Article

JIS

Journal of Information Science
38(4) 313–332

DOI: 10.1177/0165551512438357
jis.sagepub.com
SAGE

A social inverted index for social-tagging-based information retrieval

Kang-Pyo Lee
Seoul National University, South Korea

Hong-Gee Kim
Seoul National University, South Korea

Hyoung-Joo Kim
Seoul National University, South Korea

Abstract
Keywords have played an important role not only for searchers who formulate a query, but also for search engines that index documents and evaluate the query. Recently, tags chosen by users to annotate web resources are gaining significance for improving information retrieval (IR) tasks, in that they can act as meaningful keywords bridging the gap between humans and machines. One critical aspect of tagging (besides the tag and the resource) is the user (or tagger); there exists a ternary relationship among the tag, resource, and user. The traditional inverted index, however, does not consider the user aspect, and is based on the binary relationship between term and document. In this paper we propose a social inverted index – a novel inverted index extended for social-tagging-based IR – that maintains a separate user sublist for each resource in a resource-posting list to contain each user's various features as weights. The social inverted index is different from the normal inverted index in that it regards each user as a unique person, rather than simply count the number of users, and highlights the value of a user who has participated in tagging. This extended structure facilitates the use of dynamic resource weights, which are expected to be more meaningful than simple user-frequency-based weights. It also allows a flexible response to the conditional queries that are increasingly required in tag-based IR. Our experiments have shown that this user-considering indexing performs better in IR tasks than a normal inverted index with no user sublists. The time and space overhead required for index construction and maintenance was also acceptable.

Keywords
information retrieval; inverted index; social tagging; tags; web search

1. Introduction

Keywords have been one of the most required elements in information retrieval (IR) tasks. Searchers' information need is represented by a search query, which usually consists of a set of keyword terms. Consequently, it is critical for the searchers to formulate a good query that represents their information need as precisely as possible, in order to obtain satisfactory search results. Search engines' job is to collect and parse the text from a large number of documents in order to extract and weigh each term in a document. It is important for search engines to determine how relevant the set of terms in a document is in relation to the set of terms in the user query. In the context of this interaction between searchers and search engines, keywords act as a medium that bridges the gap between the searchers' minds and the information in the collection.

Recently, tags freely assigned by users to web resources have been gaining attention from researchers as good candidates for use as significant keywords for a document. Tags represent not only keywords but also personal ratings or other

Corresponding author:
Kang-Pyo Lee, School of Computer Science and Engineering, College of Engineering, Seoul National University, 599 Kwanak-ro, Gwanak-gu, Seoul 151-742, Korea.
Email: kplee@idb.snu.ac.kr

forms of comments or metadata [1]. Originally, when tagging services on the web began in the early 2000s, tags were merely a more flexible form of web resource categorization. Each user organized the web resources with his or her own vocabulary, or a set of tags, and when the user needed the resources later, he or she could easily retrieve them through the tags. This kind of personal vocabulary and resource set is called *personomy* [2] (this term is a personal version of *folksonomy*, which will be covered in the following paragraph).

As numerous users have participated in tagging, tags have begun to perform some intriguing social functions. Tags have enabled users to share any type of content (e.g. bookmarks, blogs, photos, and music) with others by saving the content and freely assigning several tags to it. Furthermore, users may also assign tags to other users' resources. This type of tagging is called *social tagging* or *collaborative tagging*. There is an interesting observation that, if users can see other users' tags, they are highly likely to be socially influenced by one another when they choose their own tags [3]. As the number of users increases, the formation of a stable tag distribution is observed, meaning that there might be a bottom-up user consensus around the categorization of information [4, 5]. From an ontological viewpoint, the emergent semantics resulting from socially created tags are of great value for creating and managing ontologies [6]. This bottom-up, socially created, and non-hierarchical labelling system [1] is called *folksonomy* [7].

Both personomy and folksonomy have been contributory factors in improving web search, especially in terms of indexing. In a traditional web search, index terms are automatically extracted from the text in a document by a search engine, and these terms are then used for matching with query terms. In contrast, tags are chosen directly by humans (we assume that tags are created only by humans, although sometimes they can be assigned by machine agents for bulk loading or spamming purposes), and can be used as a good substitute for or as a supplement to the index terms in a document. The tag-based web search is a new form of web search that exploits tag data for retrieving and ranking web resources and, is now being serviced by most well-known tagging systems, including Delicious[1] and Flickr.[2] Owing to the exploitation of interesting features of tagging, the tag-based web search has gained popularity among users. For example, Delicious provides a variety of bookmark search services, including keyword searches over personomy or folksonomy, browsing starting from a tag cloud, searches with date intervals, and querying assistance with related tags.

If we focus on indexing, an inverted index (or inverted file) is an index that maps each index term to a list of documents containing the index term, which is a fundamental data structure for the fast retrieval, evaluation, and ranking of documents in a collection. The inverted index has been widely accepted in the IR community as the most efficient data structure for supporting a range of web search tasks. Although the inverted index is also crucial to tag-based web search, it presents an obstacle to its use in the social-tagging environment. In a traditional web search, a document consists of terms that form a binary relationship from a document to terms; this relationship is inverted in a term-to-documents inverted index. By contrast, in a tag-based web search the user serves as an additional dimension, namely a user dimension. A resource (document) is annotated with tags (terms) by a user, creating a ternary relationship among resource, tag, and user that cannot be entirely contained in a normal inverted index. Most previous approaches that have incorporated tag data to improve web searches do not seem to treat each user in the ternary relationship individually. Instead, all user information is merely aggregated into a single numeric value, such as the number of corresponding users. In some situations, however, the ternary relationship should be preserved to generate a more meaningful value, rather than just a user count. In order to preserve the ternary relationship, a new type of inverted index needs to be designed, which should be different from the traditional term-to-documents inverted index and reflect the user aspect of tagging.

In this paper we propose a novel and extended index structure for social-tagging-based IR, namely a *social inverted index*, and present implementation-level solutions to a wide range of computations in tag-based web search by using the social inverted index. It is 'social' because it actively incorporates the social dimension in tagging.

The remainder of this paper is structured as follows. Section 2 presents related work on tag-based IR and inverted indexes. Section 3 describes the details of the social inverted index, including data structures, applications, and index construction and maintenance. Section 4 presents experimental results in terms of the cost and performance of the social inverted index. Finally, Section 5 summarizes this paper and discusses future work.

2. Related work

In this section, we present a brief overview of various approaches to tag-based IR and research issues related to the traditional inverted index.

2.1. Tag-based information retrieval

Since the launch of online social sharing services, such as Delicious (since 2003) for bookmarks and Flickr (since 2004) for photos, tagging has gained great popularity among Web 2.0 users. One stream of active studies on tagging is aimed at

improving existing web search. To date, various approaches have been proposed for enhancing web search by leveraging the unique characteristics of tagging. These approaches include adapting existing web search algorithms to the social-tagging environment [2, 8–12], analysing the applicability of tag data for web search use [13–16], social search and personalized search [17–23], and multimedia search [24–27].

Dmitriev et al. [8] integrate user tags as a particular form of user relevance feedback into an enterprise search engine, and assert that tags should be treated as metadata rather than as page content, and be used in a manner similar to the way anchor text is used. Hotho et al. [2] propose a formal model for folksonomies and a personalized and topic-specific ranking algorithm called *FolkRank*, an version of the *PageRank* algorithm adapted to folksonomies. FolkRank converts the folksonomy structure into an undirected tripartite graph structure so that the PageRank-like link structure-based ranking algorithms can be applied. Bao et al. [9] propose two ranking algorithms, *SocialSimRank* (*SSR*) for similarity ranking and *SocialPageRank* (*SPR*) for static ranking, which integrate social-tagging information with existing well-known metrics. Both algorithms are based on the interesting observation that useful information such as the semantic similarity between tags and the quality score of a resource can be extracted from the cyclic relations among user, tag, and resource. Yanbe et al. [10] calculate the popularity of a web page, called *SBRank*, from the number of users who bookmark the page. This tries to capture the popularity of resources, and shows the possibility of more complex searches that use contextual, temporal, or sentiment-related tagging information. Zhou et al. [11] propose a unified framework to combine social tag modelling with traditional language modelling-based methods for IR. Most recently, Carmel et al. [12] have proposed a framework for social bookmark weighting in order to improve social search effectiveness and derive solutions to recommendation tasks. This framework assigns a weight value to every individual bookmark according to its estimated quality.

With the introduction of tag-based search approaches, tag data's applicability to searching has been analysed. Heymann et al. [13] collect a large amount of social bookmarking data from Delicious, and conduct comprehensive experiments to analyse the positive and negative effects of social data on web search. They conclude that, despite the current lack of sufficient size and distribution of tags, social bookmarking can complement what cannot be supported from other sources. Bischoff et al. [14] analyse three different kinds of collection (music resources, pictures, and web pages). One interesting finding regarding the search in their study is that most tags can be used for searching, and that in most cases tagging behaviour is very similar to searching behaviour. Chi and Mytkowicz [15] evaluate the efficiency of the social-tagging system Delicious, based on information theory, and hold the negative opinion that social-tagging systems encouraging users to share their vocabularies do not seem to enhance navigation efficiency. Carman et al. [16] make a direct comparison between the distributions of tags assigned to URLs and the corresponding query terms used to access the URLs. They show that the vocabularies in tags and query terms are similar but not identical.

