The Pennsylvania State University
The Graduate School

MACHINE LEARNING FOR TEXT MINING: CLASSIFICATION, RETRIEVAL AND RECOMMENDATION

A Dissertation in
Computer Science and Engineering
by
Yang Song

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2009

The dissertation of Yang Song was reviewed and approved* by the following:

C. Lee Giles
Professor of Information Sciences and Technology
Dissertation Advisor, Chair of Committee

Wang-Chien Lee
Professor of Computer Science and Engineering

Jia Li
Professor of Statistics

Jesse Barlow
Professor of Computer Science and Engineering

Bing Li
Professor of Statistics

Raj Acharya
Professor of Computer Science and Engineering
Department Head of Computer Science and Engineering

*Signatures are on file in the Graduate School.

Abstract

We all witnessed the information explosion of the World Wide Web which has brought us with continuously rapid growth of information and data. However, as the amount of data grows day and night, the need for efficient and effective management of information has also increased dramatically. As a result, using intelligent computerized algorithms to discover new and useful information from existing data has become a hot-pursuit in recent research of computer and information science.

This thesis addresses the issues of discovering useful information from textual content of the data, as well as efficient management and organization of the data. These research issues are usually referred to as the task of text mining, which is a branch of the broad area of information retrieval research that contains many interesting and challenging problems and applications. In this thesis, we mainly focus on four issues of text mining: text classification (Chapter 2 & 3), text retrieval (Chapter 4), text recommendation (Chapter 5) and topic discovery (Chapter 6). Specifically, Chapter 2 proposes dimension reduction and collaborative filtering techniques to improve the scalability of text classification; Chapter 3 further addresses the performance issue of text classification by introducing a new nearest neighbor classification method; Chapter 4 deals with retrieving correct name entities from the web and textual documents where the names are ambiguous; Chapter 5 deals with text recommendation for scientific documents and webpages; Chapter 6 aims at discovering dynamic topic trends and correlations in scientific documents; Chapter 7 concludes this thesis. We will also try to answer some difficult research questions based on our study.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I am most grateful and indebted to my thesis advisor, C. Lee Giles, who has been my PhD advisor during my entire PhD study at Penn State. As an ACM & IEEE fellow and the leader in information processing and web analysis, he has introduced and guided me to the text mining research area by his great intelligence and foresightedness. I would also like to thank him for his consistent financial support during my study at Penn State.

I would like to thank Professor Hongyuan Zha for introducing me to the area of clustering techniques and matrix theories. I am indebted to Professor Jia Li in Statistics department, for her inspiration and enlightening discussions on the topics of mixture models, graphical models and Gaussian processes. I would like to thank Professor Jesse Barlow and Professor Wang-Chien Lee from Computer Science who have instructed me by their great knowledge on database and matrix related areas, which have been two fundamental parts of my dissertation research. I would like to thank Professor Bing Li from Statistics for his constructive suggestions on my PhD proposal that directly leads to this dissertation. I am so lucky and grateful to have Professor Jia Li, Jesse Barlow, Wang-Chien Lee and Bing Li in my committee.

My PhD research has been closely connected to the industry. Three of my summers during the last four years were spent at IBM TJ Watson research center, Microsoft research Redmond and AskJeeves (now Ask.com). These unforgettable and rewarding experiences have greatly inspired my research in the text mining area. I would like to thank Dr. Eric Glover and Professor Tomasz Imielinski at AskJeeves for bringing me into the search engine research. I would like to thank Dr.Aleksander Kolcz and Dr. Dennis DeCoste at Microsoft for their guidance and discussions on my summer research of email spam filtering. I would like to thank my mentor Dr. Anca Sailer and my manager Dr. Hidayatullah Shaikh at IBM, who have introduced me to the idea of using machine learning for services-related research area.

I am indebt to my dear wife Lu Zhang, who has been an indispensable part of my life and research. Her endless love to me and our family has been the greatest momentum for me to finish my PhD study at Penn State. As a PhD candidate in Statistics department at Penn State, she also devoted her time and effort to help me on my research leveraging her

expertise in stochastic processes and statistical analysis. Lu and I have worked together on several interesting and challenging interdisciplinary problems. Our research results have been published in top-tier computer science conferences.

Finally, I would like to thank my parents, Jinshan Song and Yu Jiehua, who brought me into this world and encouraged me to pursue my PhD study abroad. I am indebt to their selfless love to me in my whole life.

After graduation, my research work will be continued at Microsoft Research, Redmond, U.S.A.

Dedication

To my dear wife Lu Zhang, my dear parents Jinshan Song and Jiehua Yu.

Chapter 1

Introduction

The World Wide Web has been growing at a phenomenal rate since its emergence in the last century. A recent study has shown that the Internet contains about 9.36 billion pages[1]. These pages contain rich information of images, videos, but most importantly, text contents. According to a recent study[2], approximately 80 to 85% of all data stored in databases are texts. Due to the heterogenous nature of the WWW, the text contents in those pages are usually unstructured and thus hard to be discovered easily. Consequently, leveraging computers to automatically discover useful information from previously seen data becomes more and more desirable.

In this thesis, we address a research branch of the broad information management research — text mining. Generally, it means the process of discovering useful patterns, structures and other valuable information from unstructured natural language texts. Text mining, sometimes referred to as data mining, contains many interesting and challenging tasks. For examples, commercial search engines (e.g., Google and Yahoo!) highly leverage text mining techniques to retrieve relevant documents according to user-input queries, which is a way of showing the success of text mining for effective information extraction from massive amount of unstructured data.

The aforementioned application is an example of the broad research area of text mining, namely text retrieval. The research issues of text mining have been studied for decades, by researchers from different research areas including applied mathematics, statistics, machine learning, natural language processing and etc. Apparently, we are

[1] http://www.cs.uiowa.edu/∼asignori/web-size/

[2] http://www.edbt2006.de/edbt-share/IntroductionToTextMining.pdf

unable to cover all sub areas of text mining and thus we will focus on four important areas in this thesis: text classification, text retrieval, text recommendation and topic discovery.

1.1 Text Classification

Text classification, which is sometimes called text categorization [126, 34, 118, 116], refers to the process of automatic assignment of textual documents into one or more predefined categories. This supervised learning approach consists of two major steps: learning and prediction. During the learning step, a set of labeled training documents is presented to the algorithm, where the labels are usually acquired by human-effort. For efficiency, the documents are often represented by vectors where the elements in the vectors correspond to specific words, and the values of the elements refer to the number of appearances of the words in the documents. This representation is the most common one in text analysis, namely the bag-of-words (BOW) model. After the algorithm learns a classifier from the training documents, the classifier is then capable of categorizing new unlabeled documents into the existing categories.

The benefit of performing text classification is multifold. Obviously, text classification can reduce the cost and time of human-effort for labeling documents, which also reduces the probability of making errors during labeling. Moreover, a classifier can also act as a feature selection algorithm which selects the important word features and eliminates irrelevant ones [29, 40]. This helps reducing the dimensionality of the feature space and consequently slashes the computational cost of the learning algorithm. Finally, text classification can also be leveraged as a pre-processing step to improve the performance of other tasks. For example, classifying webpages into several topics (e.g., news, education and etc) can substantially improve the accuracy of text retrieval [28].

Text classification is arguably one of the most important research areas of text mining and machine learning. Applications of text classification include many research fields, such as query classification [63, 17], email spam filtering [68, 69], and micro-array classification [48, 135] in biological sciences.

1.2 Text Retrieval

Text retrieval, or document retrieval, aims at matching user-input keywords (or queries) with a collection of text documents. It belongs to the broad research area of information retrieval, which includes image retrieval, video retrieval and so on. Since the emergence of search engines like Yahoo! and Google, text retrieval has been widely applied in all commercial search engines and digital libraries around the world. Regardless of the applications, the general process of text retrieval usually consists of two steps: (1) find relevant documents based on the input; (2) sort the retrieved documents according to their relevance to the query. These two steps therefore pose two major challenges to the retrieval tasks, i.e., efficiency and effectiveness.

To efficiently return the query results in real-time, most retrieval systems use an index to speed up the lookup process [19]. When there is a match between the query words and the document content, the document is then treated as relevant and retrieved by the system. On the other hand, many useful metrics have been introduced to measure the rank or significance of documents / webpages to a specific query. Including the traditional term frequency (tf), inverse document frequency (idf) [62] and more recently, the pagerank score by Google [93].

1.3 Text Recommendation

Text recommendation is often used in recommender systems which suggest relevant documents (or items) based on user queries. For example, when a user buys a book on Amazon[3], the system will then suggest other relevant books based on the content of the book as well as the similarity between the user and others who also bought this book. A classic technique used in the scenario is named collaborative filtering (CF) [15], which calculates the similarity between users and recommends interesting items based on user history.

More recently, text recommendation has been applied to social bookmarking systems like delicious[4] and Flickr[5]. These social systems allow users to specify their own keywords for their collection of webpages, images and videos, which greatly facilitates

[3] http://www.amazon.com
[4] http://delicious.com/
[5] http://www.flickr.com/

the organization and sharing of resources among users. These user-defined keywords are often called tags or social bookmarks. A recent hot research issue aims at recommending relevant tags to a new resource or a new user [119, 117, 80]. Unlike the traditional taxonomy, tags are not bounded by any pre-defined vocabulary. This issue thus becomes more challenging than the traditional text recommendation tasks.

In general, text recommendation is very similar to the process of text retrieval, except that the retrieved results may not contain the user-input at all, but still quite relevant to the queries. Therefore, text recommendation is often more challenging than retrieval since it requires richer information than standard retrieval tasks, e.g., user history and so on.

1.4 Topic Discovery

Since Latent Semantic Analysis (LSA) was introduced to text mining in 1988 [31], topic analysis has become a very popular research area that attracts many computer scientists and statisticians. The simple idea behind topic analysis is that a collection of documents can be treated as a mixture of topics, where each topic contains many words that form an unknown probabilistic distribution within the topic. which essentially reduces the dimensionality of document representation from words into topics. Therefore, topic analysis can substantially reduce the effort to manage large and ever-growing collections of documents.

Recently, probabilistic graphical topic models such as the probabilistic LSA (PLSA) [54] and latent Dirichlet allocation (LDA) [12], which improve the LSA model by introducing statistical analysis, have become very useful tools for several crucial tasks in text information retrieval [152, 2, 87].

Compared to the supervised text classification methods, topic models are unsupervised learning approaches, which do not require any prior knowledge of the document labels. Thus they are used more frequently than supervised classification methods.

1.5 Text Mining Metrics

Regardless of the application type, the measurement (or the metric) of the effectiveness of a text mining algorithm is typically carried out by comparing the performance of the

algorithm in the testing set of documents after a training set. Two most frequently used metrics are *precision* and *recall* (See Figure 1.1).

- Precision: The proportion of retrieved and relevant documents to all the documents retrieved. i.e., "How many documents are assumed to be in the category truly belong to it"?
- Recall: The proportion of relevant documents that are retrieved, out of all relevant documents in a system. i.e., "How many documents that belongs to the category have been deemed as such"?

Ideally, one may want to achieve high precision and recall at the same time (e.g., Figure 1.2(left)), shown in Figure. But sometimes the algorithms may compromise one for the other as represented in Figure 1.2(middle)&(right). As a result, to measure the average performance of an algorithm, $F1$ score, defined as $2 \cdot precision \cdot recall / (precision + recall)$, is also considered a popular and effective metric.

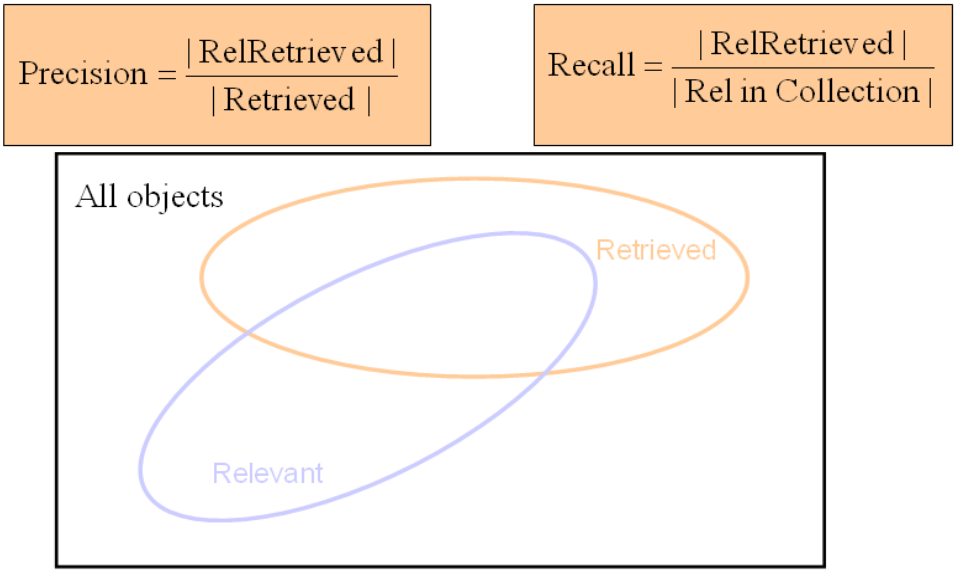


Figure 1.1. Illustration of precision and recall and their measurement. (From C. Lee Giles: IST 441, Information Retrieval and Search Engines.)

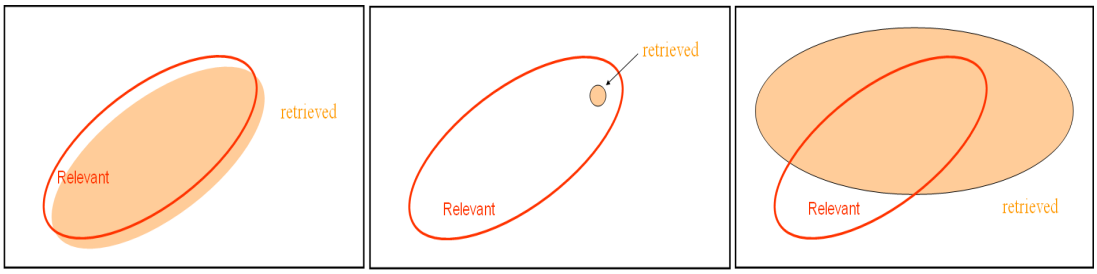


Figure 1.2. Illustration of the performance of precision and recall. (Left) High precision and high recall. (Middle) High precision with low recall. (Right) Low precision with high recall. (From C. Lee Giles: IST 441, Information Retrieval and Search Engines.)

1.6 Challenges of Text Mining Tasks

Although text mining has been studied for decades, there still exists several challenges and issues that should be addressed (and will be addressed in this thesis):

- **Performance**. Ideally, for each category the algorithm is assumed to find everything relevant in the system (high recall) and only retrieve those into that category (high precision). The accuracy of the model depends largely on how the documents are represented as well as the distribution of the documents. For example, in document classification, the documents are usually in the form of vectors. Thus the vector space is usually quite large and sparse, which is the main cause of the "curse of dimensionality" phenomena. Achieving the best performance has been the major measurement for almost all text mining tasks.

- **Scalability**. Efficiency is as crucial as performance for large-scale applications. To achieve better performance, usually more training documents are preferable, which could in turn causes prohibitively long model training time for the algorithm. For example in most scenarios of text classification, the training time is linear or quadratic to the number of training documents. e.g. the training time for support vector machine classifiers (SVMs) [145, 60, 30] is usually quadratic until a recent improvement which turns the training time to be linear [61]. Some other simple classifiers, e.g., K-nearest neighbors classifier [116], logistic regression [146], their training cost is also decided by the number of categories in the training data.

- **Adaptivity**. An algorithm may perform well in one application (e.g., image categorization) but bad for another (e.g., text categorization). More commonly, a classifier may have different performance on different data sets in the same application. Thus building a universal algorithm that is both application-independent and dataset-independent becomes quite desirable.

- **Customization(Personalization)**. Retrieving relevant documents based on user preferences has become a new research trend. For web users, the algorithms have to deal with documents with diverse content and users with diverse interests.Thus traditional algorithms (e.g., classifiers) that fix categories in advance obviously cannot cater for all user interests.

  For example, in *personalized classification*, the users are assumed to create their own personalized categories. The classifiers will then be automatically trained for classifying objects under such categories. Example applications include online news classification, book recommendation (e.g., Amazon online store) and so on. In the domain of text classification, personalized classification can be considered as *text filtering*, where one or more set of features is first constructed with each representing a different user interest domain. Based on the semantic closeness with the features, relevant documents are then retrieved from the corpus for different users. This problem have been well-studied in the Text REtrieval Conference (TREC) [1].

  In general, two types of processes are employed for personalized classification, namely *flat process* and *hierarchical process* [124]. The flat process corresponds to the case that the personalized categories are defined independently of each other, while the hierarchical process refers to the situation that each personalized category is defined within some general category.

1.7 Objective and Structure of This Thesis

The objective of this thesis is to answer the following research questions:

1. **Can dimension reduction techniques boost the performance of text classification?**

2. **Is it possible to overcome the "curse of dimensionality" for the K-nearest neighbor classifier?**

3. **Are unsupervised learning algorithms comparable with supervised learning methods for retrieving correct name entities (i.e., name disambiguation)?**

4. **Are computerized text recommendation algorithms suitable for Web2.0 applications in recommending social bookmarks to users?**

5. **Instead of simply breaking a document collection into static topics, is it possible to model the dynamic change of topics within the document collection during a range of time? Moreover, can we monitor the topic correlations between several document collections dynamically?**

In the remaining of this thesis, I will discuss the literature of this area as well as my previous work on text mining. The rest of this thesis is organized into 7 chapters. Specifically, Chapter 2 proposes dimension reduction and collaborative filtering techniques to improve the scalability of text classification; Chapter 3 further addresses the performance issue of text classification by introducing a new nearest neighbor classification method; Chapter 4 deals with retrieving correct name entities from the web and textual documents where the names are ambiguous; Chapter 5 deals with text recommendation for scientific documents and webpages; Chapter 6 aims at discovering dynamic topic trends and correlations in scientific documents; Chapter 7 concludes this thesis.

Chapter 2

Text Classification: Dimension Reduction and Collaborative Filtering

Mathematically speaking, text classification is the task of approximating the unknown *target function* $\Phi : \mathbb{D} \times \mathbb{C} \rightarrow \{T, F\}$ by means of a function $\Psi : \mathbb{D} \times \mathbb{C} \rightarrow \{T, F\}$ named the *classifier*, such that Φ can be approximated as much as possible by Ψ. Here $\mathbb{C} = \{c_1, ..., c_m\}$ ia a predefined fixed set of categories, $\mathbb{D}$ is a corpus of objects. In most cases, the categories $\mathbb{C}$ are assumed to be numerical. i.e., there is no specific meaning of a class label since it is not helpful in building a classifier.

Depending on the application, classification may be:

- Single label: In this case each object must be assigned with exactly one label. A special case of this is when $m = 2$, i.e., binary classification (labels are usually {-1, +1} or {0, 1}).
- Multiple label: Each object may have one or more labels in this case. A special case is document tagging (See Chapter 5 for details).

Meanwhile, classification may be required to perform differently depending on the application:

- Hard classification: Based on our definition, it is to provide a value in $\{T, F\}$ which indicates membership or non-membership of d_j in c_i.

- Soft classification: In this case, the classifier is only required to provide a value between $[0, 1]$ which indicates the degree of confidence of the membership of d_j in c_i.

Typically, a classifier for $\mathbb{C}$ can be built either manually or automatically. In the first case, a set of type of rules are set by people with domain expertise to decide the label of all objects. Nevertheless, due to the large-scale and ever-growing number of objects in most situations nowadays, it has become labor-intensive that causes prohibitively long time to finish the task, as well as other defect made by human-beings which makes this approach error-prone and cost-ineffective.

As a result, the second approach, namely automatic classification, is usually preferable. This process is generally carried out by supervised machine learning techniques, which leverage a set of *training* objects that is pre-classified (i.e., labelled) in $\mathbb{C}$, and make automatic predictions of the labels for the new objects, which are usually referred to as *testing* objects. Generally, the representation of objects is in the form of vectors, where the length of the vectors indicates the number of *features* that is included in at least one training object. The value (or the weights) of each feature denotes the number of occurrence in each object, which may be binary (indicating presence or absence of the feature in the object) or non-binary.

In literature, numerous supervised learning techniques have been proposed for text classification. Among all of them, successful algorithms include neural networks [104, 151, 94], decision tree [56, 86], probabilistic classifiers [58, 137, 91], nearest neighbors [5, 50, 150] and etc. More recently, support vector machines [145, 60, 30] and boosting algorithms [20, 92] are becoming more popular with generally better performance.

To overcome long training time for the classifier in large-scale applications, dimension reduction are often performed before the data is sent to the classifier. Feature selection, feature extraction and re-parameterizations are the most common used methods. In this chapter, I will focus on the issue of dimension reduction in unstructured document classification.

2.1 Challenge of Unstructured Document Classification

Industry analysts suggest that over 80% of the content within the typical Global 2000 organization is unstructured — that is, content which does not fit neatly into the rows and

columns of a relational database. The fact is, this unstructured content is key to many business processes across the organization and throughout the business value chain, from engineering drawings and specifications, to brand assets such as sales and marketing collateral, legal documents, educational videos and tutorials, online product catalogs, and customer service information.

However, experience with the CiteSeer Digital Library[1] indicates that there still exist several challenges in text classification for unstructured data on the Web, particularly when the number of classification labels is large.

In CiteSeer, several concept taxonomies exist for classifying academic materials, including the taxonomy for computer science provided by the ACM. For the purposes of this project, publication venues are used as classification labels under the assumption that each publication venue encapsulates a distinguishable concept focus. Since CiteSeer (and most search engines) automatically crawls academic documents from venue websites, author homepages and then extracts textual information from them to create metadata, false labels are inevitably assigned to many documents. Due to the increasing similarities between different venues (e.g., *SIGKDD* and *PKDD*, *ECML* and *ICML*), the effort needed to accurately classify a document into exactly one category becomes greater. Moreover, lack of keyword fields, improperly defined terms, and other feature deficiencies create unique challenges for text classification.

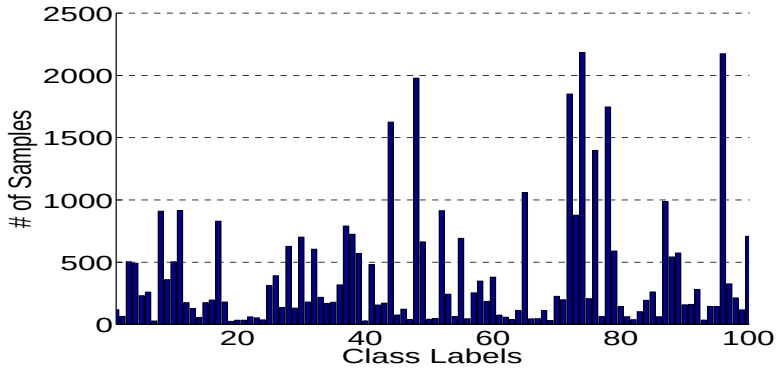


Figure 2.1. Distribution of documents w.r.t. classes in CiteSeer. In Practice, documents on the Web are also unevenly distributed.

