Automatic Tag Recommendation Algorithms for Social Recommender Systems

YANG SONG
Department of Computer Science and Engineering
The Pennsylvania State University
LU ZHANG
Department of Statistics
The Pennsylvania State University
and
C. LEE GILES
College of Information Sciences and Technology
The Pennsylvania State University

The emergence of Web 2.0 and the consequent success of social network websites such as del.icio.us and Flickr introduce us to a new concept called social bookmarking, or tagging in short. Tagging can be seen as the action of connecting a relevant user-defined keyword to a document, image or video, which helps user to better organize and share their collections of interesting stuff. With the rapid growth of Web 2.0, tagged data is becoming more and more abundant on the social network websites. An interesting problem is how to automate the process of making tag recommendations to users when a new resource becomes available.

In this paper, we address the issue of tag recommendation from a machine learning perspective of view. From our empirical observation of two large-scale data sets, we first argue that the user-centered approach for tag recommendation is not very effective in practice. Consequently, we propose two novel document-centered approaches that are capable of making effective and efficient tag recommendations in real scenarios. The first graph-based method represents the tagged data into two bipartite graphs of (document, tag) and (document, word), then finds document topics by leveraging graph partitioning algorithms. The second prototype-based method aims at finding the most representative documents within the data collections and advocates a sparse multi-class Gaussian process classifier for efficient document classification. For both methods, tags are ranked within each topic cluster/class by a novel ranking method. Recommendations are performed by first classifying a new document into one or more topic clusters/classes, and then selecting the most relevant tags from those clusters/classes as machine-recommended tags.

Experiments on real-world data from Del.icio.us, CiteULike and BibSonomy examine the quality of tag recommendation as well as the efficiency of our recommendation algorithms. The results suggest that our document-centered models can substantially improve the performance of tag recommendations when compared to the user-centered methods, as well as topic models LDA and SVM classifiers.

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

Additional Key Words and Phrases: tagging system, mixture model, graph partitioning, Gaussian processes, prototype selection, multi-label classification

1. INTRODUCTION

Tagging, or social bookmarking, 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] that support social bookmarking on web pages and images respectively, tagging services have become red-hot popular[3] among users and have drawn much attention from both academia and industry. These web sites allow users to specify keywords or tags for resources, which in turn facilitates the organizing and sharing of these resources with other users. Since the amount of tagged data potentially available is virtually free and unlimited, interest has emerged in investigating the use of data mining and machine learning methods for automated tag recommendation or both text and digital data on the web [Begelman et al. 2006; Chirita et al. 2007; Golder and Huberman 2006; Li and Wang 2006].

1.1 The Problem

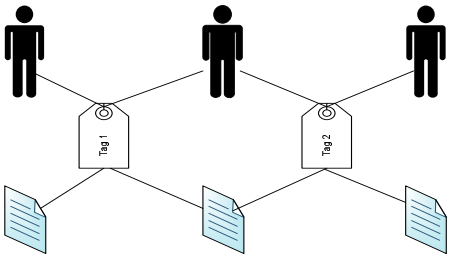


Fig. 1. A connectivity graph of users, tags and documents. In the scenario of tagging, a user annotates a document by creating a personal tag. As it can be observed, tags are not directly connected to each other, but to the users and documents instead.

Tag recommendation refers to the automated process of suggesting useful and informative tags to an emerging object based on historical information. An example of the recommendation by the Del.icio.us system is shown in Figure 2, where the user is bookmarking a webpage regarding data mapper and the system recommends relevant tags as well as popular ones for annotation. While the objects to be tagged can be images, videos or documents, we will focus on documents in this paper unless otherwise mentioned. In general, a tagged document is usually associated with one or more tags, as well as users who annotated the document by different tags. Thus, a tagging behavior to a document d performed by user u with tag t can be represented using a triplet (u, d, t). Using a graph representation where each node is one of the elements in the triplet, and edges between nodes being the degree of connection, it is obvious that both the users and the documents are highly connected to the

[1]http://del.icio.us/
[2]http://www.flickr.com/
[3]Recent statistics indicated that del.icio.us gets roughly 150,000 posts per day while Flickr gets 1,000,000 photos per day.

tags, while the relationship between tags themselves cannot be observed directly (shown in Figure 1). Consequently, recommending relevant tags to new users or new documents can only be done indirectly from the user perspective or the document perspective.

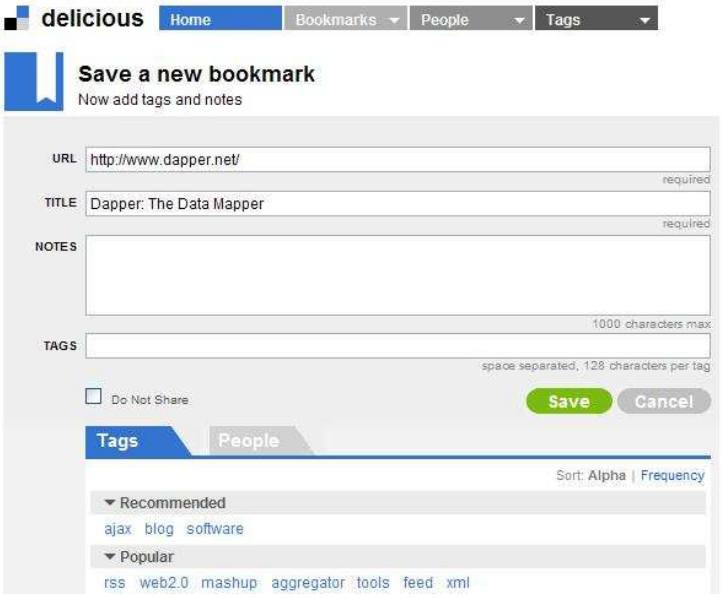


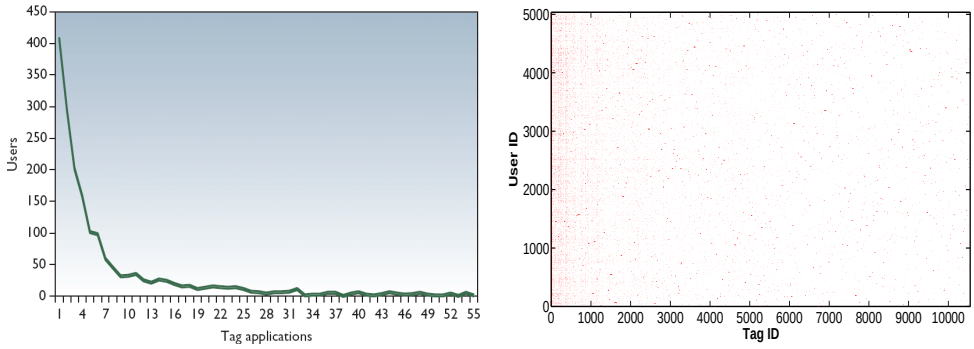
Fig. 2. An example of recommended tags by the Del.icio.us recommender system.

As it can be observed, tag recommendation can be addressed in two different aspects. i.e., user-centered approaches and document-centered approaches. User-centered approaches aim at modeling user interests based on their historical tagging behaviors, and recommend tags to a user from similar users or user groups. On the other hand, document-centered approaches focus on the document-level analysis by grouping documents into different topics. The documents within the same topic are assumed to share more common tags than documents across different topics. Theoretically, both models can be learnt by using classic machine learning approaches. For example, *collaborative filtering* (CF) techniques [Breese et al. 1998] can be applied to learn the user interests for the user-centered approaches. For document-centered approaches, both unsupervised topic models (e.g., LDA topic models [Blei et al. 2003]) and supervised classification models (e.g., SVM [Cristianini and Shawe-Taylor 2000]) are good candidates for categorizing document topic groups.

While both approaches seem to be plausible, it turns out that the user-centered approaches are not very effective due to several obvious reasons. First, according to research in [Farooq et al. 2007], the distribution of users vs. the number of tag applications follows a long tail power law distribution, meaning that only a very small portion of the users perform tagging extensively (see Figure 3 (a)). Additionally, researchers have also shown that the reusability of tags are quite low, while the

vocabulary of tags constantly grows [Farooq et al. 2007] (see Figure 3 (b)). With relatively few user information acquired, it makes the user-centered approaches difficult to find a suitable model to perform effective tag recommendation. While clustering users into interests groups can somewhat alleviate the issue of sparseness, user-centered approaches are not very flexible in monitoring the dynamic change of user interests over time.

Comparatively, the document-centered approaches are more robust because of the rich information contained in the documents. Moreover, the underlying semantics within tags and words create a potential link between topics and contents in the documents, where tags can be treated as class labels for documents in the scenario of supervised learning, or summarizations of documents as an unsupervised learning approach. This makes it flexible to apply any sophisticated machine learning algorithms for the user-centered tag recommendation approach.



(a) (b)

Fig. 3. Challenge of tag applications. (a) Number of users vs. number of tag applications. Relatively few users generated most of the tag applications. (b) Frequency matrix of tags and users, where X-axis indicates Tag ID and Y-axis is User ID, showing that the matrix is very sparse.

Additionally, while both *effectiveness* and *efficiency* need to be addressed for ensuring the performance of the tagging services, most of the existing work has focused on effectiveness [Begelman et al. 2006; Chirita et al. 2007; Golder and Huberman 2006]. Efficiency, while not being totally ignored, has only been of recent interest [Li and Wang 2006].

1.2 Our Contributions

In this paper, we propose two frameworks for addressing automatic tag recommendation for social recommender systems. From a machine learning perspective of view, we want our models to be *reusable* for different applications and systems, *scalable* to large web-scale applications, and the results are *effective* for all of them. The first approach we proposed is a *graph-based* method, in which the relationship among documents, tags, and words are represented in two bipartite graphs. A two-state framework is advocated for learning from previously seen data. 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) [Zha et al. 2001] 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 [Li and Zha 2006] 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. The two-way Poisson mixture model (PMM) applied here is very efficient for classification. Comparing to other classification methods, the two-way PMM has the advantage of modeling the multivariate distribution of words in each class, so that it is capable of clustering words simultaneously while classifying documents, which helps reducing the dimensionality of the document-word matrix. The two-way PMM is flexible in choose component distribution for each topic class, i.e., different classes may have different number of components. i.e., number of sub-topics. Moreover, this model performs a *soft* classification for new documents that allows tags to be recommended from different classes.

The second approach is a *prototype-based* method. Instead of using the entire training data, this method aims at finding the most representative subset within the training data so as to reduce the learning complexity. This supervised learning approach classifies documents into a set of pre-defined categories, which are determined by the popularity of existing tags. Similar to the graph-based method, the tags are ranked within each category and recommended to a new document based on their joint probabilities. To achieve an online speed of recommendation while selecting the best prototypes, we propose a novel sparse Gaussian processes (GP) framework for suggesting multiple tags simultaneously. Specifically, a sparse multi-class GP framework is introduced by applying Laplace approximation for the posterior latent function distribution. Laplace approximation [Rasmussen and Williams 2006] has been successfully proposed to address the intractability caused by *binary* GP classification, and we are the first to give a close-form solution for the *sparse* and *multi-class* GP classification. To find the best portion of the training data efficiently, we suggest a prototype selection algorithm that is capable of locating the most informative prototypes for each class within a few learning steps.