Recently, a wide variety of advanced search services, including social searches and personalized searches, has been presented. Social tagging provides these new search types with useful features by creating networks of entities, either explicitly or implicitly. A user's tagging activities may reflect the user's own preferences and interests. Yahia et al. [17] present the so-called *network-aware* search, which returns the top-ranked relevant answers to a query given to a user with a particular network that is formed through the user's tagging activities. Similarly, Schenkel et al. [18] propose an effective scoring model for user-centric searches in social networks, and develop an incremental top-*k* querying algorithm with both social expansion from user relations and semantic expansion from tag relations. Zanardi and Capra [19] propose *social ranking*, which exploits information extracted from a social network of users and tags in order to rank content, making it more meaningful to the querying user. Amitay et al. [20] propose the social search in the enterprise, which covers heterogeneous object types, such as document, person, and tag, based on a *unified approach*. Noll and Meinel [21] use social bookmarking and tagging information to re-rank web search results in which the searcher's tag profile is compared with the tags assigned to each result page. In [22], users and web pages are associated by a topic space that is modelled based on tags. The authors propose a *topic-adjusting* algorithm that ranks a web page by topic matching between the user's interests and the web page's topic. Carmel at al. [23] propose a personalized social search by considering each user's direct and indirect social relationships with documents, tags, and other persons or groups of persons. They then compare three types of social network: familiarity-based, similarity-based and overall network.

Tags can be leveraged as index terms for large collections of documents with non-textual multimedia data types, such as photos, audio, and videos [24]. Aurnhammer et al. [25] combine user tags with the visual features of images for improved data navigation and search. Levy and Sandler [26] represent audio tracks with both social tags and audio words (extracted audio features) for music information retrieval (MIR). Bischoff et al. [27] analyse three different tagging systems to verify the usefulness of tags for search, and propose a method for identifying emotions from music and picture resources using tagging information, thereby bridging the gap between tag terms and query terms.

It is important to note the inherent characteristics of tags regarding the vocabulary problem [28], which is said to be one of the most well-known problems in IR. The vocabulary problem arises when search engines fail to understand the

semantics of words (or tags). Golder and Huberman [5] explain three aspects of the vocabulary problem in tagging: synonymy, polysemy, and level variation. 'Synonym' refers to multiple tags sharing similar meanings (e.g. *car* and *automobile*), whereas 'polyseme' refers to a tag with multiple related senses (e.g. *head* as a part of the body and *head* as a person in charge of an organization). Level variation of tags exists because each user has his or her own semantic level in describing a resource, which is usually dependent on his or her degree of expertise (e.g. a blog post on Ajax programming may be annotated with the tag *javascript* by a programming expert, but may be annotated with just the tag *programming* by a user who is unfamiliar with programming languages). These three aspects constitute the main reasons for the vocabulary mismatch between query terms and tag terms. In fact, these problems have been also challenging for traditional IR, but they may become less serious in tag-based IR if many people participate in tagging and generate various terms for the same resource [13]. Some papers referenced earlier in this paper try to address the vocabulary problem in tagging. For example, in order to improve simple term-matching-based scoring and resource ranking, Bao et al. [9] built a global tag-to-tag similarity matrix based on the ternary relationship between tag, resource, and user. Aurnhammer et al. [25] tried to tackle the problems caused by the synonymy and homonymy of image tags by considering the relation between tags and the visual features of images.

2.2. Inverted index

As mentioned earlier, the inverted index has proven to be the most efficient data structure for retrieving and ranking documents in a large collection. Most textbooks on IR and text search discuss the details of the inverted index. Research issues related to the inverted index include its usage, indexing, index construction, maintenance and representation. Persin et al. [29] present an efficient query evaluation and ranking algorithm that filters out unimportant documents based on the frequency-sorted index. Deerwester et al. [30] propose latent semantic indexing (LSI), to reduce the dimensionality of the original term-document matrix by automated indexing based on concepts rather than words. Although LSI has proven to be a good solution to overcoming the weakness of inverted indexes – that is, the inability to handle the semantics of words – LSI also has a critical drawback, in that it does not provide indexing. Moffat and Zobel [31] propose a method for efficient index representations by compressing sequences of positive integers using various coding techniques. Most recently, Zobel and Moffat [32] have presented a comprehensive and exhaustive survey of the issues of inverted files for text search engines.

There have been few discussions about introducing the inverted index to the social-tagging environment. As previous approaches focused mainly on tag usages at an application level, they did not mention the inverted index at an implementation level. Consequently, the previous approaches were assumed to have an inverted index of their own type, and the details were not discussed in the papers. As far as we know, the only social-tagging system that revealed its whole system architecture is BibSonomy [33]. However, BibSonomy does not mention the existence of inverted structures in the paper, and seems to rely only on RDBMS for query processing. We have found some examples that consider the exploitation of inverted index in tag-based web search. Dmitriev et al. [8] build an inverted index that handles the terms separately from the contents, anchor texts, and tags of each page. However, this approach does not consider incorporating individual user-tagging information in the inverted index. Yahia et al. [17] manage one inverted list per (*tag, seeker*) pair to facilitate easy and fast access to the seeker-dependent scores that are assigned to each item. Even though this approach takes into account the user aspect of searching in the inverted index, it does not consider incorporating individual user information into the inverted index. Interestingly, approaches that incorporate the ternary relationship into the index structure are found in the field of RDF (resource description framework) triple data processing. An RDF triple (*s, p, o*) represents the ternary relationship among subject, predicate and object, which is very similar to the ternary relationship in tagging. A representative approach is Hexastore [34], which is an indexing scheme for managing a large amount of RDF triples by indexing them in six possible ways, one for each possible ordering of the three RDF elements *s, p* and *o*. Because Hexastore is designed to guarantee the scalability of RDF query processing, it cannot be exploited in search query processing.

2.3. Indexing in tag-based web search

Table 1 summarizes the differences between traditional web search engines (e.g. Google, Yahoo, and Bing) and tag-based web search engines (e.g. Delicious, Flickr, and BibSonomy) in terms of indexing. As an initial step, the document-crawling process differs, in that in traditional search engines it is done by web crawlers (or spiders) targeted at the whole web, whereas in tag-based search engines there is no need to crawl, because users themselves save and tag resources. Once the documents are collected in traditional search engines, several text operations are needed, such as tokenizing, case-folding, stopping, and stemming. In tag-based search engines, preprocessing work is also needed, but is usually

Table 1. Comparison of tag-based web search engines with traditional web search engines in terms of indexing

	Traditional web search engines	Tag-based web search engines
Example services	Google, Yahoo, Bing	Delicious, Flickr, BibSonomy
Document crawling	By web crawlers (targeted at the whole web)	No need to crawl (targeted at the resources in the system saved and tagged by users)
Document text preprocessing	Some text operations (such as tokenizing, case-folding, stopping, and stemming)	No or minimum preprocessing
Candidate index terms	Words in a document	Tags assigned to a resource
	Only from the words in the document	Not necessarily from the words in the resource
Index terms selection and weighting	Explicitly by a search engine	Implicitly by users
Index maintenance	Static and discrete	Dynamic and continuous
	More frequent document insertions than updates	Frequent document insertions and updates
Document collection size	Usually very large	Relatively small

performed to a minimal degree. One of the most marked differences between the two search engine types is that in tag-based search engines index terms are tags themselves, and thus are not necessarily derived from the words appearing on the document. This is because tags are freely assigned by humans, and are not automatically extracted from the text by machines, as is the case in traditional search engines. Likewise, index term selection and weighting are done explicitly by a traditional search engine, whereas they are implicitly done by users (e.g. the tag frequency) in tag-based search engines. Update issues for index maintenance on the tag-based web search are more complex than those on the traditional web search, because new resources and tags are continually added to the collection. The last distinct difference is the document collection size, because traditional web search engines cover the whole web, whereas tag-based web search engines cover only the tagged resources in their systems, producing a relatively small collection. This small coverage may be the major drawback of tag-based web search.

3. Social inverted index

This section introduces the social inverted index. Notations for our data structure and algorithm are defined in Section 3.1. The data structure of the social inverted index and its applications are presented in Sections 3.2 and 3.3 respectively. The processes of index construction and maintenance are described in Sections 3.4 and 3.5 respectively.

3.1. Notations

Most papers on tagging consider resource, tag, and user[3] to be the three basic elements of a formal tagging model. In this paper, we follow the common notations that have been presented in previous approaches. Let R be a set of resources, T be a set of tags, and U be a set of users. A resource $r \in R$ is annotated with a tag $t \in T$ by a user $u \in U$. A tag assignment is represented by a triplet of (1) a resource, (2) an assigned tag, and (3) a user who annotates the resource. This triplet is denoted by (r, t, u). A social-tagging system $STS = \{(r, t, u) \mid r \in R, t \in T, u \in U\}$ is a collection of triplets. $R_t = \{r_i \in R \mid (r_i, t, u) \in STS, u \in U\}$ is a set of resources annotated with tag t. $U_{t,r} = \{u_j \in U \mid (r, t, u_j) \in STS\}$is a set of users who annotate resource r with tag t.

The frequency values of any element are crucial for indexing and query evaluation. Let N be the number of (tagged) resources in the collection, and let n be the number of index tags (i.e. tags used as index terms) in the collection. Let $f_{t,r}$ be the frequency of tag t in resource r (i.e., the number of users who annotate resource r with tag t). Let $f_{t,R}$ be the resource frequency of tag t in the collection (i.e., the number of unique resources annotated with tag t). Let $F_{t,R}$ be the total frequency of tag t in the collection (i.e., the sum of all $f_{t,r}$). Let $f_{t,r,u}$ be the frequency of tag t in resource r by user u (in fact, $f_{t,r,u}$ is always 1 because each user annotates a resource with the same tag only once). Let $cof(t_1, t_2)$ be the co-occurrence frequency of tag t_1 and tag t_2.