This problem is further exacerbated due to the imbalance of documents available for

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

training in each class, i.e., the documents are unevenly distributed in different categories on the Web (for example, CiteSeer has a collection of more than 3,000 documents for *INFOCOM*, while for some other conferences, the cumulative numbers are no more than 200); Figure 2.1 gives an actual document distribution on one of the data sets used later in the paper.

Traditional *bag-of-words* approach represents each document as a feature vector and often leads to feature spaces that are sparse and large, high classification accuracies are thus hard to get. Contemporary approaches of text classification concentrate on extracting more meaningful features from structured text, e.g., adding numeric features such as timestamps [83], capturing features that share mutual information and are dependent on each other [133], as well as seeking better methods to refine the classification model based on the prediction errors from the training data sets [36, 82]. Several classifiers have been introduced to text classification, e.g., Naive Bayes [95], maximum entropy [90] and Boosting [20]. Support Vector Machines (SVMs) [14, 147, 153], which focus on finding the hyperplane that maximizes the margin between positive and negative classes, have typically been the most effective classifiers with regards to the classification errors. Forming the feature space has become for many a crucial part of using SVMs as text classifiers, since naturally there are hundreds of terms in each document and thousands of documents in each class, which results in very high-dimensional feature spaces. Yet it has been reported that SVMs can still achieve high accuracy in document classification without feature selection [125].

Research on entity extraction spans the fields of linguistics and computer science. Linguistic techniques can be employed to enhance feature selection from raw text by grouping text into semantically meaningful chunks. Developments in entity extraction technology have traditionally been concerned with the issue of computational complexity as well as extraction accuracy and domain specificity. Methods for entity extraction from unstructured data typically fall into two categories: pattern-based approaches and model-based approaches. Pattern-based extractions require extensive manual labor for detecting patterns and is generally not robust to variant data. On the other hand, model-based approaches like hidden Markov models (HMMs) [111] and SVMs [71], while requiring careful feature selection, have proved to be robust and flexible.

The major contributions of this chapter are: (1) dimension reduction by leveraging entity extraction methods, (2) using collaborative filtering technique for refining mini-

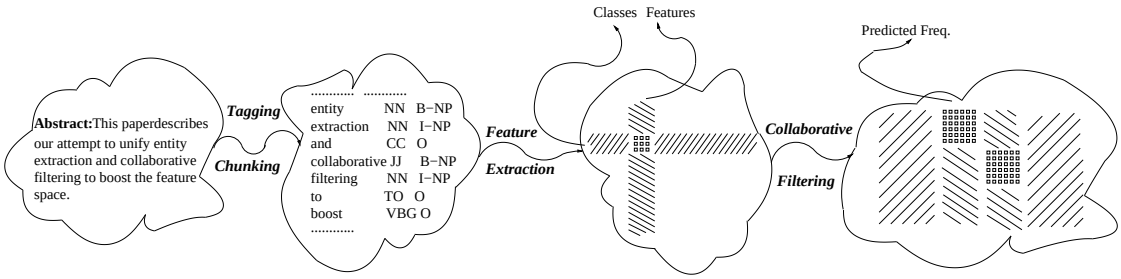


Figure 2.2. Feature space augmentation by using CF algorithm.

mal, noisy feature spaces, and (3) a comparison of the performance of SVM versus AdaBoost classifiers for the problem of categorizing academic documents by publication venue. Collaborative filtering is employed to predict the value of missing features for each class. Experimental evaluations on both real-world data set and benchmark corpus show great improvement with regard to classification accuracy compared with classification using the original feature space and the feature selection method — Information Gain (IG). Figure 2.2 shows the structure of our approach.

2.2 Entity Extraction using SVM-decision-tree

Text documents are often treated as “bags of words” for machine learning tasks, represented only by each word as a feature along with an associated frequency count. Some heuristic improvement that reduce the dimension of feature spaces include the removal of stop words, link words and punctuation from the term list. However, the “bag of words” approach does not help defining meaningful entities and results in a very large feature space. Documents on the Web usually fall into many categories, e.g., in digital libraries, academic documents span many research areas, each of which may have different meanings for the same words thus making the terms ambiguous. The feature space may be reduced by chunking text into meaningful units and treating each chunk as an individual feature. This process is known as entity extraction.

Entity extraction techniques typically fall into one of two categories: named entity recognition (NER) and phrase extraction. NER deals with identifying proper names text, extracting paper titles and author names in on-line publications and so on. Phrases, or meaningful entities, can be recognized as *signature* that best represent the main idea of papers, most of which can be found in the titles, abstracts and keyword fields in a paper.

However, only a few publication venues require keyword fields[2].

Maximum Entropy (ME) is a feature reduction approach that works by choosing the model with the most uniform probability distribution (the highest entropy), the model is described as $P(w|h) = \frac{1}{Z(h)} \cdot e^{\sum_i \lambda_i f_i(h,w)}$, where $f_i(h, w)$ denotes a binary feature function that describes a certain term; λ_i is a parameter that indicates how important feature f_i is for the model. The disadvantage of ME is that it cannot automatically select features from given feature sets thus relying on careful feature selection techniques.

Conditional random fields (CRFs) [73] is another NER technique that aims to label and segment data into phrases. It works by defining a conditional probability distribution over training data given a particular observation phrase. It usually works better than HMM and avoids the label bias problem, however, the training time of CRFs is prohibitively high.

To identify non-trivial noun phrases with semantic meanings in the documents, noun phrases (NP) chunking is adopted for this purpose. Chunking groups together semantically related words into constituents, a step beyond POS (Part-Of-Speech) tagging; but it does not identify the role of these constituents with respect to the sentence, nor their inter-relationships. In our system, we revised and implemented a previous chunking algorithm [70] as a simplified yet more efficient two-level SVM-based NP chunker.

The NP chunking problem is formalized as a three class classification problem, which assigns each word with one of the labels: B (Beginning of NP), I (Inside NP), O (Outside NP). A feature space is constructed, with dimensions representing the surrounding words, the POS tags of those words, and the already tagged chunk tags. Three SVM models (BI, IO, OB) are trained, each designed to tag a word in favor of one label over the other, for example, the BI model provides a hyperplane to tag a word with label B rather than label I.

We adopt the pairwise method that allows the SVM to classify multi-class problems. Traditional methods have considered using three SVMs together at each time, i.e., in the worst case, three comparisons need to be made in order to determine the label of a word. However, we use a method that allows us to use two SVMs instead of three, which in turn accelerates the chunking time by one third. The hierarchy of the two-level decision tree employed is shown in Figure 2.3. Furthermore, Table 2.1 shows an example to

[2]From an investigation of ACM & DBLP metadata cross-referenced with CiteSeer data, 5% of venues do not have an identifiable abstract field and more than 70% of venues lack a keyword field.

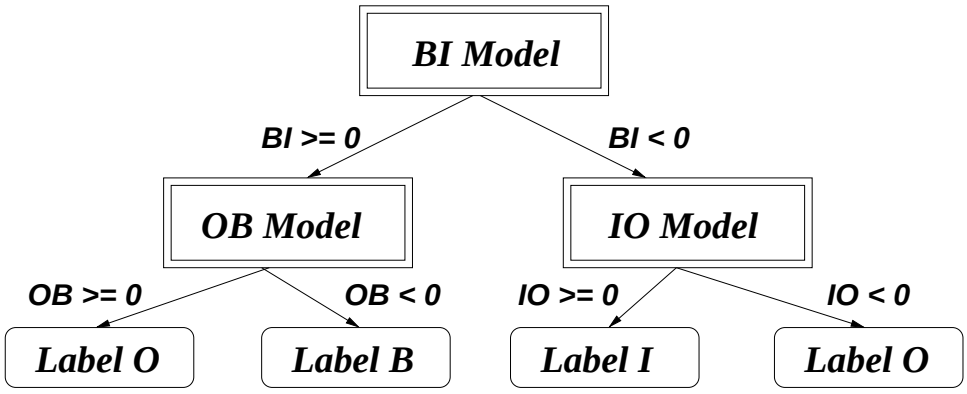


Figure 2.3. Two-level decision tree for tagging.

clarify the method we use.

Given a paragraph of unstructured text, the extraction goes through the steps of sentence segmentation, POS tagging using Brill tagger[3] and NP chunking. In this example, *collaborative* and *filtering* are labeled as adjective and noun by Brill tagger, the chunking decision for *collaborative* is based on the results of the SVMs: the result of BI model is 0.5 (in favor of label B), so the OB model is used which yields -0.6 (in favor of label B), thus B-NP is chosen as the chunk tag for this word. Totally 5 phrases are extracted in the above example shown in Figure 3, by merging the chunking tags and discarding general terms yields three meaningful entities: *entity extraction*, *collaborative filtering* and *feature space*.

2.2.1 Extract NP as Phrases

Various kinds of chunks are used in *natural language processing* (Noun Phrases, Verb Phrases, Prepositional Phrases, Adjective Phrases and Adverb Phrases) for different purposes including location extraction, noun phrase extraction and so on. However, for a specific task of text processing, it is not necessary to use all chunks together. For example, it is generally believed that in information retrieval, using only NPs and VPs in a sentence can be enough. Our approach is similar to the DAG-SVM [98] approach in essence, however, for the task of extracting meaningful entities from documents, only NP tags are considered in our case. Often *signatures* are recognized as some combina-

[3]http://research.microsoft.com/∼Ebrill/

current word	POS tag	chunk tag
This	DT	B-NP
paper	NN	I-NP
describes	VBZ	O
our	PRP	B-NP
attempt	NN	I-NP
to	TO	O
unify	VB	O
entity	NN	B-NP
extraction	NN	I-NP
and	CC	O
collaborative	JJ	B-NP
filtering	NN	I-NP
to	TO	O
boost	VBG	O
the	DT	B-NP
feature	NN	I-NP
space	NN	I-NP
.	.	O

Table 2.1. Chunk representation example. Each word is first tagged with POS tag, and POS tags are then classified into B-NP, I-NP and O tags.

tion of noun phrases in documents, which in our case are mostly denoted as B-NPs and I-NPs. As a result, instead of making use of all NP chunks for tagging, chunk labels except B-NPs and I-NPs are masked under the label of O, which loses some information during training, but significantly speeds up the training process as only three chunking tags are taken into account. Our program then simply combines B-NPs and I-NPs between O tags and treated them as *signatures* for the documents.

To summarize, our proposed *SVM-decision-tree* method finds an effective trade-off between performance and cost for extracting phrases from unstructured data, which is served as preliminary results for our final task of text classification. On one hand, we *know less* in the feature space which yields little impact on the precision of NP chunking, testing results indicate that the performance is only reduced slightly by our approach[4]; on the other hand, the algorithm slashes a lot of computational time that is required by shallow parsing methods. In our experiments, our NP chunker runs even faster than the

[4]On the Wall Street Journal data set, our model achieved precision of 90.2%, comparable with previous results in [70].

POS tagger, making it an highly efficient solution for large scale data.

2.3 Feature Space Refinement

Inspired by the analogy between *user behaviors* and *venue focuses* (i.e., different users may have similar preferences, different conferences/journals may focus on the same research areas), we employ collaborative filtering (CF) to refine the feature space by predicting missing values as well as reducing noise factors from the feature space. Here we propose two alternatives of the traditional CF algorithm. The first one is a refined instance selection algorithm, while the second algorithm aims at clustering similar users to user groups for better efficiency.

There are two major approaches of collaborative filtering, memory-based and model-based. Memory-based algorithms store users ratings/preferences in storage, and give recommendations based on known scores from existing users. Model-based algorithms describe users preference by applying descriptive models to users and/or ratings; the virtue being that once the model is established, little computation is required for prediction. However, the training requirements of both models require trade-offs between on-line and offline computation.

CF has the following issues - efficiency of the algorithms and quality of the recommendations. In general, the computational complexity of memory-based and model-based algorithms are $O(nm^2)$ and $O(nm)$, respectively, where m denotes the number of users and n denotes the number of items (e.g., movies) in the data sets where both have an upper bound of $m \times n$. Currently, data sets from large on-line stores contain millions of items as well as millions of registered users, which can be computationally expensive.

The quality of the recommendation is problematic and is primarily due to the number of items that users rated. Given the large number of items in the data sets, a single user may want to buy or rate a very small portion of the items, resulting in most items in the data sets unrated. For example, in the EachMovie[5] database, a user rates 20 movies on average, while the whole database contains more than 1,000 movies. Thus making recommendations from very limited data can lead to unacceptable errors.

[5]http://www.grouplens.org/taxonomy/term/14

2.3.1 Best Selection Algorithm (BS1) for Memory-Based CF

For the introduction to CF, we refer interested readers to the classic paper [15]. In this section, we first give a formal definition of our memory-based CF algorithm [113].

2.3.1.1 A Revised Inner Product-Based Weight Function

Given a vector space $V \in \mathbb{R}^{mn}$ and a field of scalars K (which is either in the field $\mathbb{R}$ of real numbers or the filed $\mathbb{C}$ of complex numbers), an inner product is defined as a function $< ., . >: V \times V \longrightarrow K$ which satisfies the properties of *linearity, conjugate symmetry* and *positive definite*. Generally, an inner product is a generalization of the dot product. Usually in a vector space, the inner product is used as a way to multiply vectors together, with the result of this multiplication being a scalar. In the area of information retrieval, inner product is used as a measurement of vector similarity that represents how similar two or more queries/sentences are to each other. Based on these features, we introduce our new weight function as follows:

$$\mathcal{W}(a,i) = \frac{\sum_{j=0}^{n} \langle \langle min(V_{aj}, \epsilon_a), min(V_{ij}, \epsilon_i) \rangle , Sim(a,i,j) \rangle}{min(|I_a|, |I_j|)} \tag{2.1}$$

Where function *Sim* is defined as:

$$Sim(a,i,j) = \frac{(S - |V_{aj} - V_{ij}|)}{S} \tag{2.2}$$

Here in equation (2.1) and equation (2.2), V_{ij} represents the rating of user $i \in [1,m]$ over item $j \in [1,n]$, $|I_i|$ denotes the number of items (features) that user i rated, m equals to the total number of items and S stands for the rating scale (e.g., [0-1], [0-5] and etc). The vector $\epsilon = \{\epsilon_1, \epsilon_2, ...\epsilon_n\}$ are some significant small coefficients that are a little bit smaller than the lowest feature frequencies for each sample. Specifically,

$$\epsilon(V) = \min_{r \neq 0} \frac{||Vr||_1}{||r||_1} \tag{2.3}$$

where $|| \cdot ||_1$ represents the L_1 norm (i.e., $||f||_1 = \sum |f_i|$). The reason that instead of using the true value of V_{ij}, we use $min(V_{ij}, \epsilon_i)$ in equation (2.2) for computing the weight function is that we are only concerned about whether users a and i have rated

item j or not. How close their ratings are related is decided by the function $Sim()$, which computes the similarity between two ratings by calculating the difference between user ratings divided by the rating scale resulting in vector distances (or similarity). It is evident that the higher the similarity, the closer relationships users have to each other. Obviously $Sim(a, a, i) = 1$ for all a's and i's, which implies that the similarity between a user and itself is always the highest.

The multiple inner product [15] directly computes the vector similarity between users which for some cases is similar to calculating the documents/queries similarity, but is generally not a very strict estimate for defining the weight function with respect to collaborative filtering. Given an extreme example, for three users a, i, j, suppose $V_a = (1, 1, 1)$, $V_i = (5, 5, 5)$ and $V_j = (1, 1, 1)$. In this case that user a disliked all three items and rated them all 1's, user i likes all three items, and user j disliked all. Intuitively, user a has more similarity to user j than user i. However, if we compute by multiplying vectors directly ($\langle V_a, V_i \rangle = 15, \langle V_a, V_j \rangle = 3$), the similarity between user a and i would be much higher, which is in conflict. However, by applying our Sim function, we end up having weights of values $w(a, i) = 1/5$ and $w(a, j) = 1$, showing that user a and j have more similar profiles.

2.3.1.2 Algorithm and Analysis

Based on the weight function we computed previously, we propose our first algorithm Best Selection (BS1).

In algorithm 1, a is the active user whose ratings we want to predict; m and n are used to denote the number of users and the number of items, respectively. The idea behind this algorithm is that instead of doing computation across the entire datasets only for a single prediction, we choose totally $\lceil \log m \rceil$ users that has the *best* similarity with the active user a for prediction. In other words, these candidates are the best ones to represent user a's ratings of the items in the same data set. To achieve the best result, the algorithm is iterated $\lceil \log m \rceil$ times, at each time $\lceil \log m \rceil$ users are randomly picked, and only the *best* one that has the highest similarity (judged by the weight function $\mathcal{W}(a, i)$) with the active user a is selected and inserted into the candidates list.

Once we have the candidates list and the weight function, we can predict user a's

Algorithm 1 Best Selection Algorithm (BS1)

input: user-item matrix $\mathcal{P} \in \mathcal{R}^{m \times n}$
weight matrix $\mathcal{W} \in \mathcal{R}^{m \times m}$
output: candidates list $\mathcal{C} \in \mathcal{R}^{\lceil \log m \rceil \times n}$
Initialize $\mathcal{C} \leftarrow \emptyset$
for each active user a **do**
  for i equals 0 to $\lceil \log m \rceil$ **do**
    randomly choose $\lceil \log m \rceil$ users $\mathcal{U}$ from $\mathcal{P}$
    select the candidate μ that satisfies:
    $\forall\, \upsilon \in \mathcal{U}\ and\ \upsilon \neq \mu, \mathcal{W}(a, \mu) > \mathcal{W}(a, \upsilon)$
    $\mathcal{C}_i \leftarrow \mathcal{C}_i \cup \mathcal{P}(\mu)$
  end
end
output $\mathcal{C}$

rating for item j as follows (remember $\mathcal{C}$ is the candidates list):

$$\mathcal{Q}(a, j) = \sum_{c=0}^{\lceil \log m \rceil} \mathcal{W}(a, c) * \mathcal{P}(c, j), c \in \mathcal{C} \tag{2.4}$$

Some previous memory-based approaches [15] predicted user ratings by collecting ratings from all users in the datasets. Two shortcomings are obvious here: first, lots of noise factors were introduced to the prediction process. i.e., the ratings of those users whom have very little similarities with the active user a are also added up to predict its ratings, as long as the weights between these users and a a greater than 0. Secondly, time are wasted to compute these useless ratings, which have very minor effects to the final prediction results.

By adopting instance selection, we minimize the noise factors by selecting the most representative users from the data set. As a result, on establishing the candidates list, we could get better prediction results by computing from a much smaller set, but definitely no less reliable, of related users, regardless of items. The running time is also significantly reduced to $O(n \log^2 m)$.

It has also been realized that by using random selection, we may get the optimal result for each step, but we may not achieve the optimum solution globally. i.e., those users that have the highest weights with the active user have chances of not being selected as candidates. However, trying to find the absolute best candidates each time would be labor-intensive and time-consuming. By mentioning *best* here, we mean that

our approach finds the best trade-off between computational complexity and prediction results. In other words, by selecting the best candidate for each step and repeating the same step $\lceil \log m \rceil$ times, we have minimized the standard deviation of errors that could cause mis-prediction.

2.3.2 Clustering Approach — Improved Best Selection Algorithm (BS2)

In this section, we continue to propose the second selection algorithm that unifies instance selection and feature selection techniques together, selected instances are further clustered with selected features into several clusters based on user profiles.

2.3.2.1 Select Features from Candidates List

Generally, feature selection is different from instance selection regarding selection criteria. i.e., instances are selected based on user similarities while features are chosen according to the quantity of ratings provided by users. In algorithm 2, we want to select those features that have been rated most frequently by users. At the beginning of BS2, a candidates list is retrieved by using BS1. Based on that, we count the number of items that have been rated by the users in the candidates list, and select the first $\lceil \log n \rceil$ features according to the descending order of the frequency.

The initialization part of algorithm 2 takes $O(n \log^2 m)$ time with respect to the number of users m and number of features n. The first part of the algorithm computes the number of times that the items have been rated by the selected candidates, taking $O(n \log m)$ time to finish. The second part of the algorithm selects $\lceil \log n \rceil$ features from all the items, requiring no more than $O(\log n)$ time. Considering in normal cases, $n \not\gg m$ and $m \not\gg n$, the running time of algorithm 2 is still bounded by $O(n \log^2 m)$, which makes BS2 as fast as BS1.

2.3.2.2 Clustering for Predicting User Ratings

After applied to BS2, the sample data collected have been optimized for prediction. The prediction is then carried out by clustering the training data set based on the user profiles. In practice, the size of the raw data sets varies a lot. As a result, how to

Algorithm 2 Best Selection Algorithm (BS2)

program Best-Selection-Two
input: user-item matrix $\mathcal{P} \in \mathcal{R}^{m \times n}$
output: instance-feature list $\mathcal{F} \in \mathcal{R}^{\lceil \log m \rceil \times \lceil \log n \rceil}$
Initialize $\mathcal{C} \leftarrow$ BS1($\mathcal{P}$), $\mathcal{F} \leftarrow \emptyset, \mathcal{I} \leftarrow \emptyset$
for i equals 0 to n **do**
  for j equals 0 to $\lceil \log m \rceil$ **do**
    if $\mathcal{C}(j,i) > 0$
      $\mathcal{I}(i) \leftarrow \mathcal{I}(i) + 1$
  end
end
Append active user a to $\mathcal{C}$: $\mathcal{C} \leftarrow \mathcal{P}(a)$
for i equals 0 to $\lceil \log n \rceil$ **do**
  select the feature κ from $\mathcal{C}$ that satisfies:
  $\forall\, \xi \in \mathcal{C}\ and\ \xi \neq \kappa, \mathcal{I}(\kappa) > \mathcal{I}(\xi)$
  $\mathcal{F}_i \leftarrow \mathcal{F}_i \cup \mathcal{C}(\kappa)$
  $\mathcal{C} \leftarrow \mathcal{C} - \mathcal{C}_\kappa$
end
output $\mathcal{F}$

dynamically choose the number of clusters that could lead the results of clustering to the best prediction outcome becomes the key part of employing the classical clustering algorithm.

