Intelligent Information Systems 9999
ISBN 666-666-666, pages 1–14

Personalized Search in Folksonomies with Ontological User Profiles

Noriko Tomuro and Andriy Shepitsen

College of Computing and Digital Media
DePaul University
Chicago, Illinois, USA

Abstract

This paper presents a new method for search personalization in Folksonomies which utilizes ontological user profiles. Notably, our method builds a folksonomy tag ontology in which the tags are disambiguated – each node corresponds to a single concept, and ambiguous tags are mapped to several nodes in the ontology. Our method first creates a set of unambiguous tag clusters by using an algorithm called *DSCBC*. Then we use a modified hierarchical agglomerative clustering algorithm to construct a disambiguated tag ontology. Next we match the tag profile of a target user against the ontology and derive an ontological profile of the user. Finally we feed the user's ontological profile into the modified FolkRank algorithm and retrieve web resources which are ranked and personalized to the user. We ran our system on data from two social tagging systems. The results showed our method achieved significant improvements over other approaches.

Keywords: Search Personalization, Folksonomies, Clustering, Natural Language Processing, FolkRank, Ontological User Profile

1 Introduction

Collaborative tagging systems, usually referred to as *folksonomies*, is a popular new trend where Internet users apply descriptive tags to online resources. Central to collaborative tagging are annotations - users assign personalized tags to Web resources. The last few years has shown a rapid growth of various folksonomies. Two popular folksonomies are Delicious [1] and Last.FM[2]. In Delicious, users bookmark Universal Resource Locators (URLs). Last.FM allows users to upload, share and tag media files. Folksonomies are found in other applications such as on-line digital pictures, blogs and journal publications. The content of some of these resources can be determined by computers and effectively processed by standard search techniques, but some resources are particularly difficult to automatically categorize. For example, it can be particularly difficult for standard search engines

[1] delicious.com/
[2] www.last.fm

to describe or organize video and music resources. Folksonomies applications, however, rely on the users' tags to determine the content of a resource. Thus, users find it easier to locate resources of their interest in social tagging systems than posting a query in standard search engines.

One of the main services provided by social tagging systems is searching. Searching occurs when the user enters a tag as a query and a list of related resources are returned to the user which are ranked by relevance. Even though collaborative tagging applications have many benefits, they also present unique challenges for Information Retrieval *(IR)*. The core of many search engines is the ranking algorithm. Most of the ranking algorithms currently used are those developed for IR and Library Sciences *(LS)*. However, these algorithms are not easily adaptable to Folksonomies. Most folksonomy applications permit uncontrolled tagging - users are free to use any tag they wish to describe a resource. This is often done to reduce the entry cost of using the application and make collaborative tagging more user friendly. As a result, folksonomies contain a wide variety of tags: from the factual (e.g., "Chess_engine") to the subjective (e.g., "cool"), and from the semantically obvious (e.g., "Poland") to the utterly opaque (e.g., "PS023"). Moreover, *tag redundancy* in which several tags have the same meaning or *tag ambiguity* (polysemy) in which a single tag has many different meanings can confound effective IR. The task of combating noise is made even more difficult by capitalization, punctuation, misspelling and other discrepancies, which are typical in most of the contemporary Folksonomies.

Data mining techniques such as clustering can provide a means to overcome these problems. Through hierarchical agglomerative clustering, we can build tag ontological structures created by a "collective mind" of many users contributing to the Folksonomy. After matching the obtained ontology against the user profile the system may determine the user interest presented by ontological profile, which can then be used for search personalization. In turn, personalization is used to overcome the problems of tag ambiguity and redundancy. In this work, we develop a modified hierarchical agglomerative clustering algorithm to automatically building tag ontologies for social tagging systems. The aim is to construct an ontology in which the tags (as nodes in the ontology) are disambiguated, then to use the ontology to infer a more accurate profile of the users interest and retrieve/rank the relevant resources that are better personalized to the user.

In our previous works (Shepitsen *et al.*, 2008), (Gemmell *et al.*, 2008), we used a standard hierarchical agglomerative clustering algorithm to build a tag hierarchy. Although we were able to show an improved performance over other algorithms for personalizing search and retrieval tasks, for user profiles we only used flat, unstructured vector of tags.