3.2. Data structure

A sample social-tagging system, used as an example throughout this paper, contains resources, tags, users, and triplets as follows:

$$R = \{r_1, r_2, r_3\}$$

$$T = \{apple,\ iphone,\ mobile\}$$

$$U = \{Alice,\ Bob,\ Tom\}$$

$$STS = \{(r_1,\ apple,\ Alice),\ (r_1,\ apple,\ Bob),\ (r_1,\ mobile,\ Alice),\ (r_1,\ mobile,\ Tom),$$
$$(r_2,\ apple,\ Alice),\ (r_2,\ apple,\ Bob),\ (r_2,\ apple,\ Tom),\ (r_2,\ iphone,\ Alice),\ (r_2,\ iphone,\ Tom),$$
$$(r_3,\ apple,\ Bob),\ (r_3,\ apple,\ Tom),\ (r_3,\ iphone,\ Alice),\ (r_3,\ iphone,\ Tom),\ (r_3,\ mobile,\ Bob)\}$$

When considering a situation that requires a tag-to-resources inverted index while maintaining user information (i.e. to preserve the (*r, t, u*) ternary relationship), two possible combinations of two inverted lists may be available. One is cross-references (Figure 1) that interlink the resources and users, and the other is sublists (Figure 2) that maintain a separate user sublist for each resource. The latter is better for the following reasons:

- It has a simpler structure leading to easier implementation.
- Each entry for user sublists may have its own weight that varies according to the resource.
- The sublists, if needed, may be sorted by weight.

The only drawback is the space cost owing to the redundancy of users in the sublists. This drawback is acceptable, however, when considering the efficiency gains from maintaining separate user sublists. Furthermore, numerous existing methods for index compression can be applied.

If we take the sublists as a basic structure of the social inverted index, every entry has its own weight as a numeric value. As shown in Figure 3, there are three major components in the social inverted index: a tag index, (resource) posting lists, and (user) sublists. A tag index is a set of index tags that consists of three subcomponents: a tag name, the

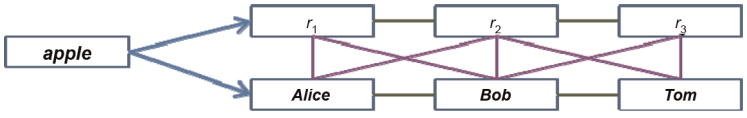

Figure 1. Cross-references.

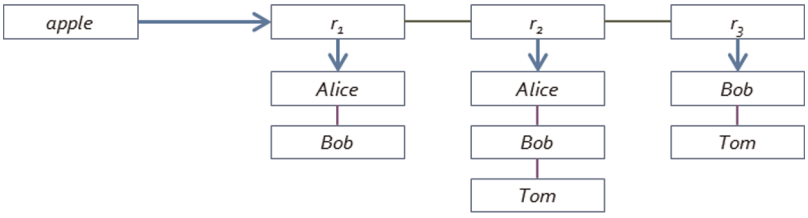

Figure 2. Sublists.

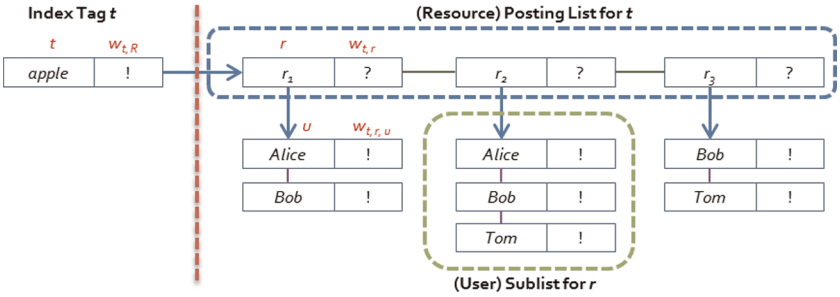

Figure 3. Part of a basic social inverted index.

weight of a tag ($w_{t,\ R}$), and a pointer to the head of the corresponding resource-posting list. Each resource-posting list represents resources that are annotated with a tag, and a node in the list has three subcomponents: a resource identifier, the weight of the resource annotated with the tag ($w_{t,\ r}$), and a pointer to the head of the corresponding user sublists. Each user sublist represents those users who annotate a resource with a tag, and a node in the sublist has two subcomponents: a user name, and the weight of the user annotating the resource with the tag ($w_{t,\ r,\ u}$). We propose this type of extended inverted index as a basic structure of the social inverted index, namely *basic social inverted index*.

Noticeably, the spaces for the $w_{t,\ r}$ values of the social inverted index may remain empty and be determined on the fly at the time of query evaluation, depending on the query types. In Figure 3, the question mark in each entry means that its value may not be determined, while the exclamation mark means that its value is determined, as will be covered shortly in this subsection.

The naïve frequency values defined in Section 3.1 can be used as weights. For example, $w_{t,\ R} = f_{t,\ R,}$ $w_{t,\ r} = f_{t,\ r,}$ and $w_{t,\ r,\ u} = f_{t,\ r,\ u}$ (= 1). In this case, all user weights equal simply 1. If all user weights are not equal – in other words, if the weight of each triplet is not treated as just 1 – any existing triplet or user-weighting scheme can be applied. To illustrate, we use an example from [12], in which every bookmark (i.e., tag assignment) triplet is weighted separately, as shown in Figure 4. The $w_{t,\ r}$ value may be calculated on the fly if there is a restriction in a query that, for example, only triplets whose weight is greater than a threshold value, say 1.000, should be counted.

As a second example, the network-aware personalized search is also supported, in which each user is assigned one (or more) label that indicates a group (or cluster) to which the user belongs. Suppose that *Alice* is assigned '*a*', *Bob* is assigned '*a*', and *Tom* is assigned '*b*' by a certain user-grouping scheme. As shown in Figure 5, every user entry for *Alice* is then set as '*a*', for *Bob* as '*a*', and for *Tom* as '*b*'. When a searcher who belongs to group '*a*' sends a query, only those users labelled with '*a*' may be counted on the fly.

As a last example, the timestamp information recorded at the time of tagging can be exploited. In other words, $w_{t,\ r,\ u} = ts_{t,\ r,\ u}$ (the *timestamp* when *r* is annotated with *t* by *u*), as shown in Figure 6. The temporal information for each triplet may be of great value in various situations: for example, when resource freshness is to be incorporated in a search [35], or to enhance personalized recommendations with the tag and time information for predicting users' preferences [36]. Again, the $w_{t,\ r}$ value may be calculated on the fly if there is a restriction in a query that, for example, only those users whose timestamp belongs to a certain period, say Q4 in year 2010, should be counted.

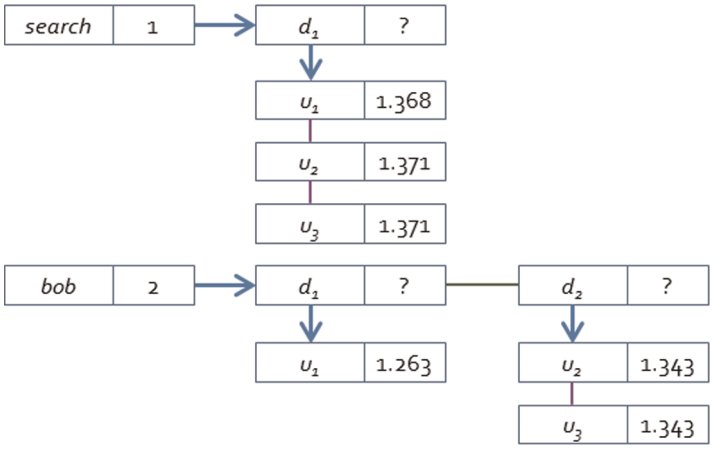

Figure 4. Part of a basic social inverted index with weighted triplets.

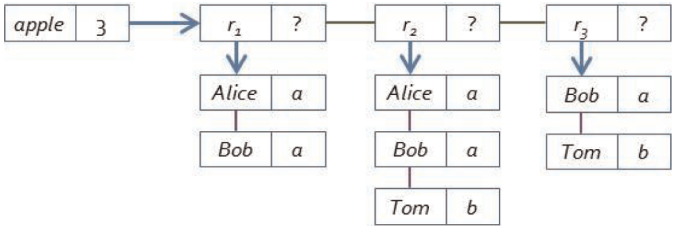

Figure 5. Part of a basic social inverted index with user labels.

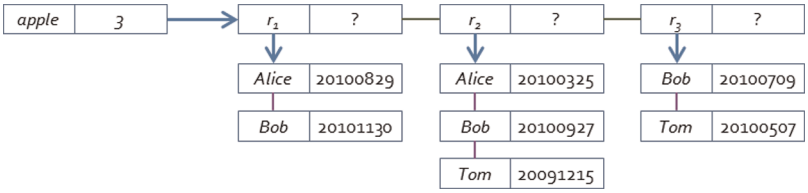

Figure 6. Part of a basic social inverted index with user timestamps.

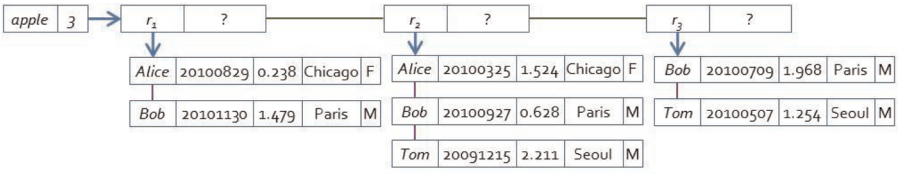

Figure 7. Part of a basic social inverted index with various user weights.

Note that all or some of the three examples above may be combined into one social inverted index. This is very useful when trying to contain various user weights from both the dynamic aspects (e.g. timestamp or triplet weight) and static aspects (e.g. group, location, or even age or gender) of a user profile. The weight values in Figure 7 are user name, timestamp, triplet weight, user location, and gender. This flexible structure facilitates the processing of any conditional queries, for example, 'retrieve the resources that were annotated (1) with tag *apple* (2) during the year 2010 (3) by male users (4) in Paris'.

When looking for users, not resources (in social search or, more specifically, people search) using tags, the positions of resources and users can be reversed. In other words, there are user-posting lists and resource sublists, as shown in Figure 8. The index structure around resource, tag, and user may be modified, depending on the goal of each search system.

