date data. The computational complexity required to compute the ranking score of all tags for a given item is $O(|T|)$, this complexity can be reduced to $O(n)$, where n represent the top N tags for a given item. In our experiments, when we consider the top 50 tags to be recommended to a given item, we generated rankings on average for 2310 tags in 0.45 seconds for CiteULike dataset and for 9749 tags in 0.75 seconds for Last.fm dataset.

4.7 Summary

In this chapter, we have presented our new algorithm *KatzBm25*, and how this algorithm can utilize the social tagging information to provide personalized tag recommendation and item annotation. The KatzBm25 improves tag-recommendation and item annotation by adapting the Katz measure, a path-ensemble based proximity measure, for the use in

social tagging systems. The algorithm model the folksonomy as a weighted, undirected tripartite graph, and then it applies the Katz measure to this graph, and exploit it to provide personalized tag recommendation for individual users.

Chapter 5 SongSter

In this Chapter, we present the proof-of-concept of our proposed algorithms named *SongSter*. The *SongSter* application is a mobile application for the Android platform that has a database of different songs from which users can get song suggestions and find similar users. Users can also contribute to the database by adding more songs and assigning tags to songs.

5.1 System Layout

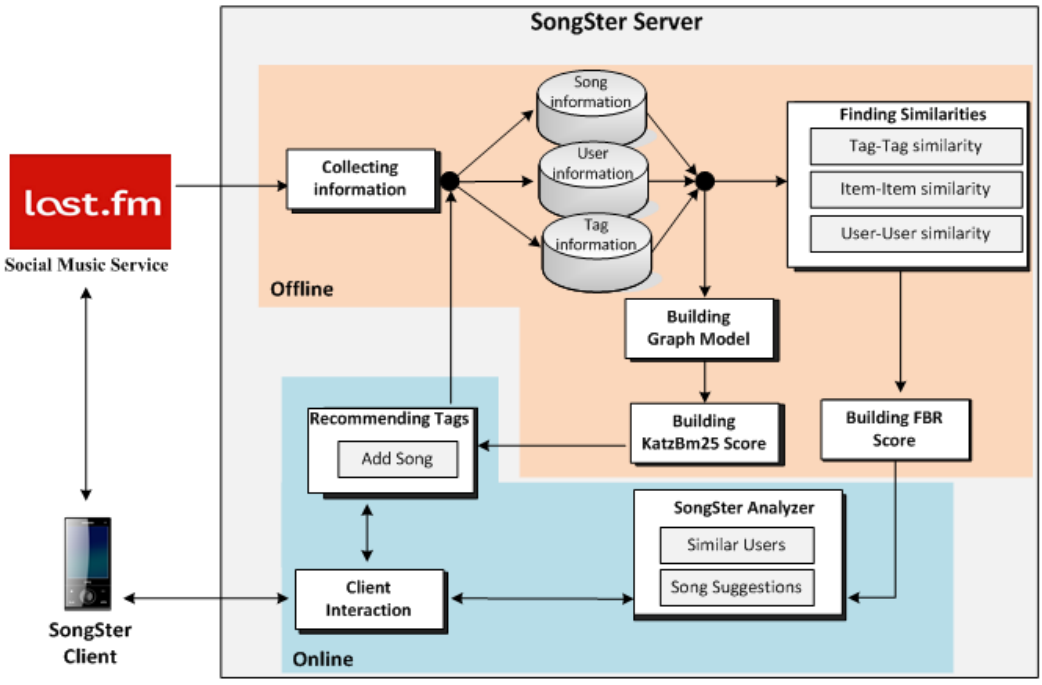

Figure 18 *SongSter* system layout

The system is a client-server application with the detail of the high level architecture for the system shown in Figure 18. Each component of the system is explained in the following text.

SongSter Client: This client is installed and runs on the user mobile device. It communicates with the Last.fm network to grant the user permission to the SongSter application and to verify the user's login information. The user can select different options from this component.

Collecting Information: This component collects and extracts the artist, song and tag information from the Last.fm social network and save the information to the designated database.

Song Information: This database stores all the extracted songs information.

User Information: This database stores all information about a user, which includes personal data and the history of their selected songs.

Tag Information: This database stores tags description and the list of songs annotated with these tags, and a list of users who used these tags.

Finding Similarity: This component is responsible to build the user-item matrix, user-tag matrix, and tag-item matrix. It also computes the user-user similarity, item-tem similarity and tag-tag similarity as described in section 3.3.

Building FBR Score: This component computes the latent tag preference model **P** as described in equation 5, that reflects how a certain user has assigned tags similar to a given tag, as well as the latent tag annotation model **W** as described in equation 8, that captures how users have tagged a certain tag to items similar to a given item. Finally this

component computes the final *FBR* ranking scores based on the two models **P** and **W** as described in equation 9.

SongSter Analyzer: It handles calculating the top best results using the ranking score returned from the Building FBR Model component, and passes the results to the client interaction component according to the user request.

Building Graph Model: This component analyzes the tripartite graph and builds the adjacency matrix as described in 4.3.

Building KatzBm25 Score: This component builds the final weighted KatzBm25 score matrix which is used to calculate the ranking score between any two nodes as indicated in equation 14.

Recommending Tags: This component recommends the suitable tags for a given user-item pair. It uses the values generated by Building Katz Score component as weights for identifying the relevant tags.

Client Interaction: This component retrieves the information from the *SongSter* Analyzer component to pass back to the client to display the results according to the user request.

5.2 Client Side Functionality

The client side application is a mobile application for the Android platform that has a multiple activities from which users can get song suggestions, find similar users, and add songs. Our application communicates with Last.fm, a social music service. To handle the interaction with the last.fm service, we used the Last.fm's provided API library. This library required registration of an application on Last.fm. Upon launching the SongeSter application for the first time, the user is promoted to register with Last.fm. If the user has

already an account with Last.fm, then the user can login by submitting a valid username and password combination.

Now once the user has logged in successfully, the client greets the user and shows the different options available in the main menu, and based on the user's selection the server responds accordingly. Assuming the server detects the user's action, the client receives back the top recommend songs or tags. The main client interfaces can be described as follows:

5.2.1 Similar Users

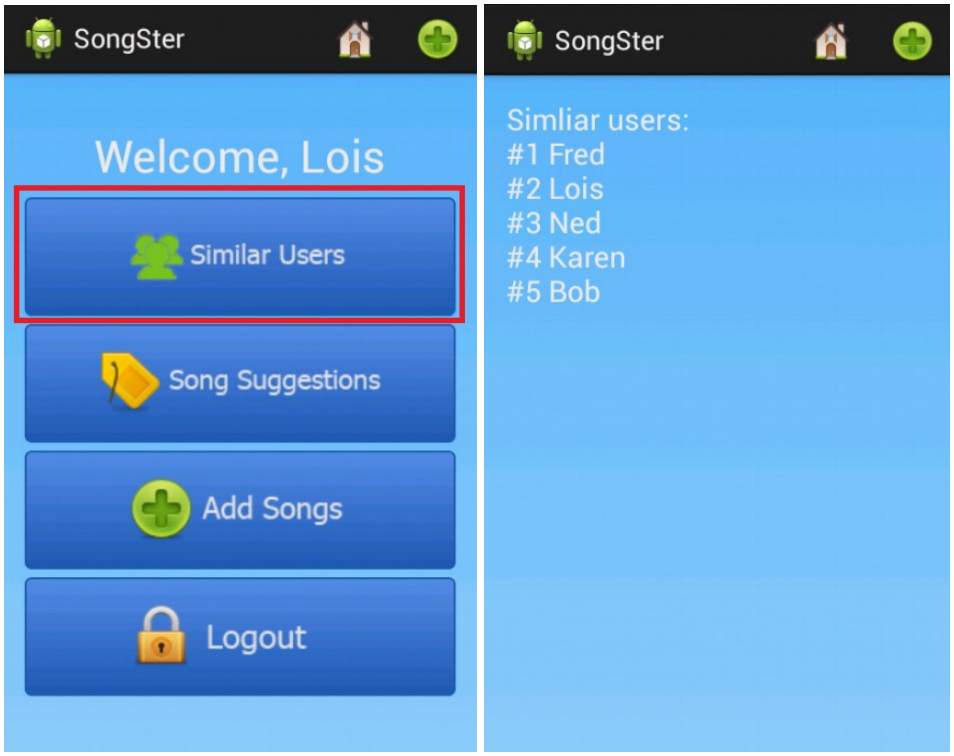

Figure 19 The top most similar users

By pressing the "Similar Users" button, the server runs a query on the database by selecting the top most similar users from the user-user similarity matrix. This similarity

matrix contains the computed cosine similarity values between all users. The server sends the top most similar users to the client as shown in Figure 19.

5.2.2 Song Suggestions

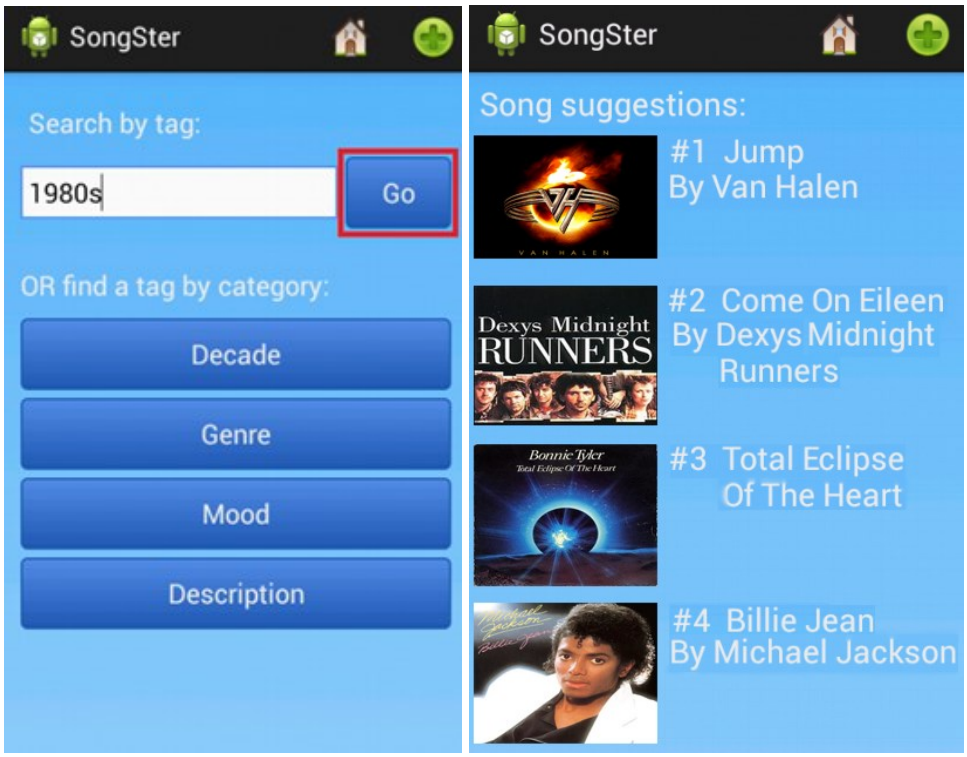

Figure 20 The top recommended songs

The user first selects the "Song Suggestions" button. Accordingly, the user can either type his own tag into the text box, or click a predefine category of tags. Then, the application searches the database for songs that are relevant to the requested tags. In Figure 20 the user Lois has inputted the tag "1980s" so the top songs are recommended based on this user's information including the tag. The ranking value of the retrieved songs is computed by the server through our FBR algorithm as described in section 3.5.

5.2.3 Add Songs

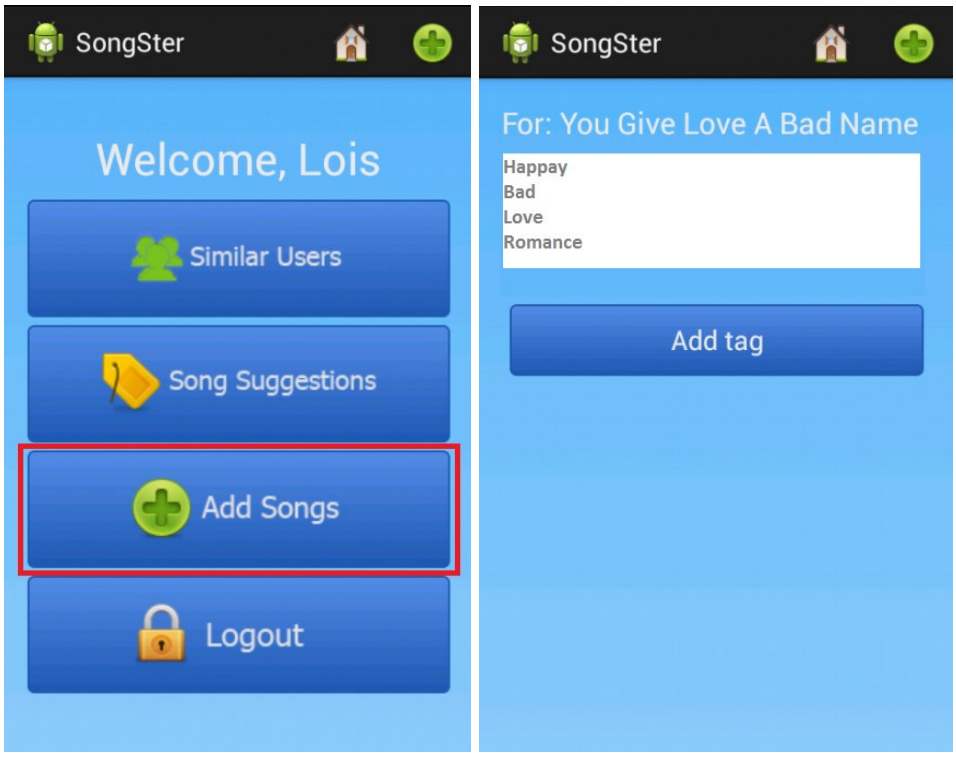

Figure 21 Adding a new song, with the top most suitable recommend tags

To add a song, a user can either click the "Add Songs" button or select the plus icon from the action bar menu, as shown in Figure 21. The user must then type in a song name and the artist name of the song into the textboxes. If this song name and artist combination already exists, then a popup message displays "Song already exists!" and the song name and artist input boxes are cleared. If no matches for the song and artist combination are found, the application suggests a tag or set of tags to be assigned to the added song. The suggested tags are recommended to the user based on the ranking score computed by our algorithm *KaztBm25* described earlier in section 4.5.