While a lot of classifiers are good candidates for the classification of tagged documents, we advocate the use of GP for tag recommendation for a couple of reasons. First, GP have become an important non-parametric tool for classification (and regression). Unlike *generative* classifiers such like Naive Bayes, GP make no assumption on the form of class-conditional density of the data, which makes it immune to any poor performance caused by a false model assumption. Another advantage of GP is that the predicted result of the model yields a probabilistic interpretation, while traditional *discriminative* classifiers such like Support Vector Machines (SVMs) [Cristianini and Shawe-Taylor 2000] usually do not consider the predictive variance of test cases[4]. For tag recommendation where the tagged data

[4]Although Platt suggested an ad-hoc probabilistic SVM in [Platt 2000], it does not consider the predictive variance of the function.

(e.g., web pages) usually does not contain any class labels, the user-assigned tags can be used as labels. In this case, GP classifiers that inherit some level of uncertainty can provide a probabilistic classification which tolerates the limitations and possible errors caused by the tags. The predictive variance also offers flexibility of making predictions to new instances.

As mentioned above, another characteristic of tagged data is the unbounded vocabulary of the tagging systems [Farooq et al. 2007]. Therefore, the tagged data sets used for empirical analysis are usually of high-dimensionality and sparseness [Song et al. 2008]. In this case, the efficiency of the model training should also be considered in addition to the performance issue. Nevertheless, massive training data often requires large memory and high computational cost for most discriminative approaches including SVMs. Ad-hoc methods have been developed to select subset for training but those approaches are somewhat heuristic and often performed outside of the model itself. Instead, the sparse GP framework we developed directly selects a subset of most informative documents from all tagged data during training. The prototype selection algorithm we developed requires no extra cost because it reuses the covariance function developed by the GP framework. Consequently, the GP model shows a very promising performance when limited training resources are available by comparing to SVMs [Rasmussen and Williams 2006].

The remaining of the paper is organized as follows: Section 2 reviews the literature of tag recommendation methods; Section 3 proposes a graph-based approach; Section 4 proceeds with a prototype-based approach; Section 5 presents the results of empirical analysis on three real-world data sets; Section 6 concludes our work.

2. RELATED WORK

For the user-centered approaches, it has been observed that by mining usage patterns from current users, *collaborative filtering* (CF) can be applied to suggest tags from users who share similar tagging behaviors [Golder and Huberman 2006; Begelman et al. 2006]. Specifically, during the *collaborative* step, users who share similar tagging behaviors with the user we want recommend tags to are chosen based on the between-user similarities, which are calculated based on the users' tagging history. This step usually requires a pre-computed look-up table for the between-user similarities, which is usually in the form of weighted symmetric matrices. After that, the *filtering* step selects the best tags from those similar users for recommendation. As discussed above, the drawback of this approach is obvious: a new user that does not have recorded history are unable to benefit from this approach at all since the similarities with existing users cannot be calculated. Moreover, calculating the between-user similarity matrix poses a quadratic computational cost to the number of users. Unfortunately, the whole matrix needs to be re-calculated whenever a new user pattern is injected into the system, making this approach infeasible for web-scale applications.

Among various unsupervised learning methods, clustering technique is of particular popularity for the document-centered approaches. In [Chirita et al. 2007], the authors suggested a method named P-TAG for automatically generating personalized tags in a semantic fashion. They paid particular attention to personalized annotations of web pages. In their document-oriented approach, a web page is

compared with a desktop document using either cosine similarity or latent semantic analysis. Keywords are then extracted from similar documents for recommendation. The second keyword-oriented approach alternatively finds the co-occurrence of terms in different documents and recommends the remaining tags from similar desktop documents to the web page. The third hybrid approach combines the previous two methods. From a collaborative filtering point of view, the first two methods can be interpreted as item-based CF with the item being documents and keywords respectively. Their methods, however, do not investigate the behaviors between different users for similar web pages.

A clustering-based approach was proposed in [Begelman et al. 2006] to aggregate semantically related user tags in to similar clusters. Tags are represented as graphs where each node is a tag and the edge between two nodes corresponds to their co-occurrence in the same documents. Tags in the same cluster were recommended to the users based on their similarities. Similarly, an automatic annotation method for images was proposed in [Li and Wang 2006]. A generative model is trained by exploiting the statistical relationships between words and images. A discrete distribution (D2-) clustering algorithm was introduced for prototype-based clustering of images and words, resulting in a very efficient model for image tagging.

3. APPROACH 1: A GRAPH-BASED METHOD

The graph-based method we proposed consists of four steps: (1) represents the relationship among words, documents and tags into two bipartite graphs, then cut the graph into sub-graphs as topic clusters, (2) ranks the tags within each topic based on their frequency, (3) trains a two-way Poisson mixture model for documents and words, (4) performs a soft classification for a new document and recommend tags with the highest probabilities.

3.1 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 4 depicts an undirected bipartite graph with 4 documents and 5 terms.

3.2 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

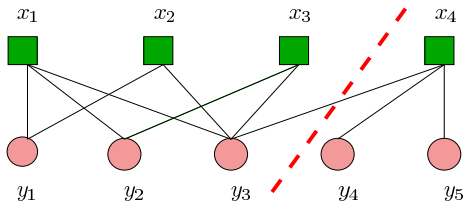


Fig. 4. A bipartite graph of X (documents) and Y (terms). Dot line represents a potential (best) cut of the graph.

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

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

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 [Golub and Loan 1996] which iteratively finds the eigenvalues and eigenvector of square matrices. Given an $n \times n$ sparse symmetric matrix A with eigenvalues:

$$\lambda_1 \geq ... \geq \lambda_n > 0, \tag{2}$$

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:

$$\|A - T\|_F \leq \epsilon \|A\|_F, \tag{3}$$

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.

3.3 Bipartite Graph Partitioning

For multi-clustering on bipartite graphs, we apply the Spectral Recursive Embedding (SRE) algorithm [Zha et al. 2001]. Traditional graph cutting algorithms aimed

at minimizing the cut loss that minimized the weighted mismatch of edges between partitions. Unfortunately, those approaches often lead to unbalanced clusters which are not desirable. Thus, 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:

$$Ncut(A,B) = \frac{cut(A,B)}{W(A,Y)+W(X,B)} + \frac{cut(A^c,B^c)}{W(A^c,Y)+W(X,B^c)}, \tag{4}$$

where

$$\begin{aligned} cut(A,B) &= W(A,B^c) + W(A^c,B) \\ &= \sum_{i\in A, j\in B^c} w_{ij} + \sum_{i\in A^c, j\in B} w_{ij}. \end{aligned} \tag{5}$$

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.

3.4 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 $C(i)$,

$$np_i = \frac{\sum_{j=1}^{n} w_{ij}\mathbb{I}[C(j)=C(i)]}{\sum_{j,k=1}^{n} w_{jk}\mathbb{I}[C(j)=C(k)=C(i)]}, j,k \neq i. \tag{6}$$

where the indicator function $\mathbb{I}[\cdot]$ equals to one if the condition satisfies and 0 otherwise. 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 (6) 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

$$nr_i = \frac{|E_i|}{|E_i| - \sum_{j=1}^{n} \mathbb{I}[C(j)=C(i)]}, \tag{7}$$

where $|E_i|$ represents the total number of edges from node i. 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:

$$Rank_i = \begin{cases} \exp\left(-\frac{1}{r(i)^2}\right) & r(i) \neq 0, \\ 0 & r(i) = 0, \end{cases}$$
$$\texttt{where } r(i) = (np_i) * \log(nr_i). \quad (8)$$

Depicted in Figure 5, our ranking function is a smoothed surrogate that is proportional to both node precision and recall, guaranteed to be in the range of $(0, 1)$. An example cluster is also shown in Figure 5 where the precision of tags $np_1 = 0.75, np_2 = 0.25$, and the recall $nr_1 = 7, nr_2 = 3$. Thus the rank of tag t_1 is higher than t_2, i.e., $t_1 = 0.8, t_2 = 0.1$, indicating that tag t_1 ranks higher in that topic cluster than tag t_2.

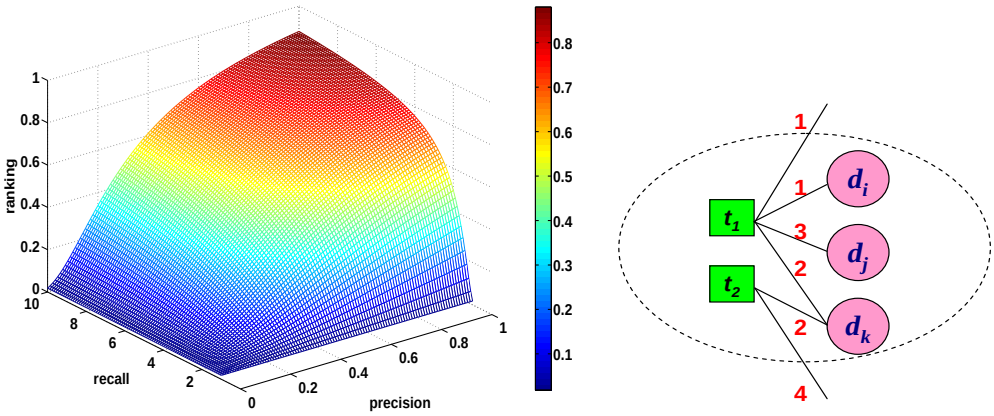


Fig. 5. Smoothed Ranking Function (left) and an example of two-tag three-document cluster (right), with the numbers on the edges showing the frequencies of tags being annotated to specific documents.

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.

3.5 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 6.

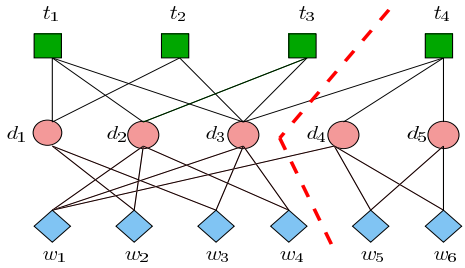


Fig. 6. Two bipartite graphs of documents, words and tags.

The weighted graph can be written as

$$W = \begin{pmatrix} 0 & A & 0 \\ A^T & 0 & B \\ 0 & B^T & 0 \end{pmatrix}, \tag{9}$$

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

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 [Breese et al. 1998]. 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. We summarize our framework in Algorithm 1.

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.

3.6 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

Algorithm 1 Poisson Mixture Model (PMM) Online Tag Recommendation

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
Represent the weighted adjacency matrix W as in eq. (9)
Normalize W using the normalized Laplacian
$L(W) = D^{(-1/2)} W D^{(-1/2)}$ (eq. (1))
Compute a low rank approximation matrix using the Lanczos:
$\tilde{W} \simeq L(W) = Q_k T_k Q_k^T$
Partition $\tilde{W}$ into K clusters using SRE [Zha et al. 2001],
$\tilde{W} = \{\tilde{W_1}, ..., \tilde{W_K}\}$
Assign labels to each document $\mathcal{D}_j, j \in \{1, ...m\}$
$C(\mathcal{D}_j) \in \{1, ..., K\}$
Compute the node rank $Rank(T)$ for each tag $T_{i,k}$ in cluster k, $i \in \{1, ..., n\}, k\{1, ..., K\}$ (eq. (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. (9))
Online Recommendation
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. (16)))
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. (17))

estimation is involved, it is capable for real-time process.

Nevertheless, traditional unsupervised learning approaches of mixture models [Figueiredo and Jain 2002; Schlattmann 2003] 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 [Li and Zha 2006] in order to simultaneously cluster word features and classify documents.

