Adapting K-Nearest Neighbor for Tag Recommendation in Folksonomies

Jonathan Gemmell, Thomas Schimoler, Maryam Ramezani, Bamshad Mobasher

Center for Web Intelligence
School of Computing, DePaul University
Chicago, Illinois, USA

{jgemmell, tschimoler, mramezani, mobasher}@cdm.depaul.edu

Abstract

Folksonomies, otherwise known as Collaborative Tagging Systems, enable Internet users to share, annotate and search for online resources with user selected labels called tags. Tag recommendation, the suggestion of an ordered set of tags during the annotation process, reduces the user effort from a keyboard entry to a mouse click. By simplifying the annotation process tagging is promoted, noise in the data is reduced through the elimination of discrepancies that result in redundant tags, and ambiguous tags may be avoided. Tag recommenders can suggest tags that maximize utility, offer tags the user may not have previously considered or steer users toward adopting a core vocabulary. In sum, tag recommendation promotes a denser dataset that is useful in its own right or can be exploited by a myriad of data mining techniques for additional functionality.

While there exists a long history of recommendation algorithms, the data structure of a Folksonomy is distinct from those found in traditional recommendation problems. We first explore two data reduction techniques, p-core processing and Hebbian deflation, then demonstrate how to adapt K-Nearest Neighbor for use with Folksonomies by incorporating user, resource and tag information into the algorithm. We further investigate multiple techniques for user modeling required to compute the similarity among users. Additionally we demonstrate that tag boosting, the promoting of tags previously applied by a user to a resource, improves the coverage and accuracy of K-Nearest Neighbor.

These techniques are evaluated through extensive experimentation using data collected from two real Collaborative Tagging Web sites. Finally the modified K-Nearest Neighbor algorithm is compared with alternative techniques based on popularity and link analysis. We find that K-Nearest Neighbor modified for use with Folksonomies generates excellent recommendations, scales well with large datasets, and is applicable to both narrow and broadly focused Folksonomies.

Introduction

Folksonomies, also known as Collaborative Tagging Systems, have emerged as a powerful trend allowing Internet users to share, annotate and explore online resources through personalized labels. Several Collaborative Tagging Systems have recently gained popularity attracting millions of users. Delicious[1] supports users as they bookmark URLs. Flickr[2] allows users to upload, share and manage online photographs. Citeulike[3] enables researchers to manage and discover scholarly references. Still other Collaborative Tagging Systems specialize in music, blogs and business documents.

At the core of Collaborative Tagging Systems is the post: a user describes a resource with a set of tags. These tags may be descriptive ("Folksonomies"), subjective ("awesome"), organizational ("toread") or completely idiosyncratic ("jfgwh"). Taken in isolation an individual annotation allows a user to organize web resources for later use: resources can be easily sorted, aggregated and retrieved. Resources may be annotated with multiple tags allowing a resource to be identified with several topic areas rather than pigeonholed in a single directory. Moreover users need not conform to a predefined vocabulary or rigid hierarchy but may annotate a resource with any tag they wish thereby reducing user effort and limiting the entry cost.

However the utility of tagging extends beyond their immediate use. Taken as a whole, the aggregation of many annotations results in a complex network of interrelated users, resources and tags often referred to as a Folksonomy (Mathes 2004). The opportunity to explore the Folksonomy unburdened by a preconceived navigational or conceptual hierarchy is crucial to the utility and popularity of Folksonomies. Users are able to share or discover resources through the collaborative network and connect to other users with related interests. Collaborative Tagging Systems can identify groups of like-minded users, catering not only to mainstream but also to non-conventional users who are often under-served by traditional Web tools. Furthermore, users may enjoy the social aspects of collaborative tagging, attracted by a sense of community not offered by either ontologies or search engines.

A distinct advantage of Folksonomies is the richness of the user profiles. If a user is interested enough in a resource to annotate it, the tag describes the user as much as it describes the resource. As users annotate resources, the system is able to track their interests. These profiles are a powerful tool for data mining algorithms.

[1] delicious.com
[2] www.flickr.com
[3] www.citeulike.org

Even though tags offer many benefits both in the short and long term, they also present unique challenges for recommendation algorithms. Most Collaborative Tagging Applications permit unsupervised 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. Unsupervised tagging can result in tag redundancy in which several tags have the same meaning or tag ambiguity in which a single tag has many different meanings. Such inconsistencies can confound users as they attempt to utilize the Folksonomy.

Tag recommendation provides a means to overcome these problems. It reduces the user effort to a mouse click rather than a keyboard entry. By reducing the effort users are encouraged to tag more frequently, apply more tags to an individual resource, reuse common tags, and perhaps use tags the user had not previously considered. Moreover, user error is reduced by eliminating redundant tags caused by capitalization inconsistencies, punctuation errors, misspellings and other discrepancies. The tag recommender can further promote a core tag vocabulary steering the user toward adopting certain tags while not imposing any strict rules. The tag recommender may even avoid ambiguous tags in favor of tags that offer greater information value. The final result is a cleaner, denser dataset that is useful in its own right or for further data mining techniques.

However the data generated through Collaborative Tagging differs from that common in recommendation algorithms. The introduction of tags manifests a third dimension which must be integrated into recommenders that traditionally incorporate only two dimensions. In this paper we demonstrate how K-Nearest Neighbor may be adapted to recommend tags in Folksonomies. We describe how both user and resource information can be directly applied in the algorithm improving both coverage and accuracy while reducing computational costs. In addition several user models are explored including vectors over the set of tags, vectors over the set of resources, combinations of these two and features derived through Hebbian deflation.

The rest of this paper is organized as follows. We begin by presenting some related work involving the use of recommendations in Folksonomies. We explore the data structure of folksonomies and discuss two feature reduction techniques: p-core processing and Hebbian deflation. We then outline the basic approach used for recommending tags in Folksonomies and propose modifications to K-Nearest Neighbor. After a discussion of the datasets and a description of the experimental methodology, we evaluate the proposed modifications using data collected from two real world Folksonomies. Finally, we compare the modified algorithm with alternative strategies based on popularity and link analysis.

Related Work

As Collaborative Tagging Applications have gained in popularity researchers have started to explore and characterize the tagging phenomenon. In (Macgregor and McCulloch 2006) and (Golder and Huberman 2006) the authors studied the information dynamics of Delicious, one of the most popular Folksonomies. The authors discussed how tags have been used by individual users over time and how tags for an individual resource stabilize over time. They also discussed two semantic difficulties: tag redundancy, when multiple tags have the same meaning, and tag ambiguity, when a single tag has multiple meanings. (Macgregor and McCulloch 2006) provide an overview of the phenomenon and explore reasons why both Folksonomies and Ontologies will have a place in the future of information access.

There have been several recent research investigations into recommendation within Folksonomies. Unlike traditional recommender systems which have a two-dimensional relation between users and items, tagging systems have a three dimensional relation between users, tags and resources. Recommender systems can be used to recommend each of the dimensions based on one or two of the other dimensions. (Tso-Sutter, Marinho, and Schmidt-Thieme 2008) applies user-based and item-based collaborative filtering to recommend resources in a tagging system and uses tags as an extension to the user-item matrices. (Nakamoto et al. 2008a) and (Nakamoto et al. 2008b) use tags as context information to recommend resources.