5.3 Server Side Functionality

The server side handles most of the functionality associated with the application. The server first extracts the social music information from Last.fm, and constructs the user-item matrix, user-tag matrix, and item-tag matrix as described in section 3.2. After that the server computes the cosine similarity matrices — user-user, item-item, and tag-tag—. It also computes the *FBR* ranking scores as described in equation 9. Currently all these functionalities are done offline. Once the user connects to the server, it verifies the user's login information and updates their information to keep the database up to date. If the user is interested to find a song, he presses the "Song suggestions" button, and specifies the query tags. The server calculates the *FBR* scores of all songs according to the user's query tags as explained in equation 9, and returns the top ranked songs to the user.

The server also builds the undirected tripartite graph from the extracted information from last.fm. It computes the BM25 weight associated with the link between the corresponding two nodes in the graph. After that it constructs the adjacency matrix **D** corresponding to the undirected tripartite graph as described in 4.3. After building the adjacency matrix, the server computes the Katz score for all pair of nodes in the adjacency matrix as explained in equation 12. Keep in mind that all these functionality are done offline. Now if the user clicks the "add song" button to add specific song, the server computes the *KatzBm25* score of all tags based on the requested user and returns the top ranked tags. The server will update the database accordingly.

The main task of the server can be described by two interaction diagrams. Figure 22 illustrates the sequence of events which the application undertakes when the user search a song based on an inputted tag or if he clicks a predefined category tag. Figure 23

illustrates the sequence of events which the application undertakes when the user is interested to add a new song; the server recommends the most suitable tags to the requested user and updates the database accordingly.

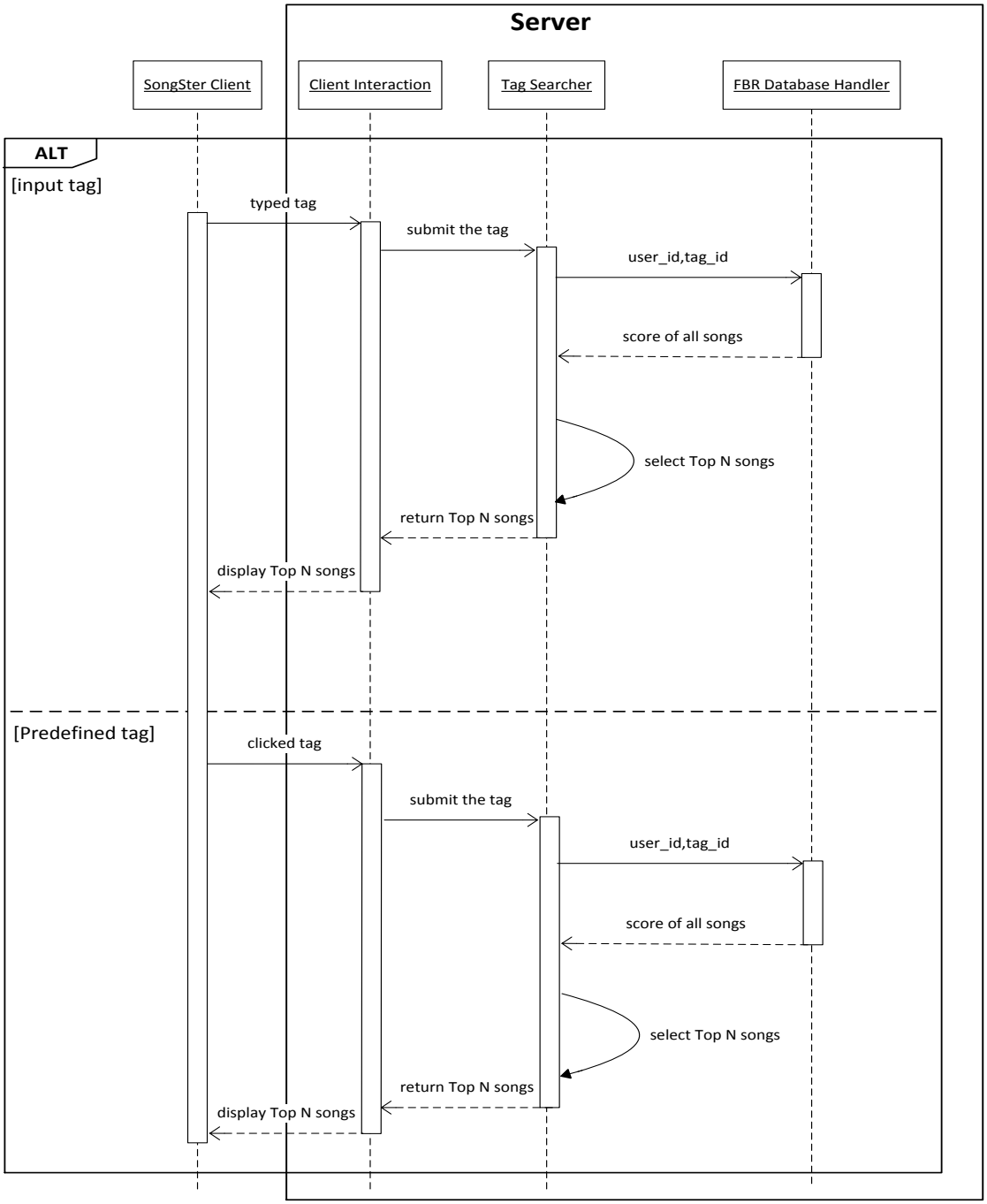

Figure 22 Interaction diagram of a user searching a song

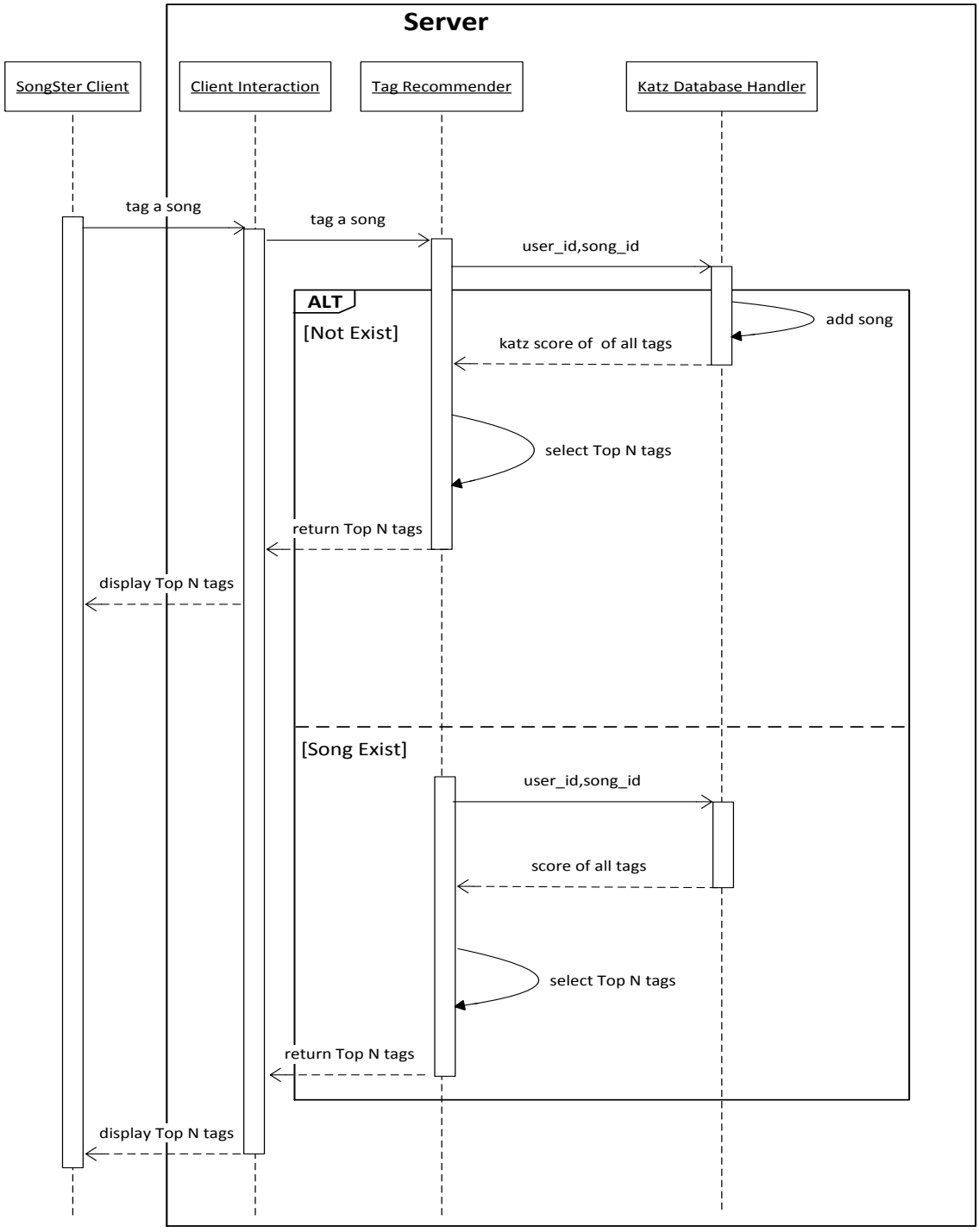

Figure 23 Interaction diagram of a user adding a song

5.3 Summary

We have presented in this chapter our android mobile application *SongSter*. The *SongSter* application has a database of different songs from which users can get song suggestions and find similar users. Users can also contribute to the database by adding more songs and assigning tags to songs. This application applied our *FBR* algorithm to compute the item ranking score; also the application applied our *KatzBm25* algorithm to compute the tag ranking score.

Chapter 6 Experiments and Results

This chapter presents the evaluation metrics used to measure the accuracy and the performance of our tag-based search algorithms *RBR* described in section 3.5, and our tag recommendation / item annotation algorithm *KatzBm25* described in section 4.5 and 4.6. The chapter also shows our algorithms comparison results with other baseline algorithms.

6.1 Evaluating the Tag-based search algorithm

In this section, we evaluate our search algorithm and compare its performance against that of state-of-the art algorithms. The experimental data came from *CiteULike*[6], a free service for managing and discovering scholarly references. The original dataset was too sparse to be used for experiments. Therefore, we cleaned the dataset to carry out experiments that were more meaningful. We selected (1) users who have been members of certain groups and have tagged at least five items; (2) items that have been labeled with at least five tags and bookmarked by at least five users; and (3) tags that at least five users have used and that have labeled to at least five items. The cleaned dataset used in this study contained 100,322 tag assignments. We projected the tag assignments onto three two-dimensional matrices: 41,596 non-zero entries in the 1121 × 14169 user-tag

[6] http://www.citeulike.org/

matrix **A** (99.7% sparsity), 35,910 non-zero entries in the 1121 × 3993 user-item matrix **C** (99.2% sparsity), and 71,900 non-zero entries in the 14169 × 3993 tag-item matrix **N** (99.9% sparsity). Table 22 briefly summarizes the dataset.

Table 22 The CiteULike dataset used in our experiments

	Users	Items	Tags	**A**	**C**	**N**	Tag assignments
CiteULike	1121	3993	14,169	41,596	35,910	71,900	100,322

6.1.1 Evaluation design and metrics

To evaluate our search *FBR* algorithm described in section 3.5, we followed the evaluation procedure described in (Wetzker et al. 2010; Zanardi et al. 2008). For each user, we randomly withheld one of their posts (i.e., an item) and their tags annotated in that post. Subsequently, those tags were used as a test query for the user. The remaining tag assignments were employed as a training set for building the models. This evaluation procedure is analogous to *leave-one-out* (Zanardi et al. 2008). For example, if user u_1 annotated item i_1 with tags, t_1 and t_2, we hid the item and the tags from all tag assignments of user u_1. We then used t_1 and t_2 to simulate the user's query and then tried to find item i_1 within the top-N ranked results. To ensure that our results were not sensitive to a particular test query for each user, we conducted five different runs, in which each run used a different test query. The experimental results reported values that were the averages of five runs with a 95% confidence interval.

To measure the retrieval accuracy, we first employed the Mean Hit Rate (MHR) at top-N, which judges how often a set of ranked results for a given query contains items that a user is actually interested in. We had one relevant item for each specific test query of a test user, so we can compute the MHR at top-N for all users in the test set by:

$$MHR@N = \frac{1}{|U|} \times \sum_{u=1}^{|U|} B_u^N \quad (17)$$

where B_u^N is a binary variable that equals 1 if an item of user *u* in the test data appears in a top-*N* ranked list for that user's test query and equals 0 otherwise. The above measure can be thought of as *recall* within a top-*N* ranked list. If B_u^N is divided by the number of returned items *N*, we can compute *precision* at top-*N*. Note that in our test scenario we treated each user's non-tagged items as non-relevant items for that user because we could not collect the entire ground truth data about which items are relevant to each user. Accordingly, precision at top-*N* is generally an underestimate of the true precision. Since the key observation from our experiment evaluations is to compare our performance against that of other algorithms, we only reported the *MHR@N* values for comparison purposes.

Since the MHR metric treats a searched item with the top-1 ranking equally with a searched item with *N*-th ranking, we also adopted the Mean Reciprocal Rank (MRR) at top-*N* to quantify the ability of ranking relevant items:

$$MRR@N = \frac{1}{|U|} \sum_{u=1}^{|U|} \left(\sum_{i \in (T_u \cap R_u^N)} \frac{1}{r(i)} \right) \quad (18)$$

where T_u is the set of test items that user *u* tagged in the test data, whereas R_u^N is the set of top-*N* returned items for user *u*'s query. Additionally, $r(i)$, $1 \le r(i) \le N$, refers to the rank of item *i* within a top-*N* list of user *u*. By measuring the accuracy with MRR, we can give a hit item appearing earlier in the top-*N* ranked list more weight than a later one received. Accordingly, the higher the *MRR@N* value, the more accurately an algorithm ranks items relevant to users. Note that the value of MHR is exactly the same as that of MRR when *N* is 1 (i.e., top-1).