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

$$P(D = d|C = k) = \sum_{m=1}^{M} \pi_m \mathbb{I}(F(m) = k) \prod_{j=1}^{p} \phi(d_j|\lambda_{j,m}), \tag{10}$$

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

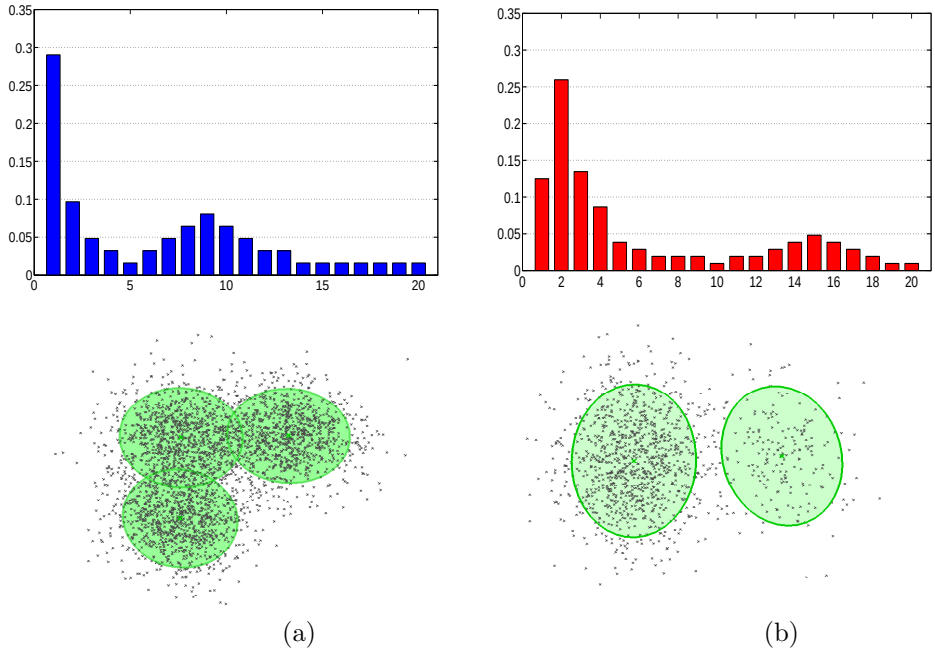


Fig. 7. An example of two mixtures of the Poisson distribution in two clusters.(Top) The histograms of mixture components. (Bottom) Mixture model classification results. (a) Three-component mixtures. (b) Two-component mixtures.

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 7 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 (10) can be simplified as follows (with $L \ll p$):

$$P(D = d|C = k) \propto \sum_{m=1}^{M} \pi_m \mathbb{I}(F(m) = k) \prod_{l=1}^{L} \phi(d_{k,l}|\tilde{\lambda}_{l,m}). \tag{11}$$

3.6.1 *Parameter Estimation.* With the classes determined, we apply EM algorithm [Dempster et al.] 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 $p_{i,m}$:

$$p_{i,m} \propto \pi_m^{(t)} \mathbb{I}(C(i)) \prod_{j=1}^{p} \theta(d(i,j)|\tilde{\lambda}_{m,i,j}^{(t)}). \tag{12}$$

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

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

and update the parameters

$$\pi_m^{(t+1)} = \frac{\sum_{i=1}^{n} p_{i,m}}{\sum_{m'=1}^{M} \sum_{i=1}^{n} p_{i,m'}}, \tag{13}$$

$$\tilde{\lambda}_m^{(t+1)} = \frac{\sum_{i=1}^{n} p_{i,m} \sum_j d(i,j) \mathbb{I}(C(i))}{|d(i,j)| \sum_{i=1}^{n} p_{i,m}}, \tag{14}$$

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:

$$c^{(t+1)}(k,j) = \arg\max_l \sum_{i=1}^{n} \sum_{m=1}^{M} \log(d(i,j)|\tilde{\lambda}_{m,l}^{(t+1)}). \tag{15}$$

3.7 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 (11) and the Bayes rule,

$$\begin{aligned} P(C = k|D = d_t) &= \frac{P(D = d_t|C = k)P(C = k)}{P(D = d_t)} \\ &= \frac{\sum_{m=1}^{M} \pi_m \mathbb{I}(F(m) = k) \prod_{l=1}^{L} \phi(d_{k,l}|\tilde{\lambda}_{l,m}) P(C = k)}{P(D = d_t)}, \end{aligned} \tag{16}$$

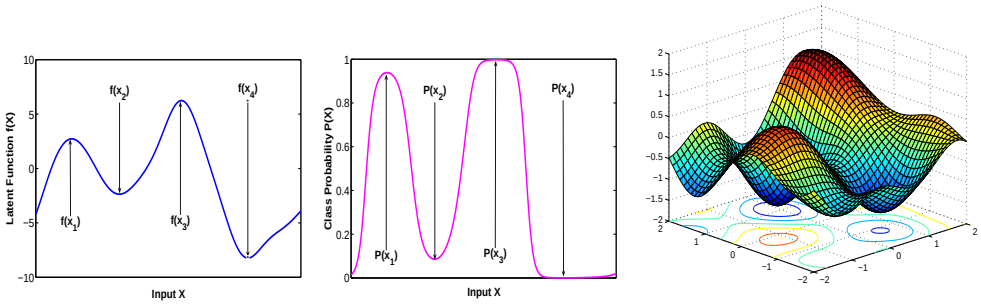
where $P(C = k)$ are the prior probabilities for class k and are set uniform. Finally, the probability for each tag $T_i, i \in \{1, ..., n\}$ to be associated with the sample is

$$R(T_i, d_t) = P(T = T_i|D = d_t) = Rank_{T_i} * P(C = x|D = d_t). \tag{17}$$

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

4. APPROACH 2: A PROTOTYPE-BASED METHOD

The second method we introduce here, a prototype-based method, is made up of three main parts: (1) train a multi-class multi-label Gaussian processes classifier,



(a) (b) (c)

Fig. 8. One-dimensional illustration of Gaussian process construction for classification. (a) A latent function $f(X)$ drawn from Gaussian Process, where $f(x_i)$ denotes the latent function value of point x_i. (b) The class probability of X after scaling $f(X)$ into $(0, 1)$ by a sigmoid function $\Phi(f_i) = 1 + \exp(-f_i)^{-1}$, where $P(x_i)$ denotes the class probability at x_i. (c) An example of two-dimensional input with an independent noise-free covariance function of each input. For the output latent function f, both dimensions are equally important.

(2) find the most informative prototypes (i.e., representatives) for each class, (3) perform a multi-label classification for a new document by assigning it to one or more class, and recommend the highest-ranked tags to the document.

4.1 Background of Gaussian Process Classification

A Gaussian process (GP) is a *stochastic* process consists of a collection of random variables $\mathbf{x}$, which forms a multivariate Gaussian distribution specified by a mean function $\mu(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$. For classification, the objective is to assign a new observation $\mathbf{x}_*$ to one or more predefined classes denoted by $y_* \in \{1, ..., C\}$. GPs can not be applied to the classification task directly because the values of y are not continuous. Consequently, a *latent function* $f(\mathbf{x})$ is employed to infer the labels. The GP prior is therefore placed over $f(\mathbf{x})$. Fig 8 (a) illustrates an one-dimensional case of the latent function with mean 0. To make a prediction given a new $\mathbf{x}_*$, one first determine the predictive distribution $p(\mathbf{f}_*|\mathbf{f})$, where $\mathbf{f}$ is obtained from the training set, $\mathbf{f}|X_{train} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$, with $\mathbf{K}$ denoting the multivariate covariance matrix. The class probability y_* is then related to the latent function $\mathbf{f}_*$.

4.2 Traditional multi-class GP model

Denote a training data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)|i = 1, ..., N\}$ with N training points $X = \{\mathbf{x}_i|i = 1, ..., N\}$ drawn independent and identically distributed (i.i.d.) from an unknown distribution, and the associated labels $\mathbf{y} = \{y_i|i = 1, ..., N\}$, where each point $\mathbf{x}_i$ is a D dimensional feature vector, $\mathbf{x}_i \in \mathbb{R}^D$ and $y_i \in \{1, ..., C\}$. Following the convention in [Rasmussen and Williams 2006], we introduce a vector

of latent function values of N training points for C classes, which has length CN

$$\mathbf{f} = (f_1^1, ..., f_N^1, ..., f_1^j, ..., f_N^j, ..., f_1^C, ..., f_N^C)^T, \tag{18}$$

where $\mathbf{x}_i$ has C latent functions $\mathbf{f}_i = (f_i^1, ..., f_i^C)$. We further assume that the GP prior over $\mathbf{f}$ has the form $\mathbf{f}|X \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$, where $\mathbf{K}$ represents the covariance matrix which is constructed from a pair-wise covariance function $\mathbf{K}(\mathbf{x}_n, \mathbf{x}_{n'}) \stackrel{\triangle}{=} [\mathbf{K}_N]_{nn'}$. Specifically, $\mathbf{K}$ is block diagonal of size $CN \times CN$ in the matrices $\mathbf{K}_1, ..., \mathbf{K}_C$, where each $\mathbf{K}_j$ represents the correlations of the latent function values within class j. A wide range of covariance functions can be chosen for GP classification [Rasmussen and Williams 2006].A commonly used function in the classification case is the *squared exponential* function, defined as:

$$[\mathbf{K}_N]_{nn'} = l \exp\left(-\frac{1}{2} \frac{\sum_{d'=1}^{D} \left(x_n^{(d')} - x_{n'}^{(d')} \right)^2}{\mathbf{\Sigma}^2} \right), \tag{19}$$

where $\theta = \{l, \mathbf{\Sigma}^2\}$ corresponds to the *hyper-parameters*.

Given the training set $\mathcal{D}$, we can compute the posterior of the latent function by plugging in the Bayes' rule,

$$p(\mathbf{f}|X, y) = \frac{p(\mathbf{f}|x)p(y|\mathbf{f})}{p(X, y)} \stackrel{i.i.d.}{=} \frac{\mathcal{N}(\mathbf{0}, K)}{p(X, y)} \prod_{i=1}^{N} p(\mathbf{y}_i|\mathbf{f}_i), \tag{20}$$

which is non-Gaussian. In eq.(20), the conditional probability $p(\mathbf{y}|\mathbf{f})$ has not been decided yet. In the multi-class case, $\mathbf{y}$ is a vector of the length CN (which is the same as $\mathbf{f}$), which for each $i = 1, ..., N$ has an entry of 1 for the class which corresponds to the label of the point $\mathbf{x}_i$ and 0 for the rest $C-1$ entries. One of the choices is a *softmax* function:

$$p(y_i^c|\mathbf{f}_i) = \frac{\exp(f_i^c)}{\sum_{c'} \exp(f_i^{c'})}. \tag{21}$$

To proceed, we compute the predictive distribution of the class probability given a new $\mathbf{x}_*$ in two steps. First, compute the latent value $\mathbf{f}_*$ by integrating out $\mathbf{f}$:

$$p(\mathbf{f}_*|X, y, \mathbf{x}_*) = \int p(\mathbf{f}_*|\mathbf{f}, X, \mathbf{x}_*) \underbrace{p(\mathbf{f}|X, y)}_{\text{eq.(20)}} d\mathbf{f}, \tag{22}$$

then $\mathbf{y}_*$ can be computed by integrating out $\mathbf{f}_*$:

$$p(\mathbf{y}_*|X, y, \mathbf{x}_*) = \int p(\mathbf{y}_*|\mathbf{f}_*) \underbrace{p(\mathbf{f}_*|X, y, \mathbf{x}_*)}_{\text{eq.(22)}} d\mathbf{f}_*. \tag{23}$$

This method takes $O(N^3)$ to train due to the inversion of the covariance matrix $\mathbf{K}$. A range of *sparse* GP approximations have been proposed [Lawrence et al. 2003; Seeger and Jordan]. Most of these methods seek a subset of M ($M \ll N$) training points which are *informative* enough to represent the entire training set. Consequently, the training cost is reduces to $O(NM^2)$ and the corresponding test cost to $O(M^2)$. Next we discuss a sparse way to reduce the computational cost in the multi-class case.

