Real-time Automatic Tag Recommendation

Yang Song[1], Ziming Zhuang[4], Huajing Li[1], Qiankun Zhao[5]
Jia Li[3], Wang-Chien Lee[1], C. Lee Giles[2]

[1]Computer Science and Engineering,
[2]Information Sciences and Technology,
[3]Department of Statistics,
The Pennsylvania State University,
University Park, PA 16802, USA

[4]Yahoo! Applied Research,
2821 Mission College Blvd,
Santa Clara, CA 95054, USA

[5]AOL Research Lab,
Beijing, China

ABSTRACT

Tags are user-generated labels for entities. Existing research on tag recommendation either focuses on improving its accuracy or on automating the process, while ignoring the efficiency issue. We propose a highly-automated novel framework for real-time tag recommendation. The tagged training documents are treated as triplets of (words, docs, tags), and represented in two bipartite graphs, which are partitioned into clusters by Spectral Recursive Embedding (SRE). Tags in each topical cluster are ranked by our novel ranking algorithm. A two-way Poisson Mixture Model (PMM) is proposed to model the document distribution into mixture components within each cluster and aggregate words into word clusters simultaneously. A new document is classified by the mixture model based on its posterior probabilities so that tags are recommended according to their ranks. Experiments on large-scale tagging datasets of scientific documents (CiteULike) and web pages (del.icio.us) indicate that our framework is capable of making tag recommendation efficiently and effectively. The average tagging time for testing a document is around 1 second, with over 88% test documents correctly labeled with the top nine tags we suggested.

Categories and Subject Descriptors

I.5.3 [**Pattern Recognition**]: Clustering—*algorithms*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation, Performance

Keywords

tagging system, mixture model, graph partitioning

SIGIR'08, July 20–24, 2008, Singapore.

1. INTRODUCTION

Tagging usually refers to the action of associating a relevant keyword or phrase with an entity (e.g. document, image, or video). With the recent proliferation of Web 2.0 applications such as Del.icio.us[1] and Flickr[2], tagging services have become popular and have drawn much attention in both academia and industry. Existing research on tagging services includes improving the quality of searching and recommendation in the tag space [2], analyzing the usage patterns of tagging systems [11], and automating the process of tag assignment [4].

Here, we address the problem of automatic tag recommendation for document search engines and digital libraries, which bears some similarities to that of query recommendation [1]. However, our problem space is arguably larger, because relevant candidate tags may not even appear in the document, while candidate queries are most likely bounded in the document term space in keyword-based search.

While automatic tag recommendation is an actively pursued research topic, to the best of our knowledge, we are the first to study in depth the problem of automatic *and* real-time tag recommendation, and propose a solution with promising performance when evaluated on two real-world tagging datasets, i.e., CiteULike[3] for scientific documents and del.icio.us for web pages.

Specifically, we advocate a two-state framework. First, the relationship among documents, tags, and words are represented in two bipartite graphs. During the offline learning stage, we use the Lanczos algorithm for *symmetric* low rank approximation for the weighted adjacency matrix for the bipartite graphs, and Spectral Recursive Embedding (SRE) to symmetrically partition the graphs into multi-class clusters. We propose a novel node ranking algorithm to rank nodes (tags) within each cluster, and then apply a Poisson mixture model to learn the document distributions for each class.

During the online recommendation stage, given a document vector, its posterior probabilities of classes are first calculated. Then based on the joint probabilities of the tags and the document, tags are recommended for this document based on their within-cluster ranking. The efficiency of the Poisson mixture model lets our model make recommendations in linear-time in practice. As an example, using reason-

[1]http://del.icio.us/
[2]http://www.flickr.com/
[3]http://www.citeulike.org/

able resources, the average tagging time for a test document is only 1.1 seconds.

It should be noted that tag suggestion is still a complicated problem that can be addresses in many aspects. e.g., examine the tag growth and reuse by user study [8]. In this paper, we address this issue from a machine learning perspective by analyzing the content of tags. We believe this approach could be a useful component that can be combined with other powerful tools to boost the performance of real-world tagging systems.

The rest of the paper is organized as follows. We briefly review the related work in Section 2. Section 3 introduces bipartite graphs, the Lanczos algorithm, and the graph partitioning algorithm as well as the node ranking method. Section 4 presents the mixture model for document classification and a online tag recommendation algorithm. Section 5 presents the experimental results on two data sets. Section 6 concludes our work.

2. RELATED WORK

Bipartite Graph Partitioning: A bipartite graph consists of two disjoint sets of vertices X and Y such that no edge has both end points in the same set. The general graph partitioning problem is NP hard so that for bipartite graphs, partitioning is optimized by minimizing a global function. Many clustering algorithms have been proposed to partition bipartite graphs. The Min-Max Cut algorithm minimizes between-cluster sum of weights and maximizes the within-cluster sum of weights [6]. The spectral clustering simultaneously clusters rows and columns of adjacency matrix of the graph [5]. However, as pointed out in [10], spectral clustering may fail in certain cases where two bipartite graphs are merged to be one tripartite graph due to the heterogeneous nature of the vertices. To address this problem, the algorithm, Consistent Bipartite Graph Co-partitioning (CBGC), is proposed [10]. CBGC applies semi-definite programming (SDP) to deal with star-structured high order heterogeneous data by representing them as several bipartite graphs, and optimizes a global function to find the best cut. However, CBGC only deals with binary clustering problems and is thus not suitable for multi-clustering tasks.

Low Rank Matrix Approximation: Low rank matrix approximation is the problem of approximating a $m \times n$ matrix A by another rank k matrix, where k is smaller than m and n. Traditional methods like Singular Value Decomposition (SVD) can be used to find such matrix but the computation time is usually too long ($O(min\{mn^2, nm^2\})$).