The work presented here is an extension of our earlier work with a focus on constructing a disambiguated ontology and improving the ranking algorithm. To construct a disambiguated ontology, we employ a clustering algorithm called *Domain Similarity Clustering By Committee (DSCBC)* (Tomuro *et al.*, 2007) to first derive a set of tag *committees*. A tag committee is a cluster of tags that have similar meanings. *DSCBC* was originally developed in Natural Language Processing (NLP) to automatically derive unambiguous clusters of word meanings from ambiguous words. Thus, a tag committee in this work is a group of tags where

members share the same or very similar concept in one of their meanings. Using the tag committees, we organize tags into an ontology by using a modified hierarchical agglomerative clustering algorithm. The core of modification is that we add the ambiguous tags to multiple places in the ontology according to the number of senses they have.

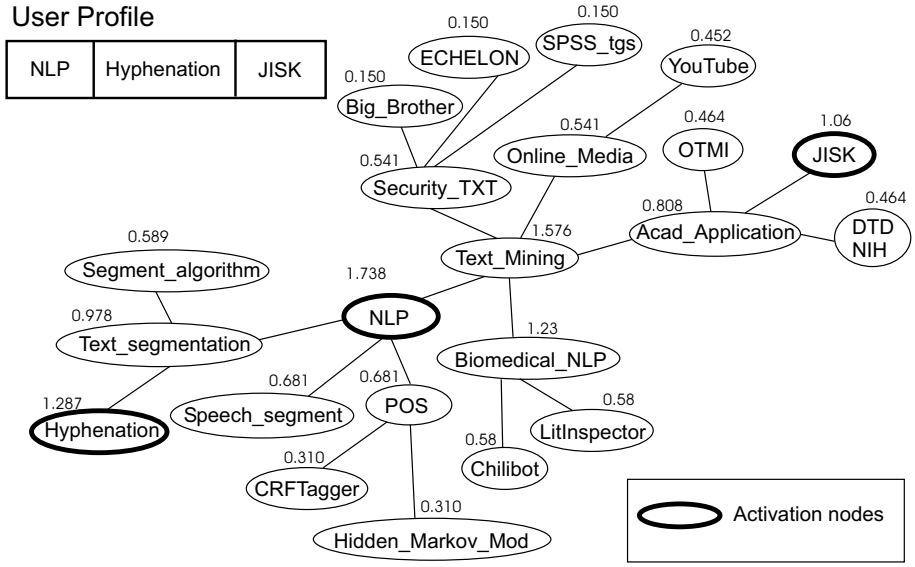

FIGURE 1: Ontological user profile in Folksonomies

For example, a tag "NLP" is ambiguous – it could mean "Natural Language Processing" or "Neuro-Linguistic Programming". Therefore the tag "NLP" will be added to two places in the ontology with different resource features. In addition, we did not use the step threshold, typical for standard hierarchical agglomerative clustering, but join the most similar tag/cluster pair in every iteration. Thus, we get an ontological tree instead of "lattice" which is more appropriate for search personalization. Then for a target user, we compute a preference vector based on his/her tag profile, matching it against the whole ontology and spreading interest weights up and down the tree. Values in a preference vector indicate the various degrees of the users interest distributed in the ontology. An example of an user ontological profile is depicted in figure 1. The depicted ontological structure has been created by hierarchical agglomerative clustering applied for folksonomy dataset. For instance, if tags "Text_Mining" and "NLP" were connected by an edge it means that those tags were frequently used for the same resources by many Folksonomy users. Suppose we got a test user who has three tags in his profile "NLP", "Hyphenation" and "JISK". We start matching the user profile with formulated ontology sequentially for every tag. Initial node interest scores are equal to zero. We start from "NLP" tag by assigning interest score equal to one. "NLP" tag has three children - "Text_segmentation", "Speech_segment" and "POS", so the interest score will be distributed equally among those children. In

addition, in this example we used a *damping coefficient* equal to 0.5, so each child node of "NLP" tag will get 0.17 interest score (0.33*0.5). Each child node will transfer obtained weight further in the structure to its children and so forth. The same approach was used during transferring weight up in the ontological tree. In this example, tag "Text_Mining" will get 0.5 from "NLP" and transfer it further to the network. So the process is similar to the transferring activation by nerve cells in the biological organism. The final weights near the nodes characterize the degree of the user interest after sequentially activating interest score inside this restricted part of ontology with the user who has in his profile tags *Hyphenation*, *NLP* and *JISK*. Notice, that some activation point tags have interest score greater than one, because the activation happens sequentially one tag after another and tags "strengthen" each other. For instance, we may conclude that the tag "Text_segmentation" got a comparatively high weight because it is between the two tags in the ontological structure that appeared in the user profile (*NLP* and *Hyphenation*). Finally we feed the preference vector to the modified FolkRank algorithm (Hotho *et al.*, 2006a) to retrieve and rank the relevant web resources which reflect the user-specific preferences. We tested our system on the data from two social tagging systems and compare the results with other algorithms, in particular the original FolkRank algorithm (which uses binary preference vectors), ranking using a vector space model and also ranking based on committee-based preference vector without building the ontological profiles. Our results showed marked improvements over the above mentioned algorithms.