Other researchers have studied tag recommendation in folksonomies. (Jaschke et al. 2007) compares user-based collaborative filtering and a graph-based recommender based on the Pagerank algorithm to recommend personalized tags. (Heymann, Ramage, and Garcia-Molina 2008) use association rules to recommend tags and introduce an entropy-based metric to find how predictable a tag is. (Lipczak 2008) uses the title of a resource, the posts of a resource and the user's vocabulary to recommend tags. The results show that tags retrieved from the user's vocabulary outperform recommendations driven by resource information. However the experiment was performed on data from Bibsonomy, a Folksonomy focused on scientific publications, and thus might not be applicable to multi-domain data that cover.

(Xu et al. 2006) presents general criteria for a good tagging system including high coverage of multiple facets, high popularity and least-effort. They categorize tags to content-based tags, context-based tags, attribute tags, subjective tags, and organizational tags and use a probabilistic method to recommend tags. (Basile et al. 2007) proposes a classification algorithm for tag recommendation. (Sigurbjörnsson and van Zwol 2008) uses a co-occurrence-based technique to recommend tags for photos in Flickr. The assumption is that the user has already assigned a set of tags to a photo and the recommender uses those tags to recommend more tags. (Adrian, Sauermann, and Roth-Berghofer 2007) suggests a semantic tag recommendation system in the context of a semantic desktop. (Song et al. 2008) uses clustering to make real-time tag recommendation.

Data Structures of Folksonomies

In this section we define the three-dimensional data structure of a Folksonomy and describe how to construct two-dimensional projections. We further explore two data reduction techniques for use with Folksonomies. The first is a data selection technique, P-core processing, that reduces the size

of the data by removing users, resources and tags that occur less than a predefined number of times. The second is a data extraction technique, Hebbian Deflation, which produces a new set of features from a two-dimensional projection of the Folksonomy.

Data Models

The data structure of a Folksonomy differs from that common to most traditional recommendation algorithms. A Folksonomy can be described as a four-tuple D:

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

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, represented as user-tag-resource triples:

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

A Folksonomy can, therefore, be viewed as a tripartite hyper-graph (Mika 2007) with users, tags, and resources represented as nodes and the annotations represented as hyper-edges connecting a user, a tag and a resource.

The tripartite nature of Folksonomies make it ill suited for many traditional data mining techniques. User based K-Nearest Neighbor for example requires a means to measure the similarity between users. The introduction of a third dimension confounds this task.

Aggregate projections of the data can be constructed, reducing the dimensionality by sacrificing information. (Schmitz et al. 2006) The relation between resources and tags can be formulated as a two-dimensional projection, RT, such that each entry, $RT(r, t)$, is the weight associated with the resource, r, and the tag, t. This weight may be binary, merely showing that one or more users have applied that tag to the resource, or it may be finer grained using the number of users that have applied that tag to the resource:

$$RT_{tf}(r, t) = |\{a = \langle u, r, t \rangle \in A : u \in U\}| \tag{3}$$

Such a measure is equivalent to *term frequency* or *tf* common in Information Retrieval. Similarly, *term frequency * inverse document frequency* or *tf*idf* (Salton and Buckley 1988) can be adapted for use with two-dimensional projections:

$$RT_{tf*idf}(r, t) = RT_{tf}(r, t) * \log(|R|/n_r) \tag{4}$$

The *tf*idf* multiplies the tag frequency by the relative distinctiveness of the tag. The distinctiveness is measured by the log of the total number of resources, $|R|$, divided by the number of resources to which that tag was applied, n_r. Similar two-dimensional projections can be constructed for *UT* in which the weights correspond to users and tags, and *UR* in which the weights correspond to users and resources.

While a portion of the information is lost through this process the result is a two-dimensional matrix which can be readily applied to existing data-mining algorithms such as K-Nearest Neighbor.

P-Core Processing

Through P-Core Processing users, resources and tags are removed from the dataset in order to produce a residual dataset that guarantees each user, resource and tag occur in at least p posts (Batagelj and Zaveršnik 2002) (Jaschke et al. 2007). Here we define a post to include a user, a resource, and every tag the user has applied to the resource.

The algorithm iterates through the posts, counting the occurrence of users, resources and tags. If the occurrence of these items does not meet the requisite value, p, all occurrences of the item and the posts in which it appears are removed from the dataset. Since removing a post based on the occurrence of one item reduces the count for the other items in the post, several passes through the dataset may be required. The result is a smaller denser dataset.

Several reasons exist to use P-Core Processing. By removing infrequent users, resources and tags noise in the data is reduced; uncommon items whether they be tags used by only a few users, unpopular resources, or unproductive users are eliminated from consideration. Because of their scarcity, these are the very items likely to confound recommenders. Moreover, by eliminating infrequent items, the size of the data may be dramatically reduced allowing the application of the data mining techniques that would otherwise be computationally impractical. In short, P-Core Processing offers a means reduce noise and focus on the dense regions of the data.

Hebbian Deflation

Whereas P-Core Processing offers a means to reduce the size of a dataset through feature selection, feature extraction techniques generate entirely new features. Many feature extraction techniques exist such as Principle Component Analysis and Singular Value Decomposition. However, despite the utility these techniques provide, their computational cost and memory requirements make them impractical for use on extremely large datasets common in Folksonomies. Hebbian Deflation (Oja and Karhunen 1985) offers a means to approximate these feature extraction techniques with reduced computational and memory needs.

Given a two-dimensional projection of the Folksonomy and a preselected number of features, *F*, Hebbian Deflation produces two smaller matrices that approximate the original projection. Consider for example the two-dimensional projection *RT*; Hebbian Deflation will produce two new matrixes, *RX* and *TX*, such that:

$$RT(r, t) \approx \sum_{i=0}^{F} RX_{ri} * TX_{ti} \tag{5}$$

Since *F* is often far smaller than either the number of resources or the number of tags the resultant matrixes require many orders of magnitude less space than the original projection. Analogous features can be extracted from *UR* and *UT*.

The algorithm requires many inputs: F, the number of features to be extracted; $epochs$ the number of epochs to train each feature; and lr the learning rate at which the features are adjusted.

Features are extracted one at a time. To begin, the features are set to random weights. Then, for each resource-tag pair in RT the actual weight is compared with the current approximation. The features are adjusted based upon the degree of the error, the learning rate and the corresponding feature in the opposing matrix. These adjustments are repeated for the predetermined number of epochs before the feature is finally adopted and the next feature is trained.

Input: RT, a projection of a Folksonomy; F, the number of features to derive; $epochs$, the number of epochs to train each feature; lr, the learning rate
Output: RX and TX, the Hebbian features
for $f = 1 \rightarrow features$ **do**
  for $epoch = 1 \rightarrow epochs$ **do**
    foreach $(r, t) \in RT$ **do**
      $aprox = \sum_{i=1}^{f} RX_{ri} * TX_{ti}$
      $error = RT(r, t) - approx$
      $RX(r, f)$ += $lr * error * TX(t, f)$
      $TX(t, f)$ += $lr * error * RX(r, f)$
    end
  end
end
return RX and TX;

Algorithm 1: Hebbian Deflation

Like many learning algorithms, Hebbian Deflation may result in overfitting. Overfitting occurs when the features over specialize in the training data at the expense of generalization. In order to combat this, a percentage of the training data may be held out. At each epoch, the Root Mean Square Error (RMSE), is calculated both for the ability of the features to approximate the training data and for their ability to approximate the holdout set. If it is observed that the RMSE is rising for the holdout data even as it is dropping for the training data, overfitting can be assumed. Then the training of the current feature may be halted and the training of the next feature can begin.