In addition to MHR and MRR, we also reported *coverage*, a measure of the ratio of the items an algorithm can find for given queries (Zanardi et al. 2008):

$$Coverage = \frac{1}{|U|}\sum_{u=1}^{|U|} B_u^{|I|} \quad (19)$$

where $B_u^{|I|}$ is a binary variable that is 1 if an item of user *u* in the test data appears in all returned results for that user's test query and 0 otherwise. That is, we returned all items for a given query unless an item's ranking score is less than or equal to zero.

6.1.2 Baseline algorithms

For comparison purposes, we experimented with four different algorithms: i) *Social Ranking* without query expansion (denoted *SRk0*) and with query expansion using the *k* most similar tags (denoted *SRk5*) described in (Zanardi et al. 2008); ii) the *User-Centric Tag Model* (denoted *UCTM*) presented in (Wetzker et al. 2010); iii) the *BM25-Based Personalization Model* (denoted *CosBM25*) described in (Vallet et al. 2010); and iv) the Latent Semantic Analysis Model (denoted *LSAM*) described in (Levy et al. 2008). Because those algorithms were mostly tailored to our search scenario with social tagging, we implemented them to the best of our knowledge, based on the published papers.

Social Ranking exploited users' similarities for improving accuracy and tags' similarities for improving search coverage by expanding users' queries. Upon performance of a query expansion, the accuracy of the resultant set tended to decrease, but its coverage increased. Thus, we tested two versions of Social Ranking when $k = 0$ (i.e., no query expansion) and when $k = 5$ (query expansion using 5 most similar tags). With respect to the *UCTM*, (Wetzker et al. 2010) demonstrated that the *UCTM* outperformed the *Adapted PageRank* (Jäschke et al 2008) and the *FolkRank* (Hotho et al.

2006). However, the *UCTM* originally considered a single-tag query whereas our study considered a set of tags as a query. Accordingly, we extended the possible *UCTM* to work a multi-tag query, as per the suggestion in (Wetzker et al. 2010). In the case of the *CosBM25*, the final ranked list was created by combining the query matching based on the cosine similarity and the topic matching based on the BM25 weighting scheme, using CombSUM as a ranking aggregation method (Shaw et al. 1994).

For the *LSAM*, we selectively experimented with four approximations, a rank-500, a rank-1000, a rank-2000, and a rank-3000 approximation of **N**, derived from singular value decomposition (*SVD*). As a result, the *LSAM* with the rank-1000 approximation generally performed better than one with the other approximations; thus we set k (largest singular values) to 1000. For a given query, items were ranked in descending order of cosine similarities. Note that the *LSAM* is a non-personalized search unlike the other baseline methods.

6.1.3 Sensitivity to parameters

Prior to running the comparison experiment, we first investigated how sensitive our accuracy was regarding the number of similar tags k and similar items k' used in building the latent models, **P** and **W**, described in section 3.4.1 and 3.4.3 respectively. As described in Section 3.3, we expected that the number of similar tags and the number of similar items could be significant factors, affecting the accuracies of the personalized searches, because we built the models **P** and **W** to depend on the values k and k', respectively. Therefore, we measured the Mean Reciprocal Rank at the top-10 and the coverage according to variations of the parameter values.

Figure 24 depicts the results having 95% confidence intervals. The *x*-axis on the graph refers to the values of *k* and *k'*. For example, the first data point on the graphed curves represents $\mathbf{E}^{10}$ and $\mathbf{H}^{10}$, implying that a tag-tag similarity matrix and an item-item similarity matrix contain, at most, ten non-zero entries, i.e., 10 most similar tags and 10 most similar items. We also plotted the result (the last point of the *x*-axis labeled "all") when using the tag-tag similarity matrix **E** and the item-item similarity matrix **H**, themselves, i.e., $\mathbf{E}^{14169}$ and $\mathbf{H}^{3993}$.

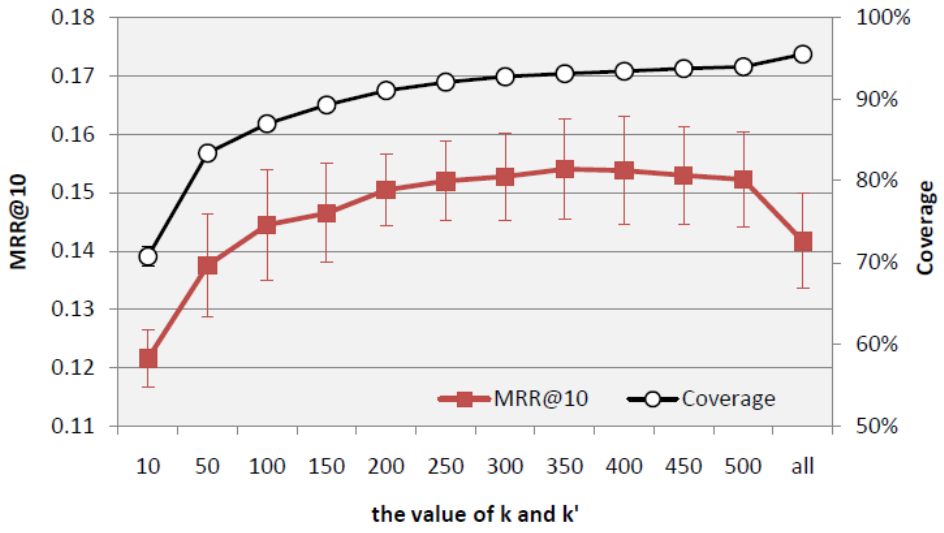

Figure 24 MRR values and coverage according to the number of similar tags and items used in building the models

The experimental results demonstrated that the MRR improved until the values reached 200; beyond that point, the curve tended to become flat though a slight variation in MRR values appeared. That is, once the numbers of similar tags and items were relatively large enough, any further increases barely changed the ranks of the searched items for each user. When all tags and items (*k* = all and *k'* = all) were used, the ranking accuracy appeared rather worse, especially as compared to small values of *k* and *k'*. When *k* and *k'* were set to 100 (less than 1% of the total tags and less than 3% of the total

items), an average MRR was 0.144 with a deviation of 0.009 whereas the MRR value was 0.142 with a deviation of 0.008 in the case of all tags and items. This result implies that superfluous, similar tags and items can include noise. This is particularly important because the computational cost can thus be reduced while the retrieval accuracy improves. The models we derived from limiting similar tags and similar items were effective and efficient in ways that avoid not only unnecessarily increasing computation costs, but also including unneeded "noise" information. As intuitively expected, we saw that the coverage value did tend to increase gradually as the values of k and k' increased. Accordingly, we should make a trade-off between ranking accuracy and coverage.

As our folksonomy-boosted ranking resulted from a combination of the models **P** and **W**, we could not present all possible combinations due to lack of space. Rather, we chose alternatively reasonable values for k and k' although these values may not be optimal. Upon considering accuracy, coverage, and computational cost, we selected k=350 and k'=350 as alternative optimal values in the subsequent comparison experiments.

6.1.4 Effect of normalization

In this experiment, we examined the effect of the matrix normalization on retrieval accuracy. To this end, we carried out experiments using a non-normalized approach that employed the matrix **A**, in Equation 3, and the matrix **N**, in Equation 6, instead of $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{N}}$, respectively. Then, we compared the MHR and MRR results of this approach to the results we obtained by the normalized approach, reported in the previous section.

.

Table 23 A comparison of MHR and MRR at top-10 obtained by a non-normalized approach and normalized approach shown with 95% confidence intervals.

	Non-normalization		Normalization	
k, k'	MHR@10	MRR@10	MHR@10	MRR@10
10	0.225±0.007	0.109±0.006	0.251±0.009**	0.122±0.005*
100	0.261±0.014	0.132±0.010	0.289±0.010***	0.144±0.009***
200	0.268±0.013	0.139±0.007	0.295±0.007**	0.150±0.006***
350	0.272±0.014	0.140±0.008	0.299±0.012**	0.154±0.009**

* Significant at $p < 0.5$, ** Significant at $p < 0.01$, *** Significant at $p < 0.001$

Table 23 shows the MHR and MRR results according to different numbers for *k* and *k'*. As a result, we saw that the normalization cases were always superior to the non-normalization cases. To analyze statistically significant improvements of the normalized approach over the non-normalized one, we also conducted two-tailed paired *t*-tests under the same conditions of *k* and *k'*. Comparing the statistical significance of these results, we observed that the method normalizing the columns of the matrix **A** and the matrix **N** performed better statistically on all occasions. The results confirmed that the matrix normalization strategy positively impacted on improvements in both retrieval accuracy and its ranking.

6.1.5 Evaluation of the proposed algorithm

As discussed in section 3.5, our personalized ranking algorithm *FBR* utilizes both latent models **P** and **W** — as discussed in section 3.4.1 and 3.43 respectively — for customizing search results to individual users. In the following experiment, we investigated how each individual model contributed to the *FBR*'s performance. For this purpose, we contrasted the performance of *FBR* that resulted from a combination of the models, **P** and **W**, with

the performance of other ranking approaches that employed 1) only **N**; 2) only **W**; 3) both **A** and **N**; 4) both **A** and **W**; and 5) both **P** and **N**. Note that, in the case of 1) and 2), ranking scores for a certain query are identical regardless of query users, i.e., non-personalized searches. All the other approaches personalize users' search results.

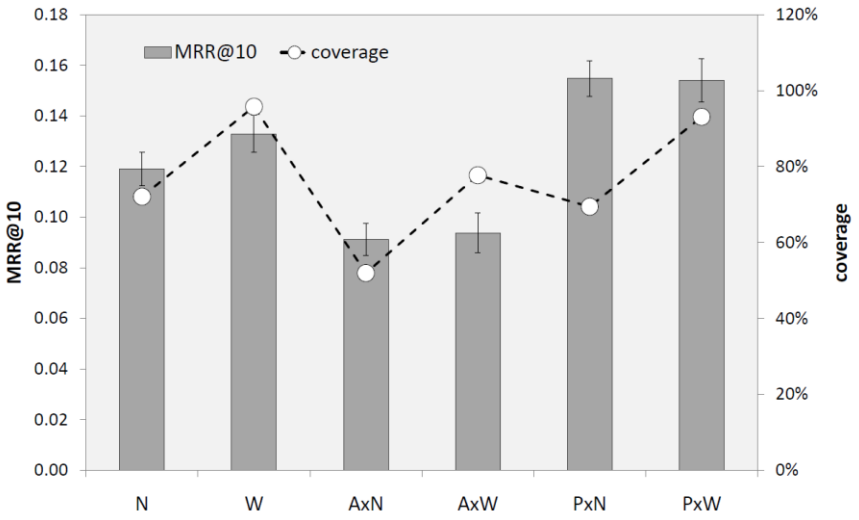

Figure 25 MRR and coverage according to different matrices used in computing raking scores

Figure 25 shows the MRR at top-10 and coverage results. The results demonstrated that our preference model **P** helped improve MRR values, implying that items in which users were actually interested ranked higher in top-10 searched lists. When **A** was used for personalized rankings, we achieved low MRR values; the non-personalized approaches performed rather better. The result was caused by the sparsity-related limitations of **A**. Tags used by each test user were only a small proportion of the total number of tags and thus users' preferences for many query tags could not be determined. In addition, items labeled only with tags that the user highly used did tend to rank higher within the top-10 result. In our dataset, the number of non-zero entries of each row (user) in **A** was 32.8 on average. However, the number of those in **P** was 7055.4, indicating that users' latent preferences for tags were remarkably extended. Interestingly, exploiting

both **P** and **N** also personalized users' search results well; the MRR value obtained by it was very close to the MRR value that was based on **P** and **W**. However, that approach failed to discover a wide range of desirable items because of **N**'s sparseness. In our dataset, the average number of non-zero entries of each column in **N** was 17.4, implying that an item was labeled with approximately 17 tags. This value increased up to 2250 in the model **W**.

Since **W** extends a description of items regarding tags, we expected that **W** could help uncover a wide variety of items in a searched list. The coverage results support this. When **W** was applied to searches, the coverage values substantially increased. More particularly, the non-personalized approach that employed only **W** performed the best with respect to coverage; it substantially outperformed another non-personalized approach that was based on **N**. Furthermore, the non-personalized approach based solely on **W** obtained 2.7% improvements on coverage compared to the personalized approach that exploited both **P** with **W**. This result might be caused by a few test users most notably having extremely insufficient personal tags. For some of such users, we were not able to infer diverse tag preferences apart from their own tags and few tags related to those tags, leading to poor coverage for their queries. In such situation, the non-personalized approach based on **W** would provide more accurate search results.

The sparsity of data could lead to uneven performance of our algorithm. As explained in section 3.3, we calculated the similarities among tags and among items by using **N**; its sparsity was 99.9% in our dataset. In our experiment, on average we generated roughly 81 similarities per tag from **N**. On the contrary, we could compute items' similarities quite well; on average 1590 similarity values per item were computed. This difference was affected by the fact that the number of tags ($|T|$=14,169) was far larger than the

number of items (|*I*|=3993). As reported earlier, overall we were able to infer quite a number of tag preferences for users from a relatively small number of similar tags. Nevertheless, in some cases, if a pair of tags has few items (or nothing) in common, we might not properly compute the similarity between the tags although they share some meanings. This is because the similarity computation is by nature based on co-occurrences of items. Analogously, it may happen that two items has no common tags and hence the similarity between those items cannot be computed, even if they are related to each other. Dimensionality reduction techniques, such as *LSA* (Deerwester et al. 1990) and Probabilistic LSA (Hofmann 1999; Hofmann 2004), could address this issue and thus improve our retrieval accuracy. We intend to incorporate the dimensionality reduction technique in building our models **P** and **W** in the future although it could increase computational cost for building the models.

6.1.6 Comparison with other ranking algorithms

The following experiment compares our ranking algorithm *FBR* to *LSAM*, *CosBM25*, *SRk0*, *SRk5*, and *UCTM*. To examine how well each algorithm positioned a relevant item at a higher rank for any given user's query, we selectively varied the number of returned items *N*: 1, 5, 10, and 20.