2 Related Work

The closest re-ranking/recommendation algorithm to the one presented in this work appeared in (Middleton *et al.*, 2004) presenting the algorithm for using ontological profiles and generating recommendations. The authors used a scientific publication ontology for expressing the broadness of the user scientific interest. Afterwards, the collaborative filtering recommender was used to generate recommendations. The recommendation generated with ontological user profiles was more relevant than the ones obtained with "flat" user vectors. However, the authors used an ontology of scientific papers manually created by experts – which simply is not the case with Folksonomies. In addition, the authors used an explicit user feedback, which is also not viable with existing social tagging systems.

One of the first attempts for automatically creating an ontological structure from Web documents appeared in (Sanderson and Croft, 1999). The main idea for determining the child-parent relation is subsumption: a term is a child of another term if the set of documents in which the (child) term occurred is a subset of the set of documents in which the (parent) term occurred. The authors tested their algorithm by posting a random query term and examined the first 500 returned documents from a large corpora. They reported that in 70% of the cases the derived hierarchy coincided with the judgment by the experts.

In our previous work (Shepitsen *et al.*, 2008), (Gemmell *et al.*, 2008), we used a hierarchical agglomerative clustering algorithm to build a tag hierarchy for Folksonomies. To build a hierarchy bottom-up, we gradually decreased a similarity

threshold to join tags/clusters whose similarity were above that the threshold. Then we used the obtained ontology to personalize search and retrieval for a target user. However, in our previous algorithms we did not present users with ontological profiles for expressing their interest, but used ontological tree as a method to find relevant branches for cluster extraction. This work is our continuing effort toward improving the relevance of the search results in Folksonomies. To the best of our knowledge, our system was the first in using an automatically constructed tag ontology in a ranking algorithm for Folksonomies.

Manually created ontologies have been widely used in IR systems as well. The most well-known ontology would be WordNet (Miller, 1995). WordNet has been used in many IR and NLP tasks, such as query extension (Moldovan and Mihalcea, 2000), (Sieg *et al.*, 2004), word sense disambiguation (Banerjee and Pedersen, 2002), clustering (Hotho *et al.*, 2006a) (Pedersen *et al.*, 2004) and semantic indexing (Mihalcea and Moldovan, 2000). However, manually constructed ontologies are subjective and known to contain inconsistencies. In addition, the manual creation of ontologies requires a lot of resources and does not react fast enough on changes. A better approach would be to automatically derive ontologies from empirical data.

3 Search Personalization Algorithm in Folksonomies

3.1 Annotations for Folksonomies

In this paper, we use the following notation to define Folksonomies. A Folksonomy D is denoted as a four-tuple

$$D = \langle U, R, T, A \rangle \tag{1}$$

where U is a set of users, R is a set of resources, T is a set of tags, and A is a set of annotations where each annotation is defined by a three-tuple consisting of a (specific) user, a tag and a resource:

$$A \subseteq \{\langle u, r, t \rangle : u \in U, r \in R, t \in T\} \tag{2}$$

3.2 Modern approaches of Search Ranking in Folksonomies

Folksonomies may very in the way they ranked the returned results. Although very few social tagging systems/companies have disclosed the detailed information on their techniques, we hypothesize that the search engine in *Delicious* returns mainly the most recent resources for a query tag. In $Flickr$, the ranking seems to combine the number of tags and comments assigned to the picture. On the other hand, $Last.fm$ seems to use a vector space model for ranking resources.