Recently, *near-optimal* low rank matrix approximation methods have become increasing popularity. If we denote A_k as an optimal rank k approximation of matrix A, the goal is to find a near-optimal matrix A_k^* that minimizes the error ϵ:

$$\|A - A_k^*\| \leq \|A - A_k\| + \epsilon \quad (1)$$

CUR [7] decomposition is one such algorithm that approximates A by $A = CUR$, where C is a matrix consisting of a small number of columns of A, R is a matrix consisting of a small number of rows of A, and U is an appropriately-defined low-dimensional encoding matrix. Thus, a CUR matrix decomposition provides a dimensionally-reduced low-rank approximation to the original data matrix A that is expressed in terms of a small number of actual columns and rows of the original matrix. Both linear and constant time CUR algorithms have been proposed to efficiently approximate large sparse matrices. However, since the rows and columns are picked randomly, CUR can not guarantee the symmetry of a matrix, which makes it not suitable for bipartite graphs since the weight matrices are always symmetric.

3. BIPARTITE GRAPH REPRESENTATION

We define a graph $G = (V, E, W)$ as a set of vertices V and their corresponding edges E, with W denoting the weight of edges. e.g., w_{ij} denotes the weight of the edge between vertices i and j.

A graph G is *bipartite* if it contains two vertex classes X and Y such that $V = X \cup Y$ and $X \cap Y = \emptyset$, each edge $e_{ij} \in E$ has one endpoint (i) in X and the other endpoint (j) in Y. In practice, X and Y usually refer to different types of objects and E represents the relationship between them. In the context of document representation, X represents a set of documents while Y represents a set of terms, and w_{ij} denotes the number of times term j appears in document i. Note that the weighted adjacency matrix W for a bipartite graph is always symmetric.For example, Figure 1 depicts an undirected bipartite graph with 4 documents and 5 terms.

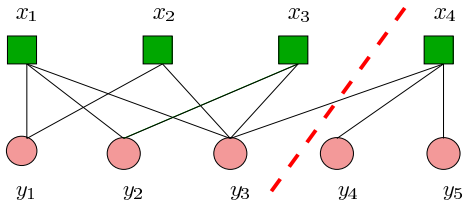


Figure 1: A bipartite graph of X (documents) and Y (terms).

3.1 Normalization and Approximation

Normalization is usually performed first for the weight matrix W to eliminate the bias. The most straightforward way to normalize W is row normalization, which does not take into account the symmetry of W. However, to consider the symmetry of W, we propose to use normalized graph Laplacian to approximate W. The normalized Laplacian $L(W)$ is defined as:

$$L(W)_{ij} = \begin{cases} 1 - \frac{w_{ij}}{d_i} & \text{if } i = j, \\ -\frac{w_{ij}}{\sqrt{d_i d_j}} & \text{if } i \text{ and } j \text{ are adjacent,} \\ 0 & \text{otherwise,} \end{cases}$$

where d_i is the out degree of vertex i, i.e., $d_i = \sum w_{ij}, \forall j \in V$. We can then define a diagonal matrix D where $D_{ii} = d_i$. Therefore, the normalized Laplacian can be represented as

$$L(W) = D^{(-1/2)} W D^{(-1/2)}. \quad (2)$$

For large-scale datasets such as the Web corpora and image collections, their feature space usually consists of millions of vectors of very high dimensions (e.g., $x = 10^6, y = 10^7$). Therefore, it is often desirable to find a low rank matrix $\tilde{W}$ to approximate $L(W)$ in order to lower the computation cost, to extract correlations, and remove noise. Traditional matrix decomposition methods, e.g., Singular Value Decomposition (SVD) and eigenvalue decomposition

(when the matrix is symmetric), require superlinear time for matrix-vector multiplication so they usually do not scale to real-world applications.

For symmetric low rank approximation, we use the Lanczos algorithm [12] which iteratively finds the eigenvalues and eigenvector of square matrices. Given an $n \times n$ sparse symmetric matrix A with eigenvalues:

$$\lambda_1 \geq ... \geq \lambda_n > 0, \tag{3}$$

the Lanczos algorithm computes a $k \times k$ symmetric tridiagonal matrix T, whose eigenvalues approximate the eigenvalues of A, and the eigenvectors of T can be used as the approximations of A's eigenvectors, with k much smaller than n. In other words, T satisfies:

$$\|A - T\|_F \leq \epsilon \|A\|_F, \tag{4}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, with ϵ as a controlled variable. For example, to capture 95% variances of A, ϵ is set to 0.05.

3.2 Bipartite Graph Partitioning

For multi-clustering on bipartite graphs, we apply the Spectral Recursive Embedding (SRE) algorithm [16]. SRE essentially constructs partitions by minimizing a normalized sum of edge weights between unmatched pairs of vertices, i.e., $\min_{\Pi(A,B)} Ncut(A,B)$, where A and B are matched pairs in one partition with A^c and B^c being the other. The normalized variant of edge cut $Ncut(A,B)$ is defined as:

$$Ncut(A,B) = \frac{cut(A,B)}{W(A,Y)+W(X,B)} + \frac{cut(A^c,B^c)}{W(A^c,Y)+W(X,B^c)}, \tag{5}$$

where

$$\begin{aligned} cut(A,B) &= W(A,B^c) + W(A^c,B) \\ &= \sum_{i\in A, j\in B^c} w_{ij} + \sum_{i\in A^c, j\in B} w_{ij}. \end{aligned} \tag{6}$$

The rationale of $Ncut$ is not only to find a partition with a small edge cut, but also partitions that are as dense as possible. This is useful for our application of tagging documents, where the documents in each partition are ideally focused on one specific *topic*. As a result, the denser a partition is, the better that relevant documents and tags are grouped together.