Several approaches have been applied to cluster the data sets of user-item matrices, which were mentioned in [33] as repeated clustering. In our approach, users who have similar preferences of the same items are clustered together. To determine the quality of clustering results, two metrics namely the average intra-class compactness and inter-class looseness of clusters are employed:

$$\mathcal{C}p(\mathcal{K}) = \frac{1}{\mathcal{K}} \sum_{i=1}^{\mathcal{K}} \sum_{j=1}^{N_i} \left(\mathcal{C}(i) - \mathcal{U}(i)\right)^2 \tag{2.5}$$

$$\mathcal{L}s(\mathcal{K}) = \frac{1}{\mathcal{K}} \sum_{i=1}^{\mathcal{K}} \sum_{j=1}^{\mathcal{K}} \left(\mathcal{C}(i) - \mathcal{C}(j)\right)^2 \tag{2.6}$$

In equation (2.5) and (2.6), $\mathcal{U}(i) \in \mathcal{F}$, where $\mathcal{F}$ is the result from BS2 and $\mathcal{F} \in \mathcal{R}^{\lceil \log m \rceil \times \lceil \log n \rceil}$, $\mathcal{K}$ denotes the number of clusters, N_i represents the total number of users in class i, and $\mathcal{C}(i)$ is the centroid of class i which is defined as:

$$\mathcal{C}(i) = \frac{1}{N_i} \sum_{j=1}^{N_i} \mathcal{U}(i) \tag{2.7}$$

The purpose of clustering is to minimize the average intra-class compactness while simultaneously maximizing inter-class looseness. In Algorithm 3, two initial thresholds ϵ_1, ϵ_2 are chosen for intra-class compactness and inter-class looseness, respectively. The number of clusters $\mathcal{K}$ are initialized to be half the number of instances selected from BS1. Each time when the KMeans algorithm is called, we get the labels for instances ι, cluster centroids c and the modified instance-feature list $\mathcal{F}'$ which appends the labels to the last column of $\mathcal{F}$. After that, new $\tilde{\rho}$ and $\tilde{\tau}$ are computed by using the new labels. If both conditions are satisfied, we take the clustering results and compute the prediction value of the active user a. However, if the computed $\tilde{\rho}$ is larger than the initial threshold, we recognize that the current clusters are not compact enough, i.e., users that are clustered in the same class may still exhibit different user profiles, indicating that the current clusters still need to be split.

In the third case, if the value of $\tilde{\rho}$ is small enough, but $\tilde{\tau}$ is smaller than the threshold (which means we have "over-cluster" the users and clusters are not maximally separated), then clusters are merged to get larger inter-class looseness values.

The running time of clustering, however, is still bounded by $O(n \log^2 m)$, which is equal to the running time of BS1. The worst situation happens when we need to try all $\mathcal{K}$'s from 2 to $\lceil \log m \rceil$, in which situation the selection of initial thresholds ϵ_1 and ϵ_2 may not be optimal.

To summarize, choosing the right initial values for intra-class compactness and inter-class looseness becomes crucial to both the program running time and clustering results. During the experiments, we found that for intra-class compactness, $(\lceil \log m/2 \rceil) \times maxdist(\mathcal{F})$ would be an optimal choice, where $maxdist(\mathcal{F})$ stands for the maximum pairwise distances between instances in the instance-feature matrix. The value for inter-class looseness, however, varies a lot due to the distances between centroids of clusters, which are greatly dependent on the distribution of the selection matrix. In practice, we figure out that the optimal value for $\mathcal{L}s$ is always around $\mathcal{C}p(\lceil \log m \rceil)$ / $\Re$, where $\Re$ is a value between of (2, 4).

Algorithm 3 Clustering for Prediction

program Cluster-Prediction
input: instance-feature list $\mathcal{F} \in \mathcal{R}^{\lceil \log m \rceil \times \lceil \log n \rceil}$
output: rating-prediction list $\mathcal{P}' \in \mathcal{R}^{\lceil \log m \rceil}$
Initialize $\mathcal{K} \leftarrow \lceil \log m/2 \rceil, \rho \leftarrow \epsilon_1, \tau \leftarrow \epsilon_2,$
$\iota \leftarrow \emptyset, c \leftarrow \emptyset, \mathcal{P}' \leftarrow \emptyset$
while $\mathcal{K} > 2$ and $\mathcal{K} < \lceil \log m \rceil - 1$ **do**
  $(\iota, c, \mathcal{F}') \leftarrow$ KMeans($\mathcal{F}, \mathcal{K}$)
  for i equals 1 to $\mathcal{K}$ **do**
    for j equals 1 to N_i **do**
      $\tilde{\rho} \leftarrow \mathcal{C}p(\mathcal{F}')$
      $\tilde{\tau} \leftarrow \mathcal{L}s(\mathcal{F}')$
    end
  end
  if $\tilde{\rho} < \rho$ and $\tilde{\tau} > \tau$
    break
  else if $\tilde{\rho} > \rho$
    $\mathcal{K} \leftarrow \mathcal{K} + 1$
  else if $\tilde{\rho} < \rho$ and $\tilde{\tau} < \tau$
    $\mathcal{K} \leftarrow \mathcal{K} - 1$
end while
for i equals 1 to $\lceil \log m \rceil$ **do**
  for j equals 1 to n **do**
    if $\iota_i = \iota_{\lceil \log m \rceil + 1}$
    $\mathcal{P}'_j \leftarrow \mathcal{P}'_j + \mathcal{W}(i, \lceil \log m \rceil + 1)\mathcal{P}(\lceil \log m \rceil + 1, j)$
  end
end
output $\mathcal{P}'$

2.4 Classifiers for Text Classification

In this section, we continue to discuss the classifiers we employed for text classification. We begin by first showing how we use SVM as a classifier, and then present the algorithm AdaBoost.MH for multiclass classification.

2.4.1 SVM for Text Classification

Support Vector Machines (SVM) classify training examples by using the strategy that maximizes the margin between critical examples and the separating hyperplane. Given a set of training examples $S = \langle (x_1, y_1), ..., (x_N, y_N) \rangle$, where each example $x_i \in \mathbb{R}^n$ and each label $y_i \in \{+1, -1\}$. Our task is to find a hyperplane that optimally separates

the positive and negative examples which can be interpreted as $w \cdot x + b = \pm 1$ ($w \in \mathbb{R}^n, b \in \mathbb{R}$). By optimal we mean that the hyperplane maximizes the Euclidean distance to the closest examples. In order to maximize the margin, we need to minimize $||w||$.

Generally, SVM is employed as a binary classifier. Nevertheless, two approaches can be used to extend SVM for multi-class classification [42]. *pairwise classification* trains SVMs for each pair of classes, thus totally $\frac{n(n-1)}{2}$ SVMs need to be trained for n classes, these SVMs are then arranged in trees where each node denotes an SVM. On the other hand, *one-against-all* approach only needs to train n SVMs. The trick is to treat only one class as a positive class (with label +1) at a time and all remaining as negative classes (with label -1). Recently, multiclass SVMs have emerged to be an alternative and reported to be more effective than the predecessors. In this paper, we employ [25] as an implementation of multiclass SVM.

2.4.2 AdaBoost.MH

AdaBoost.MH (cf Algorithm 4) is used for multiclass multi-label text classification. Given a sequence of training examples $S = \langle (x_1, y_1), ..., (x_N, y_N) \rangle$ where each example $x_i \in \mathcal{X}$ and each label $y_i \in \mathcal{Y}$, for $Y \subseteq \mathcal{Y}$ which is the set of labels assigned to x_i, $Y[\ell]$ ($\ell \in \mathcal{Y}$) is defined as 1 if $\ell \in Y$ and 0 otherwise.

In each round, the distribution $D_t(i)$ over all instances is dynamically maintained and updated. $D_1(i)$ is initially set to be uniform. During the t-th round, the distribution $D_t(i)$ and the example sequence S are sent to the weak learner which later returns a weak hypothesis h_t that minimizes the *Hamming loss*, i.e., to minimize the probability of the number of examples (i, ℓ) whose sign of $f(x_i, \ell)$ differs from its observed sign $Y_i[\ell]$. As a result, $D_t(i)$ is updated in the manner that more weight is given to the examples that were misclassified by h_t during the t-th round. After T iterations or a termination condition is met, the final hypothesis $H(x, \ell)$ is calculated. For each example, the label can also be computed by $L(x) = sign\,(H(x, \ell))$.

2.4.2.1 Weak Hypotheses for Text Classification

Boosting is well-suited as a general purpose method that can be combined with any classifier. The weak hypotheses h we use here is as simple as a one-level decision tree. For example, a possible phrase could be *computer graphics*, the corresponding predictor

Algorithm 4 AdaBoost.MH for Multiclass Classification

procedure AdaBoost.MH
input: N labeled documents $\langle (x_1, y_1), ..., (x_N, y_N) \rangle$
    where $y_i \in Y = \{1, ..., k\}$
    distribution D over the N documents
    weak learning algorithm **WeakLearn**
    number of iterations T

Initialize $D_1(i, \ell) = 1/(mk)$. /* *uniform distribution* */
for $t = 1, 2, ..., T$ **do**
  1. Call **WeakLearn**, providing with distribution D_t
  2. Get back a weak hypothesis $h_t : X \times Y \to \mathbb{R}$.
  3. Choose optimal update step $\alpha_t \in \mathbb{R}$
  4. Update the new distribution
  5. Set the new weights vector:

    $D_{t+1}(i, \ell) = \frac{D_t(i,\ell) \exp(-\alpha_t Y_i[\ell] h_t(x_i,\ell))}{Z_t}$
    for $i = 1, ..., N, y \in Y - \{y_i\}$.
  where $Z_t = \sum_{i=1}^{m} \sum_{\ell \in \mathcal{Y}} D_t(i,\ell) \exp(-\alpha_t Y_i[\ell] h_t(x_i,\ell))$
end
Output the hypothesis: $H(x, \ell) = \sum_{t=1}^{T} \alpha_t h_t(x, \ell)$.

is "If *computer graphics* appears in the document then predict that the document in the class *SIGGRAPH* with high confidence; predict that the document in the class *ICCV* with low confidence; and predict that it does not belong to any other classes with low confidence." Formally, the weak hypotheses h can be defined as:

$$h(x, \ell) = \begin{cases} c_{0\ell} & \text{if } w \notin x; \\ c_{1\ell} & \text{if } w \in x. \end{cases}$$

Where $w \in x$ denotes that a possible phrase w occurs in document x, which was categorized based on a binary feature for w: $\mathcal{X}_0 = \{x : w \notin x\}$ and $\mathcal{X}_1 = \{x : w \in x\}$.

The process of constructing weak hypotheses is as follows: each round the weak learners check all possible phrases, for each of which the values $c_{j\ell}$ are selected with respect to some criteria, and a score is given for the resulting weak hypothesis. When the search of all phrases are done, the weak hypothesis with the lowest score is returned by the weak learner.

To simplify the weak hypotheses, we choose discrete values for $c_{j\ell}$, i.e., the value of

$c_{j\ell}$ is either +1 or -1. Thus we set the prediction $c_{j\ell} = sign(W_+^{j\ell} - W_-^{j\ell})$[6]. Here $W_+^{j\ell}$ and $W_-^{j\ell}$ are the weights of documents in X_j that are labeled and not labeled ℓ, respectively. Each round they are updated as follows:

$$W_b^{j\ell} = \sum_{i=1}^{m} D_t(i,\ell)\,[x_i \in X_j \wedge Y_i[\ell] = b], b \in \{-1,+1\}. \tag{2.8}$$

To minimize D_t, the corresponding optimal parameter update is given by

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1+r_t}{1-r_t}\right), \tag{2.9}$$

where r_t is a weighted major vote over examples in block X_j, specified as:

$$\begin{aligned} r_t &= \sum_{j\in\{0,1\}}\sum_{\ell\in\mathcal{Y}} |W_+^{j\ell} - W_-^{j\ell}| \\ &= \sum_{\ell\in\mathcal{Y}} |W_+^{0\ell} - W_-^{0\ell}| + |W_+^{1\ell} - W_-^{1\ell}|. \end{aligned} \tag{2.10}$$

Notice that when α_t is positive, the distribution D_t is updated in the way that the weight of misclassified example-label pairs always increases. With the normalization of weight distribution in mind, this means that more weights are put to the samples that are not correctly classified by h in t's iteration.

2.5 Empirical Analysis of Collaborative Filtering Algorithms

To evaluate our proposed CF algorithms, we evaluate the performance based on computational complexity and prediction accuracy. In this section, we first introduce the data sets used for experiments, then present the protocols and the metrics for evaluation. Subsequently, we show the results of our experiments.

[6]The alternative is to use real values for $c_{j\ell}$, where the prediction $c_{j\ell} = \frac{1}{2}\ln\left(\frac{W_+^{j\ell}}{W_-^{j\ell}}\right)$

2.5.1 Data Sets Preparation

We use both self generated data set and real data sets to perform a range of experiments. For the self generated data, two random matrices are created that contain 200 rows and 500 columns, and intensionally prevent 0 from user ratings. The reason is to intensionally create very detailed user profiles regarding the items in the data sets, which can't be observed from the real data sets. In the same time, we also use two open data sets from GroupLens, the first of which contains 943 users and 1,682 movies, with totally 100,000 ratings, about 6.3% of the entries in the matrix are non–empty. The second data set consists of 6,040 users of MovieLens and approximately 3,900 movies, with totally 1,000,209 ratings. It is even sparser than the first data set — only 4.2% entries have ratings. Table 2.2 summarizes the statistics.

2.5.2 Experiment Setup

To better visualize the outcome of our proposed algorithms, three representative algorithms in three different categories are selected for comparison. The first one is from Breese et al. [15], the classical memory-based CF algorithm that applies vector similarity as its weight function (VSIM); the second is Fast Correlation-Based Filter [148] (FCBF) that uses feature selection technique; the last is Personality Diagnosis [97] (PD) that unifies memory and model-based approaches.

Name	Instances	Features
Synthetic data	200	500
Movielen1	943	1,682
Movielen2	6,040	3,900

Table 2.2. Summary of benchmark datasets

Three protocols are employed: *all but one*, which simply chooses a random rating for each user that is not equal to 0 and withhold it; *given two* and *give ten*, which withholds all user ratings except the given number of ratings.

For evaluation metrics, we employ two classical metrics, *MAE* and *RS*, plus another effective measurement *MUG*.

- MAE (Mean Absolute Error) – MAE represents how much the mean predicted

values deviate from the actual/observed values of all users in the dataset. $MAE_a = \frac{1}{m_a} \sum_{j \in P_a} |p_{a,j} - o_{a,j}|$. Obviously, the lower the MAE is, the better the prediction.

- RS(Ranked Scoring) – RS multiplies the utility of an item by the likelihood that the item may be rated by the user. It estimates the probability that an item will be viewed by the user. $RS_a = \sum_j max(o_{a,j} - d, 0) * \frac{1}{2^{(j-1)/(\alpha-1)}}$. The higher RS score is, the better the prediction is.

- MUG(Mean User Gain) – MUG computes the average quality of recommendations for the predicted values for users. $MUG_a = \frac{1}{n_a} \sum_{j \in P_a} UG(p_{a,j})$.

2.5.3 Results and Discussions

Table 2.4, Table 2.5 and Table 2.6 exhibit the experiment results with regard to MSE scores, RS scores and MUG scores, respectively. BS1 and BS2 perform better than PD, FCBF and VSIM in all three protocols for all test data sets. The average improvements of BS1 and BS2 regarding MSE, RS and MUG scores are 12.7%, 20.4% and 13.5%, by comparing to the average scores of the three algorithms. Figure 2.4 shows two results. (We only pick one algorithm that performs best in that situation, for comparison to our approach in each graph)

Datasets	Running Time (in ms)				
	VSIM	PD	FCBF	BS1	BS2
Synthetic	3765	2688	3211	**2166**	**2238**
MovieLen1	4833	3200	4587	**2544**	**3517**
MovieLen2	15669	5644	10238	**4233**	**5291**

Table 2.3. Algorithm Running Time.

Table 2.3 shows the running time of these five algorithms. BS1 and BS2 require significantly less time than the other three algorithms, especially for the large data sets. Figure 3 depicts how running time changes as the data sets become larger, where increments for both BS1 and BS2 are small, which indicates that our algorithms are scalable and the performance gain will be greater for larger data sets.

While BS1 and BS2 outperform others, it is necessary to make comparison between these two. Table 2.7 summarizes the performance comparison between BS1 and BS2.

We use the measurement of how much can BS2 improve from BS1.

We notice that BS2 performs better for the metrics MSE and MUG, while BS1 still have advantage on RS scores and running time. Since BS2 uses the results (candidate lists) from BS1, it is then reasonable for BS2 to cost a little bit more time than BS1.

The main cost of BS2 is the KMeans function. In BS2, we call KMeans several times with different value of $\mathcal{K}$ that equals to the number of clusters. During the experiment, we find out that the number of iterations for the clustering algorithm is usually very small, which results in a fast converge of the algorithm.

Datasets	Algorithms	Protcols		
		AllBut1	*Given10*	*Given2*
Synthetic	PD	0.710	0.756	0.698
	FCBF	0.875	0.924	0.933
	VSIM	1.324	1.368	1.297
	BS1	**0.724**	**0.788**	**0.655**
	BS2	**0.692**	**0.724**	**0.633**
MovieLen1	PD	0.964	0.986	1.039
	FCBF	0.999	1.069	1.296
	VSIM	2.136	2.235	2.113
	BS1	**0.825**	**0.826**	**0.878**
	BS2	**0.814**	**0.835**	**1.022**
MovieLen2	PD	1.023	1.011	1.125
	FCBF	1.001	1.068	1.265
	VSIM	2.345	2.274	2.256
	BS1	**1.078**	**1.079**	**1.079**
	BS2	**0.802**	**0.811**	**0.978**

Table 2.4. MSE Scores

2.6 Text Classification Experiments and Discussions

In this experimental evaluation, we ran a series of experiments to compare our proposed text classification approach with traditional methods on two data sets: CiteSeer Digital Library and WebKB benchmark corpus[26]. Specifically, three kinds of experiments are carried out:

First, we make comparison between entity extraction techniques in terms of the di-

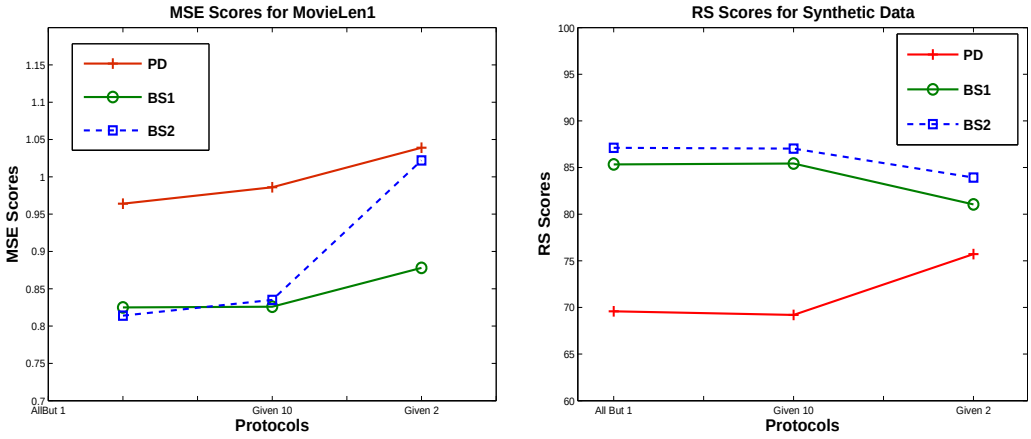


Figure 2.4. MSE scores and RS scores of PD, BS1 and BS2.

Datasets	Algorithms	RS Results		
		AllBut1	*Given10*	*Given2*
Synthetic	PD	69.58	69.20	75.72
	FCBF	76.81	71.17	59.70
	VSIM	68.33	62.34	57.22
	BS1	**85.33**	**85.43**	**81.05**
	BS2	**87.11**	**87.03**	**83.92**
MovieLen1	PD	65.22	65.08	61.22
	FCBF	73.45	70.23	59.27
	VSIM	62.51	60.31	58.11
	BS1	**74.11**	**77.25**	**75.33**
	BS2	**72.35**	**75.42**	**69.21**
MovieLen2	PD	62.11	64.35	59.22
	FCBF	65.33	68.24	62.77
	VSIM	61.24	53.27	55.78
	BS1	**68.25**	**69.77**	**60.23**
	BS2	**69.33**	**64.44**	**62.91**

Table 2.5. RS Scores

mensionality of the feature space. We compare our proposed *SVM-decision-tree* approach to the *bag-of-words* method with the standard TFIDF approach as an extension. To be more convincing, *Information Gain* (IG) is applied to the *bag-of-words* approach as a feature selection criteria. A feature y is deemed useful if its expected IG exceeds the threshold.[7] Comparisons are also made between IG and *SVM-decision-tree*.

[7]Experimentally, the threshold is usually chosen to maximize the F-measure on a validation set.

Datasets	Algorithms	MUG Results		
		AllBut1	*Given10*	*Given2*
Synthetic	PD	0.62	0.68	0.12
	FCBF	0.81	0.61	0.42
	VSIM	0.34	0.32	0.08
	BS1	**0.83**	**0.72**	**0.32**
	BS2	**0.89**	**0.87**	**0.55**
MovieLen1	PD	0.58	0.54	0.09
	FCBF	0.78	0.60	0.37
	VSIM	0.35	0.37	0.11
	BS1	**0.81**	**0.75**	**0.68**
	BS2	**0.78**	**0.62**	**0.60**
MovieLen2	PD	0.49	0.45	0.11
	FCBF	0.72	0.55	0.54
	VSIM	0.31	0.30	0.07
	BS1	**0.75**	**0.65**	**0.69**
	BS2	**0.72**	**0.60**	**0.55**

Table 2.6. MUG Scores

	Synthetic	MovieLen1	MovieLen2	Overall
MSE	5.6%	4.7%	9.2%	6.5%
RS	-2.4%	-2.8%	-1.9%	-2.4%
MUG	6.5%	5.8%	6.3%	6.2%
Running Time	-3.2%	-29%	-19.9%	-17.4%

Table 2.7. Performance Improvement of BS2 over BS1.

Furthermore, to illustrate that the CF algorithm indeed boosts the feature space, we compare the distribution of features in each class Before Collaborative Filtering (B-CF) and After Collaborative Filtering (A-CF). To be more convincing, we also calculate the distribution of features from prediction results by using the classic Inner Product approach (I-CF) proposed by Breese et al. in [15], where the weight function is calculated as $w(a,i) = \sum_j \frac{v_{a,j}}{\sqrt{\sum_{k \in I_a} v_{a,k}^2}} \frac{v_{i,j}}{\sqrt{\sum_{k \in I_i} v_{i,k}^2}}$, and the prediction score is computed through the whole data set.

Finally, we use multiclass SVM [25] and AdaBoost.MH [23] to classify the feature space extracted by (1) not using CF (B-CF-SVM and B-CF-Boost), (2) using IG feature selection (IG-SVM and IG-Boost), and (3) using CF (A-CF-SVM and A-CF-Boost).