Feature extraction through Hebbian Deflation offers many benefits. It may offer a means to represent the data in a smaller space, but the features themselves may offer insights into the data. Domain experts can analyze features for their characteristics. Users may navigate over the reduced feature space rather than the larger Folksonomy. Users, resources and tags may be modeled as a vector over the set of features allowing reduced computation. Finally, Hebbian Deflation may reveal similarities among items in a Folksonomy that remained hidden when the data was expressed as a projection using *tf* or *tf*idf*.

Tag Recommendation in Folksonomies

Recommendation algorithms serve a vital role in Web applications allowing users to focus on a few relevant items rather than being overwhelmed by a large unordered set of mostly inappropriate options. Tag recommendation in Folksonomies reduces the user effort to a mouse click rather than a keyboard entry. This reduction in effort encourages tagging more resources, promotes the application of multiple tags to a resource, and may present the user with useful tags the user had not considered. Moreover by eliminating the keyboard entry tag recommendation reduces capitalization inconsistencies, punctuation errors, misspellings, and other discrepancies that add noise to the data.

The recommendation of high quality tags benefits the user beyond the annotation process itself. Resources are often tagged for future reference; by providing the most relevant tags retrieval of the resources are made easier. Moreover, when resources are tagged in order to characterize content, the recommendation of clear descriptive tags can improve the online experience for other users that are navigating the site. This is particularly relevant for resources that are not easily evaluated by computers such as photos, videos, and music.

Tag recommendation may also be used to assert control on tag usage without encumbering the user with a strict vocabulary. Ambiguous tags such as can be avoided in preference of less ambiguous tags Moreover, the recommender can offer tags with a higher level of detail. The overuse of redundant tags can also be thwarted by consistently recommending highly used tags while eschewing their less used counterparts. The result is cleaner denser dataset that is useful in own right for navigation, or for further data mining techniques.

Basic Recommendation

In traditional recommendation algorithms the input is often a user, u, and the output is a set of items, I. The user experience is improved if this set of items is relevant to the user's needs.

Tag recommendation in Folksonomies however differs in that the input is both a user, u, and a resource, r. The output remains a set of items, in this case a recommended set of tags, T_r. One of the difficulties presented by tag recommendation is the means to incorporate both user and resource information into the recommendation algorithm.

Perhaps the simplest recommendation strategy is merely to recommend the most commonly used tags in the Folksonomy. However such a strategy ignores both user and resource information.

Alternatively given a user-resource pair a recommender may ignore the user and recommend the most popular tags for that particular resource. This strategy is strictly resource dependent and ignores the tagging habits of the user. In a similar fashion a recommender may ignore the resource and recommend the most popular tags for that particular user. While such an algorithm would include tags frequently applied by the user, it ignores the resource information and may recommend tags irrelevant to the current resource.

An algorithm for tag recommendation in Folksonomies therefore requires a means to include both user and resource information in the process so that the recommendation set includes tags that are relevant to the resource and also represent the user's tagging practice.

K-Nearest Neighbor

User Based K-Nearest Neighbor is a commonly used recommendation algorithm in Information Retrieval that can be

modified to include both user and resource information. Traditionally it finds a set of users similar to a query user. From these neighbors a set of recommended items it constructed.

We can modify this approach by ignoring users that have not tagged the query resource. Once a neighborhood of similar users has been discovered, the algorithm considers only on those tags that have been applied to the query resource and calculates a weight for each tag, w_t, the average similarity of the neighbors that have applied the tag to the query resource. Thus the algorithm is resource driven through both the selection of neighbors and the selection of tags. Still it remains user driven in that neighbors are determined through a user model.

Input: u_q, a query user; r_q, a query resource; k, number of neighbors to consider; n, the number of tags to recommend
Output: T_r, a set of recommended tags
foreach $u \in U$ *that has annotated* r_q **do**
  s_u = similarity(u, u_q)
end
Let N be k nearest neighbors to u_q;
foreach $u \in N$ **do**
  foreach t *that* u *applied to* r_q **do**
    w_t += s_u/k
  end
end
Sort tags by w_t;
Let T_r be the top n tags;
Return T_r

Algorithm 2: K-Nearest Neighbor Modified for Folksonomies

K-Nearest Neighbor is considered a lazy algorithm; the bulk of its computation takes place after the query. Traditional approaches would require a comparison between the query user and every other user. However, since the adapted algorithm for K-Nearest Neighbor considers only those users that have annotated the query resource, the number of similarities to calculate is drastically reduced. The popularity of resources in Folksonomies follows the power law and the great majority of resources will benefit from this reduced reduction in computation, while a few will require additional computational effort. As a result the adapted K-Nearest Neighbor scales well with large datasets, a trait not shared by many other recommendation algorithms.

User Models

Applications vary in the way they model users. Possible methods include recency, authority, linkage or vector space models. In this work we focus on the vector space model (Salton, Wong, and Yang 1975) adapted from the Information Retrieval discipline to work with Folksonomies. Each user, u, can be modeled as a vector over the set of tags, where each weight, $w(t_i)$, in each dimension corresponds to the importance of a particular tag, t_i.

$$\vec{u_t} = \langle w(t_1), w(t_2)...w(t_{|T|})\rangle \tag{6}$$

In calculating the vector weights a variety of measures can be used: binary, *term frequency* or *term frequency*inverse document frequency*. In this work we focus on *term frequency*. Similarly a user can be modeled as a vector over the set of resources where each weight, $w(r_i)$, corresponds to the importance of a particular resource, r_i.

$$\vec{u_r} = \langle w(r_1), w(r_2)...w(r_{|R|})\rangle \tag{7}$$

Both of these models however ignore a portion of the user profile. A user model consisting merely of tags does not consider to which resources those tags have been applied. And a user model consisting only of resources does not include the tags applied to them.

The user model may be extended to include both tags and resources. A new vector can be obtained by concatenating the two previously mentioned vectors.

$$\vec{u_{t+r}} = \langle w(t_1)...w(t_{|T|}), w(r_1)...w(r_{|R|})\rangle \tag{8}$$

While this model does include both tags and resources, the model does not specify which tags were applied to which resources. However, the tags and resources may be tightly coupled in a vector over all tag-resource pairs where each weight, $w(tr_i)$, is one if the user has applied tag, t, to the resource, r, and zero otherwise.

$$\vec{u_{(tr)}} = \langle w(t_1r_1), w(t_1r_2)...w(t_{|T|}r_{|R|})\rangle \tag{9}$$

However these user models risk becoming exceedingly large and extremely sparse. Hebbian features may be used to combat this sparsity. For example, features extracted from either *UR* or *UT* may be used to construct the user model:

$$\vec{u_{Hebbian}} = \langle f_1, f_2...f_{|F|}\rangle \tag{10}$$

These extracted features can greatly reduce the computational costs of calculating similarities since the number of features is far smaller than the size of the original matrix. Moreover feature extraction may discover hidden relationships and identify similarities among users that the previously described models would not capture.