In an inverted index, list ordering is crucial for effective query evaluation. Possible orderings include sorting by name (alphabetical order), integer ID (ascending order), or weight (descending order). The ordering is very useful when the lists are so long that their insignificant tails need to be cut out [29]. Figure 9 shows that resources are sorted in

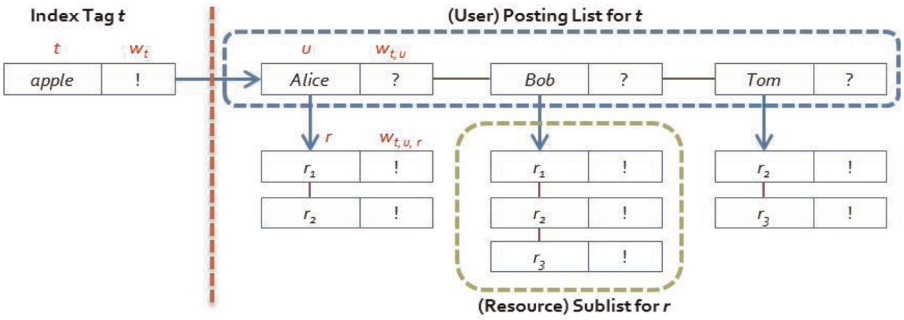

Figure 8. Part of an adaptive social inverted index.

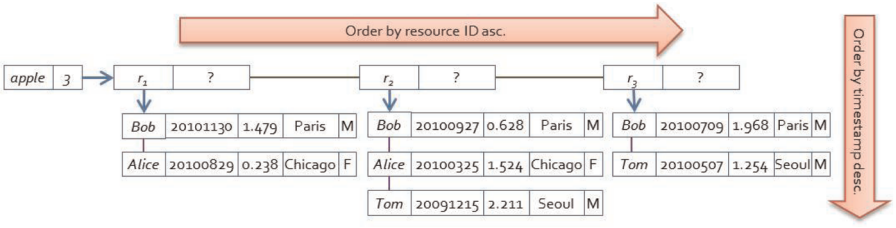

Figure 9. Part of a sorted social inverted index.

descending frequency order (or in some well-known ranking order such as FolkRank [2] or SocialPageRank [9]), and users are sorted in reverse chronological order with respect to timestamps.

In summary, the main advantages of the social inverted index are as follows:

1. They preserve the resource–tag–user ternary relationship of tagging.
2. They can support a wide variety of existing triplet or user-weighting schemes.
3. They are flexible enough to support conditional queries that use varying weight values in the lists, depending on the query.
4. They facilitate several critical computations, such as calculations of tag frequencies and tag co-occurrence frequencies (which will be covered in the following subsection).

3.3. Applications

Two measures reflecting the social semantics of tagging are tag frequency and tag co-occurrence frequency. The tag frequency represents the number of times a tag is used by people in a resource (or over the collection), and thus reflects the popularity of the tag or the degree of consensus among users about describing or categorizing the resource that is annotated with the tag. The tag frequency is one of the most widely used measures in many tag-based applications. The tag co-occurrence frequency represents the number of times two tags co-occur in a resource (or over the collection). The tag co-occurrence is important, because we can assume that those two tags that co-occur in the same resource are likely to be semantically related to each other. Tag co-occurrence therefore reflects the degree of semantic similarity between tags, and is said to be meaningful because it is human-generated semantics. Traditionally, the term 'co-occurrence' has already been used in IR, but these tag co-occurrences can be also applied to other applications, such as tag recommendations [37, 38] or query expansion [18, 39].

3.3.1. Tag frequency. As a social inverted index is being built, tag frequencies of any kind can be computed in a straightforward way. The frequency of t in r is the number of users who annotate r with t. The resource frequency of t in the collection is the number of resources that are annotated with t. The total frequency of t in the collection is the sum of the frequencies of t in all resources.

$$f_{t,\,r} = |U_{t,\,r}|$$

$$f_{t,\,R} = |R_t|$$

$$F_{t,\,R} = \sum_{r \in R_t} f_{t,\,r}$$

Usually these types of tag frequency value are stored at the corresponding entry of inverted index.

3.3.2. Tag co-occurrence frequency. Tag co-occurrences can be viewed at two levels: the macro level and the micro level. Two tags form a macro-level co-occurrence if they are assigned to the same resource, and they form a micro-level co-occurrence if they are assigned to the same resource by the same user. In other words, the criterion for a micro-level co-occurrence is stricter than that for a macro-level co-occurrence because the two tags must co-occur (1) within a resource and (2) by the same user. Accordingly, the frequencies of two levels are different. The macro-level tag co-occurrence frequency is the number of resources that are annotated with both tags (i.e., the number of common resources), whereas the micro-level tag co-occurrence frequency is the number of users who annotate the same resource with both tags (i.e., the number of common users in common resources). The social inverted index makes it easy to calculate the tag co-occurrence frequency at both levels.

$$macrocof(t_1, t_2) = |R_{t_1} \cap R_{t_2}|$$

$$microcof(t_1, t_2) = \sum_{r \in R_{t_1} \cap R_{t_2}} |U_{t_1,\,r} \cap U_{t_2,\,r}|$$

Using the two lists in Figure 10, the tag co-occurrence frequencies of *apple* and *iphone* at both levels are calculated as follows: $macrocof(apple,\ iphone) = |\{r_2, r_3\}| = 2$ and $microcof(apple,\ iphone) = |\{Alice,\ Tom\}| + |\{Tom\}| = 2 + 1 = 3$.

The two metrics above depend mainly on the intersection ($\cap$) operation between two lists. The time complexity of the intersection operation between two sorted lists using the Merge-Join (or Sort-Merge-Join) algorithm in relational databases is $O(m + n)$, where m and n are the lengths of the two lists respectively. The Merge-Join method is efficient because each item in the sorted order needs to be read only once.

Figure 10. Two lists for calculating tag co-occurrence frequencies of *apple* and *iphone*.

3.4. Index construction

During the tag-indexing process there is no need for tokenizing, because tags are already delimited by white spaces or commas. Additional preprocessing work, such as case-folding, stopping, stemming, and compound word decomposition, depends on the strategy (for this work, only the case-folding was done). We assume that the input to the index construction engine of a social inverted index is a set of resources that contains (r, t, u) triplets. We use the tag names and user names (mostly the user IDs used in the system) themselves as the identifiers of tags and users respectively, and we assign an ordinal number to each resource as a resource identifier. Consequently, we need to maintain an additional mapping table that maps resource IDs to their corresponding URLs or URIs. The primary concern is then how to construct the social inverted index efficiently from a set of resources with triplets.

The index construction process of a social inverted index is similar to that of a normal inverted index, except that user information in each triplet should be considered. Note that this process is done resource by resource, each of which contains a set of triplets. The process for the index construction of a basic social inverted index with naïve frequency weights is as follows. First, retrieve the set of input resources R, each with (r, t, u) triplets. For each resource $r^* \in R$, retrieve the (r^*, t, u) triplets and sort and group these triplets by t. For each t^* of the sorted (r^*, t^*, u) triplets, first check whether t^* is new to the system. If t^* is new, add t^* to the tag index, initialize a resource-posting list of t^*, and set $f_{t^*,\,R} = 0$. If t^* is not new, retrieve the corresponding resource-posting list of t^*. Next, check whether r^* is new to the resource-posting list of t^*. If r^* is new (in fact, it is always new, because r^* is new to the system), add r^* to the resource-posting list of t^*, increase $f_{t^*,\,R}$ by 1, initialize a user sublist of r^*, and set $f_{t^*,\,r^*} = 0$ and $f_{t^*,\,r^*,\,u} = 0$. Then, for each u^* in $U_{t,\,r}$, add u^* to the user sublist of r^* and increase $f_{t^*,\,r^*}$ and $f_{t^*,\,r^*,\,u^*}$ by 1 respectively. The algorithm is shown below.

Algorithm: *BuildBasicSocialInvertedIndex*
Input: A set of resources, each with (r, t, u) triplets in a social-tagging system
Output: A set *BasicSocialInvertedIndex*
< **Notation** >
Let R, T, and U be the set of resources, tags, and users respectively.
Let *TagIndex* be a set of index tags.
Let R_t be a set of resources annotated with t, and $U_{t,\,r}$ be a set of users who annotate r with t.
Let $f_{t,\,r}$ be a frequency of t on r, $f_{t,\,R}$ be a resource frequency of t, and $f_{t,\,r,\,u}$ be a frequency of t in r by u.
Let $ResourceList_t$ be a list of resources that are annotated with t, each of which is a triplet of < resource, frequency, address of user sublist > and $UserList_{t,\,r}$ be a list of users who annotate r with t, each of which is a pair of < user, frequency >.
< **Index Building** >
01 – Initialize $TagIndex \leftarrow \emptyset$.
02 – For each resource $r^* \in R$,
03 —— Read the (r^*, t, u) triplets.
04 —— Sort and group the (r^*, t, u) triplets by t.
05 —— For each t^* of (r^*, t^*, u) triplets,
06 ——— If $t^* \notin TagIndex$, then
07 ———— Set $TagIndex \leftarrow TagIndex \cup \{t^*\}$.
08 ———— Initialize $R_{t^*} \leftarrow \emptyset$, $f_{t^*,\,R} \leftarrow 0$, $ResourceList_{t^*} \leftarrow \emptyset$.
09 ——— Else, then read $ResourceList_{t^*}$.
11 ——— If $r^* \notin R_{t^*}$, then
12 ———— Set $R_{t^*} \leftarrow R_{t^*} \cup \{r^*\}$, $f_{t^*,\,R}++$.
13 ———— Initialize $U_{t^*,\,r^*} \leftarrow \emptyset$, $f_{t^*,\,r^*} \leftarrow 0$, $f_{t^*,\,r^*,\,u} \leftarrow 0$, $UserList_{t^*,\,r^*} \leftarrow \emptyset$.
14 ———— Execute $Add(ResourceList_{t^*}, < r^*, f_{t^*,\,r^*}, Address(UserList_{t^*,\,r^*}) >)$.
15 ——— For each u^* of (r^*, t^*, u^*) triplets,
16 ———— Set $U_{t^*,\,r^*} \leftarrow U_{t^*,\,r^*} \cup \{u^*\}$, $f_{t^*,\,r^*}++$, $f_{t^*,\,r^*,\,u^*}++$.
17 ———— Execute $Update(ResourceList_{t^*}, f_{t^*,\,r^*})$, $Add(UserList_{t^*,\,r^*}, < u^*, f_{t^*,\,r^*,\,u^*} >)$.
18 – Return $BasicSocialInvertedIndex = \{ < t, f_{t,\,R}, Address(ResourceList_t) > \mid t \in TagIndex \}$.