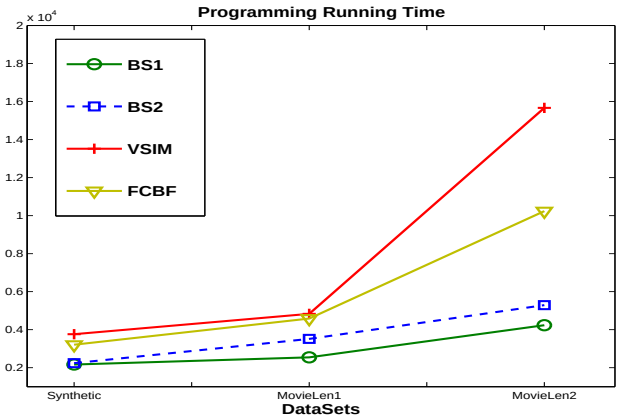


Figure 2.5. Comparison between algorithm running time. BS1 and BS2 require much less time than other algorithms.

The Vector Similarity method (VSIM) is used as baseline for comparison. Additionally, since it was shown that SVMs can perform well even without feature selection [125](SVM-No), it is also compared in the experiment. We apply *Precision*, *Recall* and *F-measure* as measures for our text classification.

2.6.1 Information Gain

Despite the existence of many successful feature selection methods, Information Gain (IG) has been experimentally proved to be among the most popular approach for feature ranking [39]. It is a measure based on Entropy. Formally, given the set of all training examples $X = \{x_1, ..., x_m\}$ and the number of features $Y = \{y_1, ..., y_n\}$, the IG of a feature y is defined as:

$$\begin{aligned} IG(X, y) &= H(X) - H(X|y) \\ &= -\sum_{j=1}^{m} p(j) \log_2 p(j) + p(y) \sum_{j=1}^{m} p(x_i|y) \log p(x_i|y) \end{aligned} \tag{2.11}$$

Given a training corpus, the IG of all features are computed after extracted by *bag-of-words approach* (with TFIDF extension). A feature y is then deemed useful if its expected IG exceeds the threshold. Experimentally, the threshold is usually chosen to maximize the F-measure on a validation set.

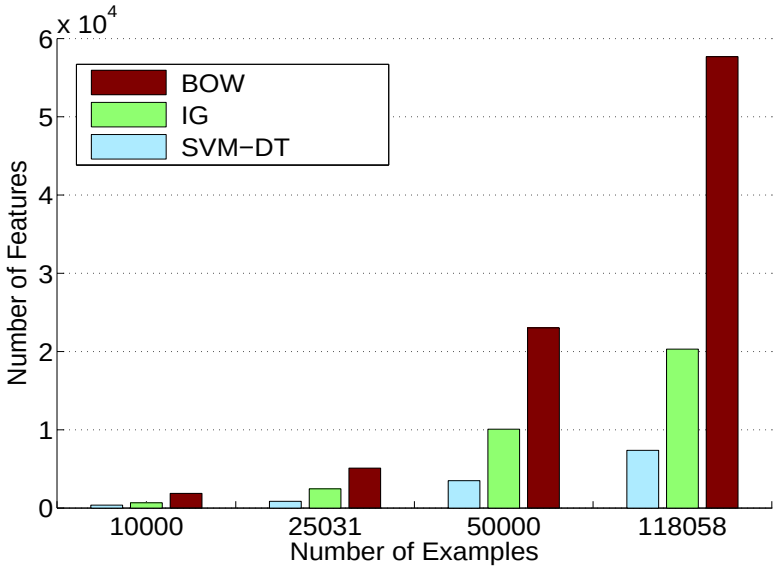


Figure 2.6. Features extracted by *bag-of-words* (BOW) and *SVM-decision-tree* (S-D) from the summarizing parts of the documents in CiteSeer data set, where S-D creates a much smaller feature space as a function of example size. The number of examples chosen by IG is decided by maximizing the F-measure on the validation set.

2.6.2 CiteSeer Data Preparation

The data we used for experiments are from CiteSeer, one of the largest digital libraries which now holds about 747,588 documents primarily in the domain of computer science, and the number is ever-growing. Several kinds of data formats are indexed concurrently (*txt, pdf, ps, archives* and so on), for the purpose of text extraction, we only make use of plain text files or convert non-text formats into text format by programming. As mentioned in Section 3, for the purpose of current experiments, we only consider extracting entities from the summarizing parts of the documents, i.e., the titles, abstracts and keyword fields. Documents that do not contain either abstracts or keywords are not under consideration.

Document class labels are obtained from the *venue impact page*[8] which lists 1,221 major venues whose titles are named according to DBLP[9] format. For the purpose of experiments, we only consider the top 200 publication venues listed in DBLP in terms of impact rates, each of which was referred as a class label[10]. Furthermore, we

[8]http://citeseer.ist.psu.edu/impact.html
[9]http://dblp.uni-trier.de/
[10]We manually merged venues with the same names but different volumes, like *ECCV(1), ECCV(2),*

number of examples	118,058
number of classes	193
number of examples per class	590
average file size	40KB
total file size	4728MB

Table 2.8. Statistics of CiteSeer data set.

	examples	VSIM	SVM-No	Before CF		Info Gain		After CF	
				SVM	Boosting	SVM	Boosting	SVM	Boosting
P	10,000	24.31	62.17	46.44	65.74	53.77	56.29	80.25	**85.24**
	25,031	25.17	85.77	68.33	82.33	85.63	87.21	91.01	**94.08**
	50,000	25.24	86.02	70.47	82.53	86.31	88.24	92.53	**93.22**
	118,058	27.96	89.42	82.77	85.32	89.32	89.33	**95.77**	94.66
R	10,000	10.23	13.75	11.43	11.77	11.85	12.11	**14.53**	12.11
	25,031	25.72	33.23	26.22	30.25	28.53	31.74	**42.77**	40.69
	50,000	34.81	**50.25**	35.79	29.88	42.79	40.01	**50.25**	49.00
	118,058	72.38	84.88	74.25	77.91	83.99	72.53	**85.27**	73.00
F	10,000	14.40	22.52	18.34	17.73	20.11	18.35	24.61	**25.21**
	25,031	25.44	47.90	37.89	44.24	38.29	48.32	**58.19**	56.81
	50,000	29.26	63.25	47.47	43.88	56.77	60.11	64.13	**64.24**
	118,058	40.34	87.09	78.28	81.45	79.52	83.23	**90.22**	81.14

Table 2.9. Experimental results of CiteSeer data set in terms of Precision (P), Recall(R) and F-measure(F), averaged over all classes. VSIM is compared as a baseline approach. Our approach (A-CF) shows competitive results on both classifiers. IG chooses top k features to maximize the F-measure of the validation set. For the entire data set (118,058), k is around 20,000.

intentionally filtered those classes that contain too few examples (i.e., less than 100 documents). Overall, the total number of documents we used for the experiments is up to 118,058, divided into training set and testing set by doing 10-fold cross-validation. Notably, we keep the imbalance of the classes, i.e., some classes have more training examples than others. A few statistics are shown in Table 2.8. Table 2.10 shows the number of documents in the top 20 classes.

class	samples	class	samples	class	samples
PLDI	1259	*SOSP*	816	*SIGCOMM*	1416
MICRO	812	*POPL*	1741	*ICML*	1924
HPCA	895	*ICCV*	1316	*KDD*	1236
VLDB	2431	*AAAI*	1268	*INFOCOM*	3366
MOBICOM	505	*SIGGRAPH*	1511	*SIGIR*	878
CVPR	1611	*PODS*	842	*NIPS*	3421
SIGMOD	1843	*WWW*	382		

Table 2.10. Distribution of samples of top 20 classes in terms of sample numbers

2.6.3 Metrics Setup

We use three most widely used metrics in information retrieval as measures for our text classification.

$$\begin{aligned} Precision &= \frac{\text{number of relevant documents retrieved}}{\text{number of documents retrieved}} \\ Recall &= \frac{\text{number of relevant documents retrieved}}{\text{number of relevant documents}} \\ F-measure &= \frac{2 \cdot precision \cdot recall}{precision + recall} \end{aligned}$$

Notice that the general $F - measure$ is defined as $(\beta^2 + 1) * p * r / ((\beta^2 * p) + r)$, here we set $\beta = 1$.

2.6.4 Classification Results on CiteSeer Data Set

Figure 2.6 presents the number of features extracted by the three techniques. We ran the experiments with the number of documents, D, equal to 10,000, 25,031, 50,000 and 118,058. Using *SVM-decision-tree* approach yields a much lower-dimensional feature space compared with the *bag-of-words* method (with TFIDF), especially when the number of examples are very large. Information Gain successfully reduces the feature space to half the dimension of *bag-of-words*, but when the training data size becomes larger (118,058), it still creates a feature space of more than 20,000 features, while our approach ends up with a feature space with a little more than 7,000 features. We also notice that the dimension of feature space generated by our approach is almost linear in

so the total number of venues actually used for classification is 196.

the number of examples, indicating nice scalability of our entity extraction technique.

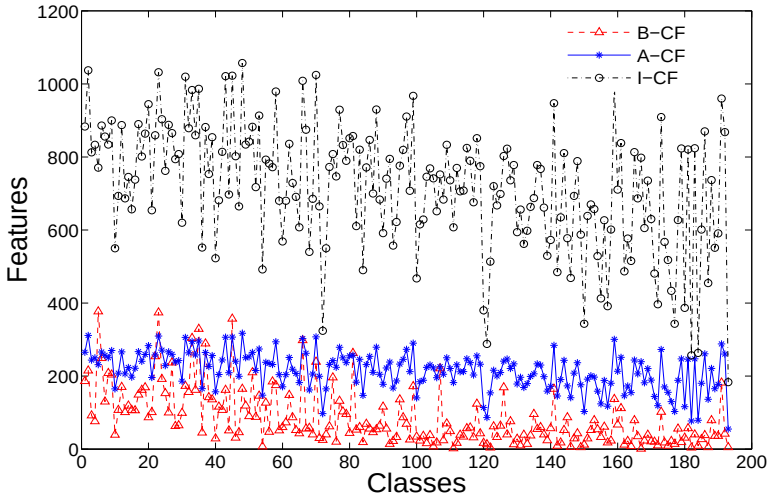


Figure 2.7. Feature distribution where B-CF denotes the feature space before applying CF, I-CF the feature space augmented by Inner-Product method, and A-CF the feature space augmented by our CF algorithm.

In Figure 2.7, we depict the distribution of features for three approaches that applied to the feature space extracted by *SVM-decision-tree* approach. Before applying CF algorithm (B-CF), the features are unevenly distributed in each class due to the random distribution of training examples in different categories. By using the Inner Product algorithm (I-CF) it first computes the correlations between each pair of examples, and then predicts the feature frequencies from the knowledge of all examples. As a result, I-CF generates too many features for each class that inevitably causes overlapping in the feature spaces, which leads to reduction of classification accuracy. Finally, by employing the CF algorithm we proposed (A-CF), the feature space is boosted to a reasonably dense level that yields a nearly even distribution of features in each class. The virtue of the boosted feature space is not only that it contains enough features within each class which makes it easy to classify, but also results in very little overlapping of different classes in the feature space, which reduces the misclassification rate significantly in comparison with I-CF. Figure 2.8 compares the feature spaces for 2 classes by applying I-CF and A-CF, respectively. We use *Singular Value Decomposition* (SVD) to get the first 3 *principal components* of the matrix and visualize in a 3-D graph. It is not hard to see that I-CF leads to a much more overlapping space than our approach, which

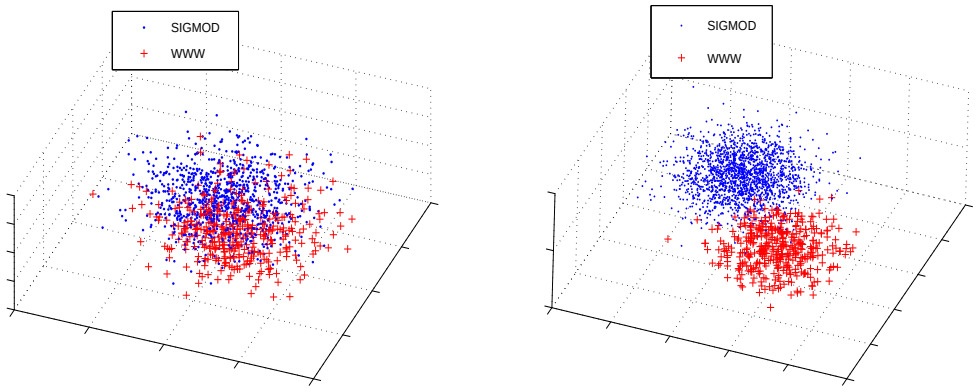


Figure 2.8. Visualization of SVD feature distribution of classes (*SIGMOD, WWW*). The left figure shows features by the I-CF inner-product, the right figure the boosted feature space by our A-CF algorithm.

generally separates two classes very well.

Table 2.9 summarizes experimental results for the three metrics averaged over all classes. With regard to *precision*, our approach achieves significant improvement on both classifiers. When the number of examples is small (10,000), A-CF-SVM and A-CF-Boost improve the precision over the VSIM baseline approach by nearly 4 times, and nearly twice as much as the results of Information Gain (IG-SVM and IG-Boost). When the whole data set is applied to the experiment (118,058), A-CF-SVM and A-CF-Boost achieve the best results of 95.77% and 94.66% respectively, about 5% more than IG. Meanwhile, without feature selection, SVM (SVM-No) shows almost the same precision as IG-SVM, with a slightly better result when the number of examples is small.

In terms of *recall*, all methods have very close performances. Comparatively, SVM performs slightly better than AdaBoost regardless of data size and entity extraction techniques. Especially when the data size is large (118,058), the baseline approach achieves recall of 72.38 %, almost the same as A-CF Boost method (73.00%). However, both of which are nearly 13% lower than A-CF-SVM approach. Both SVM-No and A-CF-SVM achieve the best recall among all when the number of examples equals 50,000.

Our approach outperforms IG for both classifiers in terms of *F-measure*, with an exception when the data size is 118,058, IG-Boost outperforms A-CF-Boost by 2%. Both IG-Boost and A-CF-Boost are almost 10% less than that of A-CF-SVM method, which shows the best performance of all. During the experiments, we also noticed that

No. of pages	*BOW+TFIDF*	*IG*	*SVM-DT*
1,000	2,413	1,533	**977**
2,000	4,987	2,422	**1,777**
4,000	7,400	5,324	**3,599**
8,282	10,322	6,891	**5,111**

Table 2.11. Number of features exacted by three techniques w.r.t. number of pages for the WebKB data set. SVM-DT approach yields a much smaller feature space.

the training time of SVM and AdaBoost are almost the same with SVM slightly better in some cases.

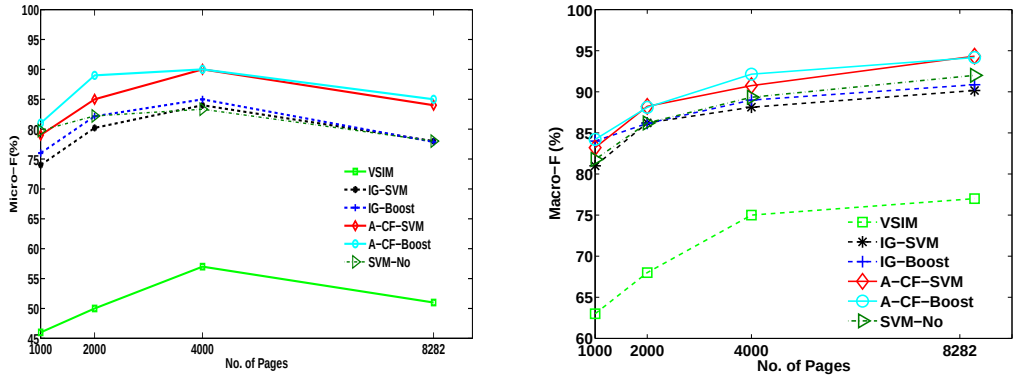
2.6.5 WebKB:World Wide Knowledge Base

The WebKB data set contains web pages collected from cs departments of many universities by the World Wide Knowledge Base project of the CMU text learning group in January 1997. For performance evaluation, we divide the data into training and testing set with the proportion of 4:1. A series of experiments were performed with the number of documents equal to 1,000, 2,000, 4,000 and 8,282. The number of iterations T for AdaBoost is set to 500.

Table 2.11 summarizes the number of features with regard to the training data size. The *SVM-decision-tree* approach creates a much smaller feature space than *bag-of-words* and *Information Gain*—20% less than the IG and 50% less than the BOW when the total WebKB collection is used.

Figure 2.9(a) shows the result of the *Micro-F* scores. When the number of training pages is small, our approach has almost the same performance as IG for both classifiers, with less than 2% improvement. As the page size get larger, the performance improvement of our approach becomes greater. When the whole collection is used, our approach outperforms IG by more than 5%, but the performance decreases for both methods as the best results are achieved when the page size is 4,000. Note that SVM-No has almost the same performance as IG-SVM.

Macro-F scores are shown in Figure 2.9(b). Clearly, the baseline approach VSIM performs the worst regardless of data size. SVM-No again performs nearly the same as IG-SVM. Note that with the increase of pages, the macro-F scores increase as well for all methods. Our approach generally outperforms IG, and the advantage becomes larger



(a) (b)

Figure 2.9. Micro-F(a) and Macro-F(b) results for WebKB w.r.t. data size.

with the increase of data size. Our approach achieves a significant improvement by 8% over IG for both classifiers when the whole WebKB collection is applied.

2.7 Related Work

Collaborative Filtering

Predicting user preferences and giving useful recommendations to users with limited information gained from the users has become a key challenge for E-commerce companies. Collaborative filtering (CF) has been widely adopted as a technique for constructing recommender systems, but is also well-suited to handle research issues in areas such as artificial intelligence (AI) and human computer interaction (HCI).

GroupLens [102] became the first open architecture for news-related collaborative filtering. Users rated articles upon reading them and the news rating server of GroupLens would use that information to automatically make recommendations to others. Breese et al. [15] described various CF algorithms such as model-based Bayesian algorithms, memory-based correlation coefficient algorithms, etc. Matrices such as Mean Absolute Error (MAE) and Ranked Scoring (RS) were used for evaluation purposes. Additionally, their data set EachMovie has become one of the most widely used databases. Memory-based and model-based algorithms were combined together to form a new algorithm, personality diagnosis (PD) [97]. Recent approaches include integrating collaborative filtering with content-based filtering algorithms [6], applying statistical and mathematical methods such as feature selection, matrix factorization and Gaussian

processes for high dimensional data [148, 100, 120], and using clustering algorithms (EM, Gibbs sampling) to group existing users in data sets into different clusters for recommendations[130], as well as the determination of better weight functions for more robust prediction [89].

Entity Extraction

Entity extraction techniques typically fall into one of two categories: named entity recognition (NER) and phrase extraction. NER deals with identifying proper names text, extracting paper titles and author names in on-line publications and so on. Phrases, or meaningful entities, can be recognized as *signature* that best represent the main idea of papers, most of which can be found in the titles, abstracts and keyword fields in a paper. However, only a few publication venues require keyword fields[11].

Maximum Entropy (ME) is a feature reduction approach that works by choosing the model with the most uniform probability distribution (the highest entropy), the model is described as $P(w|h) = \frac{1}{Z(h)} \cdot e^{\sum_i \lambda_i f_i(h,w)}$, where $f_i(h, w)$ denotes a binary feature function that describes a certain term; λ_i is a parameter that indicates how important feature f_i is for the model. The disadvantage of ME is that it cannot automatically select features from given feature sets thus relying on careful feature selection techniques.

Conditional random fields (CRFs) [73] is another NER technique that aims to label and segment data into phrases. It works by defining a conditional probability distribution over training data given a particular observation phrase. It usually works better than HMM and avoids the label bias problem, however, the training time of CRFs is prohibitively long.

Traditional *bag-of-words* approach represents each document as a feature vector and often leads to feature spaces that are sparse and large, high classification accuracies are thus hard to get. Contemporary approaches of text classification concentrate on extracting more meaningful features from structured text, e.g., adding numeric features such as timestamps [83], capturing features that share mutual information and are dependent on each other [133], as well as seeking better methods to refine the classification model based on the prediction errors from the training data sets [36, 82]. Several classifiers have been introduced to text classification, e.g., Naive Bayes [95], maximum entropy [90] and Boosting [20]. Support Vector Machines (SVMs) [14, 147, 153], which fo-

[11] From an investigation of ACM & DBLP metadata cross-referenced with CiteSeer data, 5% of venues do not have an identifiable abstract field and more than 70% of venues lack a keyword field.

cus on finding the hyperplane that maximizes the margin between positive and negative classes, have typically been the most effective classifiers with regards to the classification errors. Forming the feature space has become for many a crucial part of using SVMs as text classifiers, since naturally there are hundreds of terms in each document and thousands of documents in each class, which results in very high-dimensional feature spaces. Yet it has been reported that SVMs can still achieve high accuracies in document classification without feature selection [125].

Research on entity extraction spans the fields of linguistics and computer science. Linguistic techniques can be employed to enhance feature selection from raw text by grouping text into semantically meaningful chunks. Developments in entity extraction technology have traditionally been concerned with the issue of computational complexity as well as extraction accuracy and domain specificity. Methods for entity extraction from unstructured data typically fall into two categories: pattern-based approaches and model-based approaches. Pattern-based extractions require extensive manual labor for detecting patterns and is generally not robust to variant data. On the other hand, model-based approaches like hidden Markov models (HMMs) [111] and SVMs [71], while requiring careful feature selection, have proved to be robust and flexible.

Chapter 3

Informative KNN Classification

3.1 Pattern Recognition Leveraging K-nearest Neighbor Algorithm

The K-nearest neighbor (KNN) classifier has been both a work-horse and benchmark classifier [24, 4, 3, 96, 150]. Given a query point x_0 and a set of N labeled points $\{x_i, y_i\}_1^N$, KNN classifier tries to predict the class label of x_0 on the predefined P classes by finding the K nearest neighbors of x_0 and applying a majority vote to determine its label. Without prior knowledge, the KNN classifier usually applies Euclidean distances as the distance metric. Nevertheless, this simple method can usually yield competitive results even compared to other sophisticated machine learning methods. It has been well-used in applications include image categorization, face recognition, document classification and etc.