4.3 Our Multi-class Sparse GP Model

Our model involves several steps. First, we choose M ($M \ll N$) points (denote as $\bar{X} = \{\bar{\mathbf{x}}_m\}_{m=1}^{M}$) from the training set. Then we generate their latent functions $\bar{\mathbf{f}}$ from the prior. The corresponding $\mathbf{f}$ for the entire training set is thus drawn conditionally from $\bar{\mathbf{f}}$. See Figure 9 for details.

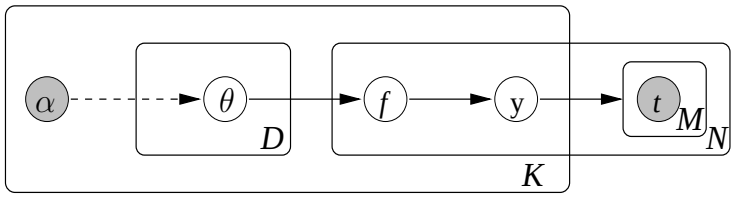


Fig. 9. Graphical representation of our sparse multi-class GP model. θ is the hyper-parameter that define the latent function f. α denotes the extra parameter for placing a distribution over θ.

First, assume that the M points have already been chosen. Then place a GP prior on $\bar{X}$, which uses the same covariance function as shown in eq. (19), such that these points have a similar distribution to the training data,

$$p(\bar{\mathbf{f}}|\bar{X}) = \mathcal{N}(\bar{\mathbf{f}}|\mathbf{0}, \mathbf{K}_M). \tag{24}$$

Given a new $\mathbf{x}_*$, we utilize M latent functions $\bar{\mathbf{f}}$ for prediction. We compute the latent values $\mathbf{f}_*$ by integrating the likelihood with the posterior:

$$p(\mathbf{f}_*|\mathbf{x}_*, X, \mathbf{y}, \bar{\mathbf{f}}, \bar{X}) = \int \underbrace{p(\mathbf{f}_*|\mathbf{x}_*, \bar{\mathbf{f}}, \bar{X})}_{A} \underbrace{p(\bar{\mathbf{f}}|X, \mathbf{y}, \bar{X})}_{B} d\bar{\mathbf{f}}, \tag{25}$$

where A represents the single data likelihood by applying to the reduced set of points. With $\bar{\mathbf{f}}$ determined, the likelihood can be treated as a bivariate normal distribution, which follows a normal distribution:

$$\mathbf{f}_*|\mathbf{x}_*, \bar{\mathbf{f}}, \bar{X} \sim \mathcal{N}(\mathbf{f}_*|\mathbf{k}_{\mathbf{x}_*}^T \mathbf{K}_M^{-1} \bar{\mathbf{f}}, K_{\mathbf{x}_*\mathbf{x}_*} - \mathbf{k}_{\mathbf{x}_*}^T \mathbf{K}_M^{-1} \mathbf{k}_{\mathbf{x}_*}), \tag{26}$$

where $\mathbf{k}_{\mathbf{x}_*} = \mathbf{K}(\bar{\mathbf{x}}, \mathbf{x}_*)$ and $[\mathbf{K}_M]_{ij} = \mathbf{K}(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j)$.

Nevertheless, the problematic form of posterior B does not follow a normal distribution and has to be approximated.

4.4 Laplace Approximation for the Posterior

Our method to approximate B in eq.(25) is based on the Laplace approximation, which were used in [Rasmussen and Williams 2006] for binary classification. Using the Bayes' rule,

$$\begin{aligned} p(\bar{\mathbf{f}}|X, \mathbf{y}, \bar{X}) &= \frac{p(\bar{\mathbf{f}}|\bar{X}) p(\mathbf{y}|\bar{\mathbf{f}}, X, \bar{X})}{p(\mathbf{y}|X, \bar{X})} \\ &= \frac{p(\bar{\mathbf{f}}|\bar{X}) \int \overbrace{p(\mathbf{f}|\bar{\mathbf{f}}, X, \bar{X}) p(\mathbf{y}|\mathbf{f})}^{C} d\mathbf{f}}{p(\mathbf{y}|X, \bar{X})}. \end{aligned} \tag{27}$$

The detail of the derivation is long and available at the Appendix. The approximated mean and variance of eq.(27) is:

$$\begin{aligned} \boldsymbol{\mu}_* &\simeq \boldsymbol{\mu}_p, \\ \boldsymbol{\Sigma}_* &= \mathbf{K}_M + \boldsymbol{\Sigma}_p. \\ \text{where } \boldsymbol{\mu}_p &= \frac{\mathbf{Q}^{-1}\mathbf{P}}{2}, \boldsymbol{\Sigma}_p = \mathbf{Q}^{-1}, \\ \mathbf{Q} &= (\mathbf{K}_{NM}\mathbf{K}_M^{-1})^T \boldsymbol{\Lambda}^{-1}(\mathbf{K}_{NM}\mathbf{K}_M^{-1}) + \mathbf{K}_M, \\ \mathbf{P} &= \hat{\mathbf{f}}^T \boldsymbol{\Lambda}^{-1}(\mathbf{K}_{NM}\mathbf{K}_M^{-1}). \end{aligned} \tag{28}$$

4.4.1 *Determine the class label of test documents.* The final step is to assign a class label to the observation $\mathbf{x}_*$, given the predictive class probabilities by integrating out the latent function f_*:

$$p(\mathbf{y}_*|\mathbf{x}_*, X, \mathbf{y}, \bar{\mathbf{f}}, \bar{X}) = \int \tilde{p}(\mathbf{f}_*|\mathbf{x}_*, X, \mathbf{y}, \bar{\mathbf{f}}, \bar{X}) p(\mathbf{y}_*|\mathbf{f}_*) df_*, \tag{29}$$

which again cannot be solved analytically. One way to approximate is to use cumulative Gaussian likelihood. In [Rasmussen and Williams 2006], the authors estimated the mean prediction by drawing S samples from the Gaussian $p(\mathbf{f}_*|\mathbf{y})$, softmax and averaging the results. Once the predictive distribution of the class probability is determined, the final label of $\mathbf{x}_*$ can be decided by choosing the maximum posterior (MAP):

$$t(\mathbf{x}_*) = \arg\max_c p(y(\mathbf{x}_*)^c|\cdot), \qquad c = 1, ..., C. \tag{30}$$

4.5 Informative Points Selection

It remains to optimize the parameters $\Theta = \{\theta, \bar{X}\}$, which contain the hyperparameters $(l, \boldsymbol{\Sigma})$ for the covariance matrix $\mathbf{K}$ as well as finding the subset $\bar{X}$ of M points. Traditionally, they are optimized jointly by optimizing the marginal likelihood of the training data. In our approach, we instead treat them individually.

4.5.1 *Parameter Inference for the Covariance Matrix.* The marginal likelihood of $\mathbf{y}$ can be obtained by integrating out $\bar{\mathbf{f}}$,

$$p(\mathbf{y}|X, \bar{X}, \Theta) = \int p(\mathbf{y}|X, \bar{X}, \bar{\mathbf{f}}) p(\bar{\mathbf{f}}|\bar{X}) d\bar{\mathbf{f}} = \int \exp(\mathcal{L}(\bar{\mathbf{f}})) d\bar{\mathbf{f}}. \tag{31}$$

With a Taylor expansion of $\mathcal{L}(\bar{\mathbf{f}})$ around $\hat{\bar{\mathbf{f}}}$ we find

$$\mathcal{L}(\bar{\mathbf{f}}) \simeq \mathcal{L}(\hat{\bar{\mathbf{f}}}) + \underbrace{(\bar{\mathbf{f}} - \hat{\bar{\mathbf{f}}}) \nabla_{\bar{\mathbf{f}}} \mathcal{L}(\bar{\mathbf{f}})}_{=0} + \frac{1}{2}(\bar{\mathbf{f}} - \hat{\bar{\mathbf{f}}})^T \nabla\nabla_{\bar{\mathbf{f}}} \mathcal{L}(\bar{\mathbf{f}})(\bar{\mathbf{f}} - \hat{\bar{\mathbf{f}}}).$$

Therefore, the approximation of the marginal likelihood can be written as

$$\begin{aligned} &p(\mathbf{y}|X, \bar{X}, \Theta) = \\ &\exp(\mathcal{L}(\hat{\bar{\mathbf{f}}})) \int \exp\left(\frac{1}{2}(\bar{\mathbf{f}} - \hat{\bar{\mathbf{f}}})^T \nabla\nabla_{\bar{\mathbf{f}}} \mathcal{L}(\bar{\mathbf{f}})(\bar{\mathbf{f}} - \hat{\bar{\mathbf{f}}})\right) d\bar{\mathbf{f}}. \end{aligned} \tag{32}$$

The log marginal likelihood can be obtained by taking logarithm on both sizes

of the above equation,

$$\log p(\mathbf{y}|X,\bar{X},\Theta) = \mathcal{L}(\hat{\bar{\mathbf{f}}}) - \frac{CN}{2}\log 2\pi - \frac{1}{2}\log|\nabla\nabla_{\bar{\mathbf{f}}}\mathcal{L}(\bar{\mathbf{f}})|, \tag{33}$$

which can be maximized w.r.t. the parameters Θ to obtain $\hat{l}$ and $\hat{\mathbf{\Sigma}}$. Note that each $\mathbf{\Sigma}_c$ is a $D \times D$ symmetric matrix, where D is the number of dimensions. We assume that each dimension is independent, thus simplifies $\mathbf{\Sigma}_c$ to be a diagonal matrix. However, this still yields DC parameters to estimate for $\mathbf{\Sigma}$. Therefore, we further assume that within each class c, the covariance of each dimension is the same, so that the total number of parameters for $\mathbf{\Sigma}_c$ is reduced to C.