We first measured the *MHR@N* obtained via the six algorithms. Figure 26 depicts the results showing how *FBR* outperformed the baseline algorithms. As shown in the graph, it is clear that *FBR* performed better than the baseline algorithms at a small number of *N*, such as the top-1 and top-5. In the top-1 case, we found that, on average, *FBR* improved MHR by 1.1% over the *LSAM*, by 5.3% over the *CosBM25*, by 3.5% over the *SRk0*, by 3.8% over the *SRk5*, and by 2.5% over the *UCTM*, respectively. Based on two-tailed

paired *t*-tests, the differences between *FBR* and the baseline algorithms appeared to be statistically significant at the 1% level ($p < 0.01$), except for *LSAM*. The result implies that our search algorithms can provide items that users would like the most with a higher rank. As top-*N* increased, the differences grew slightly narrower. In particular, *CosBM25* provided a slightly better MHR than *FBR* did at top-20 although the difference appeared comparatively insignificant.

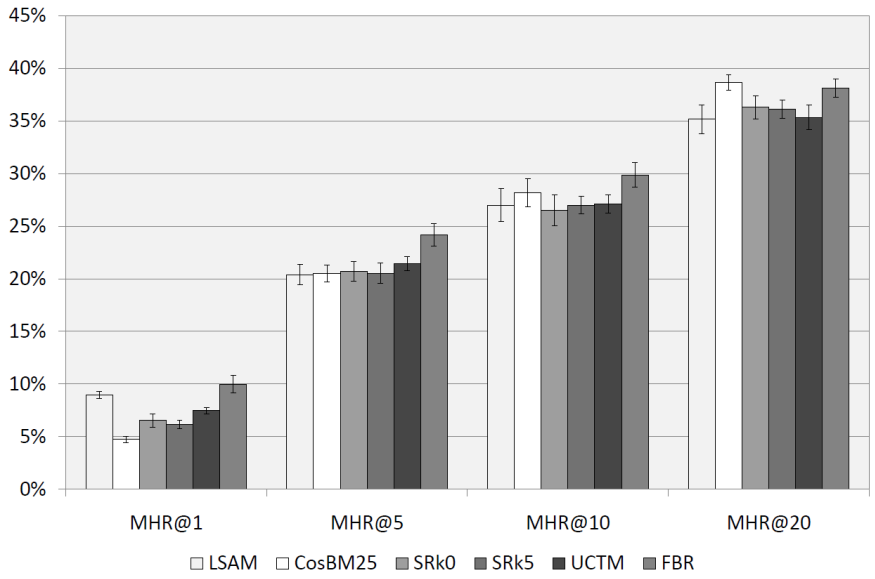

Figure 26 MHR with 95% confidence intervals obtained using the six algorithms

To provide insight of our algorithm's advantages regarding personalized ranking ability, we also compared the MRR results obtained by the *FBR* to those obtained by baseline algorithms. Figure 27 shows that *FBR* continued to substantially outperform the other algorithms in the majority of cases. Comparing results for all cases, a significant performance disparity was apparent between our algorithm and the other baseline algorithms, although there were few cases where *LSAM* achieved comparable results. Interestingly, *LSAM* that employed the dimensionality reduction also provided relevant items with a higher rank, as compared to the other baseline algorithms. It turned out that

the differences between *FBR* and *LSAM* were not statistically significant for MRR. However, *LSAM* performed the worst in terms of uncovering all relevant items as the following result of coverage shows. For the other baseline algorithms, all MRR differences were statistically significant ($p < 0.01$) based on two-tailed paired *t*-tests. These MRR improvements are more remarkable considering that users generally tend to be concerned with items having higher ranks. Upon comparing the *CosBM25* results regarding *MHR*@20 and *MRR*@20, we observed interesting results. In contrast to the MHR result, the *CosBM25* performed the worst in terms of MRR. This result indicated that *CosBM25* could not provide items that users would like the most with a higher rank in a searched list. However, our algorithms produced consistently good performance.

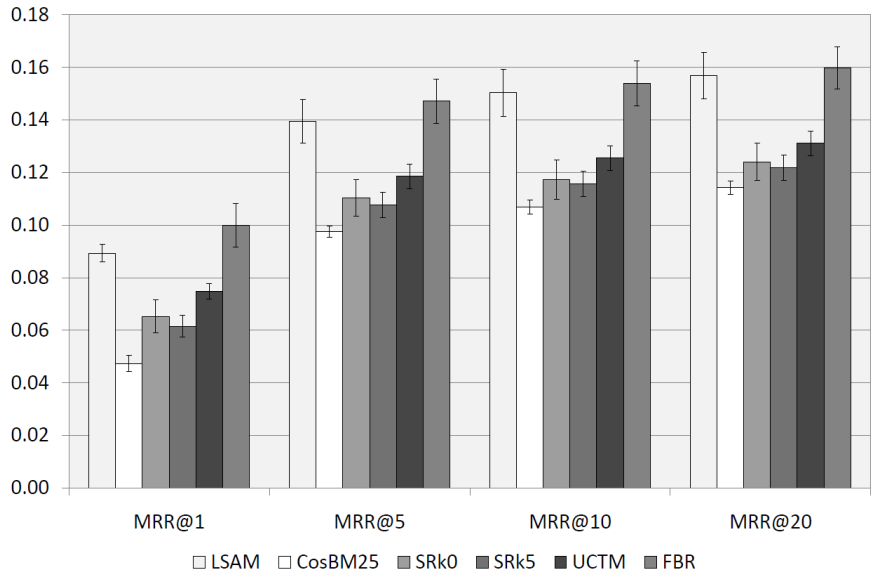

Figure 27 MRR with 95% confidence intervals obtained using the six algorithms

Ideally, search algorithms should provide a wide range of desirable items for users' queries. Therefore, we further examined each algorithm's coverage. Table 24 shows the results with 95% confidence intervals. Our algorithms obtained approximately 22%, 8%,

21%, 14%, and 16% improvements on coverage compared to *LSAM*, *CosBM25*, *SRk0*, *SRk5*, and *UCTM*, respectively.

Table 24 Coverage of the six algorithms shown with 95% confidence intervals

	LSAM	CosBM25	SRk0	SRk5	UCTM	FBR
Coverage (%)	71.36±1.6	85.07±0.67	72.07±1.32	79.27±1.14	77.56±1.04	93.10±0.41

Basically, *SRk0* can only find items labeled with at least one of the tags contained in a query. On the other hand, it is theoretically possible that the other methods, including our algorithm, can retrieve some items not previously tagged with query tags. In *LSAM*, a query vector is transformed into a new query vector in a reduced semantic space, enabling to compute the cosine similarity between the query and an item in the reduced dimensionality. Due to the nature of the singular value decomposition, *LSAM* often achieved negative ranking scores for relevant items in the reduced space. Since we did not returned items having the negative scores, *LSAM* produced the worst coverage as shown in the table. When we tested *LSAM* with other approximations, coverage became even worse. *SRk5* expands a query by utilizing the five most similar tags to each tag contained in the query; thus, it could find items annotated with tags contained in this extended query.

As can be seen from the results, such query expansion indeed helped improve coverage, as compared to the *SRk0*. The *CosBM25* performs two ranking processes, a query term matching process and a topic-matching process, to generate a final ranked list. The topic-matching process is based on how a search user is interested in items in terms of the tags labeled to the items. As a result, items annotated with tags used by a search

user could be searched, probably with low ranks, even though the items had not been previously tagged with any query tags.

As for the *UCTM*, it first translates a search user's query tag to co-occurrence tags, representable by a vector. Thus, items labeled with the translated vector could be retrieved. Consequently, *UCTM* did not produce the satisfactory coverage we had expected. In our algorithm, although a particular item was not annotated by a query tag, we could search that item if the query tag once had been labeled to items similar to that item. Analogously, we could infer personal preference of a search user for each tag in a query from the user's tags that were similar to the query tag; contrary to *UCTM*, accordingly, our algorithm could easily represent query tags as a vector regardless of whether the searcher had used them.

These comparison experiments clearly showed that our algorithm *FBR* could offer considerable performance in terms of both retrieval accuracy and coverage as compared to the state-of-the art baseline algorithms.

6.2 Evaluating the Tag Recommendation Algorithm

Table 25 Characteristics of the datasets

Dataset	users	items	tags	tag assignments
CiteULike	2614	4096	2310	161,395
Last.fm	1892	12,523	9749	186,479

As Table 25 summarizes, the datasets used to evaluate our tag recommendation model was taken from CiteULike and Last.fm[7]. In order to focus on a dense portion of the CiteULike dataset, we pruned the dataset as discussed in section 6.1. The pruned

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

CiteULike dataset contained 2614 users, 4096 items, 2310 tags, and 161,396 tag assignments. We projected the tag assignments onto three two-dimensional matrices, resulting in 62,112 non-zero entries of the user-tag matrix (1.03% density), 65,325 non-zero entries of the user-item matrix (0.61% density), and 72,619 non-zero entries of the tag-item matrix (0.77% density).

The second dataset is Last.fm, a social music service that assists users to discover, tag, and share music. The Last.fm dataset[8] used in this study was collected by the Informational Retrieval Groups at Autónoma University of Madrid (Cantador et al. 2011). This dataset contains 186,479 tag assignments on 12,523 items (i.e., music artists) from 1892 users with 9749 tags. We also projected the tag assignments onto three two-dimensional matrices: 35,816 non-zero entries of the user-tag matrix (0.19% density), 71,064 non-zero entries of the user-item matrix (0.3% density), and 109,750 non-zero entries of the tag-item matrix (0.09% density).

Note that the CiteULike evaluation dataset was relatively more clean (and dense) data compared to the Last.fm evaluation dataset. This means that the former would include few noise tags, whereas the latter would contain many personal and self-referential tags. Additionally, the CiteULike dataset did provide relatively strong connectivity between users and tags, between users and items, and between tags and items, as compared to the Last.fm dataset.

6.2.1 Experimental design and metrics

To evaluate our tag recommendation algorithm (*KatzBm25*) described in section 4.5, we followed the evaluation procedure described in (Jäschke et al. 2008; Zanardi et al. 2008;

[8] http://www.grouplens.org/node/462

Rendle et al. 2010) which has widely used for evaluating tag recommender algorithms. For each user, we randomly eliminated one item and his/her tags assigned to that item from the training set. This eliminated set was used as the test set. For example, if user u_1 annotated item i_1 with two tags, t_1 and t_2, we withheld the item i_1 and the tags t_1 and t_2 from the entire tag assignments associated to user u_1. Then, by using the training set, we examined whether a tag recommender algorithm could recommend t_1 and t_2 for u_1 associated with i_1 within the top-N ranked list. We repeated this procedure five times with different test/training datasets. Thus, the reported values in the experimental results are the mean performance averaged over these five runs. In the Last.fm dataset, some users had only one tagged item (252 out of 1892 users). We therefore did not carry out the evaluation procedure for such users, as there were no training data for them.

As evaluation metrics, we first employed the mean average precision (MAP) (Krestel et al. 2009):

$$MAP = \frac{1}{|U|}\sum_{u=1}^{|U|}\left(\frac{1}{|T_u|}\sum_{k=1}^{|T_u|} B_k \times P@k\right) \tag{20}$$

where T_u is the set of tags that user u annotated the item in the test data, $P@k$ is precision at top k, and B_k is a binary variable that is 1 if the tag with rank k in the recommended list appears in T_u and 0 otherwise.

In addition, we also measured the mean reciprocal rank (MRR) (Vallet et al. 2010) for evaluating the rank of the relevant tags:

$$MRR = \frac{1}{|U|}\sum_{u=1}^{|U|}\left(\sum_{t \in T_u} \frac{1}{rank(t)}\right) \tag{21}$$

where $rank(t)$ refers to the rank of relevant tag t within the ranked list for user u. If relevant tags appear at the very top of the ranked list made by an algorithm, it achieves a

higher MRR value. While we focused mainly on MAP and MRR for empirical analyses, we also reported *precision* and *recall* so as to compare the performance of our method against that of other methods.

6.2.2 Choice of parameter α

Before implementing our *KatzBm25* algorithm, a crucial question is what value we should choose for the parameter α that allows for the lower efficacy of longer paths during the Katz score computation. By tuning this value, we may improve the recommendation performance. Therefore, this section investigates how our recommendation performance is sensitive to the α parameter as discussed in equation 12. Because the Katz score also depends on weights — *Binary*, *Frequency*, Bm25— on links as discussed in section 4.3, we assessed our *KatzBm25* algorithm performance according to the different values of α with different weights. As mentioned earlier, for the convergence of the Katz score, the value of α should satisfy the condition $\alpha < 1/\lambda_{max}(\mathbf{A})$ in which $\lambda_{max}(\mathbf{A})$ is the largest absolute value of any eigenvalue of the adjacency matrix **A**. Table 26 shows this constraint of α associated with different weights as presented in section 3.3, in the case of the BM25 weight, α has to be less than 0.0069 on the CiteULike dataset and less than 0.0059 on the Last.fm dataset in order to achieve the convergence.

Table 26 The largest absolute eigenvalue associated with different weights

	Weight	$\lambda_{max}(\mathbf{A})$	$\alpha < 1/\lambda_{max}(\mathbf{A})$
CiteULike	Binary	117.61	0.0085
	Frequency	580.70	0.0017
	BM25	144.51	0.0069
Last.fm	Binary	138.37	0.0072
	Frequency	1219.61	0.0008
	BM25	168.82	0.0059

We first measured MRR according to different values of α with respect to different weights in order to choose the best α for each weight. Figures 28, 29 and 30 show MRR results on the CiteULike dataset. The best α value was achieved at 0.0005, 0.0005, and 0.006 for the binary, frequency and BM25 respectively.