Since it is well known that by effectively using prior knowledge such as the distribution of the data and feature selection, the performance of KNN classifiers can be significantly improved, researchers have attempted to propose new approaches to improving the performance of the KNN method, e.g., Discriminant Adaptive NN [53] (DANN), Adaptive Metric NN [35] (ADAMENN), Weight Adjusted KNN [50] (WAKNN), Large Margin NN [140] (LMNN) and so on. Despite the success and rationale of these methods, most have several limitations in practice, including the effort to tune numerous parameters (DANN introduces two new parameters, K_M and ϵ; ADAMENN has six input parameters in total that could potentially cause overfitting), the required knowledge

in other research fields (LMNN applies semidefinite programming for the optimization problem), the dependency on specific applications (WAKNN is designed specifically for text categorization) and so on. Additionally, choosing the proper value of K is still a crucial task for most KNN extensions, making it more compounded.

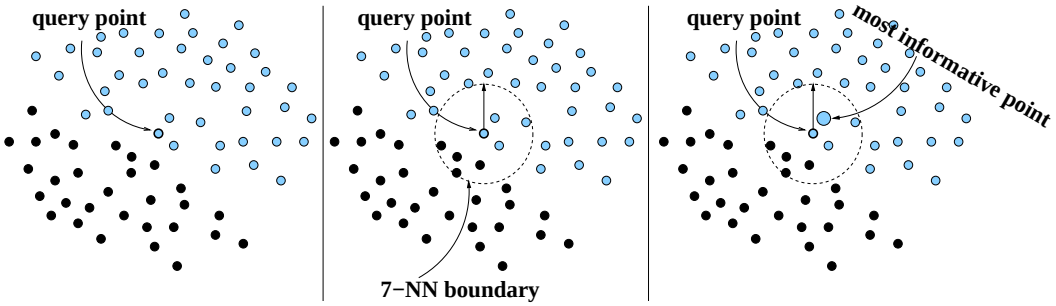


Figure 3.1. A toy classification problem. **(Left)** The original distribution of two classes. **(Middle)** Results of KNN ($K = 7$) method where the query point is misclassified. **(Right)** One of our proposed methods LI-KNN uses one informative point for prediction.

Therefore, it is desirable to enhance the performance of KNN without compromising its efficiency by introducing much overhead to this simple method. We thus propose two KNN methods which are ubiquitous and the performances are insensitive to the change of input parameters. Figure 3.1 gives an example that shows the motivation of our approach, in which the traditional KNN method fails to predict the class label of the query point with $K = 7$. Meanwhile, one of our methods (LI-KNN) finds the most *informative* point ($I = 1$) for the query point with the same K according to the new distance metric, and makes a correct prediction.

3.1.1 Our Contribution

In this chapter, we propose two novel extensions to the KNN method, whose performances are relatively insensitive to the change of parameters. Both of our methods are inspired by the idea of *informativeness*. Generally, a point is treated *informative* if it is *close to the query point and far away from the points with different class labels*. Specifically,

(1) We introduce a new concept named informativeness to measure the importance of points, which can be used as a distance metric for classification. (2) Based on the new distance metric, we propose an efficient *locally informative* KNN (LI-KNN) method. (3)

By learning a weight vector from the training data, we propose our second method that finds the *globally informative* points for KNN classification (GI-KNN). (4) We perform a series of experiments on real world image data sets by comparing with several popular classifiers including KNN, DANN, LMNN, SVM and Boosting. (5) We discuss the optimal choice of the input parameters (K and I) for LI-KNN and GI-KNN and demonstrate that our methods are relatively insensitive to the change of parameters.

3.2 Locally Informative KNN (LI-KNN)

Without prior knowledge, most KNN classifiers apply Euclidean distances as the measurement of the *closeness* between examples. Since it has been shown that treating the neighbors that are of low relevance as the same importance as those of high relevance could possibly degrade the performance of KNN procedures [41], we believe it to be beneficial to further explore the information exhibited by neighbors. In this section, we first propose a new distance metric that assesses the informativeness of points given a specific query point. We then proceed to use it to augment KNN classification and advocate our first method, LI-KNN.

3.2.1 Definition of Informativeness

We use the following naming conventions. Q denotes the query point, K indicates the K nearest neighbors according to a distance metric, and I denotes informative points based on equation 3.1. x_i denotes point i's feature vector, x_{ij} its j-th feature and y_i its class label. N represents the number of training points, each of which has P features.

Definition 1. *Specify a set of training points $\{x_i, y_i\}_1^N$ with $x_i \in \mathbb{R}^P$ and $y_i \in \{1, ...m\}$. For each query point x_i, the **informativeness** of each of the remaining N-1 points $\{x_j, y_j\}_1^N$ ($j = 1, ...N, j \neq i$) is defined as:*

$$\mathcal{I}(x_j|Q = x_i) = -\log(1 - \mathcal{F}(x_j|Q = x_i)) * \mathcal{F}(x_j|Q = x_i), \tag{3.1}$$

where $\mathcal{F}(x_j|Q = x_i)$ is the weight of point x_j (with respect to Q), and can be defined as:

$$\mathcal{F}(x_j|Q=x_i) = \frac{1}{Z_i}\left\{ \mathtt{W}(x_j|Q=x_i)^{\eta} \left(\prod_{n=1}^{N} \left(1 - \mathtt{W}(x_j|Q=x_n)\mathbb{I}_{[y_j \neq y_n]}\right) \right)^{1-\eta} \right\} \quad (3.2)$$

The first term $\mathtt{W}(x_j|Q = x_i)^{\eta}$ in equation 3.2 can be interpreted as the likelihood that point x_j is close to the Q, while the second part indicates the possibility that x_j is far apart from dissimilar points. The indicator $\mathbb{I}[\cdot]$ equals to 1 if the condition is met and 0 otherwise. Z_i is a normalization factor and η is introduced as a balancing factor that determines the emphasis of the first term. Intuitively, η is set to $\frac{N_{x_j}}{N}$, where N_{x_j} represents the number of points in the same class of point x_j.

The rationale of informativeness is that two points are likely to share the same class label when their distance is sufficiently small, assuming the points have a uniform distribution. This idea is the same as KNN classification. In addition to measuring the pairwise distances between the query point and its neighbors, our metric also considers that the informative points should have a large distance from dissimilar points. This guarantees that the locations of the informative points are of the most density in the same class.

Figure 3.2(left) provides a clarification, in which point 1 and point 2 (with the same class label) both have the same distance d from Q, but point 1 is closer to the real class boundary. Thus, point 1 is more likely to be closer to the points in other classes. As such we claim that point 1 is less informative than point 2 for Q by Definition 1. Since assuming the distribution over the concept location is uniform, it is more likely that points (e.g., 3 $\&$ 4) having the same label as points 1 $\&$ 2 will more likely distribute around point 2.

3.2.2 Informativeness Implementation

To define $\mathtt{W}(x_j|Q = x_i)$ in equation 3.2, we can model the weight of an individual point on Q as a function of the distance between them:

$$\mathtt{W}(x_j|Q=x_i) = f(\|x_i - x_j\|_p) \quad (3.3)$$

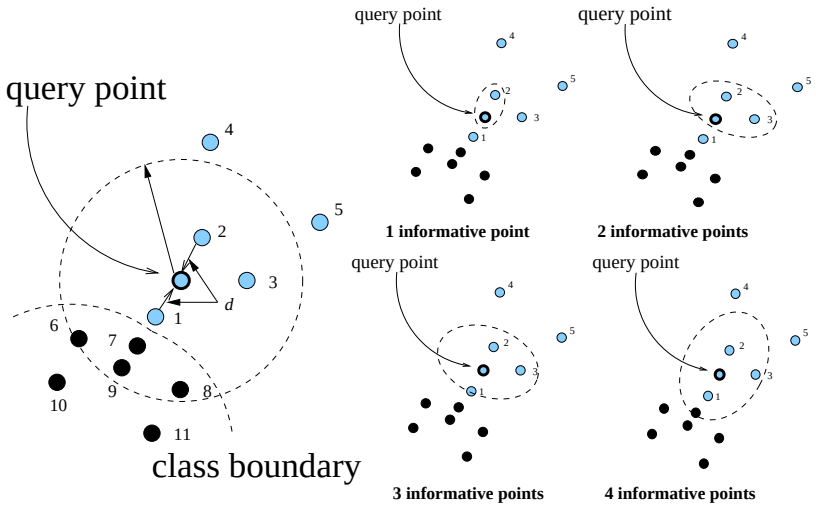


Figure 3.2. An illustration of 7-NN and the corresponding i informative points for the query point. **(Left)** 7-NN classification and the real class boundary (in dash lines). **(Right)** $i(i = \{1, 2, 3, 4\})$ informative points for the same query point.

where $\|x_i - x_j\|_p$ denotes the p-norm distance between x_i and x_j. To achieve higher possibility when two points are close to each other, we require $f(\cdot)$ to be a function inverse to the distance between two points. The generalized Euclidean distance metric satisfies this requirement. In this paper, we implement equation 3.3 as follows:

$$\mathtt{W}(x_j|Q = x_i) = \exp(-\frac{||x_i - x_j||^2}{\gamma}) \qquad \gamma > 0 \tag{3.4}$$

In practice, it is very likely that the features have different importance, making it desirable to find the best weighting of the features. Specifically, we define $||x_i - x_j||^2 = \sum_p w_p(x_{ip} - x_{jp})^2$, where w_p is a scaling factor that reflects the relative importance of feature p. One way to specify the scaling factor w_p is as follows:

$$w_p = \frac{1}{m}\sum_{k=1}^{m} w_{pk} = \frac{1}{m}\sum_{k=1}^{m} \mathtt{Var}_{\mathbf{x}_k}(\mathbf{x}_{pk}) \tag{3.5}$$

We obtain w_p by averaging over all classes' weights w_{pk}, each of which is calculated using the variance of all points in each class k at feature$p$, denoted by $\mathtt{Var}_{\mathbf{x}_k}(\mathbf{x}_{pk})$.

The normalization factor Z_i in equation (3.2) ensures the well-defined probabilities of neighbors for a given query point x_i. Specifically, $Z_i = \sum_{j=1}^{N} \mathtt{W}(x_j|Q = x_i)$. In this way the normalization is guaranteed, i.e., $\sum_{j=1}^{N} \mathcal{F}(x_j|Q = x_i) = 1$.

Based on the implementation, we have the following proposition regarding the informativeness metric:

Proposition 1. *Given a query x_0, $\forall\, x_i, x_j$ that satisfies $\|x_i - x_0\|^2 = kd$ and $\|x_j - x_0\|^2 = d$ with $d \in \mathbb{R}^+, k > 1$, x_j is guaranteed to be $\exp((k-1)d)^\eta$ times more informative than x_i, i.e., $\mathcal{I}(x_j|x_0) > \exp((k-1)d)^\eta \mathcal{I}(x_i|x_0)$.*

Proof. For simplicity, we only consider the case that x_i and x_j are in the same class, i.e., $y_i = y_j$. Without loss of generality, we let $\gamma = 1$ for equation 3.4. We have

$$\begin{aligned}
\frac{\mathcal{F}(x_j|Q=x_0)}{\mathcal{P}(x_i|Q=x_0)} &= \frac{\mathtt{W}(x_j|Q=x_0)^\eta H(x_j)^{1-\eta}}{\mathtt{W}(x_i|Q=x_0)^\eta H(x_i)^{1-\eta}} \\
&= \frac{\exp(-d)^\eta H(x_j)^{1-\eta}}{\exp(-kd)^\eta H(x_i)^{1-\eta}} \\
&= \exp((k-1)d)^\eta \frac{H(x_j)^{1-\eta}}{H(x_i)^{1-\eta}} \qquad (3.6)
\end{aligned}$$

where $H(\mathtt{x}) = \left(\prod_{n=1}^{N} \left(1 - \mathtt{Pr}(\mathtt{x}|Q=x_n)\mathbb{I}_{[\mathtt{y}\neq y_n]}\right)\right)$. Since $H(\cdot)$ is independent of the query point, its expected value (taken over $\mathtt{x}$ and each x_n) can be defined as

$$\begin{aligned}
E(H(\mathtt{x})) &= E\left(\prod_{n=1}^{N}\left(1 - \mathtt{W}(\mathtt{x}|Q=x_n)\mathbb{I}_{[\mathtt{y}\neq y_n]}\right)\right) \\
&= \prod_{n=1}^{N}\left(E\left(1 - \mathtt{W}(\mathtt{x}|Q=x_n)\mathbb{I}_{[\mathtt{y}\neq y_n]}\right)\right) \\
&= \prod_{n=1}^{N}\left(E(1 - \exp(-\|\mathtt{x}-x_n\|^2)\mathbb{I}_{[\mathtt{y}\neq y_n]})\right) \\
&= \prod_{n=1}^{N}\left((1 - E\exp(-\|\mathtt{x}-x_n\|^2)\mathbb{I}_{[\mathtt{y}\neq y_n]})\right)
\end{aligned}$$

Recall that x_i and x_j are in the same class, thus the set of dissimilar points (say $\{x'_n, y'_n\}_1^q$) should be the same. The above equation can then be simplified by removing the indicator variables:

$$E(H(\mathtt{x})) = \prod_{n=1}^{q}\left((1 - E\exp(-\|\mathtt{x}-x'_n\|^2))\right)$$

$$= \prod_{n=1}^{q} \left(1 - \int_{1}^{N} \exp(-\|\mathtt{x} - x'_n\|^2) d\mathtt{x}\right)$$

with $N \to \infty$, it is easy to verify that $E(H(x_i)) = E(H(x_j))$. Applying the results to equation (3.6), we have

$$\frac{\mathcal{F}(x_j|Q = x_0)}{\mathcal{F}(x_i|Q = x_0)} = \exp((k-1)d)^{\eta} > 1 \ \ (\mathtt{with}\ k > 1) \tag{3.7}$$

Applying equation (3.7) to equation (3.1), we finally have:

$$\frac{\mathcal{I}(x_j|Q = x_0)}{\mathcal{I}(x_i|Q = x_0)} = \frac{\log(1 - \mathcal{F}(x_j|Q = x_0))}{\log(1 - \mathcal{F}(x_i|Q = x_0))} \cdot \exp((k-1)d)^{\eta}$$

$$= \log_{(1-\mathcal{F}(x_i|Q=x_0))}(1 - \mathcal{F}(x_j|Q = x_0)) \cdot \exp((k-1)d)^{\eta}$$
$$> \exp((k-1)d)^{\eta} \quad \square$$

3.2.3 LI-KNN Classification

So far we have proposed to compute the informativeness of points in the entire data distribution for a specific query Q. However, considering the large number of data points with high dimensionality in practice, the computational cost could be prohibitively high. We propose to make use of the new distance metric defined in equation 3.1 by restricting the computation between the nearest neighbors in an augmented *query-based* KNN classifier.

Algorithm 5 gives the pseudo-code of LI-KNN classification. Instead of finding the informative points for each x_i by going over the entire data set, LI-KNN retrieves I *locally* informative points by first getting the K nearest neighbors (we consider the Euclidean distance here). It then applies equation (3.1 to the K local points and the majority label between the first I points are assigned to x_i. Specifically, when $I = 1$, LI-KNN finds only the most informative point for x_i, i.e., $y_i = \arg\max_{y_k, k \in \{1,\ldots,K\}} \mathcal{I}(x_k|Q = x_i)$. In this way the computational cost of finding the most informative points is reduced to a local computation. Note that when K equals to N, the locally informative points are exactly the optimal informative points for the entire data distribution as in Definition 1. Likewise, when I equals to K, LI-KNN performs exactly the same as the KNN method.

Algorithm 5 LI-KNN Classification

Input: (S, K, I)
target matrix: $S = \{x_i, y_i\}_1^N$
number of neighbors: $K \in \{1, ..., N-1\}$
number of informative points: $I \in \{1, ..., K\}$
Initialize $err \leftarrow 0$
for each query point x_i ($i = 1$ to N) **do**
  find K nearest neighbors $\mathcal{X}_K$ using Euclidean distance
  find I most informative points among K neighbors (equation (1))
  majority vote between the I points to determine the class label of x_i
  if x_i is misclassified
    $err \leftarrow err + 1/N$
  end if
end for
Output: err

Although our LI-KNN method introduces one more parameter I for the KNN method, it is not hard to figure out that LI-KNN is relatively insensitive to both parameters K and I. (1) Regardless of the choice of K, the points that are closest (in Euclidean distance) to Q are always selected as neighbors, which by equation 3.2 have a high possibility to be informative. (2) On the other hand, given a fixed number of K, the informativeness of the local points are fixed which insures that the most informative ones are always chosen. For example, in Figure 3.2(left), point 2 $\&$ 3 are selected as the neighbors for Q with K increasing from 3 to 7. Meanwhile, when K equals to 7 and I ranges from 1 to 3, the informative sets (Figure 3.2(right)) are $\{2\}$,$\{2, 3\}$ and $\{2, 3, 1\}$ respectively, which include the most informative points in all cases that ensures Q is classified correctly.

3.3 GI-KNN Classification

The LI-KNN algorithm classifies each individual query point by learning informative points separately, however, the informativeness of those neighbors are then discarded without being utilized for other query points. Indeed, in most scenarios, different queries may yield different informative points. However, it is possible that some points are more informative than others, i.e., they could be informative neighbors for several different points. Thus, it would seem reasonable to put more emphasis on *globally* informative points. Since it has been shown that KNN classification [140] can be improved by

learning a distance metric from the training examples, we enhance the power of the informativeness metric and propose a boosting-like iterative method, namely *globally informative* KNN (GI-KNN).

3.3.1 Algorithm and Analysis

The goal of GI-KNN is to obtain an optimal weight vector A from all training points for classification of testing points. The algorithm iterates M predefined steps to get the weight vector, which was initially set to be uniform. In each iteration, an individual point is classified in the same way as LI-KNN by finding I informative neighbors, with the only exception that in GI-KNN the distance metric is a *weighted* Euclidean distance whose weight is determined by A (line 5 & 6 in Algorithm 6, where $D(x_i, \mathbf{x})$ denotes the Euclidean distance between x_i and all the remaining training points, and $D_A(x_i, \mathbf{x})$ is the weighted distance). $\mathcal{N}_m^i(r)$ denotes the r's informative points for x_i according to the informativeness metric. We use $\epsilon_m^i \in (0, 1)$ to denote the *weighted* expected weight loss of x_i's informative neighbors during step m. The cost function C_m^i is a smooth function of ϵ_m^i, which guarantees to be in the range of (0,1) and positively related with ϵ_m^i. Here we use a tangent function as the cost function, depicted in Figure 3.3[1]. The weight vector A is updated in the manner that if x_i is classified incorrectly, the weights of its informative neighbors which have different labels from x_i are decreased exponentially to the value of C_m^i (line 9, $e(x_i, x_\ell) = C_m^i$ `if` $y_i \neq y_\ell$; line 13, $A(x_\ell) \leftarrow A(x_\ell) \cdot \exp(-e(x_i, x_\ell))$). Meanwhile, the weights remain the same for neighbors in the same class with x_i even if x_i is misclassified (line 9, $e(x_i, x_\ell) = 0$ `if` $y_i = y_\ell$). Clearly, the greater the weight the query point is, the higher the penalty of misclassification will be for the selected neighbors. A is then normalized before the next iteration.

While GI-KNN has several parallels to Boosting such as the structure of the algorithm, GI-KNN differs from Boosting in the way weights are updated. Specifically, Boosting assigns high weights to points that are misclassified in the current step, so that the weak learner can attempt to reduce the error in future iterations. For GI-KNN, the objective is to find globally informative points, thus higher weights are given to the points that seldom make wrong predictions. Notice that the weight of the query remains unchanged.

[1] In practice, we did not find much difference in performance for different τ. Therefore, we choose $\tau = 1$ for our implementation.

Algorithm 6 GI-KNN Training

Input: (T, K, I, M)
  training set: $T = \{\mathbf{x}, \mathbf{y}\} \in \mathbb{R}^{N \times P}$
  number of neighbors: $K \in \{1, ..., N-1\}$
  number of informative points: $I \in \{1, ..., K\}$
  number of iterations: $M \in \mathbb{R}$
Initialization: $A = \{1, ..., 1\} \in \mathbb{R}^{N \times 1}$ [*the weight vector*]
for $m = 1$ to M **do**
  for each query point x_i ($i = 1$ to N) **do**
    $D_A(x_i, \mathbf{x}) = \frac{D(x_i, \mathbf{x})}{A}$ [*calculate the weighted distance*]
    $\mathcal{N}_m^i \leftarrow I$ most informative points according to $D_A(x_i, \mathbf{x})$
    $\epsilon_m^i = A(x_i) \cdot E_A[\mathcal{N}_m^i] = A(x_i) \cdot \frac{1}{I} \sum_{r=1}^{I} A(\mathcal{N}_m^i(r))$
    $C_m^i = \frac{1}{2}(1 + tanh(\tau * (\epsilon_m^i - \frac{1}{2})))$

$$e(x_i, x_\ell) = \begin{cases} C_m^i & \text{if } y_i \neq y_\ell; \\ 0 & \text{if } y_i = y_\ell. \end{cases}$$

    if point x_i is classified incorrectly [*update the neighbors weights*]
      $err_m \leftarrow err_m + \frac{1}{N}$
      for each x_ℓ ($\ell \in \mathcal{N}_m(x_i)$) **do**
        $A(x_\ell) \leftarrow A(x_\ell) \cdot \exp(-e(x_i, x_\ell))$
      end for
    renormalizes A so that $\sum_{i=1}^{N} A(i) = N$
  end for
  $\xi_m \leftarrow err_m - err_{m-1}$
end for
Output: the weight vector A

3.3.2 Learning the Weight Vector

At completion, the learned vector A can be used along with a distance metric (e.g. L_2 distance metric) for KNN classification at each testing point $\mathbf{t_0}$. Specifically, given the training set $T = \{x_i, y_i\}_1^N$, the distance between $\mathbf{t_0}$ and each training point x_i is defined as

$$D(\mathbf{t_0}, x_i) = \|\mathbf{t_0} - x_i\|_{A_i} = \frac{\sqrt{(\mathbf{t_0} - x_i)^T(\mathbf{t_0} - x_i)}}{A_i} \tag{3.8}$$

By adding weights to data points, GI-KNN in essence is similar to learning a Mahalanobis distance metric $D(x_i, x_j)$ for k-nearest neighbor classification. i.e., $D(x_i, x_j) = D_{\mathcal{A}}(x_i, x_j) = \|x_i - x_j\|_{\mathcal{A}} = \sqrt{(x_i - x_j)^T \mathcal{A}(x_i - x_j)}$, where $\mathcal{A}$ is a covariance matrix that determines the similarity between features. In our case, A measures the importance of each training point rather than their features.