Several techniques exist to calculate the similarity between vectors such as Jaccard similarity or Cosine similarity (Van Rijsbergen 1979). In this work we focus on cosine similarity.

Boosting Tags

In most traditional recommendation approaches the recommendation set would not include an item the user has already used. However in Collaborative Tagging Applications users often reuse tags. Previously used tags are then an important clue for the recommendation algorithm.

We propose a boosting factor, b, that can be used to promote tags in the user profile. As an additional step to the modified K-Nearest Neighbor recommender, b is added to the weight of the tag if the user has previously applied that tag to another resource.

foreach $t \in T$ *that any* $u \in N$ *has applied to* r_q **do**
  if *(*u_q *has applied* t*)* **then**
    $w_t = w_t + b$
  end
end

Algorithm 3: Optional Step Including Boost K-Nearest Neighbor

FolkRank

In (Hotho et al. 2006) the authors proposed an adaptation of link analysis to the Folksonomy data structure. They have called this technique Folkrank since it computes a Pagerank vector from the tripartite graph induced by the Folksonomy. This graph is generated by regarding $U \cup R \cup T$ as the set of vertices. Edges are defined by the two-dimensional projections, *UT*, *UR* and *RT*.

If we regard the adjacency matrix of this graph, W, (normalized to be column-stochastic), a damping factor, d, and a preference vector, p, then we compute the Pagerank vector, w, in the usual manner:

$$w = dAw + (1 - d)p \tag{11}$$

However due to the symmetry inherent in the tripartite graph, this basic Pagerank can too easily focus on the most popular elements in the Folksonomy. The Folkrank vector is taken as a difference between two computations of Pagerank: one with a preference vector and one without the preference vector.

In order to generate tag recommendations Folkrank utilizes the preference vector to bias the algorithm towards the query user and resource(Jaschke et al. 2007). These elements are given a substantial weight in the preference vector where all other elements have uniformly small weights.

We have included this method as a benchmark as it has been shown to be an effective method of generating tag recommendations. However it has a distinct disadvantage in that it requires a complete computation of the Pagerank vector for each query. This makes the method problematic when working with data from large Folksonomies.

Experimental Evaluation

Here we describe the methods used to gather data for the experiments and provide details of our datasets. We then discuss modifications to N-Fold Cross Validation for Folksonomies and describe our experimental methodology. We briefly discus the common metrics recall and precision and then detail the results of our experiments.

Data Sets

We validate our approach through extensive evaluation of the proposed modifications using data from two real Collaborative Tagging Applications: Delicious and Citeulike.

Delicious is a popular Website in which users annotate URLs. On 10/19/2008, 198 of the most popular tags were taken from the user interface. For each of these tags the 2,000 most recent annotations including the contributors of the annotations were collected. This resulted in 99,864 distinct usernames.

For each user, the "Network" and "Fans" were explored recursively collecting additional usernames. A user's Network consists of the other users that the user has explicitly chosen to watch. Conversely a Fan is another user that has explicitly chosen to watch the user. This resulted in a total of 524,790 usernames.

From 10/20/2008 to 12/15/2008 the complete profiles of all 524,790 users were collected. Each user profile consisted of a collection of annotations including the resource, tags and date of the original bookmark. The top 100 most prolific users were visually inspected; twelve were removed from the data because their annotation count was many orders of magnitude larger than other users and were therefore suspected to be Web-bots.

Due to memory and time constraints, 10% of the user profiles was randomly selected. A P-core of 20 was derived such that each user, resource and tag appear in at least 20 posts where a post is defined as a user, resource and all tags that user applied to the resource.

The result was a dataset with 18,105 users, 42,646 resources and 13,053 tags. There are 2,309,426 annotations and 8,815,545 triples. The average number of tags in a post is 3.82.

Citeulike is a popular online tool used by researchers to manage and discover scholarly references. They make their dataset freely available to download[4]. On 2/17/2009 the most recent snapshot was downloaded with data extending back to 5/30/2007. The data contains anonymous user ids and posts for each user including resources, the date and time of the posting and the tags applied to the resource. The original dataset contains 41,689 users, 1,370,729 resource and 284,389 tags.

Because of its relatively small size and sparse data, the Citeulike data cannot support a P-core of 20. Instead a P-core of 5 was derived. This reduced the size of the data to 2,051 users, 5,376 resource and 3,343 tags. There are 42,277 annotations and 105,873 triples. The average number of tags in a post is 2.50.

Folksonomy	Delicious	Citeulike
Users	18,105	2,051
Resources	42,646	5,376
Tags	13,053	3,343
Posts	2,309,427	42,277
Annotations	8,815,545	105,873

Table 1: Datasets

An important distinction between the two datasets is their focus. Users in Delicious are able to tag any URL available on the Web. As such an individual's interests are often varied encompassing many topics. In Citeulike however researchers tag scholarly publications and their tagging is often focused in their area of expertise.

[4]http://www.citeulike.org/faq/data.adp

The data available for Delicious is also far more abundant. Using only a fraction of the data scraped from the Website, the Delicious dataset still has more than fifty times the annotations in the Citeulike dataset. Moreover, the Delicious is far denser supporting a P-core of 20 rather than a P-core of 5.

Experimental Methodologies

We implemented an extension of N-Fold Cross Validation for Folksonomies. Each user profile was divided among n folds, each fold containing approximately $1/n$ of each user's posts. A post includes the user, a resource and all tags the user applied to that resource. Models were built using $n-1$ folds of the data, while the posts in the remaining fold served as test cases.

Each test case consists of a user, u, a resource, r, and all the tags the user has applied to that resource. These tags, T_h, are analogous to the holdout set commonly used in Information Retrieval. The tag recommendation algorithms accept the user-resource pair and return an ordered set of recommended tags, T_r. From the holdout set and recommendation set utility metrics were calculated.

For each evaluation metric the average value was calculated across all test cases of an individual fold. The average was then calculated across all folds. Experiments completed on Delicious consisted of 10 folds, while experiments on Citeulike had 5 folds.

The exception to this methodology are the experiments completed for Folkrank. Due to the steep computational required for this approach only one post from each user was placed in the testing set. Experiments were then run on this single testing set as described in (Hotho et al. 2006).

Experimental Metrics

Recall is a common metric for evaluating the utility of recommendation algorithms. It measures the percentage of items in the holdout set that appear in the recommendation set. Recall is a measure of completeness and is defined as:

$$recall = |T_h \cap T_r|/|T_h| \tag{12}$$

Precision is another common metric for measuring the usefulness of recommendation algorithms. It measures the percentage of items in the recommendation set that appear in the holdout set. Precision measures the exactness of the recommendation algorithm and is defined as:

$$precision = |T_h \cap T_r|/|T_r| \tag{13}$$

Hebbian Features

A fundamental assumption of user models based upon Hebbian Deflation is that meaningful features can be extracted from the two-dimensional projections. In order to affirm this assumption we have taken the two-dimensional projection of the Delicious dataset, ***RT***, using the tag counts as the weights and performed Hebbian Deflation to generate RX and TX. A learning rate of .001 was chosen as this value allows the features to converge quickly and smoothly. Initially 100 features were built, but examination of the RMSE on a 2% overtraining-holdout set showed little improvement over 50 features. Additionally, 2000 epochs were selected to train each feature, but in all cases the algorithm halted the training when it detected overfitting and proceeded to the next feature.