3.3 Within Cluster Node Ranking

We define two new metrics *N-Precision* and *N-Recall* for node ranking. N-Precision of a node i is the weighted sum of its edges that connect to the nodes within the same cluster, divided by the total sum of edge weights in that cluster. Denote the cluster label of i as $C(i)$,

$$np_i = \frac{\sum_{j=1}^{n} w_{ij}\mathbb{I}[C(j)=C(i)]}{\sum_{j,k=1}^{n} w_{jk}\mathbb{I}[C(j)=C(k)=C(i)]}, j,k \neq i. \tag{7}$$

For the unweighted graph, the above equation equals to the number of edges associated with node i in cluster $C(i)$, divided by the total number of edges in cluster $C(i)$. Generally, N-precision measures the importance of a node to the cluster, in comparison with other nodes. In the context of text documents, the cluster is a topic set of documents and the weight of the word nodes shows the frequency of the words appearing in that topic. With the cluster determined, the denominator of equation (7) is constant, so that the more weight the node has, the more important it is.

In contrast, N-recall is used to quantify the posterior probability of a node i to a given cluster and is the inverse fraction of i's edge associated with its cluster

$$nr_i = \frac{|E_i|}{\sum_{j=1}^{n} \mathbb{I}[C(j)=C(i)]}. \tag{8}$$

It is evident that N-Recall is always no less than 1. The larger N-Recall is, the more probable that a word is associated with a specific topic.

Given np_i and nr_i, we can estimate the ranking of i:

$$\begin{aligned} Rank_i &= \begin{cases} \exp\left(-\frac{1}{r(i)^2}\right) & r(i) \neq 0, \\ 0 & r(i) = 0, \end{cases} \\ \texttt{where } r(i) &= (np_i) * \log(nr_i). \end{aligned} \tag{9}$$

Depicted in Figure 2, our ranking function is a smoothed surrogate that is proportional to both node precision and recall, guaranteed to be in the range of $(0,1)$.

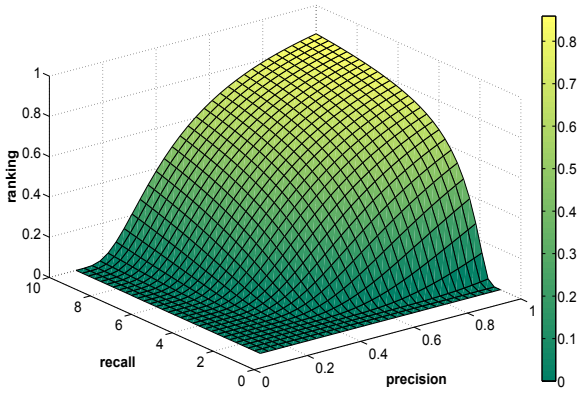


Figure 2: Smoothed Ranking Function.

Potential applications of the aforementioned bipartite graph node ranking methodology include interpreting the document-author relationship. i.e., determine the social relations (e.g., "hub" and "authority") of authors in the same research topic, and finding the most representative documents in the topic. In what follows, we apply this framework to tag recommendation by ranking nodes that represent tags in each cluster.

4. ONLINE TAG RECOMMENDATION

A typical document of concern here consists of a set of words and several tags annotated by users. The relationship among documents, words, and tags can then be represented by two bipartite graphs as shown in Figure 3.

The weighted graph can be written as

$$W = \begin{pmatrix} 0 & A & 0 \\ A^T & 0 & B \\ 0 & B^T & 0 \end{pmatrix}, \tag{10}$$

where A and B denote the inter-relationship matrices between tags and docs, docs and words, respectively.

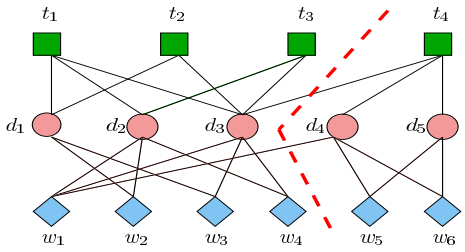


Figure 3: Two bipartite graphs of documents, words and tags.

Given the matrix representation, a straightforward approach to recommend tags is to consider the similarity (e.g., cosine similarity) between the query document and training documents by their word features, then suggest the top-ranked tags from most *similar* documents. This approach is usually referred to as collaborative filtering [3]. Nevertheless, this approach is not efficient for real-world scenarios. To take the advantage of the proposed node ranking algorithm, we propose a Poisson mixture model that can efficiently determine the membership of a sample as well as clustering words with similar meanings.

Before presenting the mixture model, we first summarize our framework of tag recommendation in Algorithm 1.