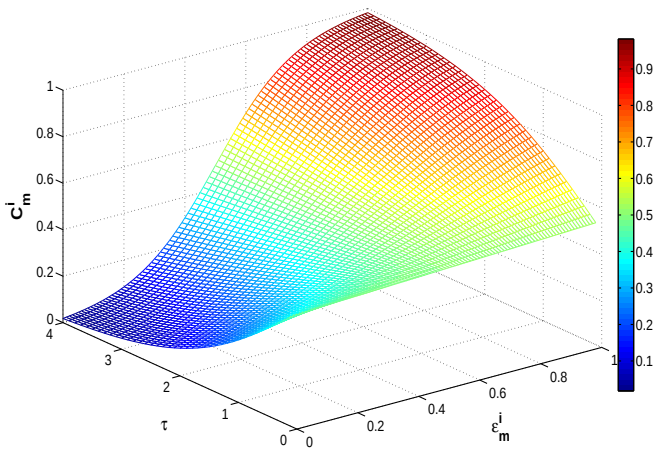


Figure 3.3. Cost function (C) used for GI-KNN.

3.4 Experiments

In this section, we present experimental results with both benchmark data and image data to demonstrate the merits of LI-KNN and GI-KNN. We start by testing on 10 standard UCI data sets to assess the performance of the two algorithms. Then our proposed methods are applied to image categorization by using three extensively bench-marked data sets, namely ORL[2], COIL-20[3] and MINST[4].

For performance evaluation, several classifiers are used for comparison. The classic KNN [24] classifier is used as the baseline algorithm. We implemented DANN [53] as an extension of KNN[5]. To be more convincing, we also compare with one of the newest KNN extensions – Large Margin Nearest Neighbor Classification (LMNN)[6]. Two discriminant classifiers are also compared: a Support Vector Machine (SVM) and a Boosting classifier. We use the AdaBoost.MH [106] and the Multi-class SVM [25] software (K.Crammer et al.[7]) for multi-class classification.

[2]http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html
[3]http://www1.cs.columbia.edu/CAVE/software/softlib/coil-20.php
[4]http://yann.lecun.com/exdb/mnist/
[5]During the experiment, we set the parameters $K_M = max(N/5, 50)$ and $\epsilon = 1$ according to the original paper.
[6]The code is available at http://www.seas.upenn.edu/∼kilianw/lmnn/
[7]See http://www.cis.upenn.edu/∼crammer/code-index.html

Data	N	D	C	KNN	DANN	LMNN	**LI-KNN**	**GI-KNN**	SVM	Boosting
Iris	150	4	3	0.044 (9)	0.040 (5)	0.053 (3)	0.013 (9, 5)	**0.010** (25)	0.042	0.038 (45)
Wine	178	13	3	0.263 (3)	0.250 (7)	**0.031** (5)	0.137 (9, 1)	0.137 (13)	0.205	0.192 (135)
Glass	214	10	2	0.372 (5)	0.436 (5)	0.356 (3)	**0.178** (7, 3)	0.198 (202)	0.222	**0.178** (304)
Iono	351	34	2	0.153 (5)	0.175 (7)	0.100 (5)	0.127 (5, 3)	0.127 (8)	**0.090**	0.092 (156)
Breast	699	9	2	0.185 (7)	0.120 (11)	0.927 (5)	0.080 (4, 1)	**0.045** (48)	0.052	0.048 (657)
Heart	779	14	5	0.102 (3)	0.117 (5)	0.092 (5)	**0.078** (7, 1)	**0.078** (192)	**0.078**	0.080 (314)
Digit	2000	649	10	0.013 (3)	0.010 (3)	0.009 (3)	**0.005** (19, 1)	**0.005** (137)	0.010	**0.005** (175)
Isolet	7797	617	26	0.078 (11)	0.082 (11)	0.053 (5)	0.048 (13, 3)	**0.042** (175)	0.044	**0.042** (499)
Pen	10992	16	10	0.027 (3)	0.021 (5)	**0.020** (3)	**0.020** (9, 1)	**0.020** (42)	0.033	0.038 (482)
Letter	20000	16	10	0.050 (5)	0.045 (3)	0.042 (5)	0.045 (5, 3)	0.040 (22)	**0.028**	0.031 (562)

Table 3.1. Testing error rates for KNN, DANN, LMNN, SVM, Boosting, LI-KNN and GI-KNN of 10 UCI Benchmark data sets. N, D and C denote the number of instances, dimensionality and number of classes respectively. Numbers in the parentheses indicate the optimal neighbors K for KNN, DANN and LMNN, (K, I) for LI-KNN, and number of iterations M for GI-KNN and Boosting.

3.4.1 UCI Benchmark Corpus

We evaluate our algorithms by using 10 representative data sets from UCI Machine Learning Repository[8]. The size of the data sets ranges from 150 to 20,000 with dimensionality between 4 and 649, including both two classes and multi-class data ($C = 3, 26$ etc). For evaluation, the data sets are split into training sets and testing sets with a fixed proportion of 4:1. Table 3.1 reports the best testing error rates for these methods, averaged over ten runs. Our methods on these data sets exhibit competitive results in most scenarios.

Figure 3.4 shows the stability of LI-KNN on the testing errors rates of the Iris data set. KNN always incurs higher error rates than our algorithms. The performance of LI-KNN is depicted for four different values of I. It is obvious that even with different values of I (given the same K), the results are similar, indicating that the performance of LI-KNN does not degrade when the number of informative points changes. In addition, with the change of K, LI-KNN is relatively stable regarding the error rate. The variation of LI-KNN is roughly 3%, meaning that K does not have a large impact on the results of LI-KNN.

Figure 3.5 compares Boosting and GI-KNN on the Breast Cancer data for the first 1,000 iterations. Overall, GI-KNN incurs lower error rates. From 620 to about 780

[8]http://www.ics.uci.edu/∼mlearn/MLRepository.html

iterations GI-KNN's error rates are slightly higher than Boosting. However, the error rates of Boosting fluctuate quite a bit from 0.048 to 0.153, while GI-KNN is relatively stable as iterations increase and the performance varies only between (0.043, 0.058). Moreover, our algorithm obtains the optimal results significantly earlier than Boosting.

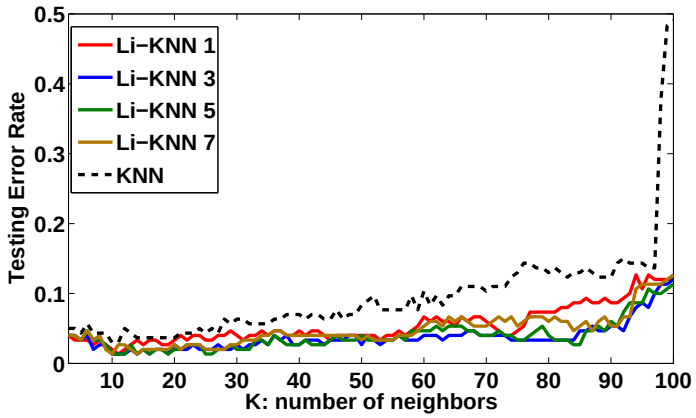


Figure 3.4. Results on Iris for K from 1 to 100. LI-KNN chooses the number of informative points (I) to be 1, 3, 5 and 7.

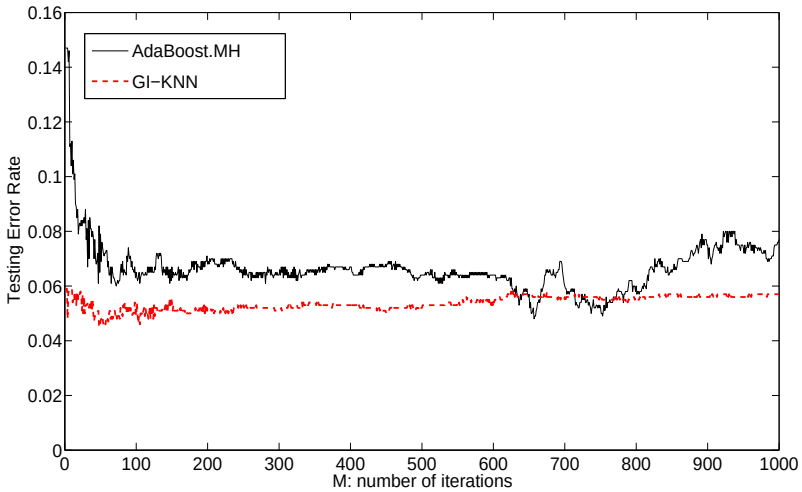


Figure 3.5. Results on Breast Cancer for AdaBoost.MH and GI-KNN (with $K = 5$ and $I = 1$). The best result for GI-KNN is slightly better (0.045) than that of AdaBoost.MH (0.048).

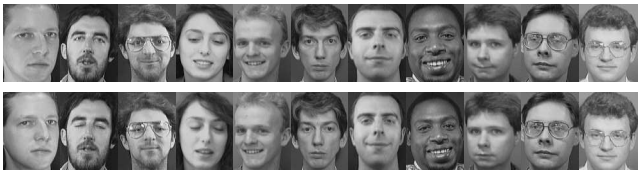


Figure 3.6. Results on ORL data set for randomly generated samples. **(Top)** Random samples. **(Bottom)** LI-KNN results.

3.4.2 Face Recognition on ORL

The ORL Database of Faces consists of ten different images, each of which has 40 distinct faces. The dimension was reduced by subsampling examples to 23×28 pixels. Figure 3.6 shows the one nearest neighbor of LI-KNN for several randomly generated samples. Clearly, each of them are from the same class as the random samples.

Figure 3.7 depicts the box plots of the *macro-F* error rates. The optimal parameters are estimated by 4-fold cross-validation on the training set. It is evident that the spread of the error distribution of our algorithms is very close to zero, which clearly indicates that LI-KNN and GI-KNN obtain robust performance over different classes.

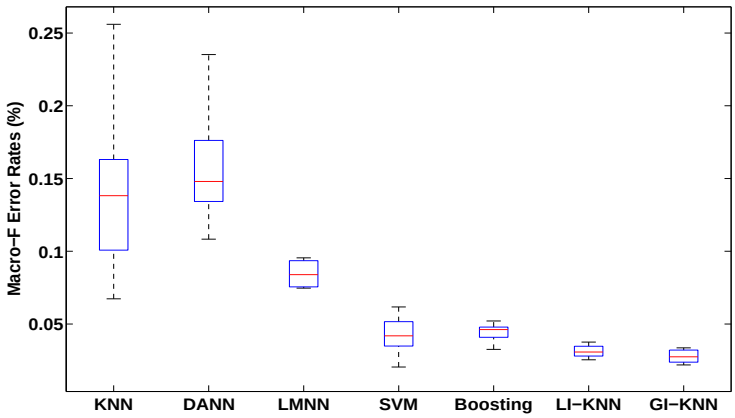


Figure 3.7. Box plots of macro-F error rates on the ORL data set.

3.4.3 Object Recognition on COIL-20

We use the processed version of COIL-20 database for object recognition. The database is made up with 20 gray-scale objects, each of which consists 72 images with size 128

Figure 3.8. Randomly generated images from each object in the COIL-20 database.

$\times$ 128. Figure 3.8 shows 20 sample images for each object.

We treat each object as one class, spliting the data into training and testing set with the proportion of 3:1. Figure 3.9 shows the classification errors regarding the 5 algorithms, where K ranges from 1 to 11. GI-KNN and LI-KNN generally outperform others with the best parameters, while both show stable results with the change of K.

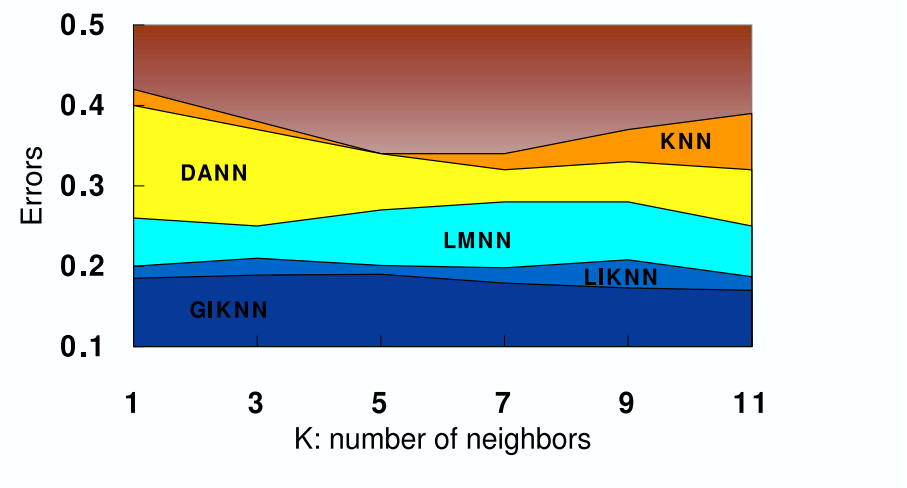


Figure 3.9. Results on COIL-20 with different number of neighbors.

3.4.4 MNIST Handwritten Digits

MNIST handwritten digits database has been used extensively to test various pattern recognition methods. We use the preprocessed data[9] that contains 8-bit grayscale images of 0 through 9; there are roughly 6,000 training examples of each class and 1,000 test examples. We further reduce the dimension by subsampling examples to 16 $\times$ 16 pixels. KNN (with L_2 distance metric) and DANN incur 4.2% and 4.15% testing error

[9]Available at http://www.cs.toronto.edu/$\sim$roweis/data.html

Original Data	
KNN results	
LI-KNN results	

Figure 3.10. Results on MNIST data set. Examples (top) are misclassified by 1-NN with Euclidean distance (middle), while classified correctly by LI-KNN with $I = 1$ (bottom).

rates respectively, while LI-KNN with one informative neighbor yields 2.1% error rate, improving the performance by roughly 50%. LMNN in this case also has good performance of incurring only 2.5% error rate[10]. Figure 3.10 shows some examples that are misclassified by KNN, while LI-KNN successfully classifies these points by finding the most informative point ($I = 1$).

3.4.5 Discussion of Experimental Results

Although we did not prove optimal choices for either K or I, our empirical studies with different values on several data sets permits rules of thumb. Basically, the value of K should be reasonably large. The larger K is, the more information can be gathered to estimate the distribution of neighborhood for the query point. However, as K increases, the computational complexity of the informativeness of neighbors (equation (3.2)) grows exponentially. In practice, the choice of a range of $K \in (7, 19)$ gives good trade-off regardless of data size. In contrast, a smaller I is preferable for the best predictions. Experimental results indicate that $I = 1, 3$ usually yield the best results.

What is the appropriate choice of the cost function C_m^i for GI-KNN training (line 8 in Algorithm 6)? Since GI-KNN has a different objective (to find the best weight vector) than boosting and other machine learning algorithms (to minimize a smooth convex surrogate of the 0-1 loss function), we did not compare the performance between different loss functions like exponential loss, hinge loss, etc. We speculate that performance will not be significantly improved for different loss functions.

[10]The authors reported an error rate of 1.3% in their paper [140] by using a different pre-processing method.

There are questions regarding the GI-KNN algorithm that are still open for discussion. Can the convergence of GI-KNN be proved, or is there an upper-bound given specific K and I? In practice, is it possible to specify an early stopping criteria? Since this is a boosting-like algorithm, can we replace the 0-1 loss function with a smooth convex cost function to improve the performance? Furthermore, it would be interesting to see whether the definition of the informativeness metric can be applied to semi-supervised learning or noisy data sets. And of course, other data sets remain to be explored.

3.5 Related Work

Nearest Neighbor Method and its Extentions

The idea of nearest neighbor pattern classification was first introduced by T. Cover and P. Hart in [24], in which the decision rule is to assign an unclassified sample point according to the labels of K nearest points. The authors proved that when the amount of data approaches infinity, the one nearest neighbor classification is bounded by twice the asymptotic error rate as the Bayes rule, independent of the distance metric applied.

T. Hastie and R. Tibshirani [53] developed an adaptive method of nearest neighbor classification (DANN) by using locally discriminative information to estimate a subspace for global dimension reduction. They estimated the values of between (B) and within (W) the sum-of-squares matrices, and used them as a local metric such as $\sum = W^{-1}BW^{-1}$. They showed that their work can be generalized by applying specialized distance measures $\sum$ for different problems.

K Weinberger et al. [140] learned a Mahanalobis distance metric for KNN classification by using semidefinite programming, a method referred to as large margin nearest neighbor (LMNN) classification. Their method seeks a large margin that separates examples from different classes, while keeping a close distance between nearest neighbors that have the same class labels. LMNN is novel in the sense that the method does not try to minimize the distance between all examples that share the same labels, but only to those that are specified as *target neighbors*. Experimental results exhibit great improvement over KNN.

By learning locally relevant features from nearest neighbors, J. Friedman [41] introduced a flexible metric that performs recursive partitioning to learn the local relevance, which is defined as $I_i^2(z) = (E[f] - E[f|x_i = z])^2$, where $E[f]$ denotes the

expected value over the joint probability density $p(x)$ of an arbitrary function $f(x)$. The most informative feature is recognized as the one having the largest deviation from $P(x|x_i = z)$.

E. Han et al. [50] proposed an application of KNN classification to text categorization by using the adjusted weight of neighbors (WAKNN). WAKNN tries to learn the best weight for vectors by measuring the cosine similarity between documents. Specifically, $cos(X, Y, W) = \frac{\sum_{t \in T}(X_t \times W_t) \times (Y_t \times W_t)}{\sqrt{\sum_{t \in T}(X_t \times W_t)^2} \times \sqrt{\sum_{t \in T}(Y_t \times W_t)^2}}$, where X and Y are two documents, W the weight vector and T the set of features (terms). Optimizations are also performed to speed up WAKNN. The experiments on benchmark data sets indicate that WAKNN consistently outperforms KNN, C4.5 and several other classifiers.

Chapter 4

A Text Retrieval Application: People Name Disambiguation

With the emergence of major search engines such as Google and Yahoo! that automate the process of gathering web pages to facilitate searching, it has become increasingly common for Internet users to search for their desired results to specific queries through search engines, with name queries making up approximately 5-10% of all searchers. Name queries are usually treated by search engines as normal keyword searches without attention to the ambiguity of particular names. For example, searching Google for "Yang Song" results in more than 11,000,000 pages with the same person's name, of which even the first page shows five different people's home pages. Table 4.1 lists the first four results which correspond to four different people. Due to this heterogeneous nature of data on the Internet crawled by search engines, the issue of identity uncertainty or *name ambiguity* has attracted significant research attention. Beyond the problem of sharing the same name among different people, name misspelling, name abbreviations and other reference variations compound the challenge of name disambiguation.

The same issue also exists in most Digital Libraries (DL), hampering the performance and quality of information retrieval and credit attribution. In DL such as DBLP[1] and CiteSeer [43], textual information is stored in metadata records to speed up field searching, including titles, venues, author names and other data. However, the existence of both *synonyms* and *polysems* as well as typographical errors makes the problem of disambiguating author names in bibliographies (citations) non-trivial. In the case of

[1] http://www.informatik.uni-trier.de/∼ley/db/index.html

Yang Song Homepage of **Yang Song**, PhD candidate of Penn State Department of Computer Sciences and Engineering. http://www.cse.psu.edu/∼yasong/
Yang Song Home page of **Yang Song**, CALTECH, Department of Electrical Engineering... http://www.vision.caltech.edu/yangs/
Yang Song's Homepage **SONG, Yang**, Department of Statistics, UW-Madison Medical Science Center... http://www.cs.wisc.edu/∼yangsong/
Song Yang the Cartoonist **Song Yang** is certainly the most successful cartoonist on the Mainland... http://japanese.china.org.cn/english/NM-e/155786.htm

Table 4.1. First 4 search results of the query "Yang Song" from Google that refer to 4 different people.

synonyms, an author may have multiple name variations/abbreviations in citations across publications, e.g., the author name "C. Lee Giles" is sometimes written as "C. L. Giles" in citations. For *polysems*, different authors may share the same name label in multiple citations, e.g., both "Guangyu Chen" and "Guilin Chen" are used as "G. Chen" in their citations. In addition to the issue of citations, authors may be inclined to use different name variations even in the title pages of their publications due to a variety of reasons (such as the change of their maiden names).

Existing approaches that address the issue of name disambiguation generally fall into two categories: supervised learning and unsupervised learning methods. In the case of supervised learning [51], the objective is to determine the *name label* by leveraging the related information (e.g., page contents and citation information). Careful labeling with specific domain knowledge is usually required for supervised learning, which makes it both error-prone and label intensive. Comparatively, unsupervised learning methods [52, 8] do not require manual labeling but instead prudently choose features (e.g., social networks, link structures, co-authorship) to classify similar instances into groups or clusters. A variety of clustering methods including K-means and spectral clustering have been extensively utilized and compared for unsupervised name disambiguation.

Nevertheless, choosing the right set of features often results in better performance than exhaustively seeking the best clustering method. However, supervised learning methods generally achieve better performance with the trade-off of expensive training time.

4.1 Our Contribution

The objective of this paper is to propose an approach of name disambiguation that includes the attractive properties of both supervised and unsupervised learning methods while trying to avoid the respective limitations. Specifically, we explore the use of a two-stage approach to address the problem of disambiguating person names in both web appearances and scientific documents (including citations). During the first stage, we present two novel *topic-based* models inspired by two generative models for documents: Probabilistic Latent Semantic Analysis (PLSA) and Latent Dirichlet Allocation (LDA). Our models differ from the general methods by explicitly introducing a variable for *persons*. After an initial model is built, person names are disambiguated by leveraging an unsupervised hierarchical agglomerative clustering method [18], which groups similar instances together in a bottom-up fashion. We empirically study our models by comparing against three other clustering methods on both web data and scientific documents.

The underlying rationale for using generative models with latent variables is to harness the unique topic distribution related to different persons. For example, the basketball player "Michael Jordan" is more likely to appear in the topic *sports*, while Professor "Michael Jordan" in Berkeley may have high probability of being associated with the topic *academics*. Likewise, for the authors of scientific papers, one may have his/her own focus, e.g. Professor "Jia Li" in the math department of Alabama and Professor "Jia Li" in the statistics department of Penn State. Moreover, even authors within the same research field should be distinguishable by topics, e.g. two researchers named "Amit Kumar" working separately at Cornell and Rice are both involved in research on networks, but with specific focus on network routing and wireless networks respectively. As a result, topic distribution may be a useful *feature set* that allows us to distinguish people from each other in a principled and efficient way.

Although both PLSA and LDA have been extensively studied and applied to various applications, there has been relatively few comparisons between their performance

in real-world studies except in [12]. Theoretically, PLSA does not need to make any assumptions regarding the document distribution, thus it is more flexible when dealing with abnormal data sets. Meanwhile, the LDA (Bayesian) approach is more robust on sparse data. With a large feature space, LDA generally exhibits better performance than PLSA as well as other probabilistic models.