4.5.2 *Prototype selection for $\bar{X}$.* The original gradient calculation in eq.(33) is very complicated. However, we can simplify it with the assumption made on the covariance matrix. Since each $\mathbf{\Sigma}_c$ is now independent of each other, we can estimate the locations of the active points regardless of the choices of l and $\mathbf{\Sigma}$. We greedily find the locations of $\bar{X}$ by stochastic gradient descent method. This is similar to finding the optimal *prototypes* for each class, which is a subset of points that contains enough information for each class. Our method for optimal prototype search is parallel to [Seo et al. 2003], which is used for K-nearest neighbor classification. We select a set of M prototypes by minimizing the misclassification rate of the training set,

$$\mathfrak{L}(X,\bar{X}) = \frac{1}{N}\sum_{n=1}^{N}\sum_{m=1}^{M} P(\bar{\mathbf{x}}_m|\mathbf{x}_n)(1 - \mathbb{I}(\bar{y}_m = y_n)), \tag{34}$$

where the indicator function $\mathbb{I}$ is 1 if the condition is hold and 0 otherwise. The likelihood $P(\bar{\mathbf{x}}_m|\mathbf{x})$ can be calculated by plugging in the normalized covariance:

$$P(\bar{\mathbf{x}}_m|\mathbf{x}) = \frac{\mathbf{k}_{\bar{\mathbf{x}}_m\mathbf{x}}}{\sum_{m'=1}^{M}\mathbf{k}_{\bar{\mathbf{x}}_{m'}\mathbf{x}}}. \tag{35}$$

We can further rewrite the loss function in eq.(34) by removing the indicator function:

$$\mathfrak{L}(X,\bar{X}) = \frac{1}{N}\sum_{n}\underbrace{\sum_{\{m:\bar{y}_m\neq y_n\}} P(\bar{\mathbf{x}}_m|\mathbf{x}_n)}_{\mathfrak{l}_m}, \tag{36}$$

where $\mathfrak{l}_m$ indicates the individual cost of misclassification, which is continuous in the interval $(0,1)$. Therefore, it can be minimized by gradient descent w.r.t. $\bar{X}$,

$$\begin{aligned}
&\bar{\mathbf{x}}_m(t+1)\\
=\ &\bar{\mathbf{x}}_m(t) - \alpha(t)\nabla_{\bar{\mathbf{x}}_m}\mathfrak{l}_m(t)\\
=\ &\bar{\mathbf{x}}_m(t) + \alpha(t)p(\bar{\mathbf{x}}_m|\mathbf{x})\left(\mathbb{I}(\bar{y}_m \neq y_n) - \mathfrak{l}_m(t)\right)\frac{\delta\mathbf{k}_{\bar{\mathbf{x}}_m\mathbf{x}}}{\delta\bar{\mathbf{x}}_m}.\\
=\ &\bar{\mathbf{x}}_m(t) + \begin{cases} \mathfrak{l}_m(1-\mathfrak{l}_m)P(\bar{\mathbf{x}}_m|\mathbf{x})(\bar{\mathbf{x}}_m - \mathbf{x}) & \text{if } \bar{y}_m \neq y_n\\ -\mathfrak{l}_m(1-\mathfrak{l}_m)P(\bar{\mathbf{x}}_m|\mathbf{x})(\bar{\mathbf{x}}_m - \mathbf{x}) & \text{otherwise}\end{cases}
\end{aligned}$$

Here $\alpha(t) > 0$ is a small enough number which specifies the step length of the descent. The program stops when a stopping criterion is reached. We further

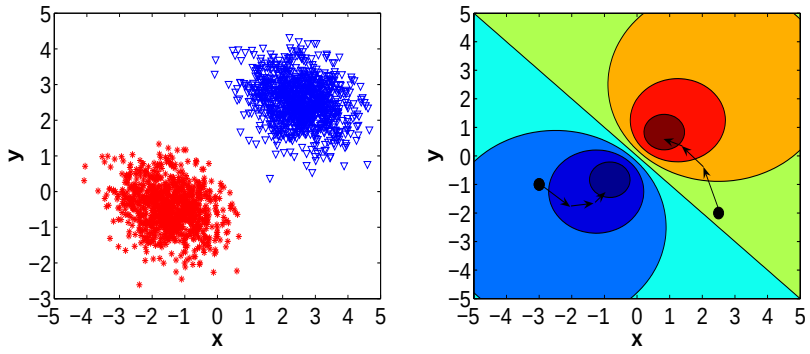


Fig. 10. An example of prototype selection with $M = 2$. Left figure shows the original distribution; right figure, contour-plots the results of descent where black dots are the starting points.

notice that only those points falling into a particular area of the input space can contribute to the update of the prototypes. This fact is explained as the *window* rule in [Kohonen 2001]. So we can speed up the prototype updates by searching over those points only. Figure 10 shows an example of two prototypes. It can be seen that after three steps of descent, our algorithm successfully finds informative points for each class.

For brevity we hyphenate our method as Sparse Gaussian process with Prototype Selection (SGPS).

4.6 Discussion of the Computational Cost

The most influential part on the computational cost is the inversion of the covariance matrix $\mathbf{K}$ which takes $O(N^3)$ time. In the sparse framework, however, it should be noticed that only the covariance matrix for the M prototypes is required to be inverted, which refers to $\mathbf{K}_M$ in our case. To be exact, $\mathbf{K}_M$ needs to be inverted when calculating $\mathbf{\Lambda}$ in eq.(40), $\mathbf{f}'$ in eq.(46), as well as $\mathbf{Q}$ and $\mathbf{P}$ in eq.(48). For efficient inversion, Cholesky decomposition is often employed [Rasmussen and Williams 2006], which ensures that for N training points distributed in C classes, the training stage can be realized in $O(M^2NC)$ time with M prototypes, likewise $O(M^2C)$ per prediction. In practice, the Cholesky decomposition is only required to be computed once for a training pass, which can then be saved and used in other equations efficiently. So it almost costs linear time for training a data set with N points.

As for the cost of prototype selection, since the updates re-uses covariance matrix in eq.(35), no additional storage and computation are required. Therefore, eq.(37) can be efficiently updated in at most $O(NC)$ time.

4.7 Application to Multi-label Tag Suggestion

So far, we have only considered the case that the each observation is single-labeled, i.e., belongs to only one class. In fact, many real-world problems are multi-labeled. In the case of tagged data, each tag associated with a document may be treated as

Algorithm 2 Multi-label Multi-class Sparse GP Classification (MMSG) for Tag Recommendation

Input: training data $\mathcal{D}: \{(\mathbf{x}_i, \mathbf{y}_i)\}_1^N, \mathbf{x}_i \in \mathbb{R}^d, \mathbf{y}_i = \{y_{i1}, ..., y_{i\tilde{K}}\}$
M: number of prototypes
$\mathbf{k}$: covariance function
begin training procedure
for $i = 1 : N$
  $c_i = \max(s(\mathbf{y}_i))$ //decide the category of $\mathbf{x}_i$
end for
Train a GP classifier given $\{(\mathbf{x}_i, c_i)_1^N, M, \mathbf{k}\}$
Output: $\bar{X}, \bar{\mathbf{f}}$
begin test procedure
Input: a test object $\mathbf{x}_*$
Decide its category probabilities $\mathbf{c}_*$ given $\bar{X}, \bar{\mathbf{f}}$ (eq.(29))
for each category $m \in \{1, ..., C\}$
  for each label $y_{ij}^{(c)} \in \{y_{i1}^{(c)}, ..., y_{i\tilde{K}}^{(c)}\}$
    $P(y_{ij}^{(c)}|\mathbf{x}_*) = Rank_{y_{ij}}^{(c)} \cdot c_m(\mathbf{x}_*)$
  end for
end for
Output: $P(\mathbf{y}_*, \mathbf{x}_*)$

a label, which may or may not refer to the same topic as other labels. Thus, the problem of tag suggestion can be transformed into a multi-label classification problem where the objective is to predict the probability of a document with all possible tags (labels) given a fixed tag vocabulary and associated training documents.

The problem of multi-label classification (MLC) is arguably more difficult than the traditional single-label classification task, since the number of combinations for two or more classes is exponential to the total number of classes. For N classes, the total number of possible multi-labeled class is 2^N, making it unfeasible to expand from an algorithm for single-label problems. Much research has been devoted to increasing the performance of MLC and generalize the framework to single-label classification; see related work for more information[Tsoumakas and Katakis 2007].

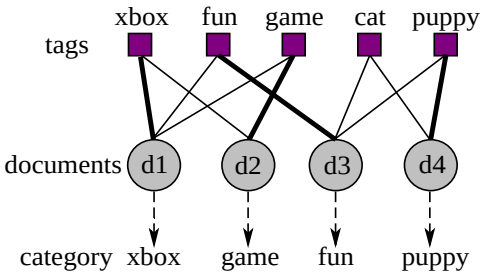


Fig. 11. Example of document-tag graph. Each document is associated with multiple tags. Tag with the highest frequency is treated as the category of that document (shown in bold line).

As pointed out in [Brinker et al. 2006], multi-label classification can be treated as a special case of label ranking, which can be realized if the classifiers provide real-valued confidence scores or a posterior probability estimates for classification outcomes. Thus, the multi-class SGPS model readily maps to this problem, since the output vector $\mathbf{y}_*$ contains real-valued scores of the posterior class probabilities. Specifically, in the multi-label case, we assume that the class label of a training instance $\mathbf{x}_i$ is no longer a binary value, but rather a vector $\mathbf{y}_i$ of binary values where each y_{ij} denotes the existence/absence of $\mathbf{x}_i$ in class j. We further assume that these class probabilities can be ranked according to their values, where $s(y_{im}) > s(y_{in})$ indicates that y_{im} is preferred to y_{in}. In the context of tags, the value of a tag is defined as the number of times it has been used to annotate the specific object. So if a document d_1 (cf Figure 11) is tagged 4 times with *game*, 3 times with *fun* and 5 times with *xbox*, we can rearrange the labels in the descending order, yielding, { *xbox*(5), *game*(4), *fun*(3) }. Note that normalization is usually required to ensure the well-defined class probability, thus the class probabilities of the above case become $\{0.42, 0.33, 0.25\}$. Figure 11 shows an example of 4 documents and 5 tags with their categories in bold lines.

In this way we can transform multi-class multi-label classification into *multi-category single-label* classification. Specifically, we first assign each $\mathbf{x}_i$ into a single *category* c which corresponds to its top-ranked label (e.g., in the above case, the category is *xbox*). Each category contains a set of labels that belong to the objects in that category. Intuitively, tags that belong to the same category are more semantically related than tags in different categories, i.e., tags in the same category have a higher co-occurrence rate. However, it should be noted that an individual tag could belong to multiple categories, e.g., in Figure 11, *fun* appears in two categories. The above two phenomenon can be roughly explained by the behavior of *polysemy* and *synonymy* in linguistics. Table I shows three ambiguous tags and their corresponding categories in one of our experiments.

tags	**categories**
apple	mac **apple** computers osx technology IT food health **apple** nutrition fruit green
tiger	photos nature animal **tiger** cute animals sports video **tiger** woods golf games
opera	music art **opera** culture design download software browser **opera** web tools internet

Table I. Example of ambiguous tags from del.icio.us.

Given a training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_1^N$, the within-category scores of all possible labels are defined as

$$Rank_{y_{i'}}^{(c)} = \frac{1}{Z^{(c)}} \sum_{i:\mathbf{x}_i \in c} \sum_j s(y_{ij}) \mathbb{I}(y_{ij} = y_{i'}), y_{i'} = \{y_{i1}, ..., y_{i\tilde{K}}\} \tag{37}$$

where $Z^{(c)}$ is a normalization factor for category c. We summarize this approach in Algorithm 2, $\tilde{K}$ refers to the total number of possible labels. During the training

phase, we train an SGPS model for C categories, as well as calculating the within-category scores for all labels. In the test phase, we use the model first to determine the probabilistic distribution of the categories given a new test case. Then combine this evidence with the within-category scores of tags in a multiplicative fashion to obtain the final label distribution. The labels are sorted in descending order based on the estimated likelihoods, the top-ranked tags are used for recommendation. Figure 12 illustrates the process.

