ISSN:0975-9646

Users Recommended Tag Prediction and Visualization for Dynamic Delicious Tag Data

Jameson Ganta[#1], G. Krishna Kishore[*2]

[#]*Post graduate student,*[*]*Assistant Professor*
V.R. Siddhartha engineering college

ABSTRACT- **Social bookmarking tools enable users to save URLs for future reference, to create tags for annotating Web pages, and to share Web pages they found interesting with others. This project presents a case study on the application of link mining to a social bookmarking Web site called del.icio.us. User feedback is obtained by iteratively presenting a set of suggested items, and users selecting items based on their own preferences either from this suggestion set or from the set of all possible items. This proposed system investigates the user bookmarking and tagging behaviors by applying robust Naïve bayes, Association rule mining techniques to find surprising patterns in the data. This study mainly focuses on predicting popular rules than others. Finally, we demonstrated the effectiveness of using data mining algorithms for predicting effective tags rules behavior for delicious web data.**

I. INTRODUCTION

With the staggering rate at which new content is produced on the Internet, it is becoming increasingly difficult for Web users to keep up to date with the new information. Social bookmarking is a tool that enables Web users to share information they found on the Internet with other users sharing similar interests. It allows users to save and organize their bookmarks on a remote Web server. Users may assign tags to each bookmark to annotate what they perceive to be the content[1]. Some of the popular social bookmarking Websites include del.icio.us, www.citeulike.org, and www.furl.net. Social bookmarking is a rich but largely unexplored domain in link mining. Many applications such as Web search and text categorization may benefit from the analysis of social bookmarking data. For example, the number of users who bookmarked a Web page can be used as a metric to measure the authoritativeness of the Web page. The tags used to annotate the bookmarks is another useful data source that can be harnessed to improve Web search or Web page classification . A social bookmarkingWeb site is also a fertile testbed to investigate many social science phenomena such as information diffusion and social selection. One approach involves measuring the deviation of a pattern from its expected frequency while the other is based on performing a temporal analysis on the user's bookmarking history. We then examined characteristics that influence the popularity of a user. At del.icio.us, a user becomes a fan of another user by adding the other user to his/her network.The number of fans each user has can be used as a measure of user popularity. Because popular users may influence the bookmarking activities of other users , it is useful to understand what makes a user more popular than others. Finally, we studied the linking behavior of users at the social bookmarking Web site. Specifically, we consider links in the form of reciprocal ties, which are pairs of users who are mutual fans of each other. Our goal is to predict the formation of such ties based on the user bookmarking and tagging activities.

We define the popularity of bookmarks, users, and tags in the following way.

1)Bookmark Popularity: The number of users who saved a given bookmark.

2)User Popularity: The number of fans associated with a user.

3)Tag Popularity: The number of posts that contain a given tag[3].

Our tag popularity measure is somewhat different than the one used by del.icio.us, which is based on the number of unique bookmarks associated with a given tag. There are two advantages for using our measure. First, it is more informative because it takes into account both the number of bookmarks and number of users who have used the tag. Second, our measure is more resilient to spam tags, which are the tags used to mislead users about the content of aWeb page or to attract users to a spamWeb site. If tag popularity is measured using the number of bookmarks only (instead of number of posts), it would be easier to promote a spam tag into a popular tag..

Motivation:

Most classification algorithms perform batch classification, that is, they work on the dataset directly in memory. More complex methods often build large data structures which give them a larger memory footprint. This larger footprint prevents them from being applied to many of the larger datasets. Even when these datasets can be used, the complexity of the algorithm can make the classification task take an inordinately long time. There are solutions to these problems in the form of adding more memory or waiting longer for experiments to finish. However both have a cost in money and time and both may not ultimately solve the problem if the dataset cannot in memory when the memory is at a maximum. A feature of most large datasets is the large number of data points that contain similar information. So one solution is to remove redundant information from the dataset to allow a classifier to concentrate on data points that represent a larger group of points[4]. This allows the construction of a classier that correctly models the relationship expressed by the data in the dataset. Random sampling is a simple way to remove redundancy and can be very effective on uniformly distributed data.

II. RELATED WORK

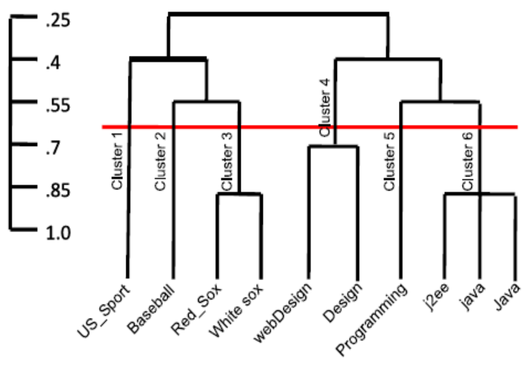
A typical Flickr provides a variety of information about the photo: who uploaded it and when, what groups it has been submitted to, its tags, who commented on the image and when, how many times the image was viewed or bookmarked as a “favorite”. Clicking on a user’s name brings one to their photo stream, which shows the latest photos they have uploaded, the images they have marked as their “favorite,” and their profile, which gives information about the user, which includes a list of their contacts and the groups they belong to. Clicking on the tag shows user’s images that have been tagged with this keyword, or all public images that have been similarly tagged. Finally, the group link brings the user to the group’s page, which shows the photo pool, group membership, popular tags, discussions and other information about the group.

We used the Flickr API to download a variety of data for our study. For the data not provided through the API (for example, the number of views), we wrote specialized data scrapers to extract this information from the Web pages. Since scraping required a separate HTTP request, this had an effect on the image statistics (e.g., number of views is incremented by every HTTP request). We corrected for this effect in post-processing[2].

```
Database D of transactions, specific Web site
Output: percentage of usage of particular site
Method
1. Accept specific site
2. count = 0;
3. for ( int i= 0; i<= D.size;i++)
4. if(D_site = = accept_site)
5. count ++
6. percentage= (count/D_site)*100
7. return percentage //percentage of user from a particular site
upon total web log record
```

```
Find a particular site usage
Input: Database D of transactions, Specific Web Site
Output: Users’ IP, Date and Time list
Method
1. Accept a specific Web Site
2. count = 0;
3. for (int i=0; i<= D.size; i++)
4. while (accept_site = = D_size)
5. extract user_IP from D
6. extract corresponding date and time
7. return  list of user_IP, date and time
```

Hierarchical Agglomerative Clustering The input to the clustering module is a set of tags, T, the step and division coefficient. Tags are represented as a vector of weights over the set of resources, R. In these experiments we used both term frequency and tfidf (Equation 5). The parameter step controls the granularity of the derived agglomerative clusters. The similarity threshold, initially set to one, is reduced by the value step at each iteration until it reaches zero. Clusters of tags are aggregated together if their similarity measure meets the current threshold. The division coefficient also plays a crucial role in the agglomerative clustering routine. It defines the level where the hierarchy is dissected into individual clusters. The output of the system is a set of tag clusters.



Hierarchical agglomerative clustering technique

Ref [5]

Step 1. Assign every tag to a singleton cluster. To begin every tag in the system is placed in its own cluster, which serves as the seeds for the agglomerative clustering.

Step 2. Combine all tags in one hierarchical cluster. In this stage of the algorithm clusters of tags are aggregated together based on their similarity. Highly similar tags are aggregated together first, then less similar ones are combined. The result is a tree structure containing every tag in a hierarchy.

Step 2.1 Combine clusters. Clusters meeting the current similarity threshold are joined. There are many ways of determining the distance between clusters: single link, maximal link, average link, etc. In this paper, we determined the distance between clusters by using the centroids of the clusters.

Step 2.2 Lower similarity. The value for the similarity threshold is lower by step. The algorithm then repeated
step 2.1 until all clusters have been aggregated.

Tag frequency

Tags are used with varying frequencies in a social bookmarking system. It can be useful to evaluate the frequency of tags in order to investigate how particular tags are being used across time and what is the probability that they will be used again.

Tags per Post (TPP)

As a Describer would focus on describing her resources in a very detailed manner, the number of tags used to annotate each resource can be taken into account as an indicator to identify the motivation of the analyzed user. The tags per post measure (short TPP) captures this by dividing the number of all tag assignments of a user by the number of resources (see Equation 1). Tur is the number of tags annotated by user u on resource r, and Ru is the number of resources of a user u. The more tags a user utilizes to annotate the resources the more likely she is a Describer and this would reect in a higher TPP score.

$$TPP(u) = \frac{\sum^{r} |T_{ur}|}{|R_u|}$$

This measure relies on the verbosity of users, as it computes the average number of tags they assigned to bookmarks.

Tag Resource Ratio (TRR)

The tag resource ratio (short TRR) relates the number of tags of a user (i.e., the size of her vocabulary) to the total number of annotated resources (see Equation 4). A typical Categorizer would apply only a small number of tags to her resources and therefore score a low number on this measure.

$$TRR(u) = \frac{|T_u|}{|R_u|}$$

III. PROPOSED ARCHITECTURE

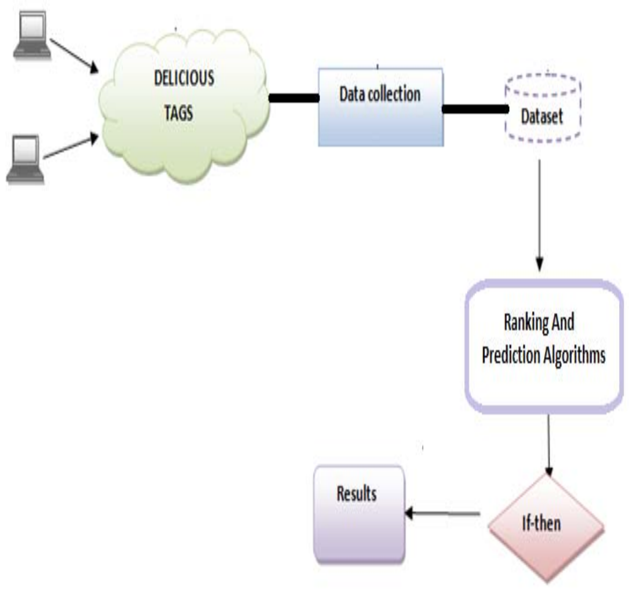


Figure1: Proposed Architecture

A. *Delicious Data*

In this paper two types of file formats are used. They are

i) CSV

ii) ARFF

i. **CSV**: It stands for Comma Separated Value. This format is obtained using MS-Excel. Spam dataset is loaded into Excel and then it is saved with an extension of csv.

ii. **ARFF**: It stands for Attribute Relation File Format. A file is an ASCII text file that describes a list of instances sharing a set of attributes. ARFF files were developed by the Machine Learning Project at the Department of Computer Science of The University of Waikato for use with the Weka machine learning softwareARFF files have two distinct sections. The first section is the Header information, which is followed the Data in The ARFF Header Section the ARFF Header section of the file contains the relation declaration and attributes declarations.

B. *The @relation Declaration*

The relation name is defined as the first line in the ARFF file. The format is:

@relation <relation-name>

where <relation-name> is a string. The string must be quoted if the name includes spaces.

C. *The @attribute Declarations*

Attribute declarations take the form of an ordered sequence of @attribute statements. Each attribute in the data set has its own @attribute statement which uniquely defines the name of that attribute and its data type[6].

D. *Delicious Tag Data*

We consider the situation where users select items from a given set

c:={1,2,3…..c} where c > 1.

We let $r = (r_1, r_2, \ldots, r_c)$ be the users' true preference over the set of items C and call r the true popularity rank scores. For an item i, we interpret ri as the portion of users that would select item I if suggestions were not made[5]. We assume that the true popularity rank scores r are such that:

1) ri is strictly positive for each item i,

2) items are enumerated such that r1>=r2>=…>=rc and

3) r is normalized such that it follows probability distribution.

First, we consider the naive algorithm which suggests a fixed number of the topmost popular items and show that this algorithm can fail to learn the true popularity ranking of items if the imitation probability in the user's choice model is sufficiently large. In particular, we find that there exists a threshold on the imitation probability below which the algorithm guarantees to learn the true popularity ranking of items, and otherwise, this may not hold. We fully specify this threshold in terms of the suggestion set size and the true popularity ranks of items[2]. This result enables us to estimate the threshold imitation probability for a given true popularity ranking. In particular, we provide estimates for the threshold using our data set of tags applied to popular Web pages in del.icio.us and found threshold to be typically around 0.1 for the suggestion set sizes ranging from 1 to 10 tags. This suggests that in real-world scenarios, using the above simple scheme may result in failing to learn the true popularity of items at small imitation rates.

E. *A Naive Algorithm*

We first introduce the simple algorithm TOP which consists of a ranking and a suggestion rule as defined below.

```
TOP ( TOP POPULAR)
Init Vi=0 for each item i
At the t-th item selection:
If item i selected:
Vi← Vi+1
S← a set of s items with largest V counts
```

The ranking rule is to set the rank score of an item equal to the number of selections of this item in the past. For this algorithm and the algorithms introduced later, we initialize Vi <0 for each item i.

Rank rule 1: A simple ranking rule is the one that we already encountered in the algorithm TOP, where the rank score for an item i is incremented by 1 whenever a user selects this item.

Tagid	Linkid	Date	Tagname	TagClass
1	16693	19:26.5	Portland	yes
2	16693	19:26.6	Maine	no
3	16693	19:26.6	meetup	yes
4	16167	02:17.1	mefi	yes
5	16167	02:17.1	deleted	no

```
Init: Vi =0 for each item i
At the t-th item selection:
If item I selected:
Vi← Vi +1
Pi ← Vi/t
```

We will see that this ranking may fail to discover the ranking order of the true popularity when combined with a suggestion rule that reinforces items that were selected early on, as it is the case under TOP.

Rank rule 2:
We noted that rank rule 1 may fail to discover the ranking order of the true popularity if used with suggestion rules such as TOP[7]. To overcome this problem, we may redefine the rank scores in the following way.

```
Init:  Ti =0,Vi=0,  for each item i
At the t-th item selection:
For each item i:
If item i not suggested:
Ti← Ti +1
If item i selected:
Vi← Vi +1
Pi ← Vi/ti
```

F. Bayesian Classification

Bayesian classifiers are statistical classifiers. They can predict class membership probabilities, such as the probability that a given tuple belongs to a particular class. Bayesian classification is based on Bayes theorem. Naïve Bayesian classifiers assume that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called class conditional independence. It is made to simplify the computations involved and, in this sense, is considered "naïve." Bayesian belief networks are graphical models, which unlike naïve Bayesian classifiers, allow the representation of dependencies among subsets of attributes. Bayesian belief networks can also be used for classification.

G. Bayes' Theorem

Bayes' theorem is named after Thomas Bayes, a nonconformist English clergyman who did early work in probability and decision theory during the 18th century. Let X be a data tuple. In Bayesian terms, X is considered "evidence." As usual, it is described by measurements made on a set of n attributes. Let H be some hypothesis, such as that the data tuple X belongs to a specified class C. For classification problems, we want to determine P(H/X), the probability that the hypothesis H holds given the "evidence" or observed data tuple X. In other words, we are looking for the probability that tuple X belongs to class C, given that we know the attribute description of X. P(H/X) is the posterior probability, or a posteriori probability, of H conditioned on X. For example, suppose our world of data tuples is confined to customers described by the attributes age and income, respectively, and that X is a 35-year-old customer with an income of $40,000. Suppose that H is the hypothesis that our customer will buy a computer. Then P(H/X) reflects the probability that customer X will buy a computer given that we know the customer's age and income. In contrast, P(H) is the prior probability, or a priori probability, of H. For our example, this is the probability that any given customer will buy a computer, regardless of age, income, or any other information, for that matter. The posterior probability, P(H/X), is based on more information (e.g., customer information) than the prior probability, P(H), which is independent of X. Similarly, P(X/H) is the posterior probability of X conditioned on H. That is, it is the probability that a customer, X, is 35 years old and earns $40,000, given that we know the customer will buy a computer[8][9]. P(X) is the prior probability ofX.Using our example, it is the probability that a person from our set of customers is 35 years old and earns $40,000. P(X/H), and P(X) may be estimated from the given data, as we shall see below. Bayes' theorem is useful in that it provides a way of calculating the posterior probability, P(H/X), from P(H), P(X/H), and P(X). Bayes' theorem is

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}.$$

H. Naïve Bayesian Classification

The naïve Bayesian classifier, or simple Bayesian classifier, works as follows:

1. Let D be a training set of tuples and their associated class labels. As usual, each tuple is represented by an n-dimensional attribute vector, X = (x1, x2, : : : , xn), depicting n measurements made on the tuple from n attributes, respectively, A1, A2, : : : , An.
2. Suppose that there are m classes, C1, C2, : : : , Cm. Given a tuple, X, the classifier will predict that X belongs to the class having the highest posterior probability, conditioned on X. That is, the naïve Bayesian classifier predicts that tuple X belongs to the class Ci if and only if P(Ci/X) > P(Cj/X). Thus we maximize P(Ci/X). The classCi for which P(Ci/X) is maximized is called the maximum posteriori hypothesis. By Bayes' theorem

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}.$$

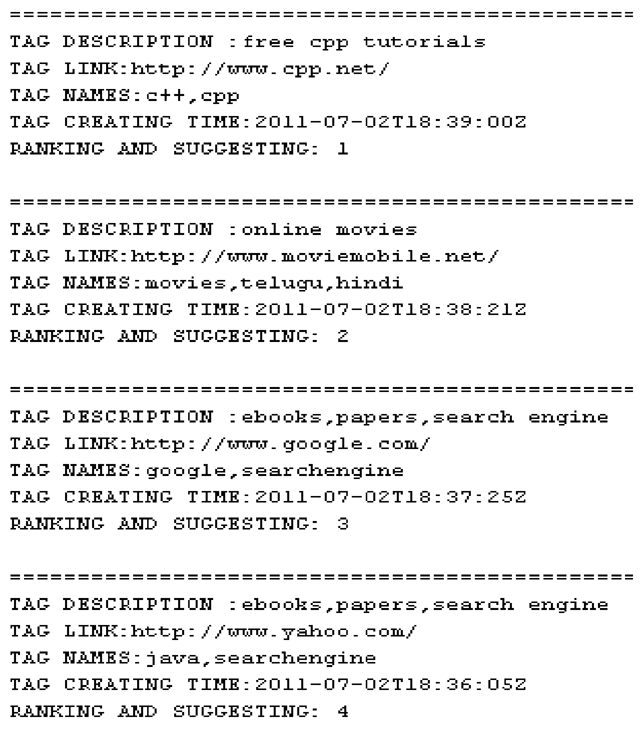
3. As P(X) is constant for all classes, only P(X/Ci)P(Ci) need be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is, P(C1) = P(C2) = …….. = P(Cm). Given data sets with many attributes, it would be extremely computationally expensive to compute P(X/Ci). In order to reduce computation in evaluating P(X/Ci), the naive assumption of class conditional independence is made. This presumes that the values of the attributes are conditionally independent of one another, given the class label of the tuple (i.e., that there are no dependence relationships among the attributes). Thus,

$$\begin{aligned} P(X|C_i) &= \prod_{k=1}^{n} P(x_k|C_i) \\ &= P(x_1|C_i) \times P(x_2|C_i) \times \cdots \times P(x_n|C_i). \end{aligned}$$

We can easily estimate the probabilities P(x1/Ci), P(x2/Ci), : : : , P(xn/Ci) fromthe training tuples. Recall that here xk refers to the value of attribute Ak for tuple X. For each attribute, we look at whether the attribute is categorical or continuous-valued.

IV. EXPERIMENTAL RESULTS

Following results gives the appropriate prediction has to be done on the delicious tag web site dynamically. Ranking algorithms gives the best results while classifying the tag dataset with the multiple link scenario.

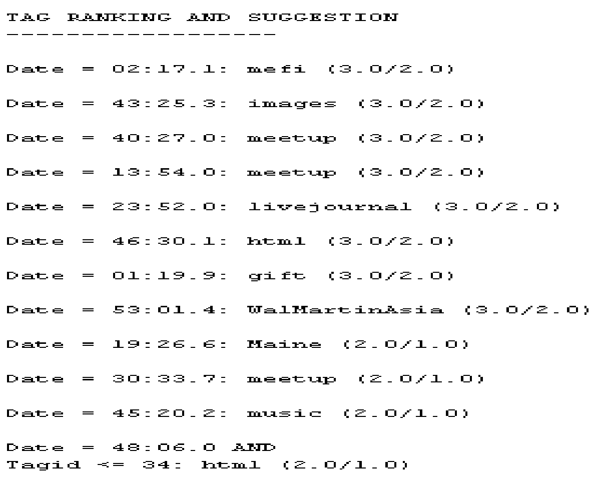


```
==============================================
TAG DESCRIPTION :free cpp tutorials
TAG LINK:http://www.cpp.net/
TAG NAMES:c++,cpp
TAG CREATING TIME:2011-07-02T18:39:00Z
RANKING AND SUGGESTING: 1

==============================================
TAG DESCRIPTION :online movies
TAG LINK:http://www.moviemobile.net/
TAG NAMES:movies,telugu,hindi
TAG CREATING TIME:2011-07-02T18:38:21Z
RANKING AND SUGGESTING: 2

==============================================
TAG DESCRIPTION :ebooks,papers,search engine
TAG LINK:http://www.google.com/
TAG NAMES:google,searchengine
TAG CREATING TIME:2011-07-02T18:37:25Z
RANKING AND SUGGESTING: 3

==============================================
TAG DESCRIPTION :ebooks,papers,search engine
TAG LINK:http://www.yahoo.com/
TAG NAMES:java,searchengine
TAG CREATING TIME:2011-07-02T18:36:05Z
RANKING AND SUGGESTING: 4
```



```
TAG RANKING AND SUGGESTION
------------------

Date = 02:17.1: mefi (3.0/2.0)

Date = 43:25.3: images (3.0/2.0)

Date = 40:27.0: meetup (3.0/2.0)

Date = 13:54.0: meetup (3.0/2.0)

Date = 23:52.0: livejournal (3.0/2.0)

Date = 46:30.1: html (3.0/2.0)

Date = 01:19.9: gift (3.0/2.0)

Date = 53:01.4: WalMartinAsia (3.0/2.0)

Date = 19:26.6: Maine (2.0/1.0)

Date = 30:33.7: meetup (2.0/1.0)

Date = 45:20.2: music (2.0/1.0)

Date = 48:06.0 AND
Tagid <= 34: html (2.0/1.0)
```

V. CONCLUSION

In this paper, we briefly describe an approach utilizes supervised ranking model for tag recommendations. Our tag prediction contains three steps. First, Dynamically delicious tags are added to the website through our application.. Second, features are decided according to categories. Then we rank the candidate tags, using the supervised ranking model, and pick the top tags as recommendation tags. The experiment results show that our tag prediction model is able to predict a considerably large portion of the stabilized tag set with less error rate.

REFERENCES

[1] Link Mining for a Social BookmarkingWeb Site Feilong Chen, Jerry Scripps, Pang-Ning Tan.

[2] Social Tag Prediction Paul Heymann, Daniel Ramage, and Hector Garcia-Molina Department of Computer Science.

[3] Evaluating Tagging Behavior in Social Bookmarking Systems: Metrics and design heuristics Umer Farooq1, Thomas G. Kannampallil1, Yang Song2, Craig H. Ganoe1, John M. Carroll1, and C. Lee Giles2.

[4] B. D. Davison. Topical locality in the web. In SIGIR, 2000.

[5] R. Fagin. Combining fuzzy information from multiple systems. J. Comput. Syst. Sci., 58(1), 1999.

[6] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. J. Comput. Syst. Sci., 66(4), 2003.

[7] S. Golder and B. A. Huberman. Usage patterns of collaborative tagging systems. Journal of Information Science, 32(2), 2006.

[8] H. Halpin, V. Robu, and H. Shepherd. The complex dynamics of collaborative tagging. In WWW, 2007.

[9] P. Heymann, G. Koutrika, and H. Garcia-Molina. Can social bookmarking improve web search? In WSDM, 2008.