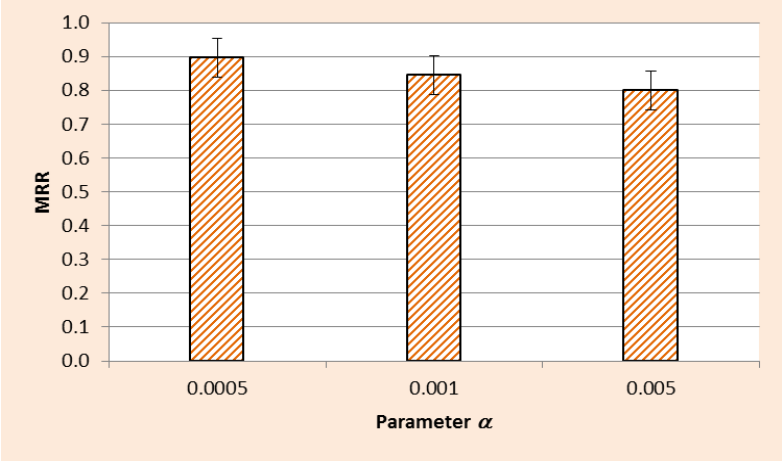

Figure 28 Different values of α using the binary weight on CiteULike

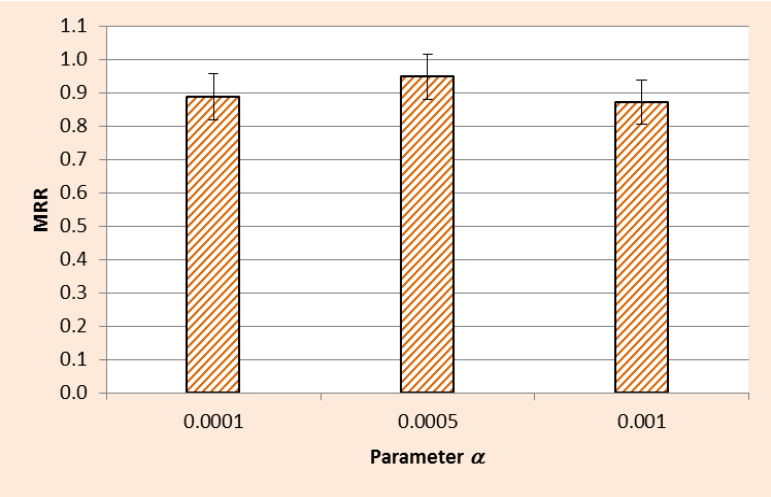

Figure 29 Different values of α using the frequency weight on CiteULike

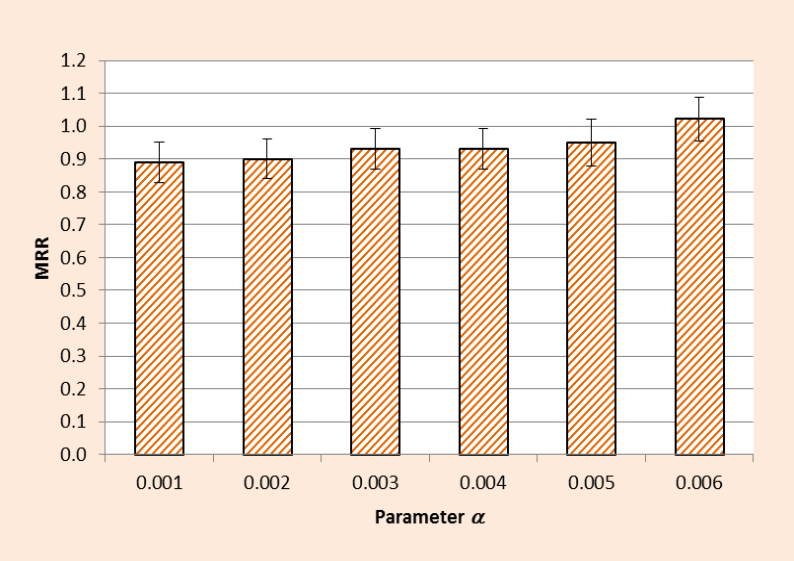

Figure 30 Different values of α using the BM25 weight on CiteULike

The MRR results on the Last.fm dataset are shown in Figures 31, 32 and 33. The best α value was achieved at 0.0005, 0.0001, and 0.005 for the binary, frequency and BM25 respectively. As a result, we identified that small values of α yielded slightly better performance than did higher values of α. It is worth mentioning that the choice of small α values brings long paths having less influence over the Katz score. In other words,

when α becomes smaller, the Katz score is profoundly affected by shortest paths such as direct links.

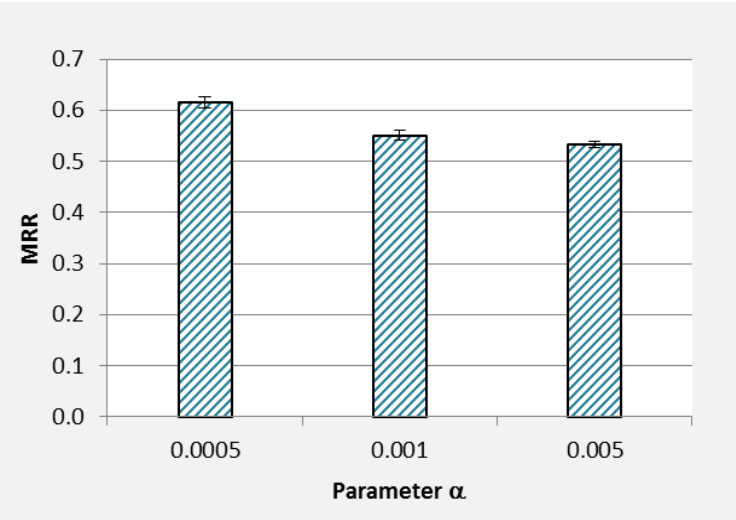

Figure 31 Different values of α using the binary weight on Last.fm

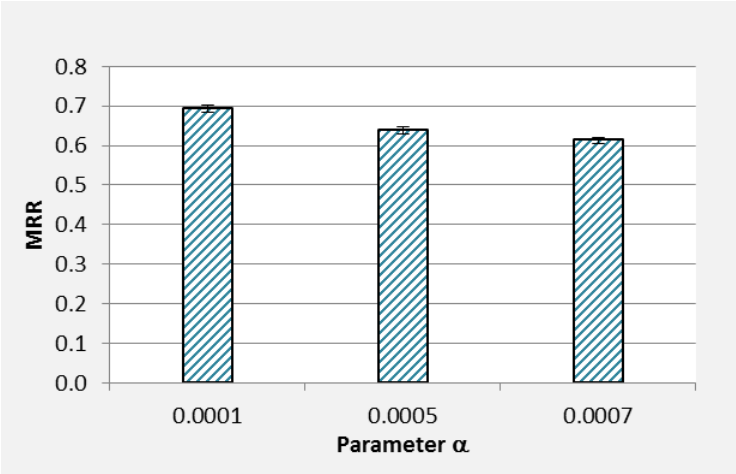

Figure 32 Different values of α using the frequency weight on Last.fm

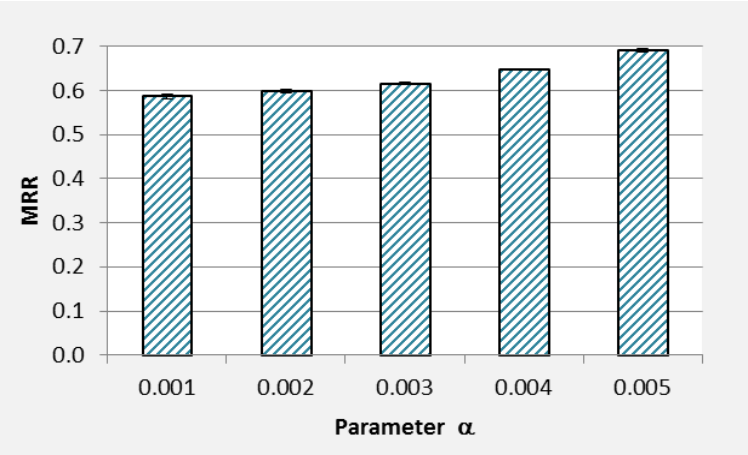

Figure 33 Different values of α using the BM25 weight on Last.fm

We further investigated the effect of the BM25 weight in comparison with the binary and frequency weights. We chose the best value of α for each weight and measured MAP and MRR values so as to compare the results. We also tested a simply directed-link-only-based Katz score using the BM25 weight (denoted as BM25-DL) to investigate whether or not indirect connections (paths) help to improve the recommendation performance. This BM25-DL approach is analogous to a mix of most popular tags recommenders described in (Jäschke et al. 2008) — a mix of the most popular tags of a given user with the most popular tags of a given item— apart from the utilization of the BM25 weight. Table 27 shows the results. The second column refers to the weights used for calculating the Katz scores and the third one refers to the best value for α producing the best performance when each weight was used.

Table 27 MRR and MAP values associated with different weights

	Weight	α	MAP±STDEV	MRR±STDEV
CiteULike	Binary	0.0005	0.317 ± 0.006	0.896 ± 0.058
	Frequency	0.0005	0.336 ± 0.003	0.948 ± 0.063
	BM25	0.006	**0.375 ± 0.005**	**1.022 ± 0.066**
	BM25-DL	-	0.338 ± 0.008	0.952 ± 0.058
Last.fm	Binary	0.0005	0.195 ± 0.003	0.615 ± 0.011
	Frequency	0.0001	0.254 ± 0.005	0.693 ± 0.009
	BM25	0.005	**0.257 ± 0.003**	**0.695 ± 0.004**
	BM25-DL	-	0.198 ± 0.010	0.582 ± 0.006

The experimental results demonstrated that the Katz score with the BM25 weight outperformed the Katz score with the other weights on both datasets. For the CiteULike dataset, improvements of approximately 5.8% and 3.9% on MAP were noted compared to the binary weight and the frequency weight, respectively. However, the MRR and MAP results obtained using the BM25 and frequency weight appeared roughly the same for the Last.fm dataset. Compared to the simply BM25-DL, the Katz BM25 method that considers the ensemble of all possible paths between nodes yielded more precise results on both evaluation datasets. This implies that indirect paths are indeed helpful for enhancing the recommendation performance, even though longer paths have less impact on the Katz score than do shorter paths.

For statistical comparisons of multiple weights over the same test users, we conducted the Friedman test, which is a non-parametric equivalent of the repeated-measures ANOVA, as there is no guarantee for normality of recommendation accuracy distributions (Demsar 2006). If the null-hypothesis—all weighting methods perform the same and the differences found are merely random—is rejected by the Friedman test, we proceeded with the Bonferroni post-hoc test to determine which pairs of methods are significantly different, and which are not

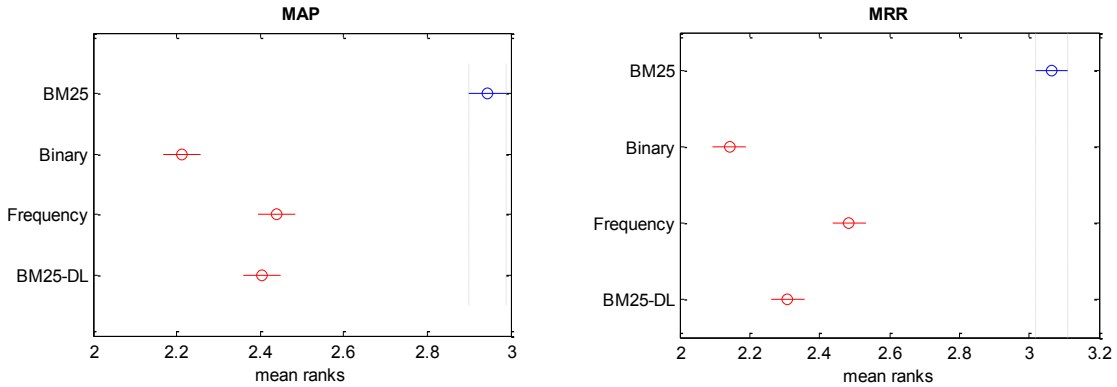

Figure 34 Comparison of BM25 against other weights on the CiteULike dataset

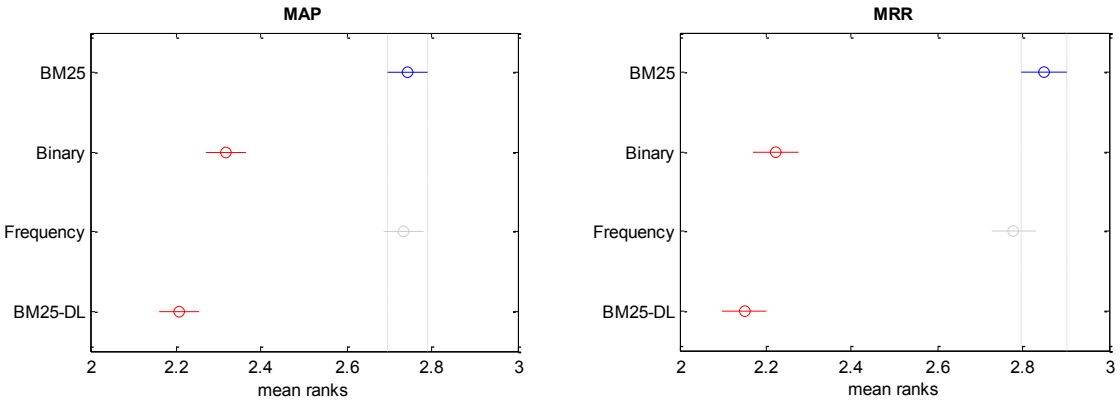

Figure 35 Comparison of BM25 against other weights on the Last.fm dataset

. Figure 34 and Figure 35 depict the results of the post-hoc tests after the Friedman tests for MRR and MAP. Note that two mean ranks which are being compared are significantly different at the 5% significance level if their confidence intervals do not overlap; otherwise, they are not significantly different. As shown, it turned out that the differences between the BM25 and the frequency weight on the Last.fm dataset were not statistically significant for both MAP and MRR. Outside of this case, all the differences of the mean ranks appeared to be statistically significant at the 5% level. Upon considering the MRR and MAP values, we chose the BM25 weigh at $\alpha = 0.006$ for the

CiteULike dataset and the BM25 weight at α = 0.005 for the Last.fm dataset in the following experiments.

6.2.3 Comparison with other baseline algorithms

In this section, we compare our tag recommendation algorithms *KaztBm25* with the following baseline algorithms: i) User-Centric Tag Model (denoted as *UCTM*) proposed by Wetzker et al. (2010), ii) the *FolkRank* algorithm which is one of the most frequently cited studies among folksonomy-based algorithms (Hotho et al. 2006), and iii) the most *Popular Tag* approach (Jäschke et al. 2008). We first calculated precision and recall obtained via the four methods by changing the number of recommended tags *N* from 1 to 10.