Each user, u, is represented as a vector over the set of tags (the tags which the user had used to annotate resources). The value for the i^{th} tag t_i is a weight on the tag $w(t_i)$, corresponding to the importance of the particular tag to the user.

$$\vec{u} = \langle w(t_1), w(t_2) ... w(t_{|T|}) \rangle \tag{3}$$

A resource is also be represented as a vector over the set of tags (the tags which were used by the users to annotate the resource). To calculate the values in the vector, a variety of measures could be used. The *tag frequency*, *tf*, for a tag, t, and a resource, r, is the number of times the resource has been annotated with the tag. We define *tf* as:

$$tf(t,r) = |\{a = \langle u, r, t\rangle \in A : u \in U\}| \tag{4}$$

Tag frequencies can also be weighted. The *tf*idf* (term frequency * inverse document frequency (Salton and Buckley, 1988)) from IR can be modified for Folksonomies to reflect the relative distinctiveness of the tags. We define *tf*idf* as:

$$tf*idf(t,r) = tf(t,r) * \log(N/n_t) \tag{5}$$

where N is the total number of resources, and n_t is the number of resources to which the tag was applied.

For either representation, a similarity measure between a query, q, represented as a vector over the set of tags, and a resource, r, also represented as a vector over the set of tags, must be calculated. Here we use the Cosine similarity, which is defined as standard:

$$cos(q,r) = \frac{\sum_{t\in T} tf(t,q) * tf(t,r)}{\sqrt{\sum_{t\in T} tf(t,q)^2} * \sqrt{\sum_{t\in T} tf(t,r)^2}} \tag{6}$$

Note that, during our experiments, we empirically observed that *tf* produced a better result for measuring the similarity in constructing an ontology, while *idf* yielded a better result for generating weights for the vector space model that was used for the final ranking.

For the retrieval and ranking algorithms, the FolkRank algorithm (Hotho *et al.*, 2006b), which was adapted from the Google PageRank algorithm in IR to Folksonomies, has been reported to produce a superior performance over the vector space model. The FolkRank algorithm uses the notion of authority in ranking: users who used "authoritative" tags for "authoritative" resources are authoritative themselves, and the same principle holds for tags and resources. So the three dimensions of folksonomies effectively reinforce each other and converge to an optimal point in the three-dimensional space.

FolkRank also incorporates the principle of random surfer, just like PageRank does: a user normally follows the links but occasionally jumps to other random pages.

$$\overrightarrow{w} \longleftarrow d\overrightarrow{w}A + (1-d)\overrightarrow{p} \tag{7}$$

where $\overrightarrow{w}$ is the weights of the vector with u, r, t dimensions, A is the adjacency matrix of size u, r, t by u, r, t, d is the damping coefficient which controls the effect of random surfing, and $\overrightarrow{p}$ is the *preference vector* which indicates the interest of the specific user (to tags or resources, whichever is applicable to the adjacency matrix at hand). In FolkRank, a preference vector contains 1s for the target object(s) and

zero in all other slots. The preference vector in the algorithm is used to effectively influence the topic ranking indirectly. The final ranking scores/weights of the retrieved results are determined by the formula $tf * (\overrightarrow{w_p} - \overrightarrow{w})$, where w_p is the weight vector found without a preference vector (and w is the vector found with preferences). By subtracting w from w_p, popular resources which do not have any relation to the query are mutually canceled and are pushed down in the ranking. Thus, the FolkRank algorithm is effectively able to personalize the retrieved results (for tags or resources) by the use of preference vectors.

3.3 Modification of FolkRank Algorithm

According to the results reported in (Hotho *et al.*, 2006b), the FolkRank algorithm generates the more relevant ranking compared to other techniques developed for Folksonomies. It significantly outperforms the two most popular techniques - the vector space model and K-Nearest Neighbor.

In this work we modified the FolkRank algorithm for further improve its effectiveness. First of all, we modified the principle of formulation of the adjacency matrix in FolkRank, while keeping the PageRank's principle which defines the amount of transferred authority to be dependent on the number of outgoing links from the page. For example, there are two web-pages A and B, both of which have references to the page C. According to PageRank, the amount of authority transfered from A and B to C depends on the number of outgoing links from those web-pages. We suggest to use the same principle in the FolkRank algorithm during building the adjacency matrix. For example, for a connection between a tag t_i and a resource r_j, we propose to normalize the number of users u, who used tag t_i and resource r_j with the number of u which used tag t_i for any resource. For instance, if the tag "Apple" was used by 10 users for the web-resource *store.apple.com* but 100 users in total used that tag for any other resources, then the value in the adjacency matrix on the intersection of the tag "Apple" and the resource store.apple.com should be 0.1. The same principle holds for all other relations among users, tag and resources, in computing the adjacency matrix. We use the following formula to compute the connection $r_i \longrightarrow t_j$:

$$r_i \longrightarrow t_j = \frac{\sum_{i=1}^{U}(< u, r_i, t_j >: u \in U)}{\sum_{i=1}^{U}(< u, r_j, t >: u \in U, t \in T)} \tag{8}$$

With such normalization we can get a non symmetrical adjacency matrix, which will solve the problem of weights bouncing back during iteration of the FolkRank algorithm. In other words in our matrix the connection from A to B is not the same as B to A.