4.2 Related Work

Generative Models for Documents

Using generative models for characterizing documents as well as images has become a recent trend in machine learning research. The first well-known model was introduced by Deerwester [31], namely *Latent Semantic Analysis* (LSA). The key idea of LSA is to map high-dimensional input data to a lower dimensional representation in a *latent semantic space* that reflects semantic relations between words, the mapping was done by Singular Value Decomposition (SVD), and thus restricted to be linear. LSA assumes that there are K underlying latent topics, to which documents are generated accordingly. Those latent topics are assumed to be approximately the same as document classes, resulting in a significant compression of data in large collections.

From a statistical point of view, Hofmann [54] presented an alternative to LSA, or Probabilistic Latent Semantic Analysis/Indexing (PLSA/PLSI), which discovers sets of latent variables with a more solid statistical foundation. The model is described as an *aspect model* that is essentially a latent class statistical mixture model, assuming the existence of hidden factors underlying the co-occurrences among two sets of objects. Thus, a single word is generated from a single topic while different words may belong to different topics within a document. *Expectation-Maximization* (EM) algorithm is applied for the inference of parameters in this model that maximize the likelihood of the data. An obvious problem of PLSA is that the model has a number of parameters that grow linearly with the size of the document collection, yielding a large potential for overfitting. Due to its efficiency and flexibility, PLSA has been widely used in many research fields, including collaborative filtering [55], image categorization [112], and web information retrieval [143, 59].

Blei et al. later introduced a Bayesian hierarchical model, *Latent Dirichlet Allocation* (LDA) [12], in which each document has its own topic distribution, drawn from a

conjugate Dirichlet prior that remains the same for all documents in a collection. The words within that document are then generated by choosing a topic from this distribution. A word is picked from that topic according to the posterior probability of the topic, which is determined by another Dirichlet prior. Inference of parameters and model learning are performed efficiently via variational EM algorithm, since exact inference is intractable in LDA due to the coupling of parameters. Essentially, this model can be statistically treated as a fully generative aspect model, which assumes an exchangeability for words and topics in documents. Experimental results indicate that LDA has better generalization performance than PLSA and a mixture of unigrams model as well as higher classification accuracies and better predictions of user preferences in the task of collaborative filtering. Successful applications and extensions of the LDA model includes unsupervised nature scene classification [123, 79], document retrieval [139] and time series analysis [136].

Name Disambiguation

Prior name disambiguation research can be categorized into supervised classification and unsupervised clustering. In [51], different classification methods such as hybrid Naive Bayes and Support Vector Machines (SVM) have been applied to a DBLP dataset. In large-scale digital libraries, however, supervised classification is inappropriate due to the unaffordable cost of human annotation for each name.

Different clustering methods have also been applied in the literature. Earlier approaches such as hierarchical clustering [85] suffered from the transitivity problem[2]. Han et al [52] used a more sophisticated K-spectral clustering method to cluster author appearances. While Han's method could find an approximation of the global optimal solution (in terms of a criteria function) for a sampled dataset, it is unsuitable for large-scale digital libraries since K is not known a priori for an ever increasing digital library and the computational complexity $O(N^2)$ is intractable for N=739,135 in CiteSeer. Lee et al. [77] successfully addressed the scalability issue by using a two level blocking framework; however, this resulted in inconsistent labeling due to the transitivity problem in such a solution. In [57], used a SVM-based distance function was used to calculate the similarity of the metadata records of author appearances, and explicitly solved the transitivity problem in labeling with the DBSCAN clustering method. [9] proposed an

[2]The transitivity problem refers to a name A that is co-referent with B, and B with C, while A is not co-referent with C. C.f. [57] for more detailed discussions.

LDA-based entity resolution method which is generative and does not require pair-wise decisions.

The aforementioned work mainly tackled the name disambiguation problem using the metadata records of the authors. This paper solves the name disambiguation problem in a novel way, by accounting for the topic distribution of the authors and adopting unsupervised methods. As such it yields an accurate and highly efficient solution to the person name disambiguation problem.

4.3 Topic-based PLSA

We use the following notations in this paper.

- A *document* d is a sequence of N words denoted by $\mathbf{w} = \{w_1, w_2, ..., w_N\}$, where w_n denotes the nth word in a document, plus a sequence of M name appearances denoted by $\mathbf{a} = \{a_1, a_2, ..., a_M\}$, where a_j represents the jth name appearances in the document. For web data, name appearances refer to the owners of their homepages or the subject of the articles. For scientific documents, it means the authors of the papers as well as the authors in the citations.

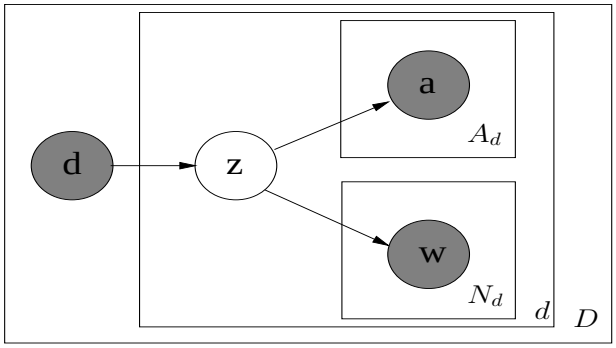
- A *corpus* is a collection of T documents denoted by $D = \{d_1, d_2, ..., d_T\}$.

- $W = \{w_1, ..., w_p\}$ represents the number of unique words (i.e., vocabulary) in a corpus with size p. $A = \{a_1, ..., a_q\}$ indicates the number of name appearances in a corpus with size q.

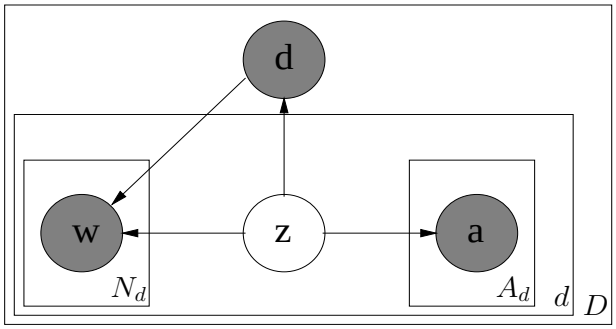
- The relationships between documents, names and words are connected by a set of latent variables $Z = \{z_1, ..., z_K\}$ with size K, each of which represents a latent topic.

In our document-name-word scenario, an observation is treated as a triplet $\{d, a, w\}$ that represents an instance that a name a appears in document d, which contains the word w. The relationship inherent in the triplets is associated by a set of topics Z. Our mixture model has a conditional independence assumption of variables, i.e., the observed objects are conditionally independent on the state of the related latent variables, which are essentially treated as persons' interests. Specifically, a document d is potentially

related to several topics Z with different probabilities, and the latent variables consequently generate a set of words $\mathbf{w}$ and name appearances $\mathbf{a}$ that are closely related to a specific topic. Figure 4.1 (a) shows the graphical illustration of the generative model.



(a) The three-way aspect model



(b) An alternative view of the aspect model

Figure 4.1. Graphical model representation. (a) The original document-name-word model, D is the number of documents, N_d is the number of words in document d and A_d is the number of name appearances in document d. (b) The alternative view of the model. Shaded nodes are observed variables.

4.3.1 The Aspect Model

The joint probability of the aspect model over $d \times a \times w$ is defined as the mixture:

$$P(d, a, w) = P(d)P(a, w|d) \tag{4.1}$$

$$P(a, w|d) = \sum_{z \in Z} P(a, w|z)P(z|d) \tag{4.2}$$

The definition of the generative model can be described in the following procedure:

1. pick a document d from the corpus D with probability $P(d)$,
2. select a latent class z_k with probability $P(z_k|d)$,
3. generate a word w with probability $P(w|z_k)$,
4. generate a name a with probability $P(a|z_k)$.

In this model, we introduce a set of latent variables z that breaks the direct relationships between documents, words and names, i.e., they are conditionally independent but still associated through latent variables. Note that by reversing the arrow from documents and words to latent topics, an equivalent symmetric model as shown in Figure 4.1 (b) can be parameterized by

$$P(d, a, w) = \sum_{z \in Z} P(z)P(d|z)P(w|z)P(a|z). \tag{4.3}$$

This paper will focus on Figure 4.1 (a) for inference unless otherwise mentioned.

4.3.2 Model Fitting with the EM Algorithm

The goal of model fitting for PLSA is to estimate the parameters $P(z), P(a|z), P(z|d)$ and $P(w|z)$, given a set of observations (d, a, w). The standard way to estimate the probability values is the Expectation-Maximization (EM) algorithm [21], which alternates two steps: (1) an expectation (E) step where posterior probabilities are estimated for the latent variables, based on the current estimates of the parameters; and (2) a maximization (M) step where parameters are estimated again to maximize the expectation of the complete data (log) likelihood. In the E-step, we compute

$$P(z|d, a, w) \propto \frac{P(z)P(a|z)P(d|z)P(w|z)}{\sum_{z'} P(z')P(a|z')P(d|z')P(w|z')}. \tag{4.4}$$

In the M-step, we aim at maximizing the expectation of the complete data likelihood, the formulas are:

$$P(a|z) \propto \frac{\sum_{d,w} n(d, a, w)P(z|d, a, w)}{\sum_{d,a',w} n(d, a', w)P(z|d, a', w)} \tag{4.5}$$

$$P(w|z) \propto \frac{\sum_{a,d} n(d,a,w)P(z|d,a,w)}{\sum_{d,a,w'} n(d,a,w')P(z|d,a,w')} \tag{4.6}$$

$$P(z|d) \propto \frac{\sum_{a,w} n(d,a,w)P(z|d,a,w)}{\sum_{d',a,w} n(d',a,w)P(z|d',a,w)} \tag{4.7}$$

where $n(d,a,w)$ denotes the number of occurrences of word w in document d with name a. The EM algorithm stops on convergence, i.e., when the improvement of the log-likelihood is significantly small:

$$\mathcal{L} = \sum_{a=1}^{A} \sum_{d=1}^{D} \sum_{w=1}^{W} n(d,a,w) \log P(d,a,w) \tag{4.8}$$

4.3.3 Predicting New Name Appearances

Despite the effectiveness of PLSA for mapping the same document to several different topics, it is still not a fully generative model at the level of documents, i.e., the number of parameters that need to be estimated grows proportionally with the size of the training set. Additionally, there is no natural way to assign probability to new documents. Therefore, to predict the topics of new documents (with potentially new names) after training, the estimated $P(w|z)$ parameters are used to estimate $P(a|z)$ for new names a in test document d through a "folding-in" process [54]. Specifically, the E-step is the same as equation (4.4); however, the M-step maintains the original $P(w|z)$ and only updates $P(a|z)$ as well as $P(z|d)$.

4.3.4 Probabilistic Inference

The PLSA model mentioned in the above section not only can derive relationships between documents, words and names, but by using probabilistic inference, it can also be used to model the topic patterns for names. Specifically, given $P(a|z)$ the probability of observing a name appearance given a certain topic, we can model the probability that a certain topic is of interest to a given name by simply applying the Bayes rule:

$$P(z|a) \propto \frac{P(a|z)P(z)}{\sum_z P(a|z)P(z)}. \tag{4.9}$$

In this way people that share similar topics can be modeled through the same pattern. By applying unsupervised learning methods, we can further cluster names for the task of name disambiguation.

4.4 Topic-Based LDA

In this section, we propose another topic-based Bayesian model. Our model is primarily an extension of the Latent Dirichlet Allocation (LDA) model proposed by Blei et al. in 2003 [12], which has quickly become regarded as one of the most efficient and effective probabilistic modeling algorithm in statistical machine learning.

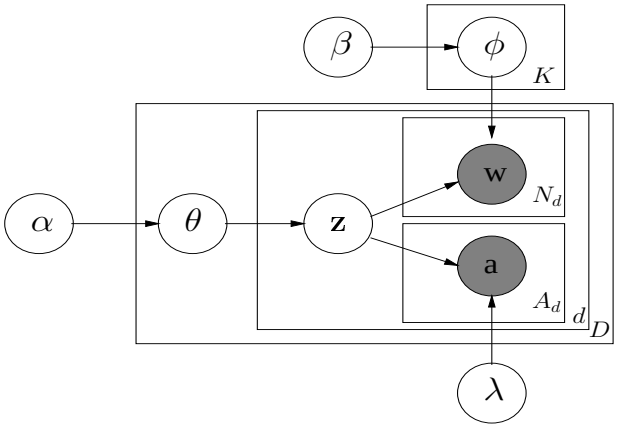
The major difference between PLSA and LDA is that in PLSA the latent variables are dependent on each document, while in LDA the topic mixture is drawn from a conjugate Dirichlet prior which remains the same for all documents. Thus LDA is able to overcome the over-fitting problem in PLSA while naturally generating new documents with consistent generative semantics.

The generative process of our topic-based LDA model can be formalized as follows:

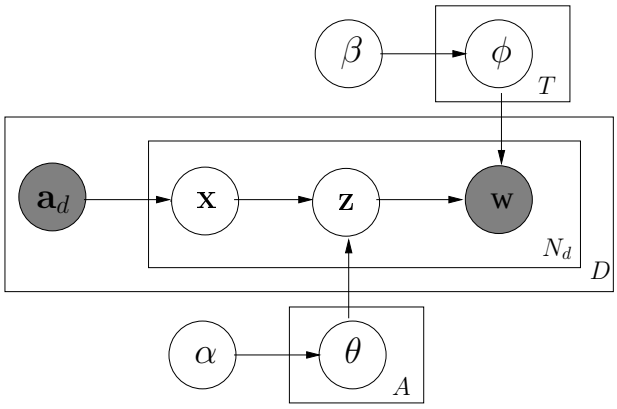
- Draw a multinomial distribution ϕ_z for each topic z from a Dirichlet distribution with prior β;
- For each document d, draw a multinomial distribution θ_d from a Dirichlet distribution with prior α;
- For each word w_{di} in document d, draw a topic z_{di} from the multinomial distribution θ_d;
- Draw a word w_{di} from the multinomial distribution $\phi_{z_{di}}$;
- Draw a name a_{di} from the multinomial distribution $\lambda_{z_{di}}$.

Figure 4.2 (a) depicts our model. Regarding the generation of parameters, α and β are corpus-level parameters and only sampled once when creating the generative corpus; θ_d are document-level variables, sampled once for each document; z_{di}, w_{di} and a_{di} are word-level variables and need to be sampled once for each word/name in the document.

Although there is resemblance between our proposed LDA model and the author-topic model [103], there exists important differences in the relationship between name



(a) Our proposed topic-based LDA model.



(b) The author-topic model [103].

Figure 4.2. Graphical model representation of the LDA model. (a) Our topic-based model. (b). The author-topic model. K is the number of topics, D is the total number of documents, N_d is the number of tokens in document d and A_d represents the number of name appearances in document d.

appearances and words. In the author-topic model, x denotes an author who is responsible for a given word. In our model, however, names (authors) and words are not directly related, i.e., each topic can generate a set of names and a set of words simultaneously with different probabilities, allowing more freedom to the model in parameter estimation.

4.4.1 Inference and Parameter Estimation

The inference problem in LDA is to compute the posterior of the (document-level) hidden variables given a document $d = (\mathbf{w}, \mathbf{a})$ with parameters α and β, i.e., $p(\theta, \phi, \mathbf{z}|\mathbf{w}, \mathbf{a}, \alpha, \beta, \lambda)$,

$$p(\theta, \phi, \mathbf{z}|\mathbf{w}, \mathbf{a}, \alpha, \beta, \lambda) = \frac{p(\theta, \phi, \mathbf{z}, \mathbf{w}, \mathbf{a}|\alpha, \beta, \lambda)}{p(\mathbf{w}, \mathbf{a}|\alpha, \beta, \lambda)}. \tag{4.10}$$

Here $p(\mathbf{w}, \mathbf{a}|\alpha, \beta, \lambda)$ is usually referred to as the marginal distribution of document d:

$$\begin{aligned} & p(\mathbf{w}, \mathbf{a}|\alpha, \beta, \lambda) \\ = & \iint p(\theta|\alpha)p(\phi|\beta) \prod_{n=1}^{N} p(w_n|\theta, \phi) \prod_{m=1}^{M} p(a_m|\theta, \lambda)\, d\theta d\phi \\ = & \iint p(\theta|\alpha)p(\phi|\beta) \left(\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta)p(w_n|z_n, \phi)) \right) \\ & \cdot \left(\prod_{m=1}^{M} \sum_{z_n} p(z_n|\theta)p(a_m|z_n, \lambda)) \right) d\theta d\phi \end{aligned} \tag{4.11}$$

By marginalizing over the hidden variable z, the name distribution $p(a|\theta, \lambda)$ can be represented as follows:

$$p(a|\theta, \lambda) = \sum_{z} p(a|z, \lambda)p(z|\theta) \tag{4.12}$$

As a result, the likelihood of a document collection D could be calculated by taking the product of the marginal probabilities of individual documents,

$$\begin{aligned} & p(D|\alpha, \beta, \lambda) = \\ & \iint \prod_{z=1}^{K} p(\phi_z|\beta) \prod_{d=1}^{N} p(\theta_d|\alpha) \left(\prod_{n=1}^{N} p(w_n|\theta, \phi) \right) \\ & \cdot \left(\prod_{m=1}^{M} p(a_m|\theta, \lambda) \right) d\theta d\phi \end{aligned} \tag{4.13}$$

Unfortunately, inference cannot be performed exactly on this model due to the problematic coupling between parameters θ, ϕ and λ. Alternative methods have been widely

developed to approximate the inference, including variational inference [12] and other methods. In the following section, we apply the Gibbs sampling framework to get around the intractability problem of parameter estimation.

4.4.1.1 Gibbs sampling for the LDA model

The Gibbs sampling algorithm was developed as a special case of the Markov Chain Monte Carlo (MCMC) algorithm, which estimates the complex joint probability distribution of several variables by generating random samples from the observed data. Note that the sampling algorithm is actually used to derive conditional probabilities for the sampler. Specifically, we need to know the conditional probabilities $p(\theta_m|\alpha, z_{m1}, ..., z_{mN})$, $p(z_{mn}|\theta_m, w_{mn}, \beta)$, where $m = 1, ..., M$ and $n = 1, ..., N$.

We construct a Markov chain that converges to the posterior distribution on $\mathbf{z}$ and then use the results to infer θ and ϕ, i.e., $p(\mathbf{z}|\mathbf{w}, \mathbf{a})$.

Based on the graphical representation in Figure 4.2, the posterior distribution can be derived as follows:

$$p(z_i = j|\mathbf{z}_{-i}, \mathbf{w}, \mathbf{a}) \propto p(z_i = j|\mathbf{z}_{-i})p(w_i|\mathbf{z}, \mathbf{w}_{-i})p(a_i|\mathbf{z}, \mathbf{a}_{-i}) \tag{4.14}$$

Notations	**Explanations**
W	number of words (vocabulary)
K	number of topics
D	number of documents
A	number of name appearances
$z_i = j$	the assignment of the ith word in a document to topic j
$\mathbf{z}_{-i}$	all topic assignments not including the ith word, i.e., $\{z_1, ..., z_{i-1}, z_{i+1}, ..., z_K\}$
H_{mj}^{WT}	number of times word m assigned to topic j, except the current instance
H_{dj}^{DT}	number of times document d contains topic j, except the current instance
ϕ_{mj}	the probability of using word m in topic j
θ_{dj}	the probability of document d contains topic j

Table 4.2. Notations used for Gibbs sampling.

$$\propto \frac{H_{dj}^{DT} + \alpha}{\sum_{j'} H_{dj'}^{DT} + K\alpha} \frac{H_{mj}^{WT} + \beta}{\sum_{m'} H_{m'j}^{WT} + W\beta}, \tag{4.15}$$

where the first two terms of Equation (15) is inferred by following the Dirichlet distribution derivation.

Note that in our case, we do not estimate the parameters α, β and λ. For simplicity and performance, they are fixed at $50/K$, 0.01 and 0.1 respectively.

Equation (13) is considered as the conditional probability of the random variables θ and ϕ. For any individual sample, we can estimate them from the latent variable $\mathbf{z}$ by

$$\hat{\theta}_{dj} = \frac{H_{dj}^{DT} + \alpha}{\sum_{j'} H_{dj'}^{DT} + K\alpha}, \tag{4.16}$$

$$\hat{\phi}_{mj} = \frac{H_{mj}^{WT} + \beta}{\sum_{m'} H_{m'j}^{WT} + W\beta}. \tag{4.17}$$

4.5 People Name Disambiguation

Learning both the PLSA and LDA models is equivalent to learning the probability distribution of the topic-word $P(w|z)$ and the topic-name $P(a|z)$ matrices. However, the topic-name matrix only reflects the relationships between names and topics, thus several people may have very similar topic interests, especially those from the same research group. For the purpose of name disambiguation, the topic-name matrix is processed further with a hierarchical clustering method. We extend the original agglomerative clustering method for our task, since it has been shown that the bottom-up clustering method performs better than the K-means method as well as other top-down clustering methods in terms of both computational cost and clustering accuracy, particularly when the number of desired clusters is not significantly smaller than the number of points.

4.5.1 Agglomerative Clustering

To distinguish people that have similar topic interests but with different names, we generate a name-name matrix that measures the pairwise similarity between names. Levenshtein distance [78] (defined as $Le(x, y)$) is used as the measurement and as a result the similarity between two names x and y can be defined as follows ($|\cdot|$ represents the

length of the string):

$$Sim(x, y) = 1 - \frac{Le(x, y)}{max(|x|, |y|)}. \tag{4.18}$$

Our modified agglomerative clustering method is shown in Algorithm 7, in which each name a_i is a vector of length K, a_{ik} reflects the probabilities of name a_i being in a specific topic k, and satisfies $\sum_k a_{ik} = 1$. We apply Euclidean distance as our point-level distance metric, i.e., $\mathcal{D}(a_i, a_j) = \sqrt{\sum_k (a_{ik} - a_{jk})^2}$. Meanwhile, to measure the distance between clusters, the complete-link metric [66] is used that considers the maximum distance of all elements in two clusters[3]. Two additional parameters should also be specified at the beginning of the algorithm, ϵ and θ, as the stopping criteria for the entire program and the merge criteria for two names/name clusters, respectively. In practice, we set $\epsilon = 0.05$ and $\theta = 0.5$.

4.6 Experiments

To evaluate the two proposed methods, we perform the experiments on two applications, i.e., disambiguation of people's web appearances and author names in scientific documents.

4.6.1 Evaluation Metrics

Instead of using a matching matrix (a.k.a. a confusion matrix in supervised learning) as in [52] (since the number of clusters K needs to be specified explicitly in advance, making it inappropriate for unsupervised learning), two sets of metrics are applied in our experiments as in [141, 57], namely **pair-level pairwise F1** score $F1P$ and **cluster-level pairwise F1** score $F1C$. $F1P$ is defined as the *harmonic mean*[4] of **pairwise precision** pp and **pairwise recall** pr, where pp is measured by the fraction of co-referent pairs in the same cluster, and pr the fraction of co-referent pairs placed in the same cluster. Likewise, $F1C$ is the harmonic mean of **cluster precision** cp and **cluster recall** cr, where cp is the fraction of totally correct clusters to the number of clusters acquired by the algorithm, and cr is the fraction of true clusters to that of the algorithm.