Algorithm 1 Online Tag Recommendation

1: **Input** $(\mathcal{D}, S, T)$, K, M, L
   Document collection: $\mathcal{D} = \{\mathcal{D}_1, ..., \mathcal{D}_m\}$
   Word vocabulary: $S = \{S_1, ..., S_k\}$
   Tag vocabulary: $T = \{T_1, ..., T_n\}$
   Number of clusters: $K \in \mathbb{R}$
   Number of components: $M \in \mathbb{R}$
   Number of word clusters: $L \in \mathbb{R}$
   Offline Computation
2: Represent the weighted adjacency matrix W as in eq. (10)
3: Normalize W using the normalized Laplacian $L(W) = D^{(-1/2)}WD^{(-1/2)}$ (eq. (2))
4: Compute a low rank approximation matrix using the Lanczos: $\tilde{W} \simeq L(W) = Q_k T_k Q_k^T$
5: Partition $\tilde{W}$ into K clusters using SRE [16], $\tilde{W} = \{\tilde{W}_1, ..., \tilde{W}_K\}$
6: Assign labels to each document $\mathcal{D}_j, j \in \{1, ...m\}$ $C(\mathcal{D}_j) \in \{1, ..., K\}$
7: Compute the node rank $Rank(T)$ for each tag $T_{i,k}$ in cluster k, $i \in \{1, ..., n\}, k\{1, ..., K\}$ (eq. (9))
8: Build a Poisson mixture model for $(\tilde{B}, C(\mathcal{D}))$ with M components and L word clusters, where $\tilde{B}$ denotes the inter-relationship matrix of documents and words in $\tilde{W}$ (eq. (10))
   Online Recommendation
9: For each test document $\mathbb{Y}$, calculate its posterior probabilities $P(C = k|D = \mathbb{Y})$ in each cluster k, and denote the membership of $\mathbb{Y}$ as $C(\mathbb{Y}) = \{c(\mathbb{Y}, 1), ..., c(\mathbb{Y}, K)\}$ ((eq. (17)))
10: Recommend tags based on the rank of tags, i.e., the joint probability of tags T and document $\mathbb{Y}$, $R(T, \mathbb{Y})$ (eq. (18))

Intuitively, this two-stage framework can be interpreted as an unsupervised-supervised learning procedure. During the offline learning stage, nodes are partitioned into clusters using an unsupervised learning method, cluster labels are assigned to document nodes as their "class labels", and tag nodes are given ranks in each cluster. A mixture model is then built based on the distribution of document and word nodes. In the online recommendation stage, a document is classified into predefined clusters acquired in the first stage by naive Bayes so that tags can be recommended in the descending orders of their ranks. To avoid confusion, we will refer to the clusters determined by the partitioning algorithm in the first stage as *classes* in the next section.

4.1 Two-way Poisson Mixture Model

We propose to use Poisson mixture models to estimate the distribution of document vectors, because they fit the data better than standard Poissons by producing better estimates of the data variance, and are relatively easy for parameter estimation. Although it takes time to fit the training data, it is efficient to predict the class label of new documents once the model is built. Because of the numerical stability of this statistical approach, the results are usually reliable. Since only probabilistic estimation is involved, it is capable for real-time process.

Nevertheless, traditional unsupervised learning approaches of mixture models [9] are not always capable of dealing with document classification. Considering the sparseness and high-dimensionality of the document-word matrix where most entries are zeros and ones, the model may fail to predict the true feature distribution (i.e. the probability mass function) of different components. As a result, word clustering is a necessary step before estimating the components in the model. In what follows, we utilize the two-way Poisson mixture model [14] in order to simultaneously cluster word features and classify documents.

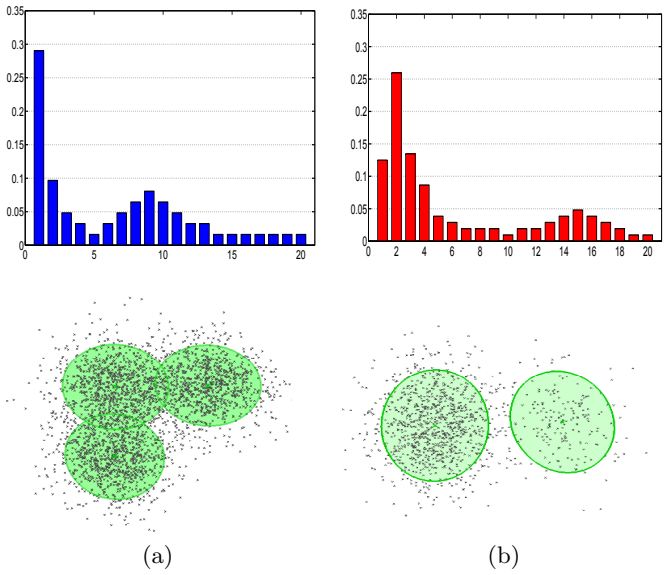


Figure 4: An example of two mixtures of the Poisson distribution in two clusters.(Top) The histograms of mixture components. (Bottom) Mixture model classification results. (a) Three-component mixtures. (b) Two-component mixtures.

Given a document $D = \{D_1, ..., D_p\}$, where p is the dimension, the distribution of the document vector in each class can be estimated by using a parametric mixture model.

Let the class label be $C = \{1, 2, ..., K\}$, then

$$P(D = d|C = k) = \sum_{m=1}^{M} \pi_m \mathbb{I}(F(m) = k) \prod_{j=1}^{p} \phi(d_j|\lambda_{j,m}), \tag{11}$$

where π_m is the prior probability of component m, with $\sum_{m=1}^{M} \pi_m = 1$. $\mathbb{I}(F(m) = k)$ is an indicator function, i.e., whether component m belongs to class k, and ϕ denotes the probability mass function (pmf) of a Poisson distribution, $\phi(d_j|\lambda_{j,m}) = e^{-\lambda_{j,m}} {\lambda_{j,m}}^{d_j}/d_j!$.

In this way, each class is a mixture model with a multivariate distribution having variables that follow a Poisson distribution. Figure 4 shows the histogram of two mixtures which can be regarded as the pmfs of two Poisson mixtures.