The preference vector formulation in the initial FolkRank algorithm is also the possibility to improve its effectiveness for increasing the ranking relevance. Intuitively, even with a three dimensional ranking system, filling the preference vector with zeros except for the target query does not reflect completely the user

interest in Folksonomies. Moreover, this approach does not address the problem of noise in ontologies caused by tag redundancy when several tags have the same meaning, and tag ambiguity when a single tag has many different meanings.

Tag redundancy and tag ambiguity problems confound the effectiveness of ranking algorithms. The task of combating noise is made even more difficult by capitalization, punctuation and misspelling. In our previous work, we tried to address the problem of tag redundancy with a relative success, by merging tags in a cluster and building the tags hierarchy (Gemmell *et al.*, 2008), (Shepitsen *et al.*, 2008).

However in our previous work, the ontology was not disambiguated, therefore the results left a room for improvements. In this work, we addressed the problem of tag ambiguity by employing a clustering algorithm called DSCBC and built an ontology in which the tags were disambiguated. This process is discussed in detail in the next section. Users' preference vectors were obtained by matching the users' tag profiles against the derived ontology. An example of a part of the ontology is depicted in Figure 1.

Input: Set of tags t ∈ T, users u ∈ U and r ∈ R, Preference vector over <U,R,T>
α, β, γ- tuning coefficients for controlling the speed and granularity of algorithm convergence
sim_{thr} - the threshold determining the stopping condition of the algorithm iteration.
n - number of iterations when we stop the further iteration.
d - damping factor
Output: Weight $w_1 < U, R, T >$ vector
$\overrightarrow{w_0} = \overrightarrow{w_1}$;
while *($n > 0$)* **do**
  $\overrightarrow{w_1} \longleftarrow \alpha * d\overrightarrow{w_1} + \beta * \overrightarrow{w_1} * A + (1-d)\gamma *$ *Pref. vector Normilize($\overrightarrow{w_1}$)*
  if *($sim(\overrightarrow{w_0}, \overrightarrow{w_1}) > sim_{thr}$)* **then**
    BREAK;
  end
  $\overrightarrow{w_0} = \overrightarrow{w_1}$;
  $n = n - 1$;
end

Algorithm 1: Modified FolkRank algorithm

3.4 Finding Ambiguous Tags with DSCBC Clustering Algorithm

One of the main obstacles for effective information retrieval is tag ambiguity, as discussed in the previous section. There are a lot of tags which have multiple meanings. For example, "run" could mean resources about jogging, or information about Unix compiler, or a popular novel of Ann Patchett.

In our previous work, during the automatic building of ontologies, we only considered the most popular sense of an ambiguous tag and ignored all other senses. For instance, "Java" was added as its most popular sense - as a computer language (For *Delicios*), and other meanings (coffee or a geographic location) was

discarded.

In this work, we adapted a clustering algorithm called DSCBC from our previous work in NLP (Tomuro *et al.*, 2007). The DSCBC algorithm is shown in Algorithm 2. In Phase I, a set of preliminary tag clusters are first created. In Phase II, some of those tag clusters are selected as committees – those which are dissimilar/orthogonal to all other committees selected so far. Then in Phase III, each tag is assigned to committees which are similar to the tag. When an ambiguous tag is assigned to one of more committees, each time the features of the assigned committee are removed from the tag. Thus, ambiguous tags are identified as those which belong to more than one committee. The tuning coefficients determine the quality and the structure of obtained committees. The n parameter determines the tightness of obtained committees and the degree of overlapping. In this work this coefficient is very important as it influences the quality of the ontology, which in turn has a great impact on the preference vector formulation. The q coefficient influences the quality of cluster centroids, by the number of features. Although using a larger number of features would be preferred to represent a centroid, it may also decrease the number of resources assigned to the cluster, thereby diminishing the generality of the clusters. The coefficient β directly controls the number of overlapping tags among committees. All coefficients were found empirically during the preliminary test runs of our experiments.