Note that the input to this algorithm is a set of resources, each with (*r, t, u*) triplets in the target social-tagging system, and that the output is a set *BasicSocialInvertedIndex* with a tag index and the corresponding resource-posting lists and user sublists. Note also that various user weights other than the naïve frequency weights can be applied in the same manner.

3.5. Index maintenance

The index maintenance of a social inverted index is more complex than that of a normal inverted index. Suppose that a social inverted index has already been constructed, and is working. A later crawling is then performed, and a new set of resources with triplets is provided to the maintenance engine of the social inverted index. Generally, there are three strategies for index maintenance: rebuild, intermittent merge, and incremental update [32]. The rebuild strategy is to rebuild an index from scratch. For the intermittent merge strategy, new resources are indexed in memory, and when the memory is full, the in-memory index is merged with the on-disk index. The incremental update strategy consists of updating the main index, term by term. A tag's resource-posting list and its user sublists are fetched from the disk, the new information is integrated into the lists, and the lists are written back to disk. Choosing a strategy among these three depends on the characteristics of the social-tagging systems.

With respect to the index update issue, we made four important observations of tagged resources:

1. The collection is highly dynamic. New resources are continually being added, and existing resources are continually being annotated with new sets of tags by new users.
2. Once users annotate a resource with tags, they rarely change their tags after that point.
3. The only update operation conducted on a resource is the additions of new sets of tags assigned by new users.
4. Temporal information of tagging can be very useful. Tags may represent the popularity or freshness of a resource, and searchers may expect to find timely content through tags.

The first and last observations indicate that tag-based search engines should reflect the dynamic nature of tagging. Furthermore, the second and third observations give us an intuition into how to update the social inverted index: when updating the social inverted index, we only need to consider the newly added triplets, regardless of whether the resource is new or existing in the social inverted index. This differs from a traditional web search, in which some parts of the content of a document are modified, not just incremented from the end of the previous version, as in tagged resources. Based on this intuition, we propose a merge-based index update method that proceeds with the following three steps:

1. Identify the new triplets that have been added since the last update using the timestamp information of tagging.
2. Build a social inverted index with these new triplets.
3. Merge this (small) social inverted index with the existing social inverted index.

In order to perform Step 1, we need to maintain an additional table that records the tagging history of each resource.

3.6. Considering the semantics of tags

One critical weakness of inverted indexes is their inability to handle tag meanings, as already mentioned in Section 2.1. Inverted indexes are used for the fast retrieval of documents containing the query terms (i.e., syntactic matching), not looking at documents containing terms whose meanings are the same as or similar to the query terms (i.e., semantic matching). For example, the social inverted index does not treat tags *nyc* and *newyorkcity* as the same, because their syntaxes (spellings) are different, although their semantics is exactly the same. In order to overcome the limitation of inverted indexes and understand the semantic relatedness between tags, we compute the semantic similarity between tags using the tag co-occurrence information. As explained in Section 3.3.2, tag co-occurrences are good indicators for calculating the semantic relatedness of two tags. Following the experimental results of Markines et al. [40], who compare a variety of similarity measures in a social-tagging environment, we compute the similarity between two tags based on *mutual information* as follows.

$$Sim(t_1, t_2) = \sum_{x_1 \in T_1} \sum_{x_2 \in T_2} p(x_1, x_2) \log \frac{p(x_1, x_2)}{p(x_1)p(x_2)}$$

where T is the row vector of tag t that contains the tag frequency in each resource, and

$$p(x) = \frac{f_{t,r_x}}{\sum_{r \in R_t} f_{t,r}}$$

$$p(x_1, x_2) = \frac{\sum_x \min(f_{t_1, x}, f_{t_2, x})}{\sum_{r \in R_{t_1} \cap R_{t_2}} f_{t_1, r} + \sum_{r \in R_{t_1} \cap R_{t_2}} f_{t_2, r}}$$

The similarity measure above is based on the fact that two tags that share more common resources get a higher score. All of the frequency values and resource intersection operations are easily calculated through the social inverted index, as illustrated in Section 3.3. This semantic similarity of two tags acquired from tag co-occurrences can be utilized in query expansion, or in more sophisticated calculations of the similarity between a query and resources.

4. Evaluation

In this section, we describe the implementation details for the social inverted index and the collected dataset, and then discuss the results of four experiments that were conducted for evaluation.

4.1. Implementation and data

General web search engines use web crawlers as a means for collecting web documents. To date, there is not an intelligent web crawler that can automatically identify tagged resources and extract tagging information without any prior knowledge of the resources. Generally, there are two approaches. The first approach is focused web crawling, based on the observation that tagged web pages have their own HTML structures according to the service they belong to (e.g. Delicious or Flickr). This approach is to make the crawlers parse the target resources and extract the appropriate tagging information using the common structure of each tagging service. The drawback to this approach is that the crawler should be adaptive to changes in the UI. The second approach uses publicly available datasets or open APIs. Some tagging services, such as CiteULike[4] and BibSonomy, provide their complete data dumps in a downloadable format for researchers. Data dumps provide the benefits of convenience in data acquisition and data completeness at the expense of using older data. Open APIs are a good option, except that they do not always provide all the data researchers want to use. We decided to utilize both approaches in this work. In focused web crawling, the crawler extracts a series of (*r, t, u*) triplets as it parses the content of a tagged resource. The available datasets used were already processed in the format of (*r, t, u*) triplets.

Tagged resources are found the same way as in general web crawling. The process starts from some seed pages, follows the outgoing links of the page, and repeats this process recursively (this method is known as a *snowball sampling* [41]). For example, in Delicious, the *Hotlist* page (which provides the list of the most popular bookmarks on Delicious) and the *Explore Everyone's Tags* page (which provides the tag cloud for Delicious) are some of the best seed pages when crawling for Delicious web pages.

For an inverted index, data structure and storage choices depend largely on the size of the term index and posting lists. In the ideal situation, both the term index and the posting lists fit into main memory. In large collections, however, the term index may be held in memory whereas the posting lists may be stored on a disk, because the lists of millions or more document IDs cannot fit in memory. In some very large collections even the term index cannot fit in memory, and is thus stored on a disk. Fortunately, our tag index can fit in memory. This is mainly because tags are a set of filtered vocabulary terms chosen directly by humans, and they are expected to be of higher quality than the vocabulary of general collections of web documents. As far as we know, the maximum number of unique Delicious tags that have been collected by researchers is 6,933,179 [42]. Therefore 10 million index tags (at most) with an average tag length of around 10 characters (i.e., 10 bytes) need approximately 100 MB for the tag index (not considering space for the weights and pointers). Our choice for the social inverted index is such that the tag index is implemented by a hash table residing in main memory, and the resource-posting lists and user sublists are implemented by linked lists stored on a disk. The hardware specification is 8-core 3.0 GHz CPU and 16 GB main memory. The implementation specification is as follows: Java SE 1.6 (JRE 6) for implementing the social inverted index; ActivePerl 5.12 for implementing the crawler; Oracle Database 11g (as a relational database); and MongoDB [43] 2.0.1 (as a NoSQL[5] database) for storing the triplets. The RDB and NoSQL DB are for comparison with the social inverted index. The Oracle databases are the most widely used relational databases, and the MongoDB is a document-oriented storage known as a simple-to-set-up and easy-to-operate NoSQL database. We implemented the MongoDB in the same standalone server as that running the social inverted index and the Oracle database, not considering the cloud-level scalability of MongoDB.

We chose three social-tagging systems for the datasets: Delicious (DL), BibSonomy (BS), and CiteULike (CU). We used the databases provided by CiteULike and BibSonomy because they are complete and unbiased. Unfortunately, Delicious does not provide public databases, so we collected the Delicious tag data from February 2011 to September 2011 using our focused web-crawling method. Table 2 summarizes the data statistics in the three datasets.

Table 2. Data statistics in Delicious, BibSonomy and CiteULike

STS	Method	No. of triplets	No. of tags	No. of resources	No. of users	Completeness
Delicious (DL)	Crawling	13,510,165	300,901	10,826	637,166	N
BibSonomy (BS)	Database	2,727,080	222,958	873,467	7,238	Y
CiteULike (CU)	Database	14,028,761	633,443	3,051,409	89,461	Y

4.2. Experiments

The goal of the experiments is to show that the social inverted index is better than (1) a normal inverted index and (2) no inverted index, despite the time and space costs required for index construction and maintenance. We used three datasets as mentioned, and for each dataset we compared two types of inverted index: N (normal) and B (basic with timestamp weights). In other words, Type N is a normal inverted index with no user sublists (i.e., weights aggregated by just counting the number of users who annotate a resource with a tag). Type B is the basic social inverted index with timestamp information as user weights. With these three datasets and two types of inverted index, we conducted four experiments: the first experiment for investigating the time and space required to build the inverted index (Type N vs Type B); the second for investigating the time required to evaluate queries (Type N with an RDB support vs Type N with a NoSQL DB support vs Type B); the third for investigating the time needed to calculate the tag co-occurrence frequencies (RDB vs NoSQL DB vs Type B); and the fourth for investigating the time needed to update the social inverted index (Type B by rebuild vs. by merge).