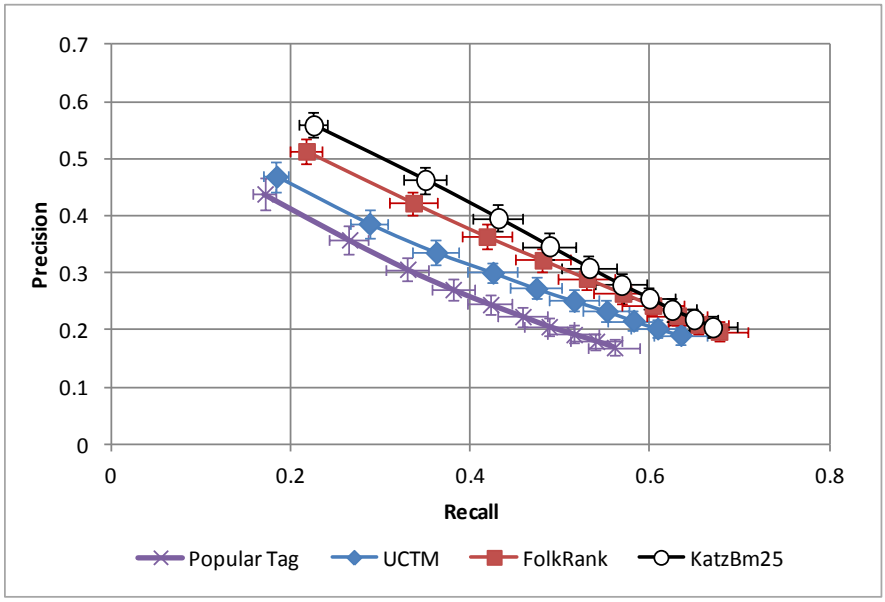

Figure 36 Precision and recall with respect to top 10 recommended tags on CiteULike

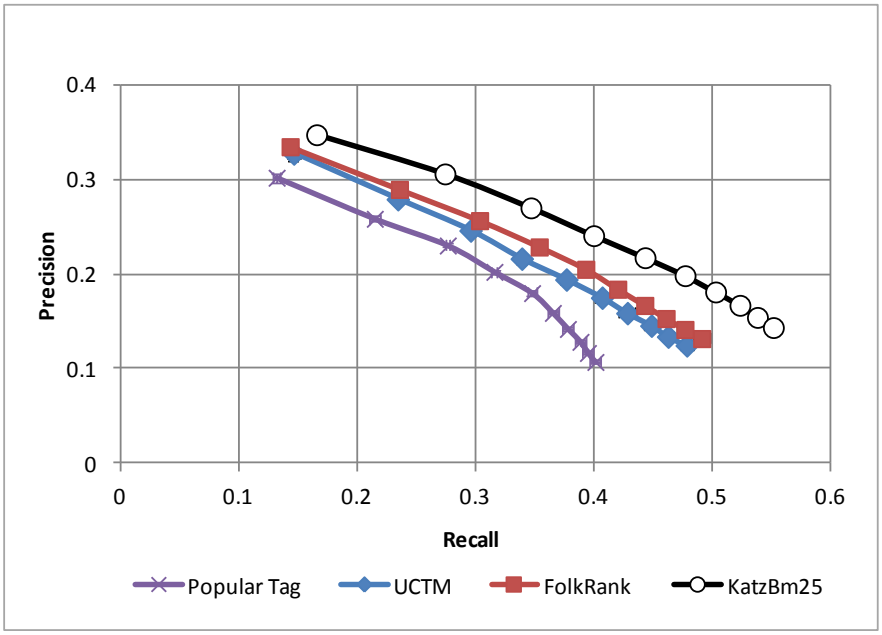

Figure 37 Precision and recall with respect to top 10 recommended tags on Last.fm

Figure 36 and 37 depict the precision-recall curves, showing how the precision and recall changes as the number of recommended tags increases. Note that the number of recommended tags is plotted on data points of the graph curves; the first point of each curve refers to the case of the top-1 whereas the last point is the case of the top-10 recommendations. Overall, we found that our algorithm *KatzBm25* outperformed the baseline algorithms in all *N* values on both datasets (except for recall at top-10 on the CiteULike dataset where FolkRank slightly performed better). Comparing the results of our algorithm *KatzBm25* and *FolkRank* on the CiteULike dataset, we observed that the differences became narrower as *N* increased. On the Last.fm dataset, however, *KatzBm25* gradually outperformed the baseline algorithms as *N* increased.

We further examined MAP and MRR of each algorithm. These evaluation measures help us see whether desirable tags are appearing at the very top of the ranked list. We note that on average users had approximately 3.2 tags and 3.5 tags in the CiteULike and Last.fm test data, respectively.

Table 28 MRR and MAP results shown with standard deviations

	Method	MAP±STD	MRR±STD
CiteULike	*Popular Tag*	0.276 ± 0.008	0.808 ± 0.058
	UCTM	0.306 ± 0.007	0.902 ± 0.065
	FolkRank	0.342 ± 0.009	0.960 ± 0.062
	KatzBm25	**0.375** ± **0.006**	**1.022** ± **0.067**
Last.fm	*Popular Tag*	0.212 ± 0.004	0.550 ± 0.003
	UCTM	0.234 ± 0.006	0.619 ± 0.007
	FolkRank	0.226 ± 0.004	0.654 ± 0.005
	KatzBm25	**0.257** ± **0.004**	**0.695** ± **0.009**

Consequently, on average we considered the top-3 or top-4 ranked tags when calculating MAP, because the number of recommended tags for each test user depends on how many tags he/she assigned to his/her test item. Table 28 shows MAP and MRR results obtained using the four algorithms. From Table 28, we can see that the performance of our algorithm *KatzBm25* is indeed substantially better than that of other alternatives that were considered. We also conducted Friedman tests for statistical comparisons of multiple algorithms in regard to MAP and MRR. As the null-hypothesis was rejected ($p < 0.05$), we thus continued to perform Bonferroni post-hoc tests. As Figure 38 and Figure 39 show, the baseline algorithms have mean ranks significantly different from our *KatzBm25* because the confidence intervals of *KatzBm25* and those of the others are all disjoint. These results confirm that MAP and MRR improvements were statistically significant.

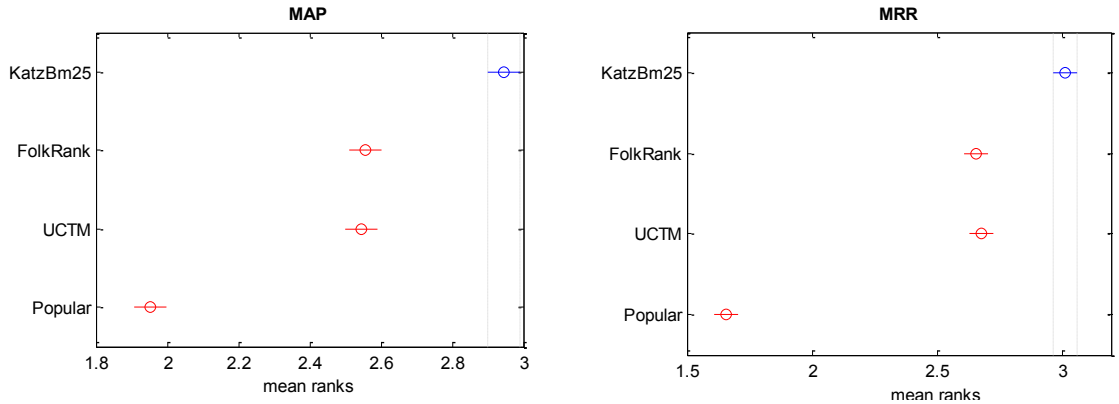

Figure 38 Statistical comparisons of KatzBm25 against other methods on CiteULike

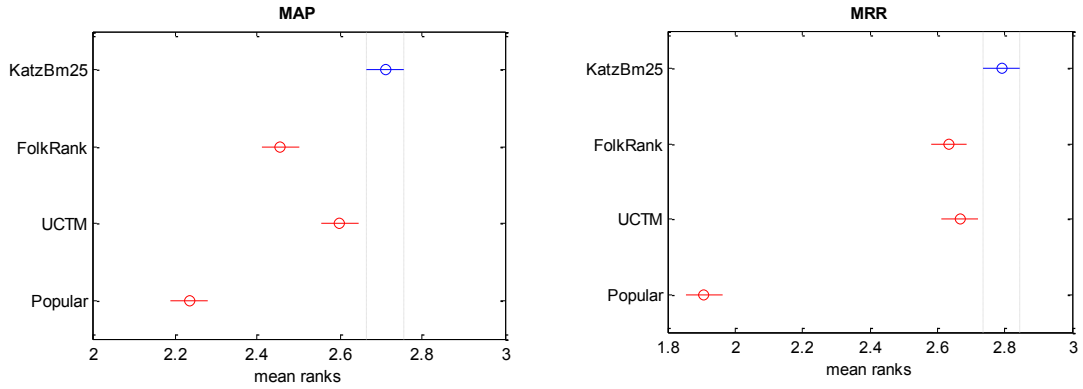

Figure 39 Statistical comparisons of KatzBm25 against other methods Last.fm

In social tagging systems, it is often that some users are very active in utilizing tags while some other users use few tags. Accordingly, the recommendation performance for individual users could be affected by how many tags each user has used. To show this impact on the recommendation performance, for every training data, we divided users into five groups according to their tagging activities: (i) *very low taggers* who used less than 5 different tags; (ii) *low taggers* who used greater than or equal to 5 tags and less than 10 tags; (iii) *medium taggers* who used greater than or equal to 10 tags and less than 20 tags; (iv) *heavy taggers* who used greater than or equal to 20 tags and less than 40 tags; and (v) *very heavy taggers* who used greater than or equal to 40 tags. We then measured the MRR and MAP values with respect to each group of users. Table 29

summarizes each group's constraint and the average number of users who belonged to each group on both evaluation datasets.

Table 29 Distribution of the users' groups based on their tagging activities

Group of users	Users' tag usage	Average number of users	
		CiteULike	Last.fm
Very Low (VL)	num. of tags < 5	455.6	370.6
Low (L)	5 ≤ num. of tags < 10	595.2	305.2
Medium (M)	10 ≤ num. of tags < 20	556.2	332.6
Heavy (H)	20 ≤ num. of tags < 40	523.4	306.8
Very Heavy (VH)	40 ≤ num. of tags	483.6	324.8

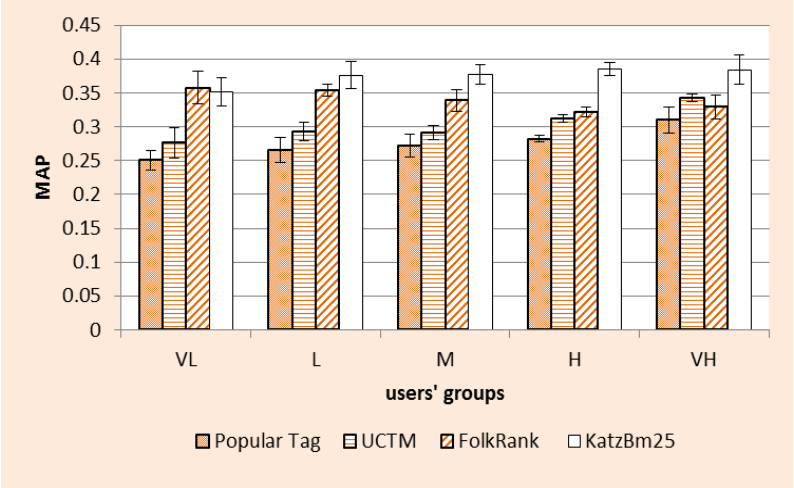

Figure 40 MAP result at different groups on the CiteULike dataset

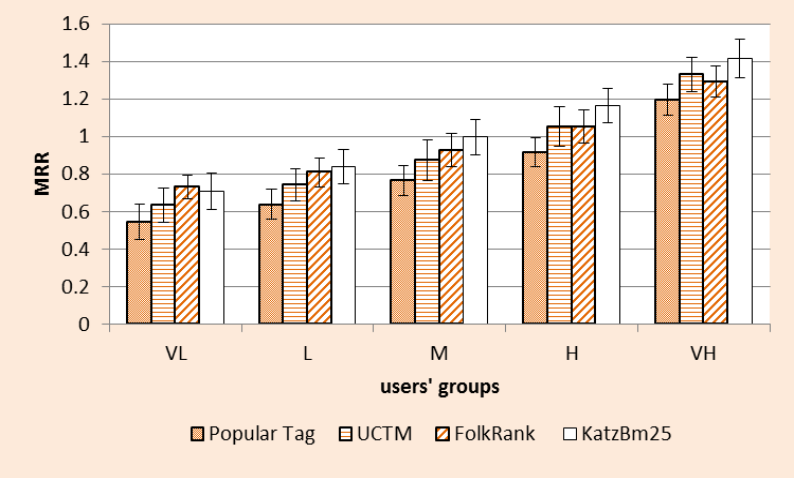

Figure 41 MRR result at different groups on the CiteULike dataset

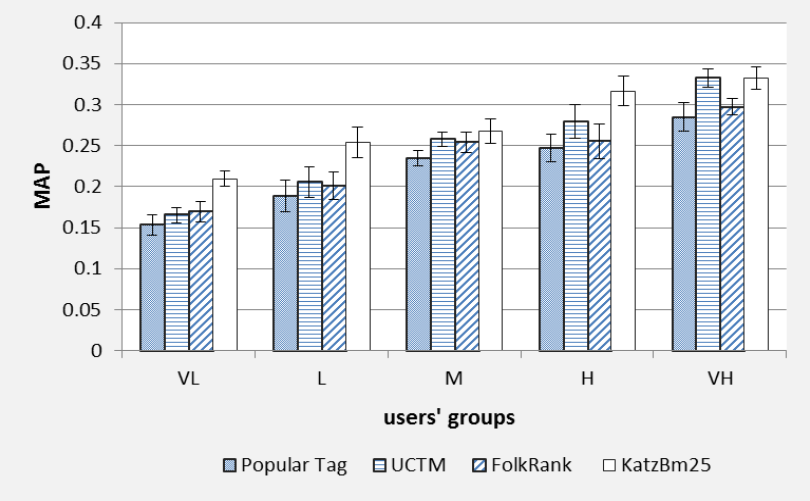

Figure 42 The MAP result at different groups on the Last.fm dataset

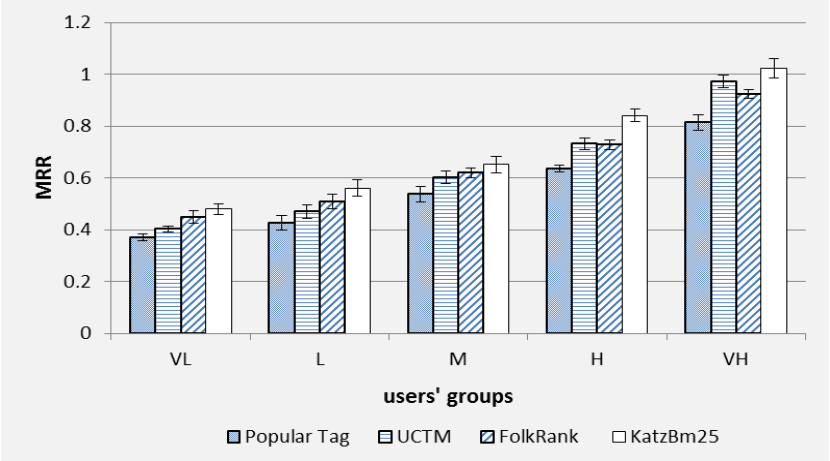

Figure 43 The MRR result at different groups on the Last.fm dataset

Figures 40-43 show the MAP and MRR results for different groups of users on the CiteULike and Last.fm dataset, respectively. When we looked at the MAP results obtained with each algorithm on the CiteULike dataset, every algorithm obtained similar MAP values for each group. This was intriguing as we intuitively expected that the level of users' tagging activities could be a significant factor, influencing the quality of tag recommender systems. This result might be caused by the characteristics of the CiteULike dataset we experimented with. As described in section 6.1, this dataset contained well-refined relations of a folksonomy. The performance of the tag