Table 2 shows selected tags along with their six most similar neighbors. (Salton, Wong, and Yang 1975) has demonstrated similar results using co-occurrence, Folkrank and context metrics. Similarities are calculated by representing each tag as a vector of weights over the Hebbian features and calculating the cosine similarity between two tags.

Initial examination of the nearest tags lends credence to the assumption that Hebbian Deflation can be used to discover meaningful features. For example the nearest tags to "photo" are clearly redundant tags. However other techniques such as stemming and thesaurus tables could provide the same utility.

The example "toread" demonstrates that this method goes beyond what other techniques might provide. Where "photo" had been a descriptive tag, "toread" is an organizational tag. The ability of the Hebbian features to match it with things that will in fact be read and other organization tags illustrated the effectiveness of Hebbian Deflation.

Moreover in the example of "folksonomies" show how highly related words can be discovered through the Hebbian features. "Tags" are of course a crucial part of "folksonomies" that provide "classification" by providing "metadata."

Domain specific words like "mac" or "osx" are difficult to relate, but again Hebbian features are able to highlight their similarity. Perhaps the most intriguing example is the subjective tag "cool." Hebbian features are able to find two other subjective tags with high similarity, "interesting" and "fun".

Tags can also be modeled from features extracted from *UT*; Similarly users and resources may be modeled from features extracted from the projections. In all cases similar trends are observed.

Table 3 shows the same experiment completed with data from Citeulike. Again RT matrix was built from count data. The learning rate is set to .001. As before, 2000 epochs were selected, but RMSE testing on a holdout set halted the training of features when overfitting was detected. 100 features were extracted, but only the first 20 showed progress in reducing RMSE.

The related tags in Citeulike show similar trends to those found in Delicious. However because of its sparsity and relatively small size, it appears to be more difficult to extract features. Nevertheless visual inspection of the tags reaffirms the assumption that Hebbian Deflation and cosine similarity offer a method to discover meaningful features.

While this paper proposes using Hebbian features to model users, these features offer many other uses for Folksonomies. Hebbian features might provide a means for users to navigate the Folksonomy. After selecting a user, resource or tag the system could present additional items based upon the Hebbian features. The user may then select an item from that list and explore other users, resources or tags that are similar. By repeating this process the user could traverse

photo	**shopping**
0.789 photos	0.803 Shopping
0.737 photography	0.780 shop
0.652 pictures	0.560 buy
0.640 foto	0.530 google
0.637 Photo	0.519 store
0.627 fotos	0.469 handmade
toread	**folksonomy**
0.886 article	0.807 tagging
0.816 articles	0.786 tags
0.811 advice	0.691 tag
0.810 Bookmarks	0.635 classification
0.808 to_read	0.574 metadata
0.805 interesting	0.523 folksonomies
mac	**cool**
0.849 osx	0.711 interesting
0.840 Mac	0.673 fun
0.778 apple	0.621 imported
0.768 OSX	0.608 how-to
0.653 macosx	0.595 article
0.650 Apple	0.571 useful

Table 2: Selected Delicious tags and their nearest neighbors using cosine similarity and 50 Hebbian features extracted from the RT_c matrix

datamining	**networks**
0.981 computational	0.840 social_networks
0.935 conceptual	0.792 classification
0.933 data-mining	0.751 community
0.930 webservices	0.740 socialnetworks
0.925 tools	0.738 graph
0.915 data_mining	0.737 functional

Table 3: Selected Citeulike tags and their nearest neighbors using cosine similarity and 20 Hebbian features extracted from the RT_c matrix

the Folksonomy or focus in on a particular domain. Hebbian features might be particularly useful in this task since they are able to uncover relationships in the Folksonomy that other methods, such as co-occurrence, are unable to discover.

The individual features themselves might be interesting. A domain expert could analyze the features in an effort to understand how the Folksonomy is growing, what aspects are dominating, and which users are having the most impact.

Moreover this reduced yet rich feature space can be utilized in a variety of data mining tasks: recommendation, personalization, search, navigation, etc.

Experimental Results

Here we present our experimental results beginning with the tuning of variables. We discuss the impact of the boost variable in the quality of the K-Nearest Neighbor algorithm, then provide an in depth comparison of the recommendation

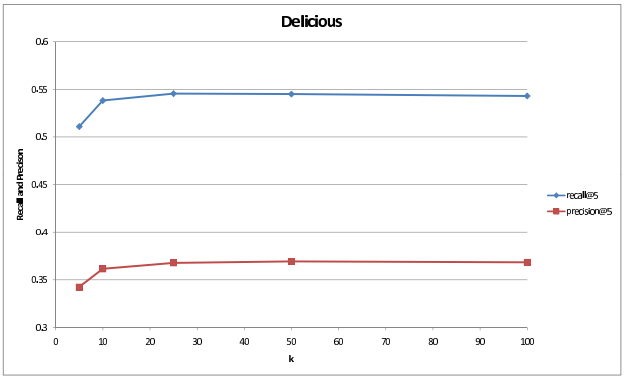

Figure 1: The effect of k in KNN on recall and precision for a recommendation set of 5 tags. Users are modeled as a vector over the tag space.

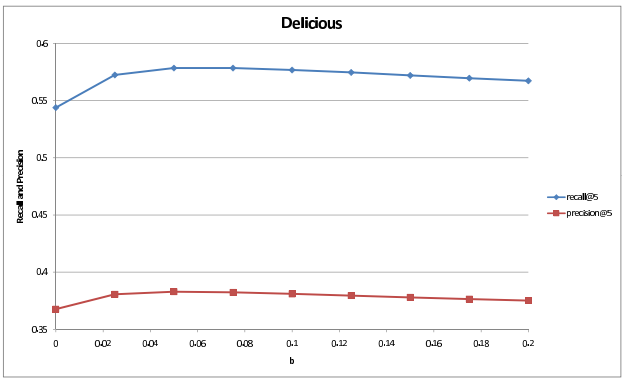

Figure 2: The effect of boosting previously used tags in KNN on recall and precision for a recommendation set of 5 tags. Users are modeled as a vector over the tag space.

techniques.

The experiments with K-Nearest Neighbor require the tuning of two key variables: k, the number of neighbors, and b, the boosting factor.

Figure 1 shows the relation between k and the the evaluation metrics recall and precision for a recommendation set of size 5. The Delicious dataset was used for this experiment. Weights in the user vector are calculated as the frequency of tags or tf. This experiment does not include any boosting factor for previously used tags. As k increases so does recall and precision. However this improvement suffers from diminishing returns until a k of 100 offers little more benefit than a k of 50. This trend was observed for all K-Nearest Neighbor experiments. As such, all K-Nearest Neighbor experiments were completed using a k of 50. Similar results are observed for the Citeulike dataset.