4.2.1. Index construction. The first experiment compared the time and space costs for building the inverted indexes of Types N and B. We assumed that a series of triplets in each resource was being sent to the construction engine as a stream. In other words, we did not take into account the process for extracting the triplets in a resource. We focused solely on how this set of triplets is converted into a social inverted index. For this reason, we created a single file for each social-tagging system that contains the whole set of triplets in each system. In this file, each triplet is of the form *resourceID* < *tab* > *userName* < *tab* > *tagName* < *tab* > *taggedDate* < *new line* > , and a set of triplets in each resource is delimited by < *RESOURCE* > and < /*RESOURCE* > tags. We also assumed that the volume of data handled for index construction could not be held in main memory, even though it safely fits into memory. This assumption is crucial for the extensibility of the social inverted index. We adopted the practical method known as merge-based inversion, by which resources with triplets are read and indexed in memory until a fixed capacity is reached. We set this capacity as k number of resources, and k was 1100, 88,000, and 310,000 in DL, BS, and CU respectively (approximately 10% of the number of resources in each system). When the memory was full (in this work when the number of processed resources becomes k), the partial index was flushed to disk as a single run and then deleted from memory. All runs are merged one by one to give a final index. As a last step of the index construction process, we sorted all resource-posting lists and user sublists such that, for Type N, the resource-posting lists were sorted by resource ID and the user sublists were sorted by user name, whereas for Type B the resource-posting lists were sorted by resource frequency and the user sublists were sorted by timestamp. This three-step (build–merge–sort) process was repeated 10 times, and the results were averaged.

Figure 11 illustrates the run-building times (the first step) of Types N and B in DL, BS, and CU. Note that Type B should be compared with Type N within each DL, BS, and CU (i.e. intra-comparisons). The comparisons across different social-tagging systems (i.e. inter-comparisons) are meaningless, because their sizes are different. The run-building times of the two types in all three social-tagging systems increase linearly as the number of runs (i.e., resources) increases. This indicates that the social inverted index can be extended to the vast amount of tag data on a web scale. Note that DL exhibited the largest gap between Type N and B among the three social-tagging systems. This means that DL is a broader folksonomy than BS and CU. In other words, in Delicious, a resource may be annotated by a larger number of users, which results in longer user sublists, than in BS and CU.

Table 3 compares the time costs (in seconds) for the three steps of index construction. As expected, Type B always requires more time than Type N during every step in all three social-tagging systems. For total time, we can see that even Type B in CU (the largest combination) shows a reasonable index construction time of 7383 seconds (approximately 123 minutes). The increase rates from Type N to Type B are 223% in DL, 36% in BS, and 37% in CU. Again, DL exhibited the maximum increase rate among the three, for the same reason as above. The increase from Type N to Type B will be enlarged as the social inverted index incorporates more user weights.

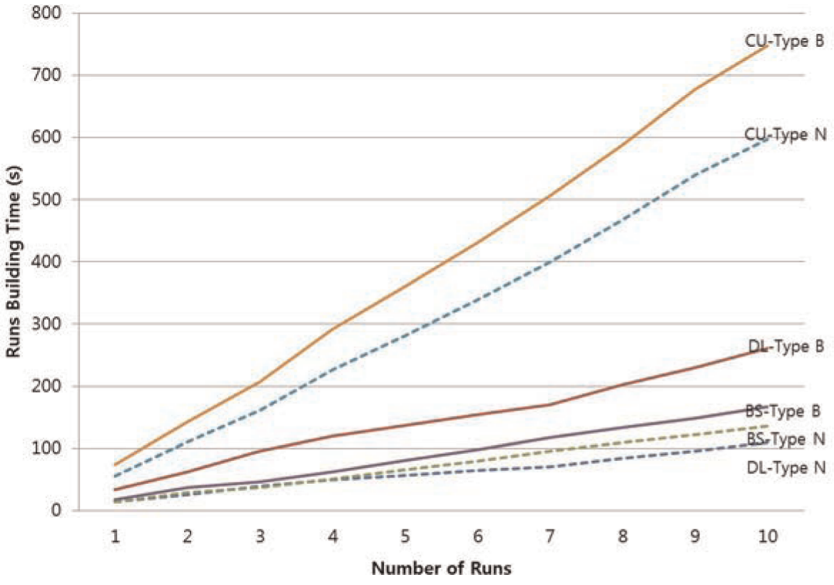

Figure 11. Run-building times of DL, BS, and CU (seconds).

Table 3. Comparison of time costs for the three steps of index construction (seconds)

		Run building	Run merging	Sorting	Total
DL	Type N	109	675	111	895
	Type B	261 (+139%)	2244 (+232%)	389 (+250%)	2894 (+223%)
BS	Type N	136	1014	180	1330
	Type B	166 (+22%)	1393 (+37%)	250 (+39%)	1809 (+36%)
CU	Type N	597	4061	731	5389
	Type B	747 (+25%)	5629 (+39%)	1007 (+38%)	7383 (+37%)

Tables 4 and 5 display the size (in megabytes) of a tag index and of resource-posting lists plus user sublists respectively. Note that, in Table 4, the sizes of the tag indexes are the same for both N and B, because the tag index residing in main memory is identical, whereas the corresponding resource-posting lists plus user sublists differ, depending on type. The sizes of the tag indexes were approximately 13 MB in DL, 10 MB in BS, and 30 MB in CU, which are very small. Table 5 shows that the sizes of resource-posting lists plus user sublists of Type N (i.e., no sublists) were approximately 11 MB in DL, 21 MB in BS, and 95 MB in CU. The space increase rates caused by adding user sublists were 2860% in

Table 4. Space costs for a tag index (megabytes)

	DL	BS	CU
Type N and B	13.01	10.00	30.17

Table 5. The space costs for resource-posting lists plus user sublists (megabytes)

	DL	BS	CU
Type N	10.51	20.80	95.21
Type B	311.05 (+2860%)	68.62 (+230%)	710.64 (+646%)

Table 6. List lengths in a social inverted index

		Average	S.D.	Minimum	Maximum
DL	Resource-posting lists	4.6	42.5	1	4,356
	User sublist	9.8	39.5	1	1,832
BS	Resource-posting lists	12.2	98.1	1	245,120
	User sublist	1.0	1.0	1	7
CU	Resource-posting lists	19.7	58.2	1	509,848
	User sublist	1.1	1.2	1	201

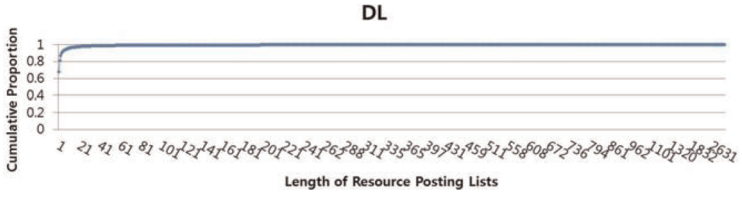

Figure 12. Distribution of lengths of resource-posting lists in DL.

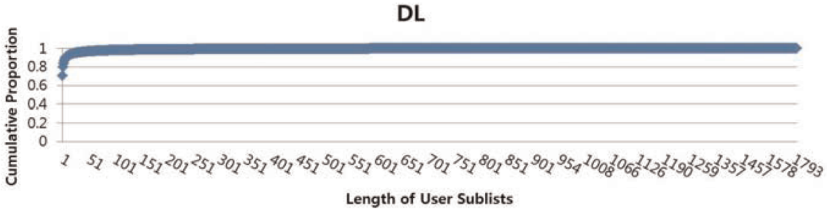

Figure 13. Distribution of lengths of user sublists in DL.

DL, 230% in BS, and 646% in CU. Again, DL exhibited the maximum increase rates among the three. Although the increase rate of 2860% may seem large, we believe that this is still acceptable, considering the explosive growth of disk and memory size.

Finally, we investigated the distribution of resource-posting lists and user sublists in a social inverted index. Table 6 displays the average lengths, the standard deviations of length, and the minimum and maximum lengths of resource-posting lists and user sublists. In DL, the average lengths of resource-posting lists and user sublists were 4.6 and 9.8 respectively, which means that, on average, a tag is used to annotate 4.6 resources by 9.8 users. The standard deviations were 42.5 and 39.5 respectively, and the maximum list lengths were 4356 and 1832 respectively. Figures 12 and 13 display, respectively, the distributions of the lengths for the resource-posting lists and user sublists of DL. They show that 95% of the lengths of the resource-posting lists and user sublists were less than 9 and 26, respectively, which also implies that, in Delicious, the distributions are skewed towards short lengths and have a long tail. Note that the distributions of BibSonomy and CiteULike are not shown, because they exhibit an almost uniform length (= 1) of user sublists. This implies that most resources in BibSonomy and CiteULike are annotated by one single user, even though BibSonomy and CiteULike are also known as broad folksonomies.

4.2.2. Query evaluation. The second experiment compared the time costs for evaluating queries of Types N and Type B. To emphasize the need to maintain the user sublists in the inverted index, we chose temporal range queries (queries targeted at a specific period of time). Type B can handle temporal queries, because the user sublist contains timestamp information as a user weight, whereas Type N cannot handle temporal queries, owing to its lack of user sublists. To assist Type N, we acquired the necessary information on the fly from the triplet table stored in an RDB and a NoSQL DB. We adopted the cosine measure from the vector space model to calculate the similarity score between a query and a resource. We used a set of randomly chosen tag pairs as queries such that the tag co-occurrence frequency of the two tags was higher than 100, which guarantees that the two tags are related to each other enough to be expected to be real query terms. We then

calculated the cosine similarity scores between 100 temporal queries (from the start date of 1 January 2010 to the end date of 31 December 2010) and the corresponding resources to get a ranked list of resources by using the normal inverted index with an RDB/NoSQL DB support and the basic social inverted index, and recorded the execution time. To get the necessary information from an RDB/NoSQL DB setting, we used the triplet table and the count function of DBMS. For example, the following SQL pseudo-query is used to calculate the frequency of a query term (i.e., tag t) in resource r.

$$\text{select count}(USER\_NAME)$$
$$\text{from } TRIPLET\_TABLE$$
$$\text{where } TAG\_NAME = QUERY\_TERM \text{ and } RESOURCE\_ID = TARGET\_RESOURCE\_ID$$
$$\text{and } TIMESTAMP >= START\_DATE \text{ and } TIMESTAMP <= END\_DATE$$

To guarantee the best performance of DBMS, we built every possible index on the columns of the *TRIPLET_TABLE*. We did the same work for the NoSQL DB setting.