Input: Set of tags T. Tuning coefficients:
n - number of the most similar tags chosen for the target tag
q - number of features for finding the centroid
β - similarity threshold for adding tags to committees
γ - similarity threshold for assigning tags to committees
Output: Set of committees C. Set of tags T where each t $\in$ T is assigned to committees in C.
Phase I. Finding set of clusters L
foreach $t_i \in T$ **do**
  Select a set k of n most similar t_j : i $\neq$ j
  add k to L if it is not already in L.
end
Phase II. Find Communities C (C is initially empty set)
foreach $c \in L$ **do**
  Find the centroid of c using only q features shared by most of tags in the cluster
  Add c to C if its similarity (measured among centroids) to every other cluster is lower than β
end
Phase III. Assign tags to committees
foreach $t \in T$ **do**
  Assign t to committee c in C if the similarity of t to centroids of c is higher than γ
end

Algorithm 2: Clustering tags using DSCBC

3.5 Modified Agglomerative Hierarchical Clustering for Extracting Tag Ontology

In our previous work, we used a standard hierarchical agglomerative clustering algorithm for building the ontological tree. However, the structure obtained from such algorithm did not resemble a standard ontology, such as $Yahoo$[3] or Open Directory Project [4]. That is why we modified the standard approach and did not use a similarity threshold for finding set tags/cluster to connect in cluster. Otherwise we will get a network where the parent node could be presented by a cluster rather than by a singular tag. That is why we sort pairs of tags or tag clusters by similarities and join the pair on the top of the list.

In addition, on every joining step we control whether the tags are ambiguous. Ambiguous tags should be included in ontology more than once, according to the number of meanings they have. For realizing this approach, we save a list of tags with a set of committees to which they belong. Afterwards, during a joining step we check whether the tag is ambiguous (belongs to more than one committee). When we join an ambiguous tag, we determine its current meaning and remove the features of the current meaning from other instances of this tag in the list for further clustering.

Therefore, ambiguous tags will be added to the ontology as many times as many senses they have. As the result we obtain the disambiguated ontological tree in Folksonomies, where the more general concepts are on the top of the tree and more specific is closer to bottom. The example of the obtained tree is shown at Figure 1. We can see in the figure that more specific nodes are located on the lower "floors".

3.6 Search Personalization with Ontology

Having an ontology built from Folksonomies we have the possibility to use it in the ranking algorithm for the specific user exploiting the hierarchical structure of the semantic networks.

Our approach offers the possibility to spread user interest from the target tag to its parents, and they in turn spread it to their parents and siblings of the target tag with some decay factor. As the result we can build a map of the user's interests and formulate an optimal preference vector.

Naturally, we should also use as an activation point the tags from the user profile, because they are sources of the user's interests. By processing the tags from the user profile sequentially, we strengthen the nodes which lie on the intersections of the user's interests.

For instance, if the user has tags "swimming_pools_NY", "flip_turns", "swim_tips", "trade_mill, "Arnold_squat", we may conclude that first - this user is interested in swimming and in bodybuilding. Sequentially spreading the interest score degree from those five tags, the algorithm will give a high weight to the sport nodes in general. Therefore, the resources related to sport will be ranked higher as they will be strengthened on every iteration of the FolkRank algorithm. In the first

[3] www.yahoo.com
[4] www.dmoz.org

Input: Set of tags T. Set of Committees C.
Output: An ontology of folksonomy tags
L is a list containing pairs of tag/clusters with associated similarity, initially empty.
foreach $t_i \in T$ **do**
  Find all t_j (i $\neq$ j) similar to t_i;
  add $\langle t_i, t_j \rangle$ pair in L, if it is not already in L.
end
while *L is not empty* **do**
  1. Sort L by the similarity of tags/clusters
  2. Pop the pair with tag t_i with the highest similarity from List L.
  if t_i *belongs to more than one committee in* C **then**
    1. Find the most similar committee c by measuring t_i pair tag/cluster with committee centroid
    2. Join t_i with its tag/cluster partner
    3. Remove all features intersecting with t_i and the most similar committee of its pair
  end
  else
    1. Join t_i with its tag/cluster
    2. Substitute all instances of t_i in L by the created cluster
  end
end

Algorithm 3: Ontology Construction Algorithm

stage we build an ontological tree with abovementioned algorithm. Afterwards, we match the tree with the user profile and generate the personalized interest score distribution among nodes in the tree. We generate a preference vector from the tag ontology, activate it with the query tag and load that profile into the extended FolkRank algorithm which in turn generate the FolkRank weights. Finally, we compute the vector space model weights for every resource and multiply them with the FolkRank weights to find the final ranking coefficients.