recommender systems would not only depend on the number of tags a target user has used, but also rely on the number of tags annotated in a target item. To explain this result, we looked into this experimental data in more detail. Accordingly, we observed that on average 88.5% of the total tags occurring in each user-item test pair also appeared in the training tag set of the corresponding user or that of the corresponding item. In other words, most of users in the refined CiteULike dataset did tend to annotate a certain item with their previously used tags or, more particularly, with tags labeled to that item. Contrary to the MAP results, the larger number of given tags per test group improved MRR for all the methods.

On the Last.fm evaluation dataset, however, the performance of all the algorithms was profoundly sensitive to the number of tags used by test users as shown in Figure 42 and Figure 43. It was determined that the more tags users used, the better recommendation quality they received. For instance, our algorithm *KatzBm25* achieved an MAP of 0.33 for the very heavy taggers while achieving an MAP of 0.21 for the very low taggers (an improvement of 12% on MAP). Similar improvements were also observed for the other algorithms.

We continued with comparisons to results obtained using each algorithm within each same group. In the case of *UCTM*, it tended to perform well particularly for users who had many tags, but provided users having few tags with poor recommendations. On the contrary, *FolkRank* made good recommendations to the very lower taggers most notably on the CiteULike dataset, especially as compared to our method. As for the *Popular Tag* algorithm, it achieved the worst performance on all occasions, thereby indicating that merely the popularity of tags annotated in a given item is not enough to fully reflect individual tagging behaviors. For both two evaluation sets, our algorithm *KatzBm25* was

consistently and clearly superior to the other methods in all cases of groups in terms of both MAP and MRR.

Not only does our algorithm perform well for the active users, but it is also capable of recommending more appropriate tags to the cold start users compared to the baseline algorithms. Since the recommendation problem for the cold start users is one of notable challenges in the field of recommender systems, the proposed algorithm *KatzBm25* can be beneficial to this problem.

6.3 Evaluating the Item Annotation

Our *KatzBm25* algorithm can be applied for tag recommendation and item annotation as discussed earlier in section 4.5 and 4.6. This section is mainly to evaluate our *KatzBm25* algorithm for the item annotation case as discussed in section 4.6. As baselines algorithms, we used the *FolkRank* algorithm again and two other algorithms: the personalized *PageRank* algorithm based on a random walk with restarts (Liben-Nowell et al. 2007) and the *LDA*-based algorithm described in (Krestel at el. 2009).

In our experiment we focused on the performance regarding the number of tags labeled to the items. To that end, we classified test items into three groups as follows: (1) *low* annotated items tagged with less than 4 different tags; (2) *medium* annotated items tagged with greater than or equal to 4 tags and less than 10 tags; and (3) *heavy* annotated items tagged with greater than or equal to 10 tags. One point that is worth noting is that our aim in this experiment was not to compare the results for one group with those for another group, due to the different sample size of the three groups and the different number of hidden test tags per group. We instead aimed to compare the performance of algorithms within each partitioned group.

Figures 44-47 plot the MAP and MRR results for the low, medium, and heavy items. As shown by the results, it turns out that our item annotation algorithm described in section 4.6 is highly effective in finding relevant tags for the items labeled with sufficient tags, as well as in doing for the items with insufficient tags, as compared to the baseline methods. For example, on CiteULike dataset we observed for the low annotated items that our approach improved MAP by 20.8% over the *LDA*, by 11.3% over the PageRank, and by 7.9% over the FolkRank, respectively. On Last.fm dataset, our algorithm *KatzBm25* improved MAP by 16.2% over the *LDA*, by 7.4% over the PageRank, and by 3.7% over the *FolkRank*, respectively. The results for the other groups validate the superiority of our algorithm as well. Interestingly, the *LDA* algorithm that models latent topics merely from the tag-item count matrix **N** produced remarkably unsatisfactory performance, particularly in situations where items were being tagged with a few tags. This result may be explained by the fact that the *LDA* cannot observe explicit relations between users and tags, and between users and items from the tag-item matrix. On the other hand, the graph-based algorithms that attempt to analyze the tripartite link structure **A**—*PageRank*, *FolkRank*, and *KatzBm25*—appear to produce rather robust performance under "cold start" cases.

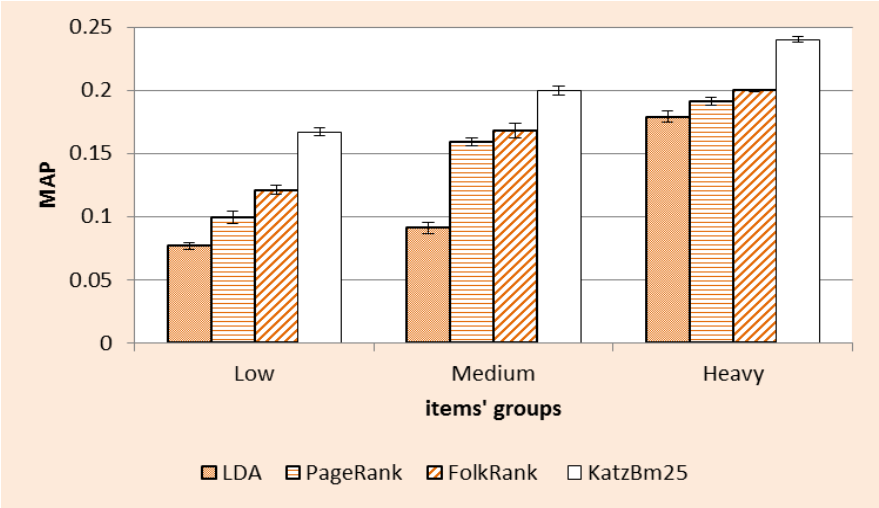

Figure 44 MAP values for tag annotation at different groups of items on CiteULike dataset

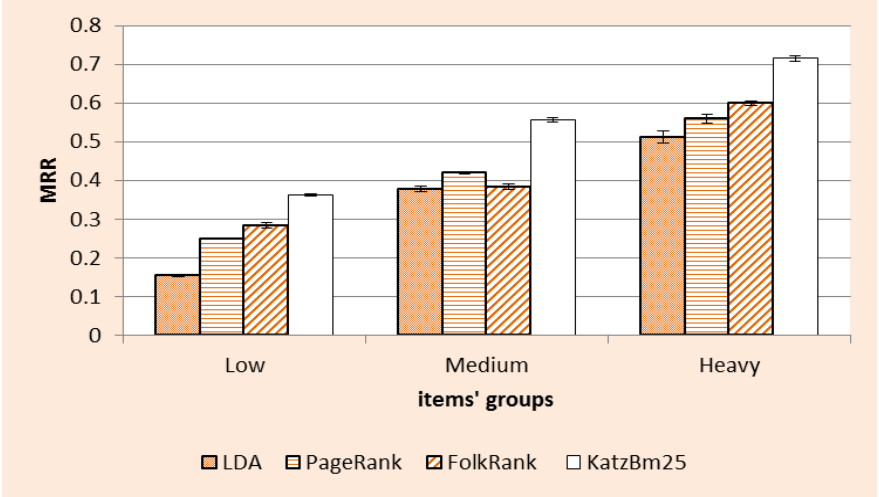

Figure 45 MRR values for tag annotation at different groups of items on CiteULike dataset

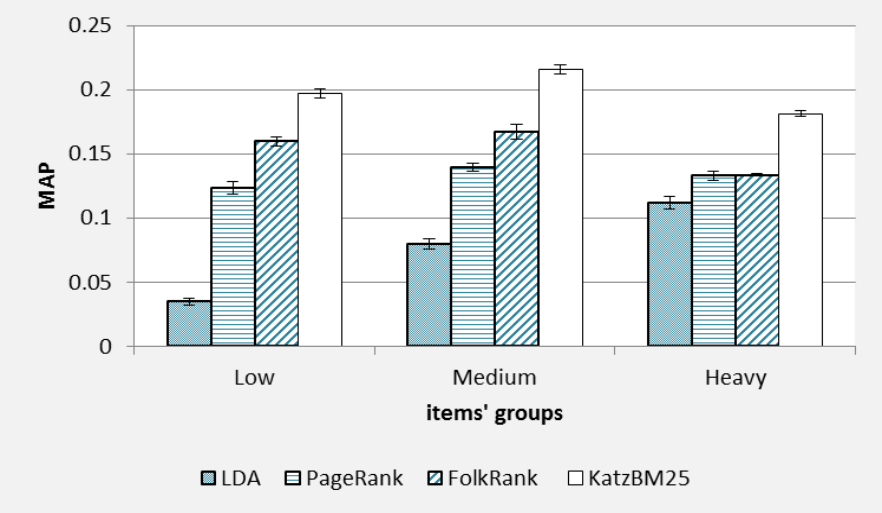

Figure 46 MAP values for tag annotation at different groups of items on Last.fm dataset

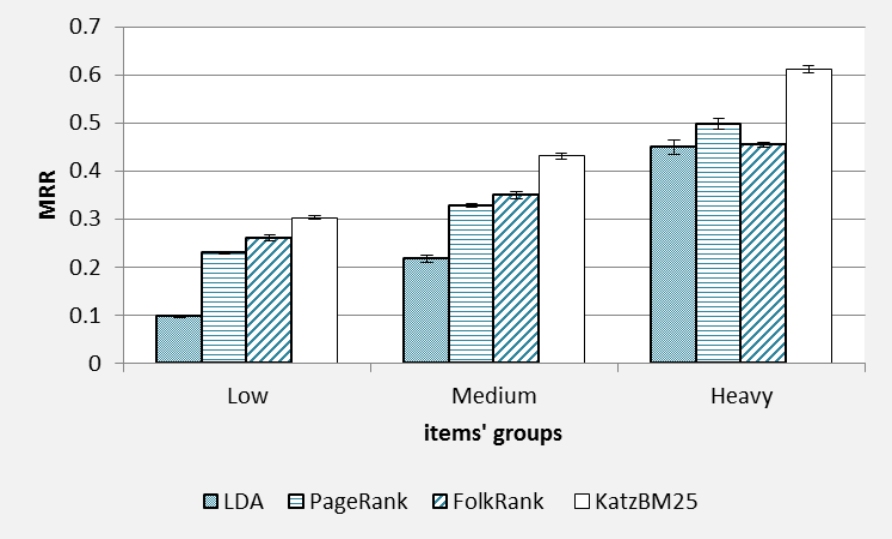

Figure 47 MRR result for tag annotation at different groups of items on Last.fm dataset

We present a summary of the experimental results in Table 30. As shown, our algorithm *KatzBm25* outperformed the baseline methods for both MAP and MRR; overall, it obtained 12.7%, 6.9%, and 4.5% improvement on MAP compared to *LDA*, *PageRank*, and *FolkRank* on Last.fm, respectively, and 12.4%, 9%, 7.6% compared to *LDA*, *PageRank*, and *FolkRank* on CiteULike, respectively.

Table 30 MRR and MAP results shown with standard deviations

	Method	MAP±STD	MRR±STD
CiteULike	*LDA*	0.115 ± 0.003	0.348 ± 0.005
	PageRank	0.149± 0.002	0.409 ± 0.007
	FolkRank	0.163 ± 0.004	0.421 ± 0.005
	KatzBm25	**0.239** ± **0.003**	**0.541** ± **0.004**
Last.fm	*LDA*	0.075 ± 0.003	0.242 ± 0.007
	PageRank	0.133 ± 0.002	0.343 ± 0.004
	FolkRank	0.157 ± 0.003	0.351 ± 0.004
	KatzBm25	**0.202 ± 0.002**	**0.440 ± 0.003**

6.4 Summary

In this section we show our experimental results on both datasets CiteULike and Last.fm. The reposted results prove that our tag-base personalized algorithm *FBR* and tag recommendation/item annotation algorithm *KatzBm25* achieved better performance than other baseline algorithms.

Chapter 7 Conclusion and Future Work

7.1 Conclusion

As a part of Web 2.0, social tagging is becoming widely used as rich information to be exploited to social media search and ranking. In this thesis, we analyzed the potential of folksonomy for searching and raking social media depending on a user. First, we proposed a new tag-based personalized search algorithm named Folksonomy-Boosted Ranking (*FBR*). The *FBR* algorithm determines the similarities among items and among tags. After that the algorithm builds two latent models: i) the latent tag preference model that reflects how a certain user has assigned tags similar to a given tag, and ii) the latent tag annotation model that captures how users have tagged a certain tag to items similar to a given item. Then the algorithm utilizes both latent models to seamlessly map the tags on the items depending on a particular user's query in order to find the most attractive media content relevant to the user needs.