Figure 2 demonstrates the effectiveness of the boosting modification for K-Nearest Neighbor. The modification gives extra weight to those tags the user has previously applied to a resource. This experiment is completed with a k of 50; b is adjusted in the range of 0 through 0.20 at 0.025 increments. Both recall and precision for a recommendation

set of 5 tags are sharply improved when b in increased to 0.05. Afterward, the effect of the boosting parameter slowly diminishes.

Table 4 provides a more detailed view of the effect boosting can have. For example without any boosting factor the precision for a recommendation set of size 3 is 43.5%. With the boosting factor, precision is increased to 45.9%, an immediate 3.4% gain. For all size recommendation sets precision is increased. The boosting factors enables K-Nearest Neighbor to become more precise.

Likewise recall increases across the board. For example recall given a recommendation set of size 10 jumps from 66.9% to 69.5%, a 2.6% increase. Boosting therefore appears to increase the completeness of the recommendation set.

Delicious					
$KNN-UT, b=0.00$			$KNN-UT, b=0.05$		
N	**Rec.**	**Prec.**	**N**	**Rec.**	**Prec.**
1	0.211	0.567	1	0.232	0.606
2	0.332	0.489	2	0.361	0.519
3	0.418	0.435	3	0.450	0.459
4	0.485	0.394	4	0.516	0.413
5	0.538	0.361	5	0.568	0.376
6	0.580	0.333	6	0.609	0.345
7	0.613	0.307	7	0.641	0.317
8	0.636	0.282	8	0.664	0.291
9	0.653	0.259	9	0.682	0.268
10	0.669	0.239	10	0.695	0.248
Citeulike					
$KNN-UT, b=0.00$			$KNN-UT, b=0.05$		
1	0.201	0.404	1	0.255	0.509
2	0.292	0.309	2	0.355	0.377
3	0.346	0.252	3	0.407	0.299
4	0.383	0.213	4	0.439	0.247
5	0.407	0.184	5	0.456	0.208
6	0.427	0.162	6	0.465	0.178
7	0.444	0.145	7	0.474	0.157
8	0.457	0.131	8	0.479	0.139
9	0.467	0.120	9	0.484	0.125
10	0.474	0.110	10	0.488	0.113

Table 4: The recall and precision for the top 10 recommended tags of K-Nearest Neighbor applied to the Delicious and Citeulike datasets. Users are modeled as vectors over the tag space. Vector weights are computed as the tag frequency. k was set to 50. N is number of tags recommended. Detailed results for two different boost values, 0.00 and 0.05, are presented.

This behavior is consistent with all K-Nearest Neighbor experiments conducted on the Delicious dataset and across all user models and all values for k; boosting results in an approximate 2.0% to 4.0% increase in recall and precision. The optimum value for b is between 0.05 and 0.075. For consistency all other K-Nearest Neighbor experiments are run using a boosting factor of 0.05.

Similar results were discovered using Citeulike data. Precision for a recommendation set of size 1 jumps from 40.4% to 50.9%, a dramatic increase of 10.5%. However, this improvement diminishes as the size of the recommendation is increased. Precision for a recommendation set of 5 climbs from 18.4% to 20.8%, a 2.4% improvement. For a recommendation set of size 10, the improvement shrinks to 0.3%.

An examination of recall shows similar signs. For a recommendation set of size 1, the improvement is 5.4%. The improvement drops to 4.7% for a recommendation set of size 5 and drops further to an improvement of 1.4% when N is increased to 10.

In general boosting tags based upon previous usage is demonstrated to add additional utility to the K-Nearest Neighbor algorithm. Yet differences in the improvements between Delicious and Citeulike offer additional insights.

First the average number of tags per posts in Delicious is 3.82. Citeulike has less with 2.5 tags per post, only 65% the number found in Delicious. As a result Citeulike presents a smaller target for tag recommendation. Moreover since the holdout set is smaller for Citeulike, boosting will have a greater impact on smaller recommendation sets than on larger sets when contrasted with Delicious. This is observed as the improvement to precision garnered from boosting drops from 10.5% when the recommendation set contains 1 tag to only 0.3% when the recommendation set consists of 10 tags. Improvements in Delicious, on the other hand, are more stable dropping from 3.9% to 1.8%. An examination of recall shows a parallel trend; in both cases recall improves as the size of the recommendation set increases subject to diminishing returns. However the effect of diminishing returns hampers Citeulike recommendations earlier and more strongly than it affects Delicious. These observations suggest that care must be taken when comparing recommendation algorithms across multiple Folksonomies.

Furthermore the two datasets have a markedly different focus. Delicious users are able to annotate any URL on the Web often tagging resources across many different topics. Citeulike users on the other hand are focused on scholarly publications and often focus primarily on their area of expertise. This observation suggests that recommendation algorithms giving added weight to resources may be more appropriate for Delicious since user information may befuddle the recommendation algorithm by incorporating unrelated tags. Conversely if a user in Citeulike has annotated items related to his research and does not stray from this topic, the user profile based on tags could offer exceptional utility.

Evidence for this analysis is provided in the difference of precision and recall between the two datasets. For a recommendation set of size 1, boosting a user's previously assigned tags offers a 3.9% gain in Delicious and a 10.5% gain in Citeulike. This trend continues as N increases until the improvements gradually diminish. This stark contrast suggests that recommendation algorithms augmented by boosting tags offer more gain for focused Folksonomies such as Citeulike than broader Folksonomies such as Delicious.

Having ascertained the optimal values for k and the boosting factor we turn our attention to the user models for K-Nearest Neighbor. Experimental results are shown in Figure 3 detailing the recall and precision for recommendation

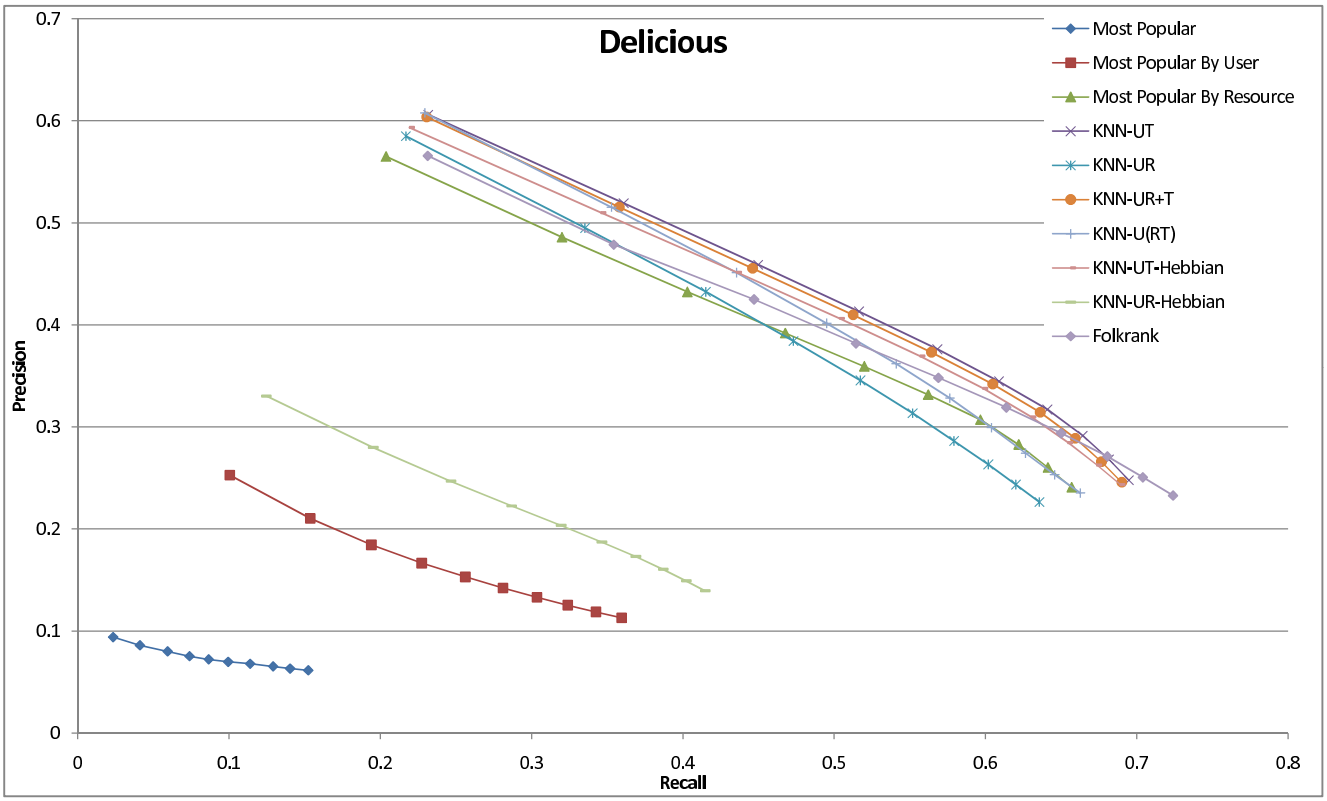

Figure 3: The comparison of tag recommender strategies for Delicious.

sets of size 1 through 10. For comparison purposes recommendation algorithms based on popularity are included. "Most Popular" recommends the most popular tags in the dataset. "Most popular by User" recommends the most popular tag for a particular user. "Most popular by Resource" recommends the most popular tags for a given resource. K-Nearest Neighbor models users as a vectors over the tag space (KNN-UT), vectors over the resource space (KNN-UR), a concatenation of the two (KNN-UR+T), a strict combination of every resource-tag pair (KNN-U(RT)) and as features derived through Hebbian deflation on either the *UT* or *UR* matrix (KNN-UT-Hebbian or KNN-UT-Hebbian). For all K-Nearest Neighbor models, k is set to 50 and the boost factor is set to .05. "Folkrank" is provided for further comparison and adapts link analysis to the folksonomy structure, recommending tags through manipulation of the preference vector.

The approach that merely recommends tags that are popular throughout the Folksonomy achieves poor results in the Delicious dataset. Recommending popular tags for a specific user fairs better, but is clearly out done by recommending popular tags given a specific resource. Nearly all user models for K-Nearest Neighbor surpass these techniques based upon popularity. In particular models that treat each user as a vector over the set of tags appear to perform best for the Delicious dataset. Folkrank offers additional completeness as seen by its superior recall, but offers less specificity as measured by it precision.

Similar trends are observed for Citeulike except for a few notable exceptions. First recommendations that rely on the popularity of a tag given a user outperform recommendations based on the popularity of a resource. This reaffirms our notion that the focused nature of Citeulike is better suited for algorithms that rely on user-tag information whereas resource-tag information is critical in broader Folksonomies where a user's annotations cover multiple topic areas.

Folkrank outperforms other methods as a measure of recall to a larger extent than in the Delicious dataset, but further trails as a measure of precision.

As in Delicious, K-Nearest Neighbor which treats users as vectors over the set of tags performs strongly in Citeulike, whereas the effectiveness of other approaches vary. The ability of this model to outperform other methods in both datasets should be noted. This is due to the inherent comprehensiveness of the *KNN-UT* algorithm.

Only those users that have tagged the query resource are considered for the neighborhood resulting in an algorithm that focuses on user-resource information. Then only those tags that have been applied to the query resource are considered for the recommendation set focusing on the resource-tag connections. Finally by treating user models as vectors over the tag space the recommender incorporates user-tag relationships. Consequently the *KNN-UT* algorithm accounts for all three aspects of the Folksonomy and is adaptable to many Folksonomies that may require an emphasis on spe-

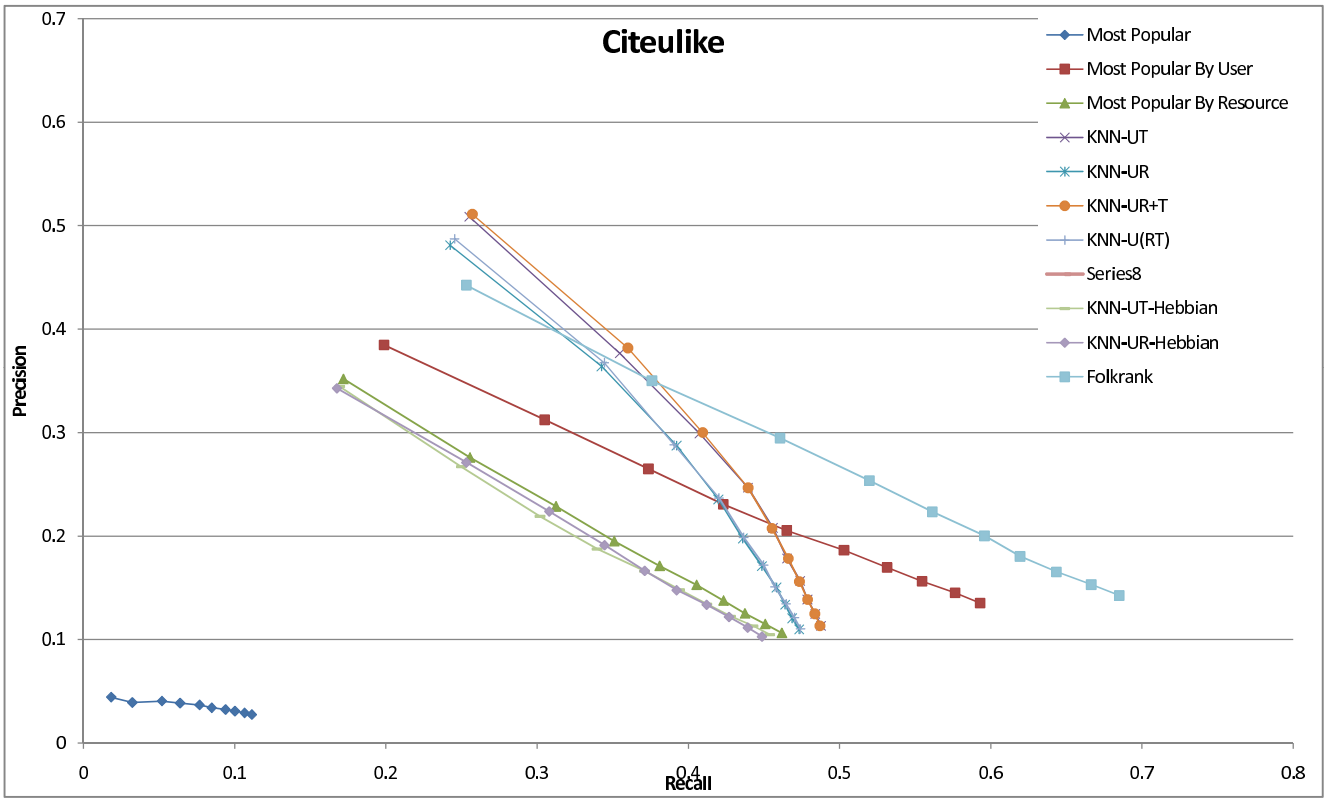

Figure 4: The comparison of tag recommender strategies for Citeulike.

cific relationships. Not surprisingly the user models that perform nearly as well as *KNN-UT* are *KNN-UR+T* and *KNN-UT-Hebbian* which share the same characteristic. Though *KNN-UT-Hebbian* does perform relatively poorly in Citeulike, likely due to the fact that this dataset is sparse and the Hebbian features are more difficult to extract.

Beyond recall and precision, time constraints should be considered when evaluating tag recommendation algorithms. Recommenders based on popularity can perform much of the computation offline thereby streamlining the recommendation process. K-Nearest Neighbor on the other hand is often referred to as a lazy algorithm since it performs the bulk of its computation during the query process. However since the proposed modifications to the algorithm limit the number of similarities that must be calculated to only those users that have tagged the query resource, the algorithm scales very well with large datasets. The computational cost may be reduced further if Hebbian features are extracted from the Folksonomy thereby reducing the length of vectors used for calculating cosine similarity. Hebbian deflation however is appropriate only when the data is dense enough that meaningful features can be extracted.

Folkrank, while it performs well in tag recommendation, is hampered by computational costs requiring a complete calculation of the Pagerank vector for each query. For example to compute 18,105 recommendations required 80 hours of computation on a modern dual-core desktop. In contrast 2,309,427 recommendations were completed in less than 1 hour using K-Nearest Neighbor.

Conclusions and Future Work

In this work we have proposed using K-Nearest Neighbor for tag recommendation in Folksonomies. Due to the unique data structure of Folksonomies, modifications are required to adapt the algorithm. Neighbors are selected only if they have tagged the query resource and tags are selected for the recommendation set only if they have been applied by the neighbor to the query resource. These modifications tie user-resource and resource-tag information into the algorithm while it dramatically reduces the computational costs. There exists a myriad of ways in which to calculate user similarity; We have found that cosine similarity between users modeled as vectors over the tag space performs well. This model incorporates user-tag information into the algorithm. By including all three relationships inherent in Folksonomies, the algorithm is robust for both broad and narrow Folksonomies. In addition, K-Nearest Neighbor can be improved by boosting tags the user has previously used. The performance of K-Nearest Neighbor exceeds that of recommendation algorithms based on popularity, while the running time makes it computationally viable for large real world Folksonomies.

In the future we plan to investigate alternative tag recommendation strategies and study resource or user recommendation algorithms. Other approaches such as association rules mining and neural networks are worth considering

for recommendation in Folksonomies. Probabilistic Latent Semantic Analysis offers an alternative means to derive features from the Folksonomy. Feature extraction of any sort presents intriguing opportunities in search, navigation, personalization and recommendation.

Acknowledgments

This work was supported in part by the National Science Foundation Cyber Trust program under Grant IIS-0430303 and a grant from the Department of Education, Graduate Assistance in the Area of National Need, P200A070536.

References

Adrian, B.; Sauermann, L.; and Roth-Berghofer, T. 2007. Contag: A semantic tag recommendation system. In Pellegrini, T., and Schaffert, S., eds., *Proceedings of I-Semantics' 07*, pp. 297–304. JUCS.

Basile, P.; Gendarmi, D.; Lanubile, F.; and Semeraro, G. 2007. Recommending smart tags in a social bookmarking system. In *Bridging the Gep between Semantic Web and Web 2.0 (SemNet 2007)*, 22–29.

Batagelj, V., and Zaveršnik, M. 2002. Generalized cores. *Arxiv preprint cs/0202039*.

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

Heymann, P.; Ramage, D.; and Garcia-Molina, H. 2008. Social tag prediction. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 531–538. New York, NY, USA: ACM.

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

Jaschke, R.; Marinho, L.; Hotho, A.; Schmidt-Thieme, L.; and Stumme, G. 2007. Tag Recommendations in Folksonomies. *LECTURE NOTES IN COMPUTER SCIENCE* 4702:506.

Lipczak, M. 2008. Tag recommendation for folksonomies oriented towards individual users. In *Proceedings of the ECML/PKDD 2008 Discovery Challenge Workshop, part of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*.

Macgregor, G., and McCulloch, E. 2006. Collaborative tagging as a knowledge organisation and resource discovery tool. *Library Review* 55(5):291–300.

Mathes, A. 2004. Folksonomies-Cooperative Classification and Communication Through Shared Metadata. *Computer Mediated Communication, (Doctoral Seminar), Graduate School of Library and Information Science, University of Illinois Urbana-Champaign, December*.

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.

Nakamoto, R. Y.; Nakajima, S.; Miyazaki, J.; Uemura, S.; Kato, H.; and Inagaki, Y. 2008a. Reasonable tag-based collaborative filtering for social tagging systems. In *WICOW '08: Proceeding of the 2nd ACM workshop on Information credibility on the web*, 11–18. New York, NY, USA: ACM.

Nakamoto, R. Y.; Nakajima, S.; Miyazaki, J.; Uemura, S.; and Kato, H. 2008b. Investigation of the effectiveness of tag-based contextual collaborative filtering in website recommendation. In *Advances in Communication Systems and Electrical Engineering*, 309–318. Springerlink.

Oja, E., and Karhunen, J. 1985. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications* 106(1):69–84.

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

Salton, G.; Wong, A.; and Yang, C. 1975. A vector space model for automatic indexing. *Communications of the ACM* 18(11):613–620.

Schmitz, C.; Hotho, A.; Jaschke, R.; and Stumme, G. 2006. Mining association rules in folksonomies. In *Proc. IFCS 2006 Conference*, 261–270. Springer.

Sigurbjörnsson, B., and van Zwol, R. 2008. Flickr tag recommendation based on collective knowledge. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, 327–336. New York, NY, USA: ACM.

Song, Y.; Zhuang, Z.; Li, H.; Zhao, Q.; Li, J.; Lee, W.-C.; and Giles, C. L. 2008. Real-time automatic tag recommendation. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 515–522. New York, NY, USA: ACM.

Tso-Sutter, K. H. L.; Marinho, L. B.; and Schmidt-Thieme, L. 2008. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, 1995–1999. New York, NY, USA: ACM.

Van Rijsbergen, C. 1979. *Information Retrieval*. Butterworth-Heinemann Newton, MA, USA.

Xu, Z.; Fu, Y.; Mao, J.; and Su, D. 2006. Towards the semantic web: Collaborative tag suggestions. *Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland, May*.