4 Experimental results

We evaluated our modified FolkRank algorithm using data from two real-world collaborative tagging systems. The first one is *Delicious* where users annotate web pages. The crawled dataset contains 29,918 users, 6,403,442 resources and 1,035,177 tags. There are 47,184,492 annotations with one user, resource and tag. We randomly picked 10% of users with all their tags and resources. The test dataset consists of 2900 users 113,443 tags and 583,137 resources.

We also tested our algorithm on the data from Last.fm, where users assign tags for musical resources: songs, albums and artists. From the Last.mm dataset, we randomly chose 2,900 users which used 37,913 tags for 261,123 media resources.

As a basis for comparison we used ranking based on the pure vector space model, and the original FolkRank algorithm with binary preference vector. In

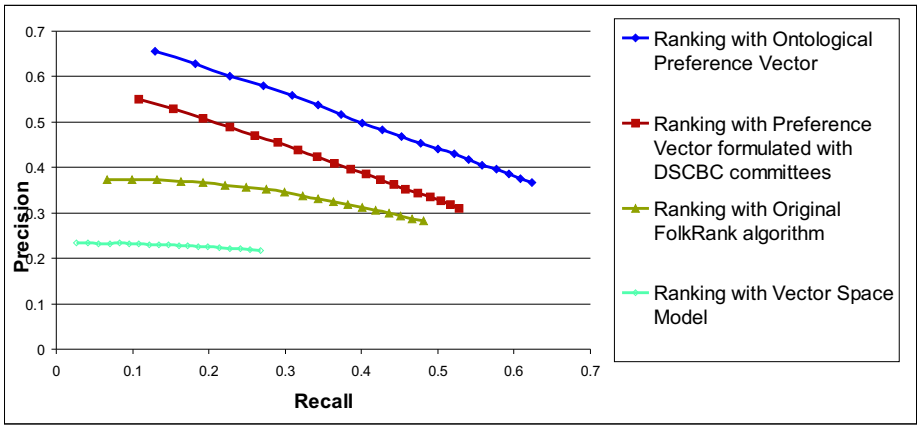

FIGURE 2: Recall vs. Precision for Delicious dataset

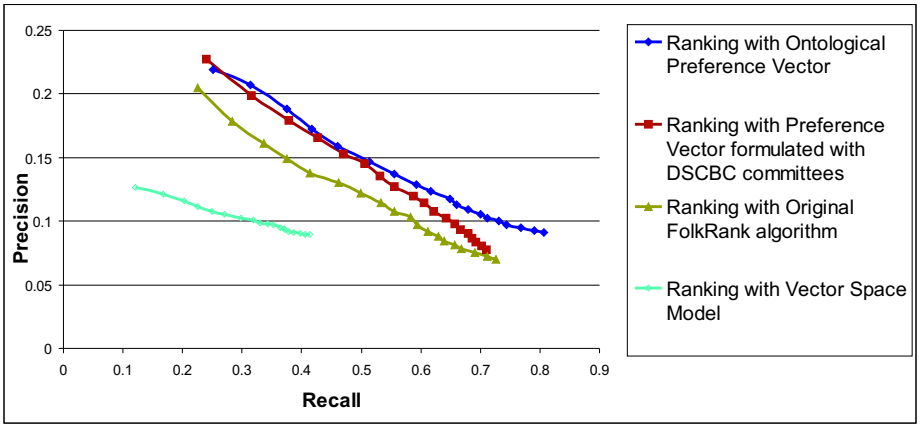

FIGURE 3: Recall vs. Precision for Last.Fm dataset

addition, we compared with the version where the tag clusters were generated by DSCBC without building an ontology - if the target tag belongs to the committee all members of that committee get a 1 in the preference vector. If the target tag belongs to several committees, we increase the weights only for the cluster with the biggest user interest. For each committee, c, the user's interest is calculated as the ratio of times the user, u, annotated a resource with a tag from that committee over the total number of annotations by that user. We denote this weight as *uc_w(u,c)* and defined it as:

$$uc\_w(u,c) = \frac{|\{a = \langle u, r, t\rangle \in A : r \in R, t \in c\}|}{|\{a = \langle u, r, t\rangle \in A : r \in R, t \in T\}|} \tag{9}$$

We used the "leave one out" scheme for generating a test dataset. For every test

users we remove one resource and all tags associated with that resource. The list of tags was our relevant set, which we try to predict using the modified FolkRank algorithm. As the measure of the effectiveness we used Recall and Precision for both datasets. We sequentially tested all users and all their resources and found the average of Recall and Precision.