Table 7 displays the total execution time for evaluating the queries (plus the comparison with Type B). It shows that Type B performs far better than Type N (from 16 times to 77 times) in all three social-tagging systems. The performance gap between the two types is natural, because in Type N with an RDB/NoSQL DB support, the necessary frequency values should be calculated from the DBMS to process the queries. We can conclude that, in order to evaluate conditional queries, such as temporal range queries, that should handle the varying weight values in the inverted index, it is very helpful to maintain a separate user sublist per resource. An additional finding is that NoSQL DB always outperformed RDB in this experiment. This shows that the newly emerging NoSQL DB may be superior to the traditional RDB, especially for the simple processing of very large datasets.

4.2.3. Tag co-occurrence frequency calculation. The third experiment compared the time costs for calculating tag co-occurrence frequencies. As discussed in Section 3.3.2, a social inverted index can be exploited in order to calculate tag co-occurrence frequencies. In comparison with the social inverted index of Type B, we used an RDB/NoSQL DB triplet table as a baseline. Considering that targeting all pairs of all tags is very costly and time-consuming (in CU, there are more than 200 billion pairs), and that not all tag pairs are interesting, we used a set of randomly chosen tags such that the resource frequency of each tag was greater than 1 and smaller than 30,000 (tags with a resource frequency greater than 30,000 are extremely rare and are usually useless for IR, such as *imported* or *no-tag*). In addition, the sample size was 1% of the original size (3009 tags from DL, 2229 tags from BS, and 6334 tags from CU). We then calculated the macro-level and micro-level co-occurrence frequencies for all pairs of the chosen tags using the basic social inverted index and an RDB/NoSQL DB triple table, and we recorded the execution time. To obtain necessary information from the RDB/NoSQL DB setting, we used a self-join for the triplet table and a count function of DBMS. For example, the two following SQL pseudo-queries were used to calculate the macro-level and micro-level co-occurrence frequencies of two tags respectively. Note that the two differences between the two queries are: (1) the counted column; and (2) the existence of the condition that $T1.USER\_ID = T2.USER\_ID$.

$$\text{select count(distinct T1.RESOURCE\_ID)}$$
$$\text{from } TRIPLET\_TABLET1, TRIPLET\_TABLET2$$
$$\text{where } T1.TAG\_NAME = TAG1 \text{ and } T2.TAG\_NAME = TAG2$$
$$\text{and } T1.RESOURCE\_ID = T2.RESOURCE\_ID;$$

$$\text{select count(T1.USER\_ID)}$$

Table 7. Comparison of execution times for query evaluations (seconds)

	Type B	Type N with RDB	Type N with NoSQL DB
DL	132	9001 (× 68)	2144 (× 16)
BS	25	1921 (× 77)	770 (× 31)
CU	54	2587 (× 48)	1033 (× 19)

Table 8. Comparison of execution times for tag co-occurrence frequency calculations (seconds)

	MacroCoF			MicroCoF		
	Type B	RDB	NoSQL DB	Type B	RDB	NoSQL DB
DL	96	13,733 (× 143)	23,411 (× 244)	172	14,025 (× 82)	28,016 (× 163)
BS	83	10,317 (× 124)	7,139 (× 86)	582	6,838 (× 11.7)	7,199 (× 12.4)
CU	848	57,796 (× 68)	86,005 (× 101)	14769	55,790 (× 3.8)	89,286 (× 6.0)

$$\text{from } TRIPLET\_TABLET1,\ TRIPLET\_TABLET2$$
$$\text{where } T1.TAG\_NAME = TAG1 \text{ and } T2.TAG\_NAME = TAG2$$
$$\text{and } T1.RESOURCE\_ID = T2.RESOURCE\_ID$$
$$\text{and } T1.USER\_ID = T2.USER\_ID;$$

To guarantee the best performance of DBMS, we again built every possible index on the columns of the *TRIPLET_TABLE*. The same was done for the NoSQL DB setting.

Table 8 displays the total execution time required for calculating all of the tag co-occurrence frequencies at the macro and micro levels (plus the comparison with Type B). It is evident that Type B performs better than RDB/NoSQL DB (from 3.8 times to 244 times) in all three social-tagging systems. This performance gap indicates that the social inverted index is useful in calculating tag co-occurrence frequencies on a large scale. An additional finding again is that for this experiment it was RDB, not NoSQL DB, that was usually superior. This result indicates that NoSQL DB may have weaker performance, especially in processing complex join queries, such as the queries used to calculate the tag co-occurrence frequencies.

4.2.4. Index maintenance. The last experiment compared the index-updating time utilized by rebuild and by our merge-based update method in order to choose an optimal strategy for maintaining a social inverted index. For the experiment, we created 12 subsets of an original triplet dataset of DL (each of which represents a monthly cumulative version) using the timestamp information of the triplets. We then subsequently recorded the whole index-updating time of Type B required for naïve rebuild from scratch, and for merge with the previous version, respectively. For example, a social inverted index for the Apr-11 version can be built either by rebuilding (i.e., by our index construction algorithm presented earlier) or by merging the new information with the previous Mar-11 version. Table 9 displays the statistics for the monthly cumulative versions of DL datasets, and shows a gradual increase in the volume of the datasets.

In general, for highly dynamic collections, rebuild may be the most plausible option, because the cost for updating a large number of inverted lists may exceed the cost for rebuilding those lists from scratch. However, as shown in Figure 14, the index-updating time for rebuild is much greater (by around three times) than the time required for merge. This indicates that our merge-based update method is more efficient than the naïve rebuild method, confirming the intuition learned from the second and third observations listed in Section 3.5. An important point to note is that choosing an update method depends mainly on the policy for the update interval. If the update interval is short (in other words, if the search

Table 9. Statistics for monthly cumulative versions of DL datasets

Version	No. of triplets	No. of tags	No. of resources	No. of users
May 2010	7,923,364	202,764	6,503	508,573
Jun 2010	8,297,092	209,840	6,671	517,776
Jul 2010	8,701,895	217,037	6,839	527,886
Aug 2010	9,115,754	224,143	7,009	537,870
Sep 2010	9,553,282	232,107	7,200	548,693
Oct 2010	9,999,912	240,442	7,402	559,802
Nov 2010	10,483,851	249,265	7,651	570,902
Dec 2010	10,874,076	256,062	7,858	579,081
Jan 2011	11,321,369	263,846	8,080	588,459
Feb 2011	11,773,890	271,840	8,351	598,726
Mar 2011	12,304,037	280,731	8,720	609,580
Apr 2011	12,711,291	287,716	9,212	617,405

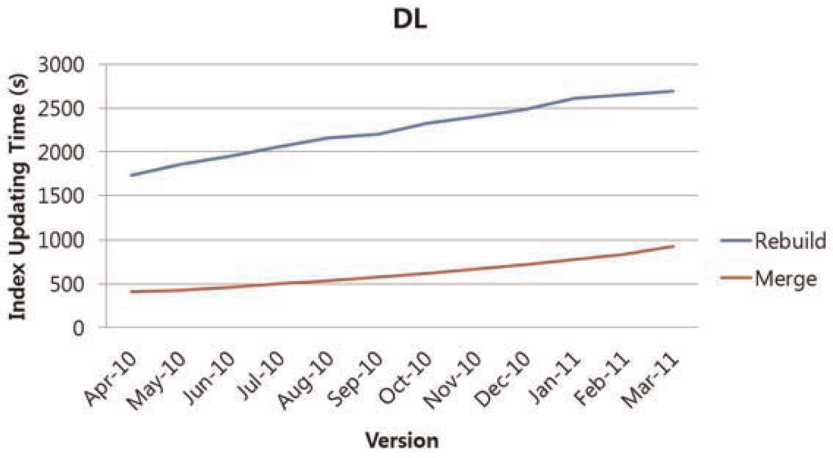

Figure 14. Index-updating times of DL for rebuild and merge.

engine wants to provide the freshest information possible), our merge-based method is superior to the rebuild method, because it is capable of handling the new fresh content at minimum cost.

5. Conclusion

In this paper, we present a novel extended inverted index, or social inverted index, for social-tagging-based IR. Our social inverted index fully supports the social dimension of social tagging by adding a user sublist to each resource in resource-posting lists. Each user in the user sublist has various weights for matching with the user query. This differs significantly from the traditional inverted index, in that it regards each user as a unique person, and does not simply count the number of users in a resource. It highlights the value of a user who participated in tagging. This extended structure facilitates the use of dynamic resource weights, which are expected to be more meaningful than weights based on simple user frequency. It also allows flexible response to various types of conditional query, an ability that is increasingly required in recent tag-based IR.

Some issues remain for future work. First, more efficient and compact index representations are needed, which are not covered in this paper. These can be achieved by compressing the resource-posting lists and user sublists, applying the existing or modified index compression methods. Although we have shown that the time and space costs for the index construction and maintenance are acceptable, the performances will probably be enhanced with more compact representations of lists.

Second, a new index structure that integrates more than one social-tagging system is also needed. Currently, the social inverted index can handle only one social-tagging system. In order to extend coverage of tag-based search engines, however, multiple social-tagging systems need to be integrated, and the problems caused by their heterogeneity need to be addressed. If more than one social-tagging system is merged in a single social inverted index, careful consideration should be given to weight normalization for query processing. For example, resources in Delicious that attract many users and tags may look more prominent than resources in BibSonomy and CiteULike, if we apply simple frequency-based weighting schemes.