The experimental results demonstrated that the proposed algorithm *FBR* indeed offers significant advantages in improving search accuracy and coverage. In addition, we believe our FBR algorithm can partially alleviate synonymy of user-generated tags, which are notable challenges in tag-based social searches.

Second, we also proposed a new personalized tag recommendation/item annotation algorithm named *KatBm25,* in order to identify relevant tags in social tagging systems. The proposed algorithm *KatzBm25* estimates proximity between users and tags, and between items and tags based on the Katz measure, and thus discovers new triangle graphs that are likely to appear within a given folksonomy .On this tripartite graph our algorithm predicts the link proximity between nodes via weighted sums over collections of possible paths connecting the two nodes.

Our experiments with the CiteULike and Last.fm datasets demonstrate that not only can the proposed algorithm *KatzBm25* accurately recommends/labels suitable tags for an item and offers more suitable items in searched results for individual users, but it is also able to successfully position such tags and items at higher ranks. Additionally, our *KatzBm25algorithm* is found to be fruitful in improving the performance for both the active taggers/heavy annotated items and the cold-start taggers/items, especially as compared to existing alternatives.

7.2 Future Work

Our research opens several tasks for interesting future work in order to successfully exploit our algorithms to a practical environment. As pointed out as a common problem in free-text tags, users make frequent use of ambiguous and synonymous tags (Golder et al. 2006). The recent systems, such as Faviki[9] and Zigtag[10], support semantic tagging that allows users to assign tags with well-defined concepts. Incorporating semantic tagging

[9] http://www.faviki.com/
[10] http://www.zigtag.com/

into our algorithms is one of the interesting works that we plan to further carry out in the future.

Since a folksonomy is 3-dimensional data (i.e. users, tags, and items), it is a natural direction to exploit a folksonomy tripartite structure for different applications, such as item recommendations for individual users, and item recommendations for a group of users together.

Our proposed approach does not require any additional information about users or items aside from tagging information, it could be potentially applicable to test various recommendation contexts—for example, TV programs, movies, and travel packages.

Though the Katz measure provides accurate proximity estimation, it is computationally expensive to calculate the ensemble of all paths between two nodes in large-scale graphs. We therefore need to study more efficient, scalable techniques that can approximate this proximity estimation for future work.

In our research, we employed the cosine-based similarity, but we plan to look into the effect of other similarity methods (Markines et al. 2009; Kim et al. 2010) on our performance in the future.

References

Bao S., Wu X., Fei B., Xue G., Su Z., Yu Y. (2007) Optimizing web search using social Annotations. In: *Proceedings of the 16th International Conference on World Wide Web*, ACM, New York, pp 501-510

Biancalana C., Micarelli A. (2009) Social Tagging in Query Expansion: A New Way for Personalized Web Search. In: *Proceedings of the 12th IEEE International Conference on Computational Science and Engineering*, IEEE Computer Society Washington, DC, pp 1060-1065

Bischoff K., Firan C. S., Nejdl W., Paiu R. (2008) Can all tags be used for search?. In: *Proceedings of the17th ACM Conference on Information and Knowledge Management,* ACM, New York, pp 193-202

Brynn M.E., Chi E.H. (2010) An elaborated model of social search. *Information Processing Management*, 46(6):656-678

Bu J., Tan S., Chen C., Wang C., Wu H., Zhang L., He X. (2010) Music recommendation by unified hypergraph: combining social media information and music content. In: *proceedings of the international conference on Multimedia*, ACM New York, pp 391- 400

Budura A., Michel S., Cudré-Mauroux P., Aberer K. (2009) Neighborhood-based tag prediction. In:*Proceedings of 6th European Semantic Web Conference*, Springer-Verlag Berlin, pp 608-622

Cantador I., Brusilovsky P., Kuflik T. (2011)Second workshop on information heterogeneity and fusion in recommender systems. In: *Proceedings of the fifth ACM conference on Recommender systems*, ACM, New York, pp 387-388

Carmel D., Zwerdling N., Guy I., Koifman S. O., Har'el N., Ronen I., Uziel E., Yogev S., Chernov S. Personalized social search based on the user's social network. (2009) In: *Proceeding of the 18th ACM conference on Information and knowledge management*, ACM New York, pp 1227-1236

Clements M., De Vries A. P., Reinders M. J. T. (2009) The influence of personalization on tag query length in social media search. *Information Processing and Management*,46(4): 403-412

Clements M., De Vries, A.P., Reinders, M.J.T. (2010) The Task-Dependent Effect of Tags and Ratings on Social Media Access. *ACM Transactions on Information Systems*, 28(4), 21

Cohen H., Kleinberg R., Szegedy B., Umans C. (2005) Group-theoretic Algorithm for Matrix Multiplication. *In proceeding of the 46th annual symposium on foundations of computer science*, IEEE computer society, pp 379-388

De Meo P., Quattrone G., Ursino D. (2009) Exploitation of semantic relationships and hierarchical data structures to support a user in his annotation and browsing activities in folksonomies. Information Systems 34(6):511-535

De Meo P., Quattrone G., Ursino D. (2010) A query expansion and user profile enrichment approach to improve the performance of recommender systems operating on a folksonomy. *User Modeling and User-Adapted Interaction*, 20 (1): 41-86

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

Demsar J. (2006) Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7: 1-30

Deshpande, M., Karypis, G. (2004) Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems*, 22(1): 143-177

Diaz-Aviles E., Georgescu M., Stewart A., Nejdl W. (2010) LDA for On-the-Fly Auto Tagging. In: *Proceedings of the 4th ACM conference on Recommender Systems*, ACM, New York, pp. 309-312

Font F., Serra J., Serra X. (2012) Folksonomy-based tag recommendation online audio clip sharing. In: *Proceedings of 13th International conference on Music Information Retrieval* (ISMIR) pp. 73-78

Foster K., Muth S., Potterat J., Rothenberg R. (2001) A faster Katz status score algorithm. *Computational & Mathematical Organization Theory*, 7(4):275-285

Gemmell J., Schimoler T., Ramezani M., Mobasher B. (2009) Adapting k-nearest neighbor for tag recommendation in folksonomies. In: *Proceedings of the 7th Workshop on Intelligent Techniques for Web Personalization & Recommender Systems*

Gemmell J., Schimoler T., Ramezani M., Mobasher B., Burke R. (2011) Tag-based resource recommendation in social annotation applications. In: *Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications*, Springer-Verlag Berlin, pp 195-206

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

Guan Z., Bu J., Mei Q., Chen C., Wang C. (2009) Personalized tag recommendation using graph-based ranking on multi-type interrelated objects. In: *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, ACM, New York, pp 540-547

Hammond T., Hannay T., Lund B., Scott J. Social (2005). Bookmarking Tools (I) – A General Review. *D-Lib Magazine*, 11(4):1082–9873

Hamouda S., Wanas N. (2011) PUT-Tag: personalized user-centric tag recommendation for social bookmarking systems. *Social Network Analysis and Mining*, 1(4):377-385

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

Herlocker J. L., Konstan J.A., Terveen L.G., Riedl J.T. (2004) Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53

Heymann P., Koutrika G., Garcia-Molina H. (2008) Can social bookmarking improve web search?. In: *Proceedings of the International Conference on Web Search and WebData Mining*, pp 195-206

Hofmann T. (1999) Probabilistic latent semantic indexing. In: Proceedings of the 22nd ACM SIGIR International Conference on Research and Development in Information Retrieval, pp. 50-57

Hofmann T. (2004) Latent semantic models for collaborative filtering. ACM Transactions on Information Systems, 22 (1):89-115

Horsburgh B., Craw S., Massie S., Boswell R. (2011) Finding the hidden gems: recommending untagged music, *In proceedings of the 22nd international joint conference of Artificial Intelligence-Volume Three*. AAAI Press, pp 2256-2261

Hotho A., Jäschke R., Schmitz C., Stumme G. (2006) Information retrieval in folksonomies: search and ranking. In: *Proceedings of the 3rd European Semantic Web Conference*, Springer-Verlag, Berlin, pp 411-426

Huang Z., Li X., Chen H. (2005) Link prediction approach to collaborative filtering. In: *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, ACM, New York, pp 141-142

Jäschke R., Marinho L., Hotho A., Schmidt-Thieme L., Stumme G. (2008) Tag recommendations in social bookmarking systems. *AI Communications*, 21(4):231-247

Katz L. (1953) A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43

Kim H.N., Ji A.T.,Ha I., Jo G.S. (2010) Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation. Electronic Commerce Research and Applications, 9 (1):73-83.

Krestel R., Fankhauser P., Nejdl W. (2009) Latent dirichlet allocation for tag recommendation. In: *Proceedings of the third ACM conference on Recommender systems*, ACM, New York, pp 61-68

Lawrenc S. (2000) Context in Web Search. *IEEE Data Engineering Bulletin*, 23(3): 25–32

Levy M., Sandler M. (2008) Learning Latent Semantic Models for Music from Social Tags. Journal of New Music Research, 37 (2):137-150

Levy M., Sandler M. (2009) Music information retrieval using social tags and audio. *IEEE Transactions on Multimedia*, 11(3):383–395

Li, X., Guo, L., and Zhao, Y. E. (2008) Tag-based social interest discovery. In: *Proceeding of the 17th international conference on World Wide Web*, ACM New York, pp 675-684

Liben-Nowell D., Kleinberg J. (2007) The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019-1031

Lipczak M., Milios E. (2010) Learning in Efficient Tag Recommendation. In: *Proceedings of the fourth ACM conference on Recommender systems*, ACM, New York, pp 167-174

Lipczak M. (2008) Tag recommendation for folksonomies oriented towards individual users. In: *Proceedings of the ECML PKDD 08 discovery challenge workshop.*

Liu D., Hua X. S., Yang L., Wang M., Zhang H. J. (2009) Tag ranking. In: *Proceedings of the 18th international conference on World Wide Web*, ACM, New York, pp 351-360

Liu F., Yu C. T., Meng W. (2004) Personalized web search for improving retrieval effectiveness. *IEEE Transactions on Knowledge and Data Engineering*, 16(1): 28-40

Ma Z., Pant G., Liu Sheng O. R. (2007) Interest-based personalized search. *ACM Transcactions on Information Systems*. 25(1), Article 5

Markines, B., Cattuto, C., Menczer, F., Benz, D., Hotho, A., Stumme, G. (2009) Evaluating similarity measures for emergent semantics of social tagging. In: *Proceedings of the 18th International Conference on World Wide Web*, ACM New York, pp 641-650

Milicevic, A. K., Nanopoulos, A., Ivanovic, M. (2010) Social tagging in recommender systems: A survey of the state-of-the-art and possible extensions. *Artificial Intelligence Review*, 33(3):187-209

Miotto R., Orio N. (2012) A probabilistic model to combine tags and acoustic similarity for music retrieval. *ACM Transactions on Information Systems*, 30(2), Article No. 8

Noll M., Meinel C. (2007) Web search personalization via social bookmarking and tagging. In: *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference*, Springer-Verlag, Berlin, pp. 365-378

Pitkow J., Schutze H., Cass T., Cooley R., Turnbull D., Edmonds A., Adar E., Breuel T. (2002) Personalized search. *Communications of the ACM*, 45(9): 50-55

Ramezani M. (2011) Improving graph-based approaches for personalized tag recommendation. *Journal of Emerging Technologies in Web Intelligence*, 3(2):168-176

Rendle S., Schmidt-Thieme L. (2010) Pairwise interaction tensor factorization for personalized tag recommendation. In: *Proceedings of the 3rd international conference on web search and web data mining*, ACM, New York, pp 81–90

Schenkel R., Crecelius T., Kacimi M., Michel S., Neumann T., Parreira J. X, Weikum G. (2008) Efficient top-k querying over social-tagging networks. In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, New York, pp 523-530

Shardanand U., Maes P. (1995) Social information filtering: Algorithms for automating word of mouth. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 210-217.

Shaw J.A., Fox E.A. (1994) Combination of multiple searches. In: Proceedings of second Text REtrieval Conference, pp. 243–252

Song Y., Zhang L., Giles C. (2011) Automatic tag recommendation algorithms for social recommender systems. *ACM Transaction on the Web*, 5(1):1-31

Sparck Jones K., Walker S., Robertson S. E. (2000) A probabilistic model of information retrieval: development and comparative experiments. *Information Processing and Management*, 36(6):809-840

Symeonidis P., Ruxanda M., Nanopoulos A., Manolopoulos Y. (2008) Ternary semantic analysis of social tags for personalized music recommendation. In: *Proceedings of 9th International conference on Music Information Retrieval (ISMIR)* pp. 219-224

Tatli I., Birturk A. (2011) A tag-based hybrid music recommendation system using semantic relations and multi-domain Information. In: *Proceedings of IEEE 11th International Conference on Data Mining Workshops* (ICDMW), pp.548-554

Vallet D., Cantador I., Joemon J. (2010) Personalizing web search with folksonomy-based user and document profiles. In: *Proceedings of the 32nd European conference on Advances in Information*, Springer-Verlag Berlin, pp 420-431

Vallet D., Cantador I., Jose J.M. (2009) Exploiting social tagging profiles to personalize web search. In: *Proceedings of the 8th International Conference on Flexible Query Answering Systems*, pp 629-640

Wang J., Clements M., Wang J., De Vries A.P., Reinders M.J.T. (2010) Personalization of tagging systems. *Information Processing and Management*, 46(1): 58-70.

Wetzker R., Zimmermann C., Bauckhage C., Albayrak, S. (2010) I tag, you tag: Translating tags for advanced user models. In: *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*, ACM, New York, pp 71-80

Wu X., Zhang L., Yu Y. (2006) Exploring social annotations for the semantic web. In: *Proceedings of the 15th International Conference on World Wide Web*, pp 417-426

Xu S., Bao S., Fei B., Su Z., Yu Y. (2008) Exploring folksonomy for personalized search. In: *Proceedings of 31st Annual International ACM SIGIR conference*, ACM, New York, pp 155-162

Yanbe Y., Jatowt A., Nakamura S., Tanaka K. (2007) Can social bookmarking enhance search in the Web?. In: *Proceedings of ACM IEEE Joint Conference on Digital Libraries,* pp 107-116

Zanardi V., Capra L. (2008) Social ranking: uncovering relevant content using tag-based recommender systems. In: *Proceedings of 2nd ACM conference on Recommender systems*, ACM, New York, pp 51-58