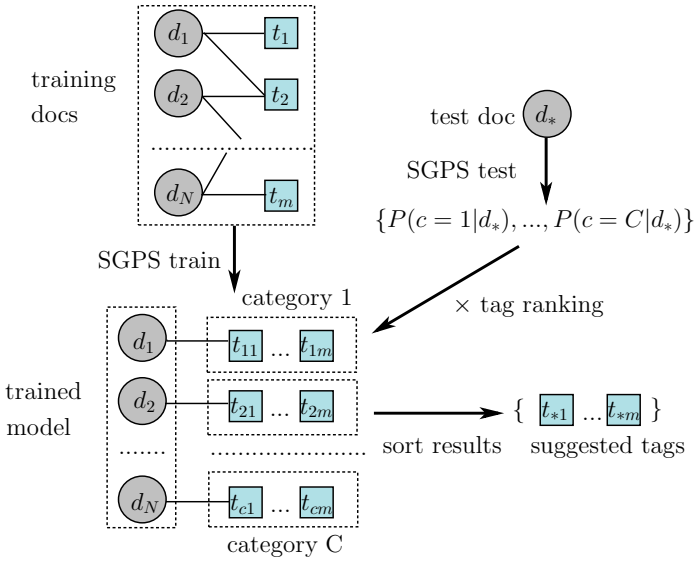


Fig. 12. The training and test processes of MMSG. Each d_i is a document and each t_i is a tag.

5. EXPERIMENTS

To assess the performance of the two proposed frameworks, we empirically analyze them using real-world data sets in this section. We will focus on the quality of the tagging results as well as the efficiency of the tagging algorithms[5].

5.1 Evaluation Metrics

In addition to the standard precision, recall, F-score and Kendall τ rank correlation metric [Kendall 1938] that measures the degree of correspondence between two ranked lists, we also propose the following metrics to measure the effectiveness of tagging performance.

—*Top-k accuracy:* Percentage of documents correctly annotated by *at least* one of the top kth returned tags.

[5]Other experimental results such as the performance of the sparse Gaussian processes model and the multi-class multi-label algorithm on bench-mark data sets are available in [Song et al. 2008].

—*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.

5.2 Data Sets

For evaluation, we made an effort to acquire three data sets from several most popular tagging websites.

CiteULike is a website for researchers to share scientific references by allowing users to specific their personal tags to the papers. We acquired the tagged data set from CiteULike for over two years from November 15, 2004 to February 13, 2007. We mapped the data set to papers that are indexed in CiteSeer[6] 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.

Del.icio.us is one of the largest web2.0 web sites that provides services for users to share personal bookmarks of web pages. We subscribed to 20 popular tags in del.icio.us, 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 data set is slightly over 2GB.

BibSonomy is a newly developed web 2.0 site which provides the sharing of social bookmarks for both web pages and scientific publications. We collected data from BibSonomy between Oct 15 2007 and Jan 10 2008. We randomly sampled 50 tags from the tag lists. For each tag, we retrieved the content of bookmarks with related tags. Overall, the BibSonomy data set contains 14,200 unique items with 37,605 words. The total number of tags is 6,321.

Table II shows top 10 tags for all three data sets[7]. For preprocessing, we considered the temporal characteristics of tags and ordered the data by time and used the earlier data for training and tested on later data. We performed experiments with training data from 10% to 90%.

5.3 Comparison to Other Methods

We compare the performance of tag recommendation of our algorithm with three other approaches.

The first unsupervised learning method we consider is the classic collaborative filtering algorithm [Breese et al. 1998]. The Vector Similarity (VS) approach is used to calculate the similarity between documents, which computes the cosine similarity between a query Q and each training document D_i, $Sim(Q, D_i) =$

[6]http://citeseer.ist.psu.edu/

[7]All data sets are available upon request.

CiteULike		del.icio.us		BibSonomy	
Tag Name	**Frequency**	**Tag Name**	**Frequency**	**Tag Name**	**Frequency**
clustering	245	internet	1743	tools	2459
p2p	220	technology	1543	computing	2294
logic	185	java	1522	software	1974
network	175	software	1473	blog	1717
learning	175	web	1429	internet	1647
haskell	166	photography	1375	web	1631
web	162	news	1328	analysis	1562
distributed	151	music	1291	data	1248
algorithm	142	business	1115	search	1196
algorithms	140	travel	1092	design	1117

Table II. Top 10 most popular tags in CiteULike, del.icio.us and BibSonomy with respective frequencies.

$\frac{\sum_i n(Q,j)n(i,j)}{\sqrt{\sum_j n(Q,j)^2}\sqrt{\sum_i n(i,j)^2}}$, where $n(i,j)$ represents the count of j's word in document 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 [Kullback and Leibler 1951] (VS+IG) to select roughly 5% of the total features.

The second method we compare to is the famous topic model by Blei [Blei et al. 2003], namely Latent Dirichlet Allocation (LDA). For tag recommendation, we first trained a n-topic LDA model [Blei et al. 2003], where n is decided by the number of tag categories. The posterior probability of $P(topic|doc)$ is then used to determine the similarity between a test document and the training ones. Tags are therefore suggested to the new document from the most similar training documents.

The last method we consider here is a variant of the supervised learning method Support Vector Machine (SVM). We choose SVM for comparison because it has been shown that SVM usually outperforms other classifiers for text classification [Cristianini and Shawe-Taylor 2000]. We first use SVM^{struct} to train a multi-label SVM model for the training documents[8], and then use the same ranking function as in eq.(37) to return top ranked tags for recommendation.

5.4 Quality of the Tagging Performance

Table III lists the top user tags for each of the top 8 papers, as well as the top 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

[8]http://www.cs.cornell.edu/People/tj/svm_light/svm_struct.html

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	*PMM:* **search**, **web**, rank, mining, **pagerank**, page, rank, www, **ir** *MMSG:* **google**, **ir**, mining, www, **pagerank**, **ranking** learning, **web**, algorithm
The Anatomy of a Large-Scale Hypertextual Web Search Engine (Sergey Brin et al.)	94	google, search, pagerank, web, engine, www, web-search, ir, graphs	*PMM:* **search**, **web**, **engine**, **www**, page, rank, **ir**, classification, mining *MMSG:* **google**, ranking, **www** **ir**, algorithm, **web** algorithms, network, social
ReferralWeb: Combining Social Networks and Collaborative Filtering (Henry Kautz et al.)	88	folksonomy, collaboration, tagging, social-networks, networks, social, filtering, recommender, network	*PMM:* **networks**, network, adhoc, mobile, mobilitymodel, **filtering**, **tagging**, **social**, socialnetwork *MMSG:* **networks**, **network**, ir, web, **social**, **recommender**, algorithm, **tagging**, learning
A Tutorial on Learning With Bayesian Networks (David Heckerman)	78	bayesian, networks, learning, network, statistics, bayes, tutorial, modeling, graphs, algorithms	*PMM:* **bayesian**, **networks**, **learning**, **network**, **bayes**, learn, **modeling**, data, **graphical** *MMSG:* **network**, **networks**, **learning**, **bayes**, algorithm, web, **algorithms**, search, mining
Maximizing the Spread of Influence through a Social Network (David Kempe et al.)	73	social, influence, network, socialnetworks, diffusion, research, spread, networking	*PMM:* **network**, networks, **social**, **socialnetworks**, adhoc, models, machinelearning, algorithm, data *MMSG:* **network**, networks, web, learning, **social**, mining, algorithm, **research**, data
Authoritative Sources in a Hyperlinked Environment (Jon M. Kleinberg)	47	ranking, hyperlink, web, search, www, ir, graphs, clustering, hub, authority, hyperlinks	*PMM:* **web**, **search**, data, query, **hyperlink**, communities, engine, information, extraction *MMSG:* **ir**, **web**, **www**, mining, learning, algorithm, **ranking**, rank, **search**
Indexing by Latent Semantic Analysis (Scott Deerwester et. al.)	45	lsi, indexing, ir, lsa, semantics, semantic, information-retrieval, latent, language, index	*PMM:* **index**, svd, data, query, theory, clustering, information, retrieval, learning *MMSG:* **ir**, data, **indexing**, web, www, algorithm, ranking, query, **index**
The Small-World Phenomenon: An Algorithmic Perspective (Jon M. Kleinberg)	43	small-world, networks, network, web, social, webgraph, power-law, ir, algorithm, graphs, graph	*PMM:* **networks**, **network**, **web**, **algorithm**, algorithms, **graphs**, data, www, **ir** *MMSG:* **network**, **networks**, **ir**, www, **web**, **algorithm**, **social**, ranking, **graph**

Table III. Top 8 most popular papers from CiteULike data. The top 9 recommended tags are listed as "Our Tags". Tags with bold font match one of the user-annotated tags.

paper "A Tutorial on Learning With Bayesian Networks", which has a Kendall τ rank of 0.78.

Algorithm	Precision	Recall	F-Score	Kendall τ rank
CiteULike				
VS+IG	25.88%	36.57%	30.18%	0.13
LDA	29.15%	43.33%	36.71%	0.19
SVM^{struct}	33.21%	50.17%	43.25%	0.29
PMM	39.17%	56.35%	49.96%	0.37
MMSG	40.27%	59.11%	51.08%	0.41
delicious				
VS+IG	27.66%	39.05%	32.16%	0.09
LDA	32.71%	48.33%	42.95%	0.18
SVM^{struct}	40.21%	61.44%	50.63%	0.25
PMM	43.52%	62.31%	52.77%	0.37
MMSG	47.38%	66.16%	54.23%	0.44
BibSonomy				
VS+IG	25.11%	40.05%	36.90%	0.13
LDA	31.75%	49.68%	42.17%	0.28
SVM^{struct}	33.45%	52.93%	45.56%	0.33
PMM	35.21%	55.72%	47.23%	0.37
MMSG	39.45%	57.01%	52.32%	0.39

Table IV. Tagging performance.

We present a summary of the experimental results in Table IV. Overall, our models PMM and MMSG exhibit better performance for all three data sets. On average, PMM and MMSG performs 3.2 times better than VS+IG, 2.1 times better than LDA, and 1.3 times better than SVM. Note that for MMSG, the performance is efficiently achieved by using only 5% of the training instances.

In addition, we also examined the performance of individual tags by looking at the top 10 suggested tags. We are interested in the difference in performance between popular tags (e.g., web, network, clustering) and rare tags (e.g., asp.net, latex, 3d). For each data set, we chose the top-5 most/least popular tags and averaged the suggesting results. Figure 13 depicts the results. It can be observed that MMSG and PMM outperform SVM and others in most cases. We notice that while SVM is comparable to MMSG and PMM for popular tags, our algorithm shows a clear edge over SVM for rare tags, with more than 18% and 15% improvement respectively. Since rare tags appear in fewer documents, this result gives credibility to the claim that MMSG works well with very few training instances.

5.4.1 *Model Selection for Tag Suggestion.* Next we quantitatively show how the model selection reflects the performance of tag suggestion. In the graph-based method, parameters include number of topic clusters K, number of mixture model components M and number of word clusters L. In our experimental setting, we select these parameters by performing cross validation on the training set.

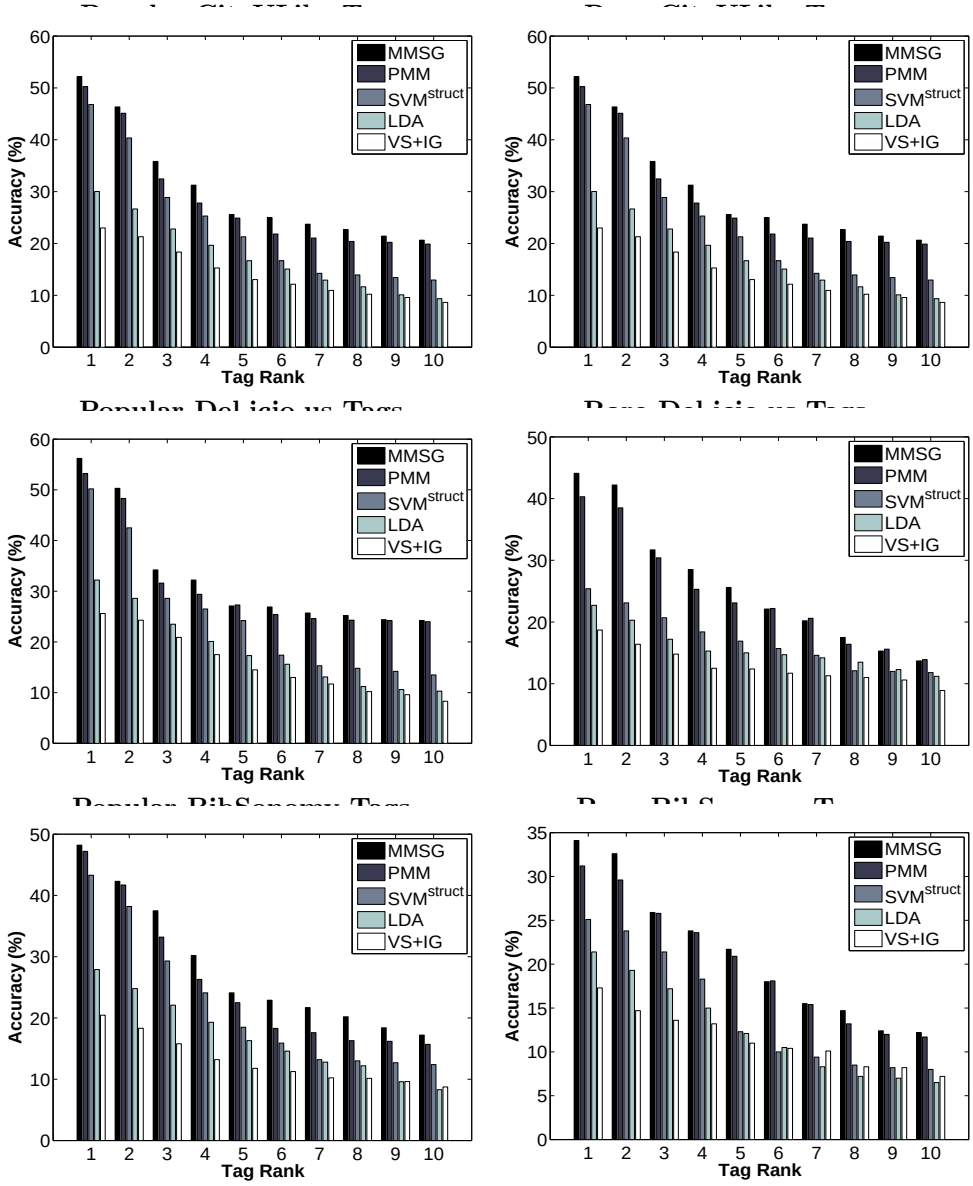


Fig. 13. Tag suggestion results on popular and rare tags for CiteULike, Delicious and BibSonomy.

In the prototype-based framework, model selection involves the decision of (1) the number of prototypes, (2) the covariance function and (3) the hyper-parameters. Since the hyper-parameters are often associated with the covariance function and can be chosen by optimizing the marginal likelihood of the training data, we then focus on how (1) and (2) affect the performance. A common covariance function

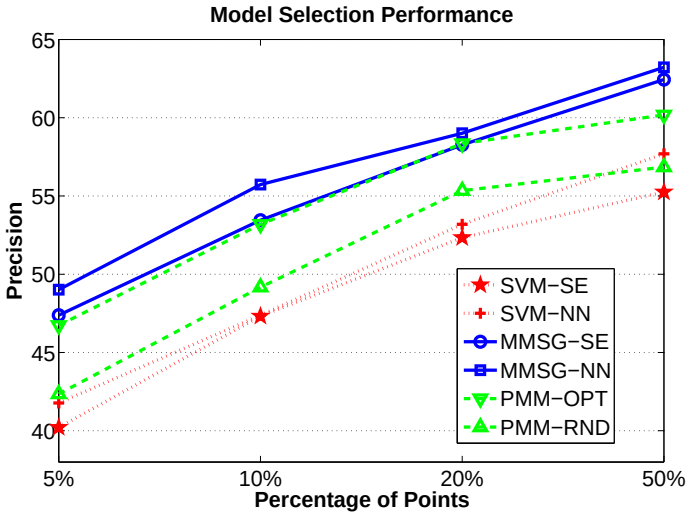


Fig. 14. Comparison of tagging performance of SVM, PMM and MMSG. Two covariance functions used: SE = squared exponential, NN = neural network.

used for classification is the squared exponential function (SE) in eq.(19). An alternative function takes the form of neural network (NN):

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = \frac{2}{\pi} \sin^{-1}\left(\frac{2\tilde{\mathbf{x}}^T \Sigma \tilde{\mathbf{x}}'}{\sqrt{(1+2\tilde{\mathbf{x}}^T \Sigma \tilde{\mathbf{x}})(1+2\tilde{\mathbf{x}}'^T \Sigma \tilde{\mathbf{x}}')}}\right), \tag{38}$$

with $\tilde{\mathbf{x}}$ being the augmented vector of the input $\mathbf{x}$.

For brevity, we only use the Del.icio.us data set to illustrate the results of model selection. We compare our results with SVM which uses the same two covariance functions. Figure 14 demonstrates the results on the three methods. We set the number of prototypes M to be $5\%, 10\%, 20\%$ and 50% respectively. It can be observed that MMSG generally outperforms SVM by roughly 10% at each point. With the number of prototypes increases, the precision also soars up from 50% to 62% for MMSG. Meanwhile, by using neural network as the covariance function, both SVM and MMSG gain about 2% precision at each point. It can also be observed that by using the optimal subset selection, the PMM method (denoted as PMM-OPT) performs almost as good as MMSG with SE kernel. Overall, MMSG-NN shows the best performance.

5.4.2 *Optimal Prototype Selection for Tag Suggestion.* To justify the use of the prototype selection (PS) algorithm for the prototype-based method, we compare with the criteria used in [Seeger and Williams 2003] which efficiently includes points into the active set based on information gain (IG). We also include a random selection (RS) method as the baseline. Figure 15 presents the results on del.icio.us. Generally, prototype selection shows better precision than IG in all four cases. To be specific, prototype selection gains more than 10% performance improvement comparing with information gain when $M = 50\%$.

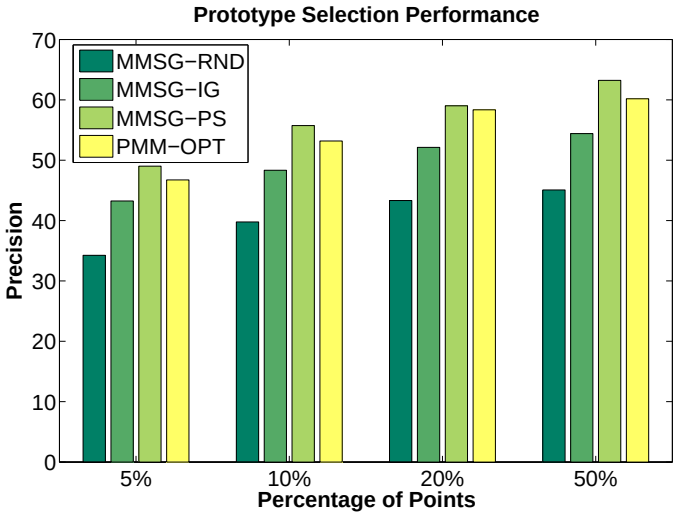


Fig. 15. Tagging performance of three selection algorithms and PMM-OPT. RND = random selection, IG = information gain, PS = prototype selection.

5.5 Discussion of the Quality of Recommendation

It has been observed in our experiment that most algorithms performed better in the CiteULike data set than the Del.icio.us data set, while the performance of the BibSonomy data is sort of in between. Remember that the CiteULike data contains mostly scientific documents, Del.icio.us has mostly web URLs with unstructured contents, while BibSonomy has both documents and web pages. We thus 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 [Golder and Huberman 2006] 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.

5.6 Efficiency of Tag Recommendation Methods

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 16 and Table V present the performance of CiteULike and del.icio.us data respectively[9]. Our approaches exhibit stable performance on both data sets with very small variance. On average, only 1.08 seconds is needed by MMSG for each test document on CiteULike and 1.23 seconds for del.icio.us. While PMM shows a slightly slower prediction speed, the time still scales linear to the number of training data. 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.

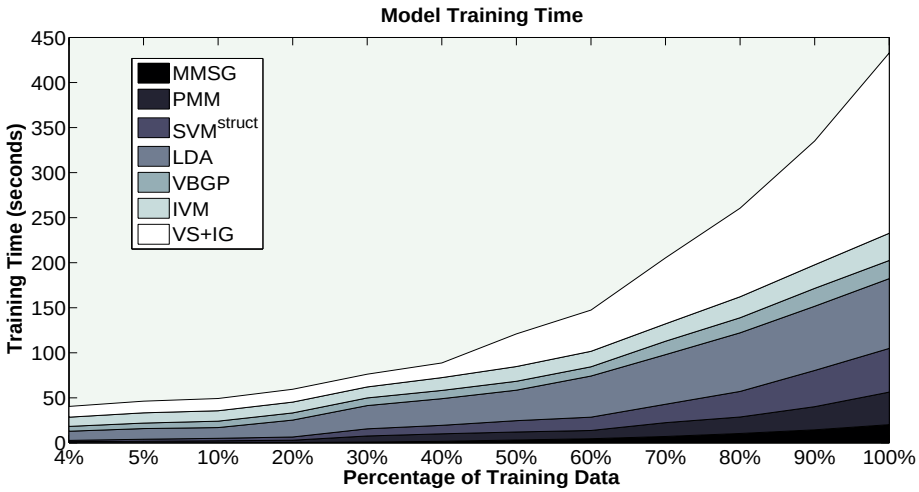


Fig. 16. Average tagging time on the CiteULike data set. Our models require the least time for making recommendations.

6. CONCLUSION AND FUTURE WORK

We presented two document-centered approaches for efficient and effective tag recommendation in social recommender systems. Our models were able to recommend good tags to users in real-time when testing on large-scale data sets. Comparing to user-centered approaches, our models were more sophisticated with better adaptability to practical systems.

Future work could access how our frameworks can be extended to including user interests into our document-centered approach to achieve more powerful predicative

[9]The experiment was performed on a 3.0GHZ sever

% Train	MMSG	PMM	SVM^{struct}	LDA	VS+IG
10	0.35 ± 0.2	0.64 ± 0.4	2.5 ± 1.7	1.7 ± 0.5	17.3 ± 10.8
20	0.38 ± 0.2	0.69 ± 0.5	2.7 ± 1.6	1.9 ± 0.5	25.8 ± 10.9
30	0.43 ± 0.2	0.72 ± 0.5	2.9 ± 1.8	2.2 ± 0.6	33.3 ± 12.7
40	0.47 ± 0.3	0.77 ± 0.5	3.3 ± 1.9	2.5 ± 0.7	46.8 ± 12.9
50	0.53 ± 0.3	0.79 ± 0.6	3.3 ± 2.0	2.6 ± 0.7	53.2 ± 13.1
60	0.56 ± 0.3	0.83 ± 0.6	3.8 ± 2.5	2.9 ± 1.1	59.0 ± 14.1
70	0.60 ± 0.4	0.88 ± 0.8	4.1 ± 2.4	3.2 ± 1.2	86.8 ± 14.6
80	0.62 ± 0.6	0.93 ± 0.7	4.4 ± 2.6	3.6 ± 1.4	106.2 ± 19.8
90	0.65 ± 0.6	0.94 ± 0.8	4.8 ± 2.8	3.7 ± 1.5	117.2 ± 25.9
Average	**0.51±0.34**	**0.80±0.60**	**3.53±2.15**	**2.70±0.91**	**60.62±14.98**

Table V. Average tagging time (seconds) for the three data sets.

performance. Statistical significance tests will be performed to examine the generalization of our methods to other data sets. We also prepare to implement our models on real-world applications such like the CiteSeer^X scientific digital library[10].