Third, as mentioned above, focused web crawling or open APIs are possible methods to collect tag data. However, we believe that a more intelligent crawling algorithm that can automatically detect, collect, and index the tag data spread on the social web should be devised. If this is possible, limitations owing to the small coverage problem of tag-based web search will be overcome sooner than expected, and a true social-tagging-based search system that integrates several social-tagging systems will be serviced. The int.ere.st website [44] and DataPortability[6] Project are good examples of attempts to integrate heterogeneous social systems.

Finally, we plan to develop a more extended index, similar to the adaptive social inverted index, that will facilitate the search for all types of entity.

Notes

1 Delicious. http://www.delicious.com/
2 Flickr. http://www.flickr.com/

3 Resource may be replaced by document, (web) page, object, or item. Tag may be replaced by annotation. User may be replaced by tagger or person.
4 CiteULike. http://www.citeulike.org/
5 NoSQL. http://nosql-database.org/
6 The DataPortability Project. http://dataportability.org/

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF), grant-funded by the Korea government (MEST) (No. 20110017480).

References

[1] Vossen G and Hagemann S. *Unleashing Web 2.0: from concepts to creativity*. Burlington, MA: Morgan Kaufmann Publishers, 2007.
[2] Hotho A, Jäschke R, Schmitz C, Stumme G. Information retrieval in folksonomies: search and ranking. In: *Proceedings of the 3rd European Semantic Web Conference: Research and Applications*. Budna, Montenegro, 2006, pp. 411–426.
[3] Fu W-T, Kannampallil T, Kang R, He J. Semantic imitation in social tagging. *ACM Transactions on Computer-Human Interactions* 2010; 17(3): 1–37.
[4] Halpin H, Robu V, Shepherd H. The complex dynamics of collaborative tagging. In: *Proceedings of the 16th International Conference on World Wide Web*. Banff, Alberta, Canada, 8–12 May 2007, pp. 211–220.
[5] Golder S, Huberman BA. Usage patterns of collaborative tagging systems. *Journal of Information Science* 2006; 32(2): 198–208.
[6] Mika P. Ontologies are us: a unified model of social networks and semantics. *Journal of Web Semantics* 2007; 5(1): 5–15.
[7] Wal TV. Folksonomy coinage and definition' http://www.vanderwal.net/folksonomy.html (2007, accessed 25 April 2011).
[8] Dmitriev PA, Eiron N, Fontoura M, Shekita E. Using annotations in enterprise search. In: *Proceedings of the 15th International World Wide Web Conference*. Edinburgh, UK, 2006, pp. 811–817.
[9] Bao S, Xue G, Wu X, Yu Y, Fei B, Su Z. Optimizing web search using social annotations. In: *Proceedings of the 16th International World Wide Web Conference*. Banff, Alberta, Canada, 8–12 May 2007, pp. 501–510.
[10] Yanbe Y, Jatowt A, Nakamura S, Tanaka K. Can social bookmarking enhance search in the web? In: *Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries*. Vancouver, Canada, 17–22 June 2007, pp. 107–116.
[11] Zhou D, Bian J, Zheng S, Zha H, Giles CL. Exploring social annotations for information retrieval. In: *Proceedings of the 17th International World Wide Web Conference*. Beijing, China, 21–25 April 2008, pp. 715–724.
[12] Carmel D, Roitman H, Yom-Tov E. Social bookmark weighting for search and recommendation. *The VLDB Journal* 2010; 19(6): 761–775.
[13] Heymann P, Koutrika G, Garcia-Molina H. Can social bookmarking improve web search? In: *Proceedings of the 1st ACM International Conference on Web Search and Data Mining*. Stanford, CA, USA, 11–12 February 2008, pp. 195–206.
[14] Bischoff K, Firan CS, Nejdl W, Paiu R. Can all tags be used for search? In: *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. New York, NY, USA, 2008, pp. 203–212.
[15] Chi EH, Mytkowicz T. Understanding the efficiency of social tagging systems using information theory. In: *Proceedings of the 19th ACM Conference on Hypertext and Hypermedia*. Pittsburgh, PA, USA, June 2008, pp. 81–88.
[16] Carman MJ, Baillie M, Gwadera R, Crestani F. A statistical comparison of tag and query logs. In: *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Boston, MA, USA, 19–23 July 2009, pp. 123–130.
[17] Yahia SA, Benedikt M, Lakshmanan LVS, Stoyanovich J. Efficient network aware search in collaborative tagging sites. *Proceedings of the VLDB Endowment* 2008; 1(1): 710–721.
[18] Schenkel R, Crecelius T, Kacimi M, et al. Efficient top-*k* querying over social-tagging networks. In: *Proceedings of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA, 2008, pp. 523–530.
[19] Zanardi V, Capra L. Social ranking: uncovering relevant content using tag-based recommender systems. In: *Proceedings of the 2nd ACM Conference on Recommender Systems*. Lausanne, Switzerland, 2008, pp. 51–58.
[20] Amitay E, Carmel D, Har'El N, et al. Social search and discovery using a unified approach. In: *Proceedings of the 20th ACM Conference on Hypertext and Hypermedia*. Torino, Italy, 2009, pp. 199–208.
[21] Noll MG, Meinel C. Web search personalization via social bookmarking and tagging. In: *Proceedings of the 6th International Conference on The Semantic Web*, LNCS 4825. 2007, pp. 367–380.
[22] Xu S, Bao S, Fei B, Su Z, Yu Y. Exploring folksonomy for personalized search. In: *Proceedings of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval*. Singapore, 20–24 July 2008, pp. 155–162.
[23] Carmel D, Zwerdling N, Guy I, et al. Personalized social search based on the user's social network. In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. Hong Kong, China, 2–6 November 2009, pp. 1227–1236.

[24] Lew MS, Sebe N, Djeraba C, Jain R. Content-based multimedia information retrieval: state of the art and challenges. *ACM Transactions on Multimedia Computing, Communications and Applications* 2006; 2(1): 1–19.
[25] Aurnhammer M, Hanappe P, Steels L. Integrating collaborative tagging and emergent semantics for image retrieval. In: *Proceedings of the Collaborative Web Tagging Workshop (WWW '06)*, 2006.
[26] Levy M, Sandler M. Music information retrieval using social tags and audio. *IEEE Transactions on Multimedia* 2009; 11(3): 383–395.
[27] Bischoff K, Firan CS, Nejdl W, Paiu R. Bridging the gap between tagging and querying vocabularies: analyses and applications for enhancing multimedia IR. *Journal of Web Semantics* 2010; 8(2–3): 97–109.
[28] Furnas GW, Landauer TK, Gomez LM, Dumais ST. The vocabulary problem in human–system communication. *Communications of the ACM* 1987; 30(11): 964–971.
[29] Persin M, Zobel J, Sacks-Davis R. Filtered document retrieval with frequency-sorted indexes. *Journal of the American Society for Information Science* 1996; 47(10): 749–764.
[30] Deerwester S, Dumais ST, Furnas GW, Landauer TK, Harshman R. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 1990; 41(6): 391–407.
[31] Moffat A, Zobel J. Self-indexing inverted files for fast text retrieval. *ACM Transactions on Information Systems* 1996; 14(4): 349–379.
[32] Zobel J, Moffat A. Inverted files for text search engines. *ACM Computing Surveys* 2006; 38(2): 1–56.
[33] Benz D, Hotho A, Jäschke R, et al. The social bookmark and publication management system BibSonomy. *The VLDB Journal* 2010; 19(6): 849–875.
[34] Weiss C, Karras P, Bernstein A. Hexastore: sextuple indexing for semantic web data management. *Proceedings of the VLDB Endowment* 2008; 1(1): 1008–1019.
[35] Huo W, Tsotras VJ. Temporal top-*k* search in social tagging sites using multiple social networks. In: *Proceedings of Database Systems for Advanced Applications (DASFAA 2010)*. Tsukuba, Japan, 1–4 April 2010, LNCS 5981, pp,.498–504.
[36] Zheng N, Li Q. A recommender system based on tag and time information for social tagging systems. *Expert Systems with Applications* 2011; 38(4): 4575–4587.
[37] Sigurbjörnsson B, Zwol Rv. Flickr tag recommendation based on collective knowledge. In: *Proceedings of the 17th International World Wide Web Conference*. Beijing, China, 21–25 April 2008, pp. 327–336.
[38] Wartena C, Brussee R, Wibbels M. Using tag co-occurrence for recommendation. In: *Proceedings of the 9th International Conference on Intelligent Systems Design and Applications*. Pisa, Italy, November 2009, pp. 273–278.
[39] Kim HH. Toward video semantic search based on a structured folksonomy. *Journal of the American Society for Information Science and Technology* 2011; 62(3): 478–492.
[40] Markines B, Cattuto C, Menczer F, Benz D, Hotho A, Stumme G. Evaluating similarity measures for emergent semantics of social tagging. In: *Proceedings of the 18th International World Wide Web Conference*. Madrid, Spain, 20–24 April 2009, pp. 641–650.
[41] Ahn Y-Y, Han S, Kwak H, Moon S, Jeong H. Analysis of topological characteristics of huge online social networking services. *Proceedings of the 16th International Conference on World Wide Web*. Banff, Alberta, Canada, 8–12 May 2007, pp. 835–844.
[42] Wetzker R, Zimmermann C, Bauckhage C. Analyzing social bookmarking systems: a del.icio.us cookbook. In: *Proceedings of the ECAI 2008 Workshop on Mining Social Data*. Patras, Greece, 21–25 July 2008, pp. 26–30.
[43] Plugge E, Hawkins T,d Membrey P. *The definitive guide to MongoDB: The NoSQL database for cloud and desktop computing*. New York, NY, USA: Apress, 2010.
[44] Kim H, Breslin JG, Yang S, Song S, Kim H. int.ere.st: building a tag sharing service with the SCOT ontology. In: *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*. Chicago, IL, USA, July 2008.