The results of the experiments are provided in Figures 2 and 3. All the tuning coefficients were empirically found on their optimal level for generating the highest Recall in Folksonomies. The results showed that, for both datasets, the modified FolkRank algorithm with the ontologically generated preference vectors outperformed the vector space model and the original FolkRank algorithm. In addition, it also showed that our proposed method produced better results as compared to the version where preference vectors are formulated by using only the DSCBC committees. The results showed the usefulness of building disambiguated ontologies for reflecting the user information needs in Folksonomies.

5 Conclusions and Future Work

In this paper we presented a modified hierarchical agglomerative clustering algorithm which uses the *DSCBC* algorithm to construct a disambiguated ontology from tag datasets. The obtained ontology, after matching with the user profile, effectively reflects the users' information needs. The preference vectors are fed into the modified FolkRank algorithm and showed its effectiveness in search personalization. Our experimental results showed ranking much more relevant to the users' interest as compared to the vector space model and the standard binary preference vector formulation technique.

Our future research will focus on the further improvement of the cluster qualities generated by the DSCBC algorithm. With that goal in mind, we are planning to experiment probabilistic models to find the optimal committee properties such as tightness, degree of tags overlap among committees and the distance among committees themselves.

Building an ontology in Folksonomies is also our future area of research. We believe that there is a possibility to further improve the subsumption relation among tags, which in turn would generate a better ontology. We believe that including the third (user) dimension in finding similarity and parent-child relation may also increase the utility of the ranking algorithm.

We also believe that the FolkRank algorithm may be further improved by including in the equation the date of annotation and putting more weights to those users who used tags earlier than others. In addition, including a similar ontological clustering approach for resources and users will probably further improve the the relevancy of personalized ranks.

References

S. Banerjee and T. Pedersen (2002), An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet, *Lecture Notes in Computer Science*, pp. 136–145.

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

A. Hotho, R. Jaschke, C. Schmitz, and G. Stumme (2006a), Folkrank: A Ranking Algorithm for Folksonomies, *Proceedings of FGIR*, 2006.

A. Hotho, R. Jaschke, C. Schmitz, and G. Stumme (2006b), Information Retrieval in Folksonomies: Search and Ranking, *The Semantic Web: Research and Applications*, 4011:411–426.

A. Hotho, S. Staab, and G. Stumme (2003), Wordnet Improves Text Document Clustering, in *Proceedings of the SIGIR 2003 Semantic Web Workshop*.

S. Middleton, N. Shadbolt, and D. De Roure (2004), Ontological User Profiling in Recommender Systems, *ACM Transactions on Information Systems*, 22(1):54–88, ISSN 1046-8188.

R. Mihalcea and D. Moldovan (2000), Semantic Indexing Using WordNet Senses, in *Proceedings of ACL Workshop on IR & NLP*, pp. 35–45.

G. Miller (1995), WordNet: A Lexical Database for English, *Communications of the ACM*, 38(11):39–41.

D. Moldovan and R. Mihalcea (2000), Using WordNet and Lexical Operators to Improve Internet Searches, *Internet Computing, IEEE*, 4(1):34–43.

M. OConnor and J. Herlocker (2001), Clustering items for collaborative filtering, in *Proceedings of SIGIR-2001 Workshop on Recommender Systems*.

T. Pedersen, S. Patwardhan, and J. Michelizzi (2004), WordNet:: Similarity-Measuring the Relatedness of Concepts, in *Proceedings of the National Conference on Artificial Intelligence*, pp. 1024–1025.

G. Salton and C. Buckley (1988), Term-weighting Approaches in Automatic Text Retrieval, *Information Processing and Management: An International Journal*, 24(5):513–523.

M. Sanderson and B. Croft (1999), Deriving Concept Hierarchies from Text, in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 206–213.

A. Shepitsen, J. Gemmell, B. Mobasher, and R. Burke (2008), Personalized Recommendation in Social Tagging Systems Using Hierarchical Clustering, in *Proceedings of the 2008 ACM Conference on Recommender systems*, pp. 259–266.

A. Sieg, B. Mobasher, S. Lytinen, and R. Burke (2004), Using Concept Hierarchies to Enhance User Queries in Web-based Information Retrieval, in *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications*.

N. Tomuro, S.L. Lytinen, K. Kanzaki, and H. Isahara (2007), Clustering Using Feature Domain Similarity to Discover Word Senses for Adjectives, in *Proceedings of the 1st IEEE International Conference on Semantic Computing*, pp. 370–377.