[3] We also tried both single-link algorithm and wards method [138], the performance are almost equally well.

[4] $H(x_1, x_2) = \frac{2x_1x_2}{x_1+x_2}$

Algorithm 7 Agglomerative Clustering

1: **Input:**
$a_1, ...a_M$: names to cluster
$\mathcal{D}(a_i, a_j)$: point-level distance metric
$\mathcal{C}(c_i, c_j)$: cluster-level distance metric
$Sim(a_i, a_j)$: name-name similarity matrix
ϵ, θ: threshold parameter
2: **Initialize**
place each name in a singleton cluster,
calculate the pairwise distance between
names according to $\mathcal{D}$,
set $\mathcal{C} \leftarrow \mathcal{D}$,
3: **Clustering Procedure**
4: **Repeat**
find two names (a_i, a_j) or name clusters (c_i, c_j) that
are closest according to $\mathcal{D}$ and $\mathcal{C}$,
randomly choose a name to represent a cluster,
if $Sim(a_i, a_j)$ is greater than θ
merge the pair to form a new cluster,
else
find the next closest pair or quit if no pair satisfy
the criteria,
update the distance between clusters according to $\mathcal{C}$,
5: **Until** the distance between the closest pair of any two clusters is greater than ϵ,
6: **Output:** Clusters $c_1, ...c_\tau$.

As the baseline method, we extracted names from the contents and formed a *name-word* matrix, which was augmented by the standard *tf-idf* method, we then applied the agglomerative clustering using inter-cluster closeness as the measure (Agglo). Our methods are further compared with two unsupervised learning approaches, the k-way spectral clustering (Spectral) [52] and the LASVM+DBSCAN approach (DBSCAN) as described in [57].

The most influential parameter on the performance as well as the scalability of our models is the number of topics K. Following convention [54, 12], we chose the values of K from the set $\{2, 5, 10, 20, 50, 100, 200\}$. For interests of space, only the best results with optimal K are reported. Meanwhile, as mentioned above, the priors α, β and λ for the LDA model are chosen as $50/K, 0.01$ and 0.1 respectively.

4.6.2 Web Appearances of Person Names

In this section, we consider the problem of automatic disambiguation of person names on the web. To be specific, when users submit name queries like "Michael Jordan" to search engines, we want to distinguish name results by the content of the retrieved web pages. We utilize the public data set[5] generated by Ron Bekkerman and Andrew McCallum [8]. 12 person names including SRI employees and professors (e.g., "David Israel" and "Andrew Ng") are submitted as queries to the Google search engine, the first 100 pages are then retrieved for each query. Post-processing is performed to clean the pages, resulting in a total of 1,085 web pages referring to 187 different people. All pages are manually labeled in the title indicating the position of the person. Among these web pages, 420 are found relevant to the 12 particular names. Some statistics can be found in [8].

For our experiment, the data set is further processed. We first translate the titles into labels with +1 indicating relevant and -1 otherwise. All URLs included in the pages are removed as well as other trivial characters. We then use the rainbow[6] tool to process the remaining text to produce the term-document matrix. Stemming and stop words removal are performed, words that appear less than twice are removed as well. Furthermore, to eliminate the bias towards longer documents, only the first 200 words are used in each example.

Table 4.3 summarizes the clustering results regarding the F1P and F1C scores. Overall, our topic-based models consistently outperform other methods for both metrics, with more than 90% on F1P score and 75% on F1C score on average. For most of the people, both PLSA and LDA achieve the best performance with 10 topics, which decrease sharply with the increase of topic numbers. The highest F1P scores for both models are achieved from the class "Leslie Pack Kaelbling", since it only has two namesakes in that class. For the "Tom Mitchell" class that has 37 namesakes, our methods are still able to achieve 85% and 82.4% F1P scores respectively, with the trade-off of using more topics (20) to disambiguate. Generally, the performance decreases and the number of topics increases with more namesakes in the class. Regarding the cluster F1 scores, since no credits will be given to clusters that are *partially* correct (i.e., either having more or less instances than the real clusters), the performance is commonly worse than the pair-wise

[5] http://www.cs.umass.edu/∼ronb
[6] http://www.cs.cmu.edu/∼mccallum/bow

	# pages	Agglo F1P	Agglo F1C	Spectral F1P	Spectral F1C	DBSCAN F1P	DBSCAN F1C	PLSA+Agglo F1P	PLSA+Agglo F1C	LDA+Agglo F1P	LDA+Agglo F1C
Cheyer	97	0.580	0.211	0.602	0.333	0.852	0.650	0.920	0.677 (10)	0.935	0.725 (20)
Cohen	88	0.515	0.208	0.500	0.210	0.742	0.520	0.888	0.625 (10)	0.850	0.625 (10)
Hardt	81	0.350	0.159	0.362	0.267	0.744	0.577	0.755	0.625 (5)	0.875	0.717 (10)
Israel	92	0.700	0.455	0.720	0.466	0.855	0.680	0.952	0.877 (20)	0.975	0.841 (20)
Kaelbling	89	0.825	0.425	0.825	0.425	0.875	0.739	0.972	0.757 (10)	0.955	0.767 (20)
Mark	94	0.396	0.208	0.475	0.340	0.575	0.500	0.855	0.717 (10)	0.871	0.704 (10)
McCallum	94	0.785	0.504	0.830	0.525	0.900	0.717	0.924	0.785 (5)	0.955	0.824 (10)
Mitchell	92	0.750	0.487	0.762	0.485	0.785	0.490	0.850	0.776 (20)	0.824	0.643 (20)
Mulford	94	0.555	0.322	0.573	0.305	0.853	0.727	0.911	0.826 (10)	0.926	0.833 (10)
Ng	87	0.750	0.542	0.785	0.575	0.915	0.845	0.951	0.925 (50)	0.953	0.911 (20)
Pereira	88	0.565	0.333	0.548	0.320	0.788	0.720	0.926	0.851 (5)	0.946	0.923 (5)
Voss	89	0.375	0.220	0.345	0.196	0.625	0.600	0.876	0.633 (10)	0.850	0.667 (10)
Mean	**90**	**0.596**	**0.340**	**0.611**	**0.371**	**0.792**	**0.647**	**0.909**	**0.756**	**0.911**	**0.765**

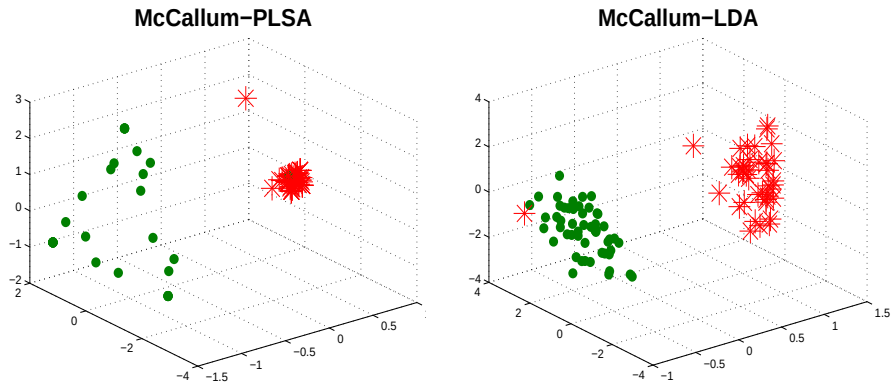
Table 4.3. Clustering results of the Web Appearances data set in terms of pair-level pairwise F1 Score(%) (F1P) and cluster-level pairwise F1 score(%) (F1C). Greedy Agglomerative Clustering is compared as a baseline approach. Our approaches (PLSA and LDA) consistently show better results than both spectral clustering and DBSCAN methods. The number of topics K is chosen from the set $\{2, 5, 10, 20, 50, 100, 200\}$. The best results with optimal K (given in parentheses) are presented here.

metrics. The best F1C scores are achieved in the class "Andrew Ng" which has 29 namesakes, larger number of topics (50 and 20 for PLSA and LDA respectively) shows better performance.

Figure 4.3 plots the result of the McCallum class for both models by projecting the data matrix on the first three eigenvectors. We choose two clusters for visualization here, one is "Andrew McCallum" from UMass and other people with the identical name for the other cluster. It is evident that both models have very high clustering accuracies and separate two clusters quite well. Specifically, PLSA only misclassified one positive instance to be negative while LDA misclassified one negative instance to be positive.

4.6.3 Author Appearances in Scientific Docs

To disambiguate author appearances in the scientific documents, we collect data from the CiteSeer Digital Library. CiteSeer is currently one of the largest digital libraries that holds more than 750,000 documents, primarily in the domain of computer and information science. CiteSeer indexes several kinds of data formats (*txt, PDF, PS*); however, for our experiment, we convert non-text formats into text and only make use of plain



(a) (b)

Figure 4.3. 3D visualization of feature distribution of the *name-topic* matrix in the web appearances data set. *'s indicate the positive class (i.e. "Andrew McCallum" from UMass) and · represents negative classes. **(a)**. PLSA result. **(b)**. LDA result.

text files. For the purpose of efficiency, extraction is performed only from the summarizing parts (title, author names, abstracts and keyword fields) and the first page of each document.

We obtained the nine most ambiguous author names from the entire data set as shown in Table 6.2, each of which has at least 20 name variations. In the worst case (C. Chen), 103 authors share the same name.

Two steps of pre-processing are performed before the experiments. First, author names are extracted from individual documents, each of which contains the author metadata associated with a unique paper identifier. Second, author references are extracted from citations by regular expressions and manual correction. Rainbow is then applied to form the document-term and document-author matrices.

Figure 4.4 plots the results of the CiteSeer data set on F1P scores and F1C scores. Clearly, our methods consistently outperform both greedy agglomerative clustering and spectral clustering, and better than DBSCAN except for the *M. Jones* class. Overall, PLSA and LDA achieve 92.3% and 93.6% pair-wise F1 metric, respectively, which shows a gain of more than 40% and 86.6% improvement over the spectral clustering and greedy agglomerative clustering. DBSCAN also achieves a comparative result (89.3%) in this case.

In terms of the cluster F1 metric, PLSA and LDA models have almost the same performance, both achieve significantly better results (more than 140%) than spectral clustering and agglomerative clustering. The relatively high F1C scores of our methods

Name	Variations	Records
A. Gupta	44	506
A. Kumar	36	143
C. Chen	103	536
D. Johnson	41	350
J. Robinson	30	115
J. Smith	86	743
K. Tanaka	20	53
M. Jones	53	352
M. Miller	34	230
Mean	49.7	336.4

Table 4.4. Summary of the 9 CiteSeer data sets of different author names and the data size. These names are most representative for the worst case scenario in author name appearances in scientific documents.

indicate that the number of unique authors can be estimated with the number of achieved clusters from the original data set.

Illustrative examples of these results are presented in Table 4.5, which summarizes the results of the PLSA model by showing the 10 highest probability words along with their corresponding conditional probabilities from 4 topics in the CiteSeer data set. Additionally, we show 3 author name variations corresponding to the same person with their probability for each topic. The appearance of new authors is handled by using the "folding-in" process discussed in Section 3.3. Clearly, the selected 4 topics reveal that the 3 name variations have very high probability to be the same author. The figure beneath depicts the probability distributions over 50 topics, of which the three names exhibit quite similar patterns.

Likewise, Table 4.6 lists the results from the LDA model. We depict several topics that show the maximum differences in probabilities to disambiguate authors with *exactly* the same name. As for the name "Yang Song", one author has very high probability of topic 4 (0.2210) while the other are highly related with topic 11 (0.2682), thus showing completely different patterns of their probability distributions over topics.

4.6.3.1 Scalability and comparison of the two models

Theoretical issue of scalability for large-scale data set has not yet been addressed for either PLSA or LDA. As a result, we empirically tested our models for the entire Cite-

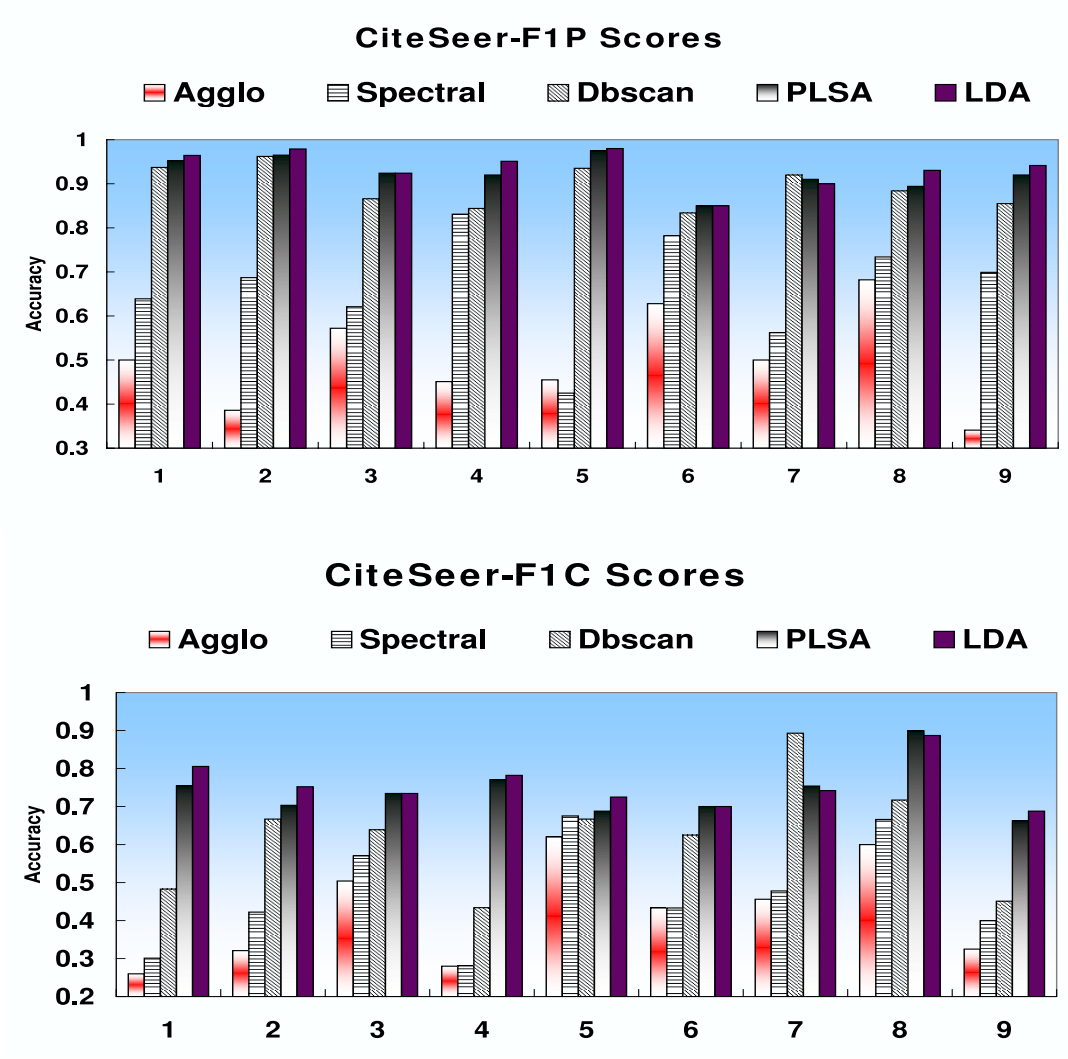


Figure 4.4. Clustering results on the CiteSeer data set. 1:A. Gupta, 2:A. Kumar, 3:C. Chen, 4:D. Johnson, 5:J. Robinson, 6:J. Smith, 7:K. Tanaka, 8:M. Jones, 9:M. Miller.

Seer data set with more than 750,000 documents. PLSA yields 418,500 unique authors in 2,570 minutes, while LDA finishes in 4,390 minutes with 418,775 authors. Both are quite consistent with previous results [57, 52]. Considering that our methods only make use of a small portion of the text for each instance (metadata plus the first page), we believe the framework can be efficient for large-scale data sets.

The results of the two models are quite close to each other in both metrics across two data sets; however, they may have different generalization capabilities. In Figure 4.5, we show the comparison between PLSA and LDA in terms of the exponential of the negative likelihood (a.k.a. *perplexity*), which is commonly used as a measure of the generalization performance of probabilistic models. Generally, lower perplexity over a

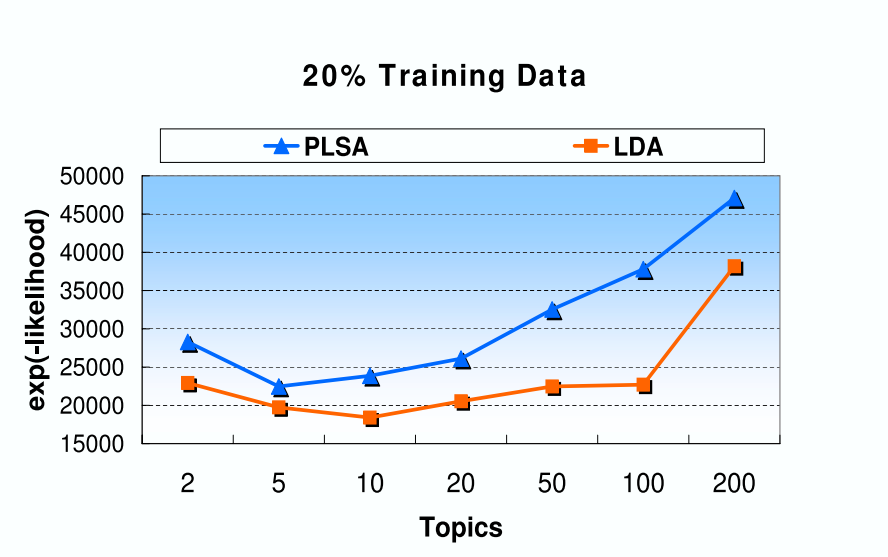


Figure 4.5. Exponential of the Negative Likelihood of the two models for the CiteSeer data set. X axis shows the number of topics. Here we show the results of using 20% training data.

set of held-out test data indicates better performance.

Figure 4.5 depicts the results for the 2 models being compared. Both models exhibit the overfitting problem when the number of topics K increases. Comparatively, LDA is less sensitive to the change of K. This probably explains why PLSA is not a *fully generative* model, since PLSA applies "folding-in" process to manage new documents. This process assumes that documents in the testing set exhibit the same topic distribution (E-step of the EM algorithm) as those in the training set, which is not essentially true in many cases. In LDA, by generating probability with predefined priors to testing documents, all documents essentially exhibit the same topic distribution, thus no assumption is required for new authors in the testing documents. Nevertheless, the best performance for both models are quite close, achieved when K is either 5 or 10.

Topic 13 “Image Categorization”		Topic 24 “Content Retrieval”	
classifiers	0.0311	feature	0.0318
region	0.0285	learning	0.0216
image	0.0211	content	0.0138
indexing	0.0157	images	0.0130
photo	0.0152	clusters	0.0130
colors	0.0133	cluster	0.0130
color	0.0123	retrieval	0.0112
extract	0.0111	location	0.0112
aesthetics	0.0103	query	0.0064
light	0.0085	classifiers	0.0061
James Wang	0.2721	James Wang	0.1478
J. Z. Wang	0.2215	J. Z. Wang	0.1362
James Ze Wang	0.2533	James Ze Wang	0.1577

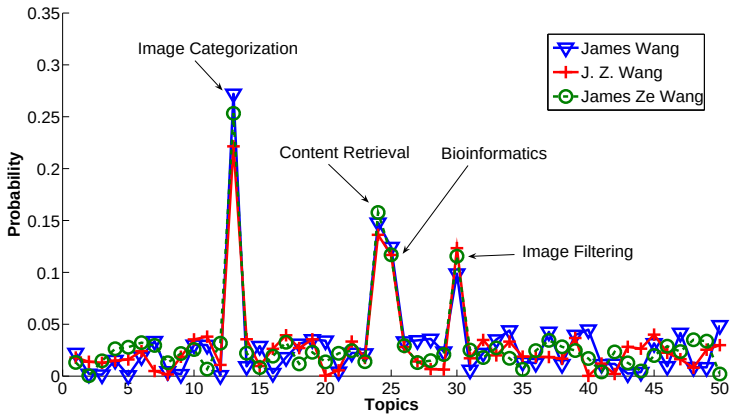


Table 4.5. An illustrative example of the author-topic relationships in the CiteSeer data set extracted by the topic-based PLSA model. 10 most corresponding words are shown for each topic. We summarize the titles of the topics to the best of our understanding. Below each topic shows the probabilities of authors with name variations. In this example three names refer to the same person.

Topic 4 “Text Classification”		Topic 11 “Vision & Motion”	
boosting	0.0473	position	0.0486
text	0.0473	motion	0.0411
classification	0.0473	perceive	0.0220
classifiers	0.0473	vision	0.0220
feature	0.0422	label	0.0162
document	0.0215	tracked	0.0162
corpora	0.0215	moving	0.0111
words	0.0116	actions	0.0111
vectors	0.0116	humans	0.0105
dimensionality	0.0116	visual	0.0105
Yang Song(PSU)	0.2210	Yang Song(PSU)	0.0320
Yang Song(Caltech)	0.0202	Yang Song(Caltech)	0.2682

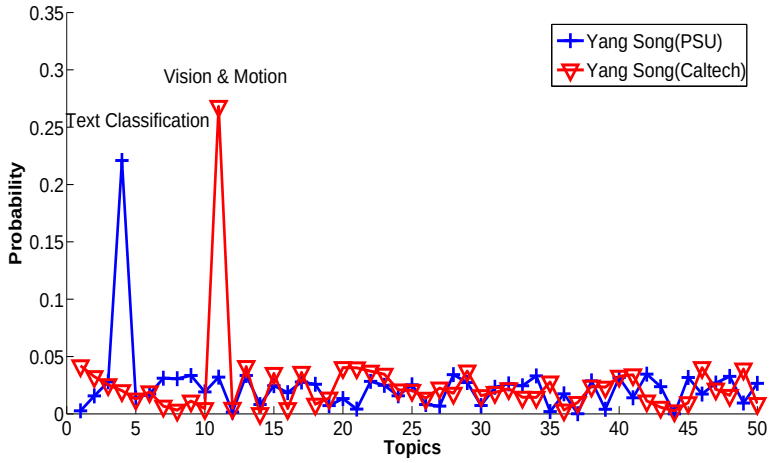


Table 4.6. LDA topic distributions of two authors with the same name “Yang Song”.

Chapter 5

Text Recommendation for Social Bookmarking Systems

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 [7, 22, 45, 80].

5.0.4 The Problem

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 5.2, