Search in Social Tagging Systems Using Ontological User Profiles

Andriy Shepitsen and **Noriko Tomuro**
College of Computing and Digital Media
DePaul University
Chicago, Illinois, USA
ashepits@cdm.depaul.edu, tomuro@cs.depaul.edu

Abstract

In this paper we present a modified hierarchical agglomerative clustering algorithm for building tag ontologies for social tagging systems. The modified algorithm first uses a clustering algorithm called *Domain Similarity Clustering By Committee (DSCBC)* (Tomuro et al. 2007) to derive a set of tag *committees*. We apply DSCBC to the tags entered by the users of social tagging systems and derive (un-ambiguous) committees of tags. Using the committees, a tag ontology is constructed in which an ambiguous tag is separated into multiple, disambiguated tags/nodes. Then a tag profile of a given user is matched against the ontology, and an ontological profile of the user is created. Finally a preference vector is fed into the (modified) FolkRank algorithm (Hotho et al. 2006a), and the web resources ordered based on the user's preferences are returned. We run our system on the data from two social tagging systems and compare the results with other algorithms. The results showed our algorithm achieved marked improvements over other algorithms.

Introduction

In recent years, there has been a rapid growth in social tagging systems – so-called *Folksonomies* where users assign keywords or tags to web resources for future references or collaboration with other users. Web resources are virtually any kind of information available on the Internet, ranging from web-pages (Delicious, Connotea), scientific articles (Bibsonomy) to media resources (Last.fm, Youtube)). One of the main services provided by social tagging systems is searching. Search mainly happens when the user enters a tag as a query, and a list of related resources, which are also ranked by relevance, is returned to the user. The core of the search engine is the ranking algorithm. Most of the ranking algorithms currently used in the social tagging systems are those developed for Information Retrieval (IR). However, those algorithms cannot be easily adapted for Folksonomies. The main reason is because most social tagging systems allow unrestricted tagging – users are free to choose any words or set of characters to formulate tags. Although those problems exist in IR as well, they are much more salient in social tagging systems due to their open nature.

In this work, we develop a modified hierarchical agglomerative clustering algorithm for 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 user's interest and retrieve/rank the relevant resources that are better personalized to the user. In our previous work (Shepitsen et al. 2008), we used a hierarchical agglomerative clustering algorithm to build a tag hierarchy and showed improved results for personalizing search and retrieval in Folksonomies. The work here is our continuing effort, focusing on the construction of a disambiguated ontology and the modification of 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*. Then we organize the tag committees into a ontology by using a modified hierarchical agglomerative clustering algorithm. Afterwards, for a given user, we compute a preference vector based on his tag profile. 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) and ranking using a vector space model. The results showed our system achieved marked improvements over other algorithms by the use of disambiguated ontology.

Related Work

The first attempt 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. They reported that in 70% of the cases the derived hierarchy coincided with the judgement by the experts.

Mika (Mika 2007) proposed an automatic creation of tags in Folksonomy networks based on the co-occurrences between the resources and users. He used a graph clustering

technique to connect tags which were used by the same users and for the same resources. Here, a connection essentially represents a user community of like-minded users. However, the derived graph/network was rather "flat" with several densely interrelated communities, thus in our opinion, not a true ontology.

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 Natural Language Processing (NLP) tasks, such as query expansion (Moldovan and Mihalcea 2000), word sense disambiguation (Banerjee and Pedersen 2002) and semantic indexing (Mihalcea and Moldovan 2000). However, not only do manually constructed ontologies require a large amount of time and effort to build, they also tend to contain inconsistencies. There has been much work in NLP focusing on automatically deriving ontologies from empirical data.

Ontological Search in Folksonomies

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:

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

Modern approaches of Ranking in Folksonomies

Most of the Folksonomies use a vector space model with different modification as the ranking algorithm. For this reason, we will use the vector space model as a baseline in evaluating our proposed approach. Each user, u, is represented as a vector over the set of 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. 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.

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.

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 Cosine similarity, which is commonly used in IR.

Note that, during the building of the tag ontology, we empirically discovered 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, 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 (Hotho et al. 2006b). FolkRank algorithm uses preference vectors for a given user and tags in order to personalize search results. The results reported in (Hotho et al. 2006b) showed that the FolkRank algorithm generated more relevant ranking compared to other techniques developed for Folksonomies. It significantly outperformed the two most popular techniques - the vector space model and K-Nearest Neighbor.

In this work, we modified the FolkRank algorithm in two ways. First, we modified the representation of the adjacency matrices (which indicate the co-occurrences of tags and users, users and resources, tags and resources) to be directed (from undirected in FolkRank). For example, for the connection from tag t_i to resource r_j, we multiplied the value (a weight calculated based on the level of importance of t_i and r_j, as used in PageRank) by the proportion of the users who used t_i for r_j over all users who used t_i for any resource. With this normalization we obtained non-symmetric adjacency matrices, thereby solving the problem of weights bouncing back during iteration in the FolkRank algorithm.

Second, we modified the formulation of preference vectors. Intuitively filling the preference vector with zeros except for the target query tag and the user does not reflect completely the user interest in Folksonomies. Using the disambiguated ontology of tags, after matching it against the user profile we were able to formulate the preference vector which reflected the user's interest in Folksonomies more accurately.

Finding Unambiguous Tags Ontology

One of the main obstacles for effective information retrieval is the abovementioned problem of tags ambiguity. There are a lot of tags which have multiple meanings. For example, "Sicilian" could mean resources with about Sicilian Mafia, or information about tourist facilities of the Italian island, or a popular chess opening. In our previous work (Shepitsen et al. 2008), 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, and other meanings (coffee or a geographic location) were 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 1. By using DSCBC, we can identify ambiguous tags as those which belong to more than one cluster/committee. Note that the algorithm contains several parameters, some of which control the tightness of a cluster (i.e., within-cluster similarity) and the distinctness of a cluster (i.e., between-cluster similarity). Those parameter values were were found empirically during the test runs of our experiment.

Input: Set of tags t ∈ T. Tuning coefficients:
n - number of the most similar tags chosen for the target tag
q - number of features for finding the centroid
sim - the similarity threshold for adding tags to committees
Output: Set of tags t ∈ T with reference of each t_i to list of Committees to which t_i belongs
Phase I. Finding set of clusters L
foreach $t_i \in T$ **do**
  Select a set k of n most similar t_j : i ≠ j
  add k into L, if it is not already in L.
end
Phase II. Find Communities C
foreach $c \in C$ **do**
  Find the centroid of c using only q features shared by most of tags in the cluster
  Add c in L if its similarity to every cluster is lower than s
end
Phase III. Assign tags to committees C
foreach $t \in T$ **do**
  Add t to set cluster C if the similarity is higher than sim
end

Algorithm 1: Clustering tags using DSCBC

Modified Agglomerative Hierarchical Clustering for Building Tag Ontology

In our previous work, we used a standard hierarchical agglomerative clustering algorithm for building the ontological tree. However, although we were able to improve a personalized search for resources for a given tag using the ontology, the derived structure did not resemble a standard ontology such as $Yahoo$ (www.yahoo.com) or Open Directory Project (www.dmoz.org).

To construct an ontology which better matches human intuitions, we first applied the DSCBC algorithm to the tags and derived a set of committees. Here, the committees are essentially representatives (or centroids) of the (disambiguated) ontological concepts. Then we assigned each tag to committees. An ambiguous tag belongs to more than one committee. Then we built an disambiguated ontology by a modified Hierarchical Agglomerative clustering approach. We first compute a pair-wise similarity between any two tags and sort those pairs according to the similarity values. Then initially we join the most similar pair and create a cluster. Afterwards, we iterate through the whole collection and substitute all the instances of joined members (if they are not ambiguous) by the obtained cluster and repeat the process until the list of pairs is empty. When we process an ambiguous tag, we first find its "core meaning" by finding the committee to which the tag is most similar, then remove all (non-zero) features (resources) that are encoded in committee from all instances left in the dataset. With this approach, we can cover all senses of an ambiguous tag, for all such tags, during ontology generation. For example, an ambiguous tag *"NLP"* could be first added to the ontology with Neuro-linguistic programming (as its first sense), then later with resources related to Natural Language Processing.

Search in Folksonomies using Ontology

After the ontology is constructed, we search in the ontology for the query tag entered by a specific user. We first match the tree with the user's profile and derive a score distribution for the nodes in the tree which reflects the user's general interest. To do so, we use the tags in the user's profile as initial activation points, then spread the activation up and down the ontology tree. Initially all nodes' interest scores are set to zeros. Then for each tag in the profile, we first add one to the interest score of the node which the tag belongs to in the tree. To spread activation from a given node, we use two parameters: *decay factor*, which determines the amount of the interest to be transfered to the parent/child of the current node; and *damping threshold* - if the interest score becomes less than this value we stop further iteration. The final score of a node may become more than one as a result of spreading activation when there are tags in the user profile that are close to each other in the nodes in the ontology tree. The resulting score distribution of the tree is effectively personalized to the user's interest.

Using this personalized ontology, we search the tree for a query tag (of this user). In the same way as the tags in the profile, we spread activation over the ontology from the node to which the tag belongs to, but this time we add a weight to emphasize the relative importance of the query tag compared to the tags from the profile. In the experiment, we used 3.5 and 1.8 for Delicious and Last.fm respectively. The resulting scores of the ontology nodes are collected in a vector, which makes the preference vector for the user for the query tag. Finally, we feed the preference vector to the modified FolkRank algorithm and get the ranked list of resources with their relevance scores.

Experimental results

We evaluated our modified FolkRank algorithm using data from two real collaborative tagging systems. The first one is *Delicious* where users annotate web pages. The 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 pure vector space model, the original FolkRank algorithm with binary preference vector. In addition, we used tag clusters generated by DSCBC without building an ontology - if the target tag belongs to a committee, all members of the committee get a 1 in the preference vector. If the target tag belongs to several committees, we increase the weight only for the cluster with the highest user interest. For each committee, c, the user's interest is calculated as the ratio of times the

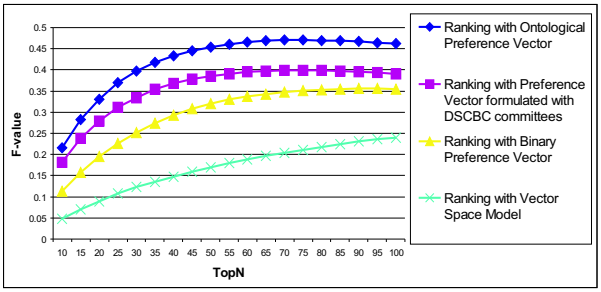

Figure 1: F-values for Delicious Social Tagging System

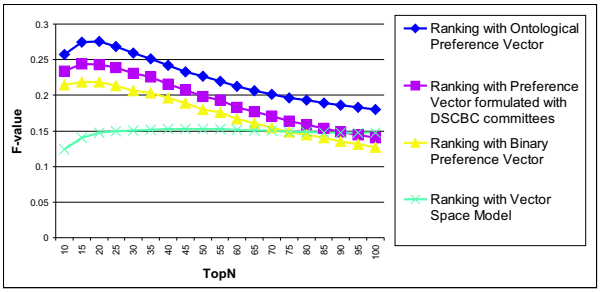

Figure 2: F-values for Last.fm Social Tagging System

user, u, annotated a resource with a tag from that committee over the total number of annotations by that user. We used the "leave one out" method to generate a test dataset. For every test user, we remove one tag and all resources associated with that tag. The list of resources was our relevant set, which we try to predict using the modified FolkRank algorithm. Then the target tag was used as a query to the search engine and a list of returned resources was used like returned set. As the measure of effectiveness, we computed *F-values* for both datasets. The *F-value* is defined as standard:

$$F = \frac{2 \cdot Precision \cdot Recall}{Precision+Recall} \quad (3)$$

The results of the experiments are shown in Figures 1 and 2. All of the tuning coefficients were set to those that yielded the highest recall during our preliminary test runs. We also used 5-fold cross validation where, in each fold/iteration, 80% of the data was used for building ontology and the remaining 20% was used in testing. The final recall and precision values were calculated as the average over 5 iterations.

The results showed that, for both datasets, the modified FolkRank algorithm with ontologically generated preference vector outperformed the vector space model and FolkRank original algorithm. In addition, it also showed better results compared to the version where preference vectors were obtained by using only the DSCBC committees. The results showed the effectiveness of using disambiguated ontologies for reflecting the users' information needs in Folksonomies.

Conclusions and Future Work

In this paper, we developed a modified hierarchical agglomerative clustering algorithm which uses the $DSCBC$ algorithm to construct a disambiguated ontology from ambiguous tags. The obtained ontology, after matching with the user profile, effectively reflects the users' information needs. Our results showed ranking much more relevant to the users' interest compared to the vector space model and the standard binary preference vector formulation technique.

For future work, we plan to investigate ways to tune the parameters in the DSCBC algorithm to improve the quality of the derived committees. Another important topic of future research is to build a more complete ontology with various relations between tags, in addition to the hierarchical relation. We anticipate that the incorporation of other relations would enhance the current system by generating preference vectors which reflect the users' preferences even better.

References

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

Hotho, A.; Jaschke, R.; Schmitz, C.; and Stumme, G. 2006a. Folkrank: A ranking algorithm for folksonomies. *Proc. FGIR* 2006.

Hotho, A.; Jaschke, R.; Schmitz, C.; and Stumme, G. 2006b. Information retrieval in folksonomies: Search and ranking. *The Semantic Web: Research and Applications* 4011:411–426.

Mihalcea, R., and Moldovan, D. 2000. Semantic indexing using WordNet senses. In *Proceedings of ACL Workshop on IR & NLP*, 35–45.

Mika, P. 2007. Ontologies are us: A unified model of social networks and semantics. *Web Semantics: Science, Services and Agents on the World Wide Web* 5(1):5–15.

Miller, G. 1995. WordNet: a lexical database for English. *Communications of the ACM* 38(11):39–41.

Moldovan, D., and Mihalcea, R. 2000. Using WordNet and lexical operators to improve Internet searches. *Internet Computing, IEEE* 4(1):34–43.

Salton, G., and Buckley, C. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing and Management: an International Journal* 24(5):513–523.

Sanderson, M., and Croft, B. 1999. Deriving concept hierarchies from text. In *In Proceedings of the 22nd annual international ACM SIGIR conference on Research and Development in Information Retrieval*, 206–213.

Shepitsen, A., Gemmell, J., Mobasher, B. and Burke, R. 2008. Personalized recommendation in social tagging systems using hierarchical clustering. In *Proceedings of the 2008 ACM conference on Recommender systems*, 259–266. ACM New York, NY, USA.

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