REFERENCES

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

Blei, D. M., Ng, A. Y., and Jordan, M. I. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res. 3*, 993–1022.

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

Brinker, K., Furnkranz, J., and Hullermeier, E. 2006. A unified model for multilabel classification and ranking. In *ECAI '06*.

Chirita, P. A., Costache, S., Nejdl, W., and Handschuh, S. 2007. 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*. ACM Press, New York, NY, USA, 845–854.

Cristianini, N. and Shawe-Taylor, J. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm.

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

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

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

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

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

Kohonen, T. 2001. *Self Organization Maps*. Springer.

Kullback, S. and Leibler, R. A. 1951. On information and sufficiency. *Annl. of Math. Stat. 22*, 79–86.

[10]http://citeseerx.ist.psu.edu

Lawrence, N., Seeger, M., and Herbrich, R. 2003. Fast sparse gaussian process methods: The informative vector machine. In *NIPS 15*. 609–616.

Li, J. and Wang, J. Z. 2006. Real-time computerized annotation of pictures. In *MULTIMEDIA '06*. 911–920.

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

Platt, J. C. 2000. Probabilities for sv machines. *Advances in Large Margin Classifiers*, 61–74.

Rasmussen, C. E. and Williams, C. K. I. 2006. *Gaussian Processes for Machine Learning*. The MIT Press.

Schlattmann, P. 2003. Estimating the number of components in a finite mixture model: the special case of homogeneity. *Comput. Stat. Data Anal. 41,* 3-4, 441–451.

Seeger, M. and Jordan, M. Sparse gaussian process classification with multiple classes. TR 661, Department of Statistics, University of California at Berkeley, 2004.

Seeger, M. and Williams, C. 2003. Fast forward selection to speed up sparse gaussian process regression. In Workshop on AI and Statistics 9, 2003.

Seo, S., Bode, M., and Obermayer, K. 2003. Soft nearest prototype classification. *IEEE Trans. Neural Networks*.

Song, Y., Zhang, L., and Giles, C. L. 2008. Sparse gaussian processes classification for fast tag recommendation. In *CIKM '08: Proceedings of the seventeenth ACM conference on Conference on information and knowledge management*. ACM, New York, NY, USA.

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*.

Tsoumakas, G. and Katakis, I. 2007. Multi-label classification: An overview. *Intl. J. of Data Warehousing and Mining 3,* 3, 1–13.

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

Appendix: Derivation of the mean and variance of the posterior in eq.(27) using Laplace Approximation

Since the denominator $p(\mathbf{y}|X, \bar{X})$ in eq.(27) is independent of $\mathbf{f}$, we only need to concern the un-normalized posterior when making the inference. We notice that for part C in the above equation, $p(\mathbf{y}|\mathbf{f})$ can be obtained from eq.(21) and is not Gaussian. Taking the logarithm of C in eq. (27), we have:

$$\mathcal{L}(\mathbf{f}) \stackrel{\triangle}{=} \log \underbrace{p(\mathbf{f}|\bar{\mathbf{f}}, X, \bar{X})}_{\mathcal{L}_1} + \log \underbrace{p(\mathbf{y}|\mathbf{f})}_{\mathcal{L}_2}, \tag{39}$$

where $\mathcal{L}_1$ corresponds to the complete data likelihood, which can be generated i.i.d. given the inputs, i.e.,

$$p(\mathbf{f}|\bar{\mathbf{f}}, X, \bar{X}) = \prod_{n=1}^{N} \underbrace{p(\mathbf{f}_n|\mathbf{x}_n, \bar{\mathbf{f}}, \bar{X})}_{\text{eq.(26)}} = \mathcal{N}(\mathbf{f}|\mathbf{K}_{NM}\mathbf{K}_M^{-1}\bar{\mathbf{f}}, \mathbf{\Lambda}), \tag{40}$$

with $\mathbf{\Lambda} = \text{diag}(\boldsymbol{\lambda}), \lambda_n = \mathbf{K}_n - \mathbf{k}_n^T \mathbf{K}_M^{-1} \mathbf{k}_n, [\mathbf{K}_{NM}]_{nm} = \mathbf{K}(\mathbf{x}_n, \bar{\mathbf{x}}_m)$. Combining eq.(40) & (21), we can evaluate eq.(39) as follows:

$$\begin{aligned} \mathcal{L}(\mathbf{f}) &= \left(-\frac{CN}{2} \log 2\pi - \frac{1}{2} \log |\mathbf{\Lambda}| - \frac{1}{2} (\mathbf{W}^T \mathbf{\Lambda}^{-1} \mathbf{W}) \right) \\ &+ \left(\mathbf{y}^T \mathbf{f} - \sum_{i=1}^{N} \log (\sum_{c=1}^{C} \exp f_i^c) \right), \end{aligned} \tag{41}$$

where $\mathbf{W} = \mathbf{f} - \mathbf{K}_{NM} \mathbf{K}_M^{-1} \bar{\mathbf{f}}$. By differentiating eq.(41) w.r.t. $\mathbf{f}$, we obtain

$$\nabla_{\mathbf{f}} \mathcal{L}(\mathbf{f}) = -\mathbf{\Lambda}^{-1} \mathbf{f} + \mathbf{\Lambda}^{-1} \mathbf{K}_{NM} \mathbf{K}_M^{-1} \bar{\mathbf{f}} + \mathbf{y} - \mathbf{m}, \tag{42}$$

where $\mathbf{m}$ is a vector of the same length as $\mathbf{y}$ and $\mathbf{m}_i^c = p(y_i^c | \mathbf{f}_i)$. At the maximum, we have the MAP value of $\mathbf{f}$:

$$\hat{\mathbf{f}} = \mathbf{K}_{NM} \mathbf{K}_M^{-1} \bar{\mathbf{f}} + \mathbf{\Lambda}(\mathbf{y} - \hat{\mathbf{m}}). \tag{43}$$

Differentiating eq.(42) again, we obtain

$$\nabla \nabla_{\mathbf{f}} \mathcal{L}(\mathbf{f}) = -\mathbf{\Lambda}^{-1} - \mathbf{M}, \quad \mathbf{M} \triangleq \text{diag}(\mathbf{m}) - \Pi \Pi^T. \tag{44}$$

According to [Rasmussen and Williams 2006], Π corresponds to a matrix of size $CN \times N$, which can be obtained by vertically stacking $\text{diag}(\mathbf{m}^c)$. Using the Newton-Raphson formula, we obtain the iterative update equation for $\mathbf{f}$:

$$\begin{aligned} \mathbf{f}' &= \mathbf{f} - (\nabla \nabla_{\mathbf{f}})^{-1} \nabla_{\mathbf{f}} \\ &= (\mathbf{\Lambda}^{-1} + \mathbf{M})^{-1} (\mathbf{M}\mathbf{f} + \Lambda^{-1} \mathbf{K}_{NM} \mathbf{K}_M^{-1} \bar{\mathbf{f}} + \mathbf{y} - \mathbf{m}). \end{aligned} \tag{45}$$

Applying the Taylor Expansion, we obtain

$$\mathcal{L}(\mathbf{f}) = \mathcal{L}(\hat{\mathbf{f}}) - \frac{1}{2} \nabla \nabla_{\mathbf{f}} \mathcal{L}(\mathbf{f}) (\mathbf{f} - \hat{\mathbf{f}})^2. \tag{46}$$

Thus the integral part in eq.(27) can be estimated analytically:

$$\begin{aligned} \int (C) d\mathbf{f} &= \int \exp(\mathcal{L}(\mathbf{f})) d\mathbf{f} \\ &= \int \exp \left(\mathcal{L}(\hat{\mathbf{f}}) - \frac{1}{2} \nabla \nabla_{\mathbf{f}} \mathcal{L}(\mathbf{f}) (\mathbf{f} - \hat{\mathbf{f}})^2 \right) d\mathbf{f} \\ &= \exp \left(\mathcal{L}(\hat{\mathbf{f}}) \right) \int \exp \left(-\frac{1}{2} \nabla \nabla_{\mathbf{f}} \mathcal{L}(\mathbf{f}) (\mathbf{f} - \hat{\mathbf{f}})^2 \right) d\mathbf{f} \\ &= \exp \left(\mathcal{L}(\hat{\mathbf{f}}) \right) \sqrt{2\pi} \left| \nabla \nabla_{\mathbf{f}} \mathcal{L}(\mathbf{f}) \right|^{-1}. \end{aligned} \tag{47}$$

Note that the above equation essentially forms a normal kernel for $\bar{\mathbf{f}}$, where the only part that contains $\bar{\mathbf{f}}$ is $\frac{1}{2}((\mathbf{f} - \mathbf{K}_{NM} \mathbf{K}_M^{-1} \bar{\mathbf{f}})^T \mathbf{\Lambda}^{-1} (\mathbf{f} - \mathbf{K}_{NM} \mathbf{K}_M^{-1} \bar{\mathbf{f}}))$. Back to eq.(27), as $p(\bar{\mathbf{f}} | \bar{X})$ follows a normal distribution according to eq.(24), the posterior also forms a normal distribution. Consequently, we only need to calculate the mean

and variance. After some matrix manipulation, we have

$$\begin{aligned} \boldsymbol{\mu}_p = \frac{\mathbf{Q}^{-1}\mathbf{P}}{2}, \boldsymbol{\Sigma}_p = \mathbf{Q}^{-1}, \\ \text{where } \mathbf{Q} &= (\mathbf{K}_{NM}\mathbf{K}_M^{-1})^T\boldsymbol{\Lambda}^{-1}(\mathbf{K}_{NM}\mathbf{K}_M^{-1}) + \mathbf{K}_M, \\ \mathbf{P} &= \hat{\mathbf{f}}^T\boldsymbol{\Lambda}^{-1}(\mathbf{K}_{NM}\mathbf{K}_M^{-1}). \end{aligned} \tag{48}$$

In this way the Laplace approximation gives an estimated results $\tilde{p}(\bar{\mathbf{f}}|X, \mathbf{y}, \bar{X})$ of the posterior in eq.(27). We can thus compute the latent values of the new $\mathbf{x}_*$ by plugging the result into eq.(25). The estimated latent values $\tilde{p}(\mathbf{f}_*|\cdot)$ now forms a Gaussian since both A and B in this equation are Gaussian. The only effect is to compute the mean and covariance, which is given by

$$\boldsymbol{\mu}_* \simeq \boldsymbol{\mu}_p, \tag{49}$$

$$\boldsymbol{\Sigma}_* = \mathbf{K}_M + \boldsymbol{\Sigma}_p. \tag{50}$$