Our assumption is that within each class, words in different documents have equal Poisson parameters, while for documents in different classes, words may follow different Poisson distributions. For simplicity, we also assume that all classes have the same number of word clusters. Denote $l = \{1, , , L\}$ to be the word clusters, words in the same word cluster m will have the same parameters, i.e., $\lambda_{i,m} = \lambda_{j,m} \equiv \tilde{\lambda}_{l,m}$, for $c(i,k) = c(j,k)$, where $c(i,k)$ denotes the cluster label of word i in class k. Therefore, Equation (11) can be simplified as follows (with $L \ll p$):

$$P(D = d|C = k) \propto \sum_{m=1}^{M} \pi_m \mathbb{I}(F(m) = k) \prod_{l=1}^{L} \phi(d_{k,l}|\tilde{\lambda}_{l,m}). \tag{12}$$

4.1.1 Parameter Estimation

With the classes determined, we apply EM algorithm to estimate the Poisson parameters $\tilde{\lambda}_{l,m}, l \in \{1, ..., L\}, m \in \{1, ..., M\}$, the priors of mixture components π_m, and the word cluster index $c(k,j) \in \{1, ..., L\}, k \in \{1, ..., K\}, j \in \{1, ..., p\}$.

The E-step estimates the posterior probability $p_{i,m}$:

$$p_{i,m} \propto \pi_m^{(t)} \mathbb{I}(C(i)) \prod_{j=1}^{p} \theta(d(i,j)|\tilde{\lambda}_{m,i,j}^{(t)}). \tag{13}$$

The M-step uses $p_{i,m}$ to maximize the objective function

$$L(\pi_m^{(t+1)}, \tilde{\lambda}_{m,l}^{(t+1)}, c^{(t+1)}(k,j)|\pi_m^{(t)}, \tilde{\lambda}_{m,l}^{(t)}, c^{(t)}(k,j))$$
$$= \max \sum_{i=1}^{n} \sum_{m=1}^{M} p_{i,m} \log \left(\pi_m^{(t+1)} \mathbb{I}(C(i)) \prod_{j=1}^{p} \theta(d(i,j)|\tilde{\lambda}_{m,i,j}^{(t+1)}) \right),$$

and update the parameters

$$\pi_m^{(t+1)} = \frac{\sum_{i=1}^{n} p_{i,m}}{\sum_{m'=1}^{M} \sum_{i=1}^{n} p_{i,m'}}, \tag{14}$$

$$\tilde{\lambda}_m^{(t+1)} = \frac{\sum_{i=1}^{n} p_{i,m} \sum_j d(i,j) \mathbb{I}(C(i))}{|d(i,j)| \sum_{i=1}^{n} p_{i,m}}, \tag{15}$$

where $|d(i,j)|$ denotes the number of j's in component l.

Once $\tilde{\lambda}_m^{(t+1)}$ is fixed, the word cluster index $c^{(t+1)}(k,j)$ can be found by doing linear search over all components:

$$c^{(t+1)}(k,j) = \arg\max_l \sum_{i=1}^{n} \sum_{m=1}^{M} \log(d(i,j)|\tilde{\lambda}_{m,l}^{(t+1)}). \tag{16}$$

4.2 Tag Recommendation for New Documents

Normally, the class label $C(d_t)$ of a new document d_t is determined by $\hat{C}(x) = \arg\max_k P(C = k|D = d_t)$. However in our case, we determine the mixed membership of a document by calculating its posterior probabilities to classes, with $\sum_{k=1}^{K} P(C = k|D = d_t) = 1$. Applying equation (12) and the Bayes rule,

$$P(C = k|D = d_t) = \frac{P(D = d_t|C = k)P(C = k)}{P(D = d_t)}$$
$$= \frac{\sum_{m=1}^{M} \pi_m \mathbb{I}(F(m) = k) \prod_{l=1}^{L} \phi(d_{k,l}|\tilde{\lambda}_{l,m}) P(C = k)}{P(D = d_t)}, \tag{17}$$

where $P(C = k)$ are the prior probabilities for class k and are set uniform. Finally, the probability for each tag $T_i, i \in \{1, ..., n\}$ to be associated with the sample is

$$R(T_i, d_t) = P(T = T_i|D = d_t) = Rank_{T_i} * P(C = x|D = d_t). \tag{18}$$

By ranking the tags in descending order of their probabilities, the top ranked tags are selected for recommendation.

5. EXPERIMENTAL RESULTS

We evaluate our proposal (PMM) by conducting two sets of experiments in different application contexts: recommending tags for scientific documents (CiteULike) and web pages (del.icio.us).

Parameters are tuned before the online step takes place, i.e., the number of clusters K, the number of components M, and the number of word clusters L. Due to the space limitation, we only present the best results here.

For comparison, the Vector Similarity (VS) approach is used as a baseline, which calculates the cosine similarity between a query Q and each training document D_i, $Sim(Q, D_i) = \frac{\sum_j n(Q,j) n(i,j)}{\sqrt{\sum_j n(Q,j)^2} \sqrt{\sum_i n(i,j)^2}}$, where $n(i,j)$ represents the count of j's word in sample i. The top t tags from s most similar documents are then considered. In our experiment, we set both t and s to be 3, resulting in 9 recommendations for each query document. To improve performance, we augment the vector similarity approach by applying information-gain (VS+IG) to select roughly 5% of the total features.

Meanwhile, we also compare with a recent developed method named *SimFusion* [15] which leveraged unified relationship matrix to iteratively calculate the similarity between objects. Details are omitted here.

5.1 Evaluation Metrics

In addition to the standard Kendall τ rank correlation metric [13] that measures the degree of correspondence between two ranked lists, we also propose the following metrics to measure the effectiveness of our algorithm.

- *Top-k accuracy:* Percentage of documents correctly annotated by *at least* one of the top kth returned tags.
- *Exact-k accuracy:* Percentage of documents correctly annotated by the kth recommended tag.
- *Tag-recall:* Percentage of correctly recommended tags among all tags annotated by the users.
- *Tag-precision:* Percentage of correctly recommended tags among all tags recommended by the algorithm.

In our experiments, we return top 9 tags for evaluation.

5.2 CiteULike

For evaluation on scientific documents, we acquired the tagging dataset from CiteULike for over two years from November 15, 2004 to February 13, 2007. We mapped the dataset to papers that are indexed in CiteSeer[4] to extract the metadata. Each entry of the CiteULike record contains four fields: user name, tag, key (the paper ID in CiteSeer), and creation date. Overall, there are 32,242 entries, with 9,623 distinct papers and 6,527 distinct tags (tag vocabulary). The average number of tags per paper was 3.35. The 5 most tagged papers are listed in Table 1 respectively.

We report the results of 50% training and 50% test data here due to space limitation[5]. The optimal number of clusters K is 30, number of component M is 40, and the number of word cluster L is 30.

Table 1 also lists the top user tags for each of the top 10 papers, as well as the top 9 tags recommended by our algorithm. The bold fonts indicate an overlap. Generally, at least one correct recommendation is made for each paper, and the first tag recommended always matches one of the user tags. In addition, although some recommended tags do not match the user tags literally, most of them are semantically relevant. e.g., "www" is relevant to "web"; "communities" is often consisted in "social networks"; "page" and "rank" together have the same meaning as "pagerank". In the best scenario, 7 of 9 recommended tags match with the user tags for the paper "A Tutorial on Learning With Bayesian Networks", which has a Kendall τ rank of 0.78.

The comprehensive performance on CiteULike data set is depicted in Figure 6. On average, the Kendall τ rank is 0.24 for PMM, indicating a positive correspondence between the two rankings. The *top-9* tag performance is shown in (a), where our algorithm makes 67.2% correct recommendation for the top-most(*top-1*) tag, for which SimFusion and VS+IG are around 58.2% and 43.3%. As the number of tags increases, the top-k performance gradually improves. The accuracy of top 9 tags for PMM reaches 93.1%, indicating that at least 1 of 9 tags recommended by our algorithm is also annotated by the users of over 4,000 test papers. Figure 6 (b) shows the accuracy at different ranks. It is clear that the top-most tags achieve the best accuracy (67.2%) for PMM and the performance decreases as the rank of the tags decreases. Finally, Figure 6 (c) shows the precision-recall graph. Our method is clearly the winner. With the number of tags increases from 1 to 9, tag-precision drops from 67.2% to 43.5%, while tag-recall goes up from 10.2% to 55.6%.

On average, the histogram of the number of correctly-labeled tags implies that 3.72 tags are correctly recommended.

5.3 Del.icio.us

Using the tagging data set from del.icio.us, we subscribed to 20 popular tags, each of which is treated as a topic. For these topics, we retrieved 22,656 URLs from March 3rd, 2007 to April 25, 2007. For each URL, we crawled del.icio.us to obtain the most popular tags with their frequencies. We also harvested the HTML content of each URL. We ended up with 215,088 tags, of which 28,457 are distinct (tag vocabulary), averaging 9.5 tags per URL. The total size of the dataset is slightly over 2GB.

We sorted the data chronically and used the first half for training, the rest for testing[5]. By filtering out stop words and using mutual information to select the most informative words, we form a feature space with 83,205 words, containing 11,300 samples. The sparseness of the training set (number of zeros vs. dimensionality) is 92%.

Figure 7 shows the results for 11,356 test URLs, with a Kendall τ value of 0.13 for PMM. Comparing with CiteULike, the performance degrades for all metrics. Again, our algorithm outperform the others significantly. The accuracy of top-1 tag is around 48%, and degrades to 22% for the 9th tag. However, the top 9 recommended tags together are still able to overlap with 88.5% of the user-annotated tags.

We give two explanations for the degraded performance on the web page tag recommendation task. First, we notice that our algorithm usually fails when the content of a specific URL contains little of the necessary information, i.e., words in our case. As an example, for the topics "photography" and "travel", many pages only contain images and short descriptions, making it hard for our model to determine the proper components for a test sample.

Second, unlike structured scientific documents with controlled vocabularies, the heterogeneous nature of web pages not only results in varied length (word count) of the html pages, but also the distribution of the tag vocabulary. In fact, for PMM, the *tag/doc* ratio for the CiteULike data is 0.68 (6,527 unique tags vs. 9,623 papers), compared with 1.26 (28,457 unique tags vs. 22,656 URLs) for del.icio.us. A previous study [11] has shown that the tag vocabulary usually does not converge for a specific user, reflecting a continual growth of interests. Thus, we believe that a large tag vocabulary could possibly compromise the recommendation performance for unstructured web pages. On average, 2.91 correct tags are recommended for each test sample.

Figure 5 depicts the user tags as well as our recommended tags that co-occurred with "ajax". It can be observed that the co-occurred tags in our model are consistent with those annotated by the users.

5.4 Efficiency of Online Recommendation

To show that our model is capable of making real-time tagging for large volumes of documents, we evaluate our model in terms of the average tagging time for query documents. Different proportions of training documents (from 10% to 90 %) are tested.

Figure 8 and Table 2 present the performance of CiteULike and del.icio.us data respectively[6]. Our approach exhibits stable performance on both data sets with very small variance. On average, only 1.08 seconds is needed for one test document on CiteULike and 1.23 seconds for del.icio.us. On the other hand, the average tagging time for SimFusion and VS+IG is 6.4 and 16 seconds respectively, expected to grow exponentially with the increase of the features.

The efficiency of our model can be explained by its linear calculation. With the model determined, calculating the membership of a new document within each class is readily automated, requiring only multiplication in equation (17) of the M components and L word clusters. Recall that $M = O(K)$ and $L \ll p$ with the process taking linear time to complete. Moreover, by taking the log over both sides of equation (17), the multiplication can be replaced by addition operations, making it even more efficient.

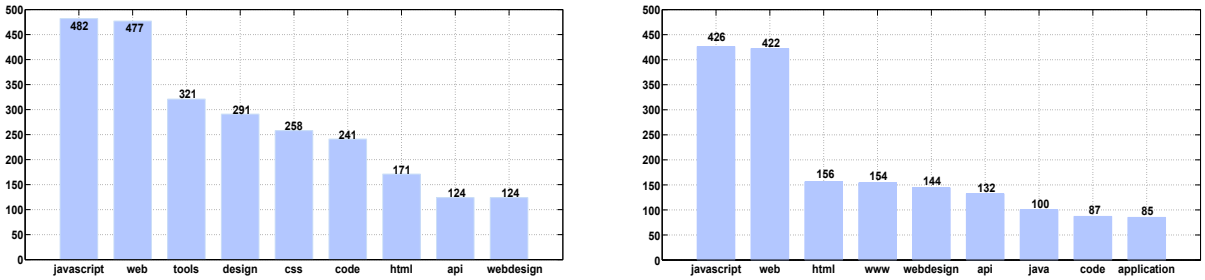
[4]http://citeseer.ist.psu.edu/oai.html

[5]We also tested our algorithm on different splitting ratios, with similar results observed.

[6]The experiment was performed on a 3.0GHZ sever.

Paper Name	Tags	Top User Tags	Our Tags
The PageRank Citation Ranking: Bringing Order to the Web (Larry Page et al.)	135	google, pagerank, search, ranking, web, social-networks, networks, socialnetworks, ir	**search**, **web**, rank, mining, **pagerank**, page, rank, www, **ir**
The Anatomy of a Large-Scale Hypertextual Web Search Engine (Sergey Brin, Lawrence Page)	94	google, search, pagerank, web, engine, www, web-search, ir, graphs	**search**, **web**, **engine**, **www**, page, rank, **ir**, classification, mining
ReferralWeb: Combining Social Networks and Collaborative Filtering (Henry Kautz et al.)	88	folksonomy, collaboration, tagging, social-networks, networks, social, filtering, recommender, socialnetworks	**networks**, network, adhoc, mobile, mobilitymodel, **filtering**, **tagging**, **social**, socialnetwork
A Tutorial on Learning With Bayesian Networks (David Heckerman)	78	bayesian, networks, learning, network, statistics, bayes, tutorial, modeling, graphs, algorithms	**bayesian**, **networks**, **learning**, **network**, **bayes**, machinelearning, **modeling**, data, **graphical**
Maximizing the Spread of Influence through a Social Network (David Kempe et al.)	73	social, influence, network, socialnetworks, diffusion, research, spread, networking	**network**, networks, **social**, **socialnetworks**, adhoc, models, machinelearning, algorithm, data

Table 1: Top 5 papers from CiteULike data in terms of popularity (number of times the paper being tagged). The top 9 recommended tags are listed as "Our Tags", ranked according to our node ranking algorithm. Tags with bold font match one of the user-annotated tags.



(a) **User tags** (b) **Our recommendations**

Figure 5: Tag co-occurrence of the del.icio.us data set shows the relationship between tags. (a) Top 9 most co-occurred user tags with "ajax", which appears 1,930 times. (b) Top 9 most co-occurred recommended tags with "ajax", which is suggested 1,325 times.

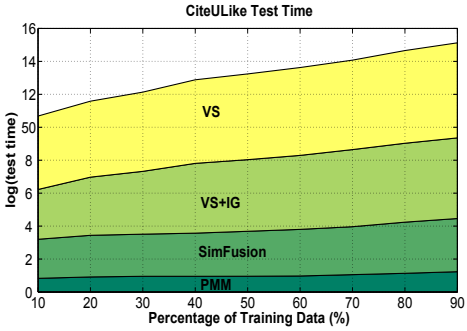


Figure 8: Average tagging time of CiteULike data.

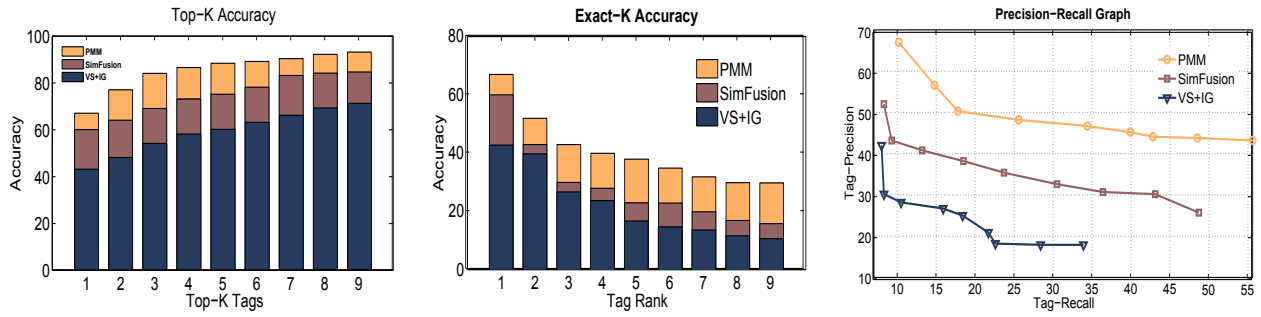
% Train	PMM	SimFusion	VS+IG
10	0.85 ± 0.7	4.3 ± 2.1	20.5 ± 11.2
30	0.90 ± 0.8	5.9 ± 2.8	45.2 ± 10.6
50	0.92 ± 0.6	6.7 ± 2.9	77.3 ± 10.7
70	1.33 ± 1.2	6.8 ± 2.9	108.0 ± 10.8
90	1.41 ± 1.4	7.8 ± 3.2	133.2 ± 9.5
Average	**1.23**	**6.44**	**77.43**

Table 2: Delicious Test Time (seconds).

6. CONCLUSION AND FUTURE WORK

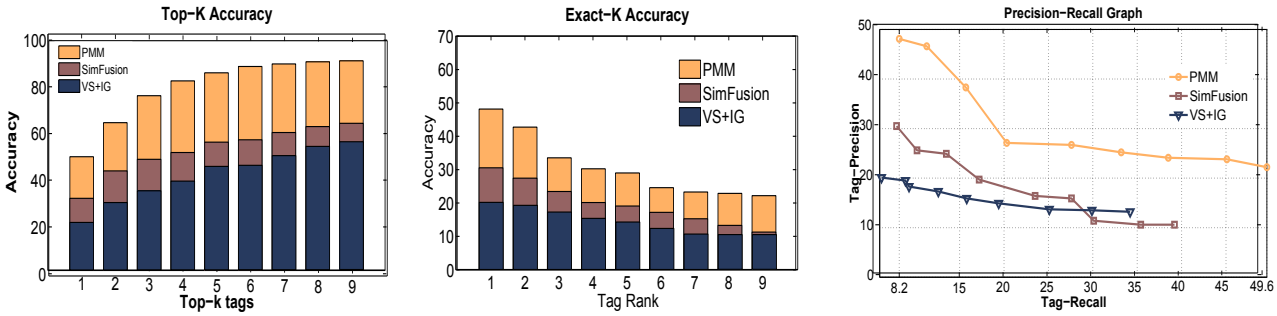
In this paper, we proposed a learning framework for tag recommendation for scientific and web documents. We proposed a Poisson mixture model for efficient document classification. We also proposed a novel node ranking method as well as several new metrics for evaluating the performance of our framework. The proposed framework demonstrates its potential in evaluations on two real-world tagging data sets, indicating its capability of handling large-scale data sets in real-time. Being runtime efficient, our proposed method can recommend tags in one second on average.

Future work would be to examine the performance of our algorithm for less popular tags/documents. A user study could be done by assigning relevance scores to each tag (e.g. 0-3), and applying Normalized Discounted Cumulative Gain (NDCG) to measure the performance. Of course it would be interesting to apply this framework to other tagging recommendations.



(a) (b) (c)

Figure 6: Performance of CiteULike data with 4,320 test documents. 9 tags are recommended for each document. (a) Top-k: percentages of documents that are correctly annotated by at least one of the top kth tags. (b) Exact-k: Percentages of documents that are correctly annotated by the kth tag. (c) Precision-Recall graph shows the change of tag-precision and tag-recall with the number of recommended tags increases.



(a) (b) (c)

Figure 7: Performance of the Del.icio.us data set with 11,300 training and 11,365 test html pages.

7. REFERENCES

[1] R. Baeza-Yates, C. Hurtado, and M. Mendoza. Query recommendation using query logs in search engines. In *International Workshop on Clustering Information over the Web (in conjunction with EDBT)*, 2004.

[2] G. Begelman, P. Keller, and F. Smadja. Automated tag clustering: Improving search and exploration in the tag space. In *Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland*, 2006.

[3] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Uncertainty in Artificial Intelligence. Proceedings of the Fourteenth Conference (1998)*, pages 43–52, 1998.

[4] P. A. Chirita, S. Costache, W. Nejdl, and S. Handschuh. P-tag: large scale automatic generation of personalized annotation tags for the web. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 845–854, New York, NY, USA, 2007. ACM Press.

[5] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274, New York, NY, USA, 2001. ACM Press.

[6] C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 107–114, Washington, DC, USA, 2001. IEEE Computer Society.

[7] P. Drineas, R. Kannan, and M. W. Mahoney. Fast Monte Carlo Algorithms for Matrices III: Computing a Compressed Approximate Matrix Decomposition. *SIAM J. Comput.*, 36(1):184–206, 2006.

[8] U. Farooq, Y. Song, J. M. Carroll, and C. L. Giles. Social bookmarking for scholarly digital libraries. *IEEE Internet Computing*, pages 29–35, Nov,2007.

[9] M. A. T. Figueiredo and A. K. Jain. Unsupervised learning of finite mixture models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(3):381–396, 2002.

[10] B. Gao, T.-Y. Liu, X. Zheng, Q.-S. Cheng, and W.-Y. Ma. Consistent bipartite graph co-partitioning for star structured high-order heterogeneous data co-clustering. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 41–50, New York, NY, USA, 2005.

[11] S. Golder and B. Huberman. Usage patterns of collaborative tagging systems. *J. Inf. Sci.*, 2006.

[12] G. H. Golub and C. F. V. Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, 1996.

[13] M. Kendall. A new measure of rank correlation. *Biometrika*, 30:81–89, 1938.

[14] J. Li and H. Zha. Two-way poisson mixture models for simultaneous document classification and word clustering. *Computational Statistics & Data Analysis*, 2006.

[15] W. Xi, E. A. Fox, W. Fan, B. Zhang, Z. Chen, J. Yan, and D. Zhuang. Simfusion: measuring similarity using unified relationship matrix. In *SIGIR '05*, pages 130–137, New York, NY, USA, 2005. ACM Press.

[16] H. Zha, X. He, C. Ding, H. Simon, and M. Gu. Bipartite graph partitioning and data clustering. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 25–32, New York, NY, USA, 2001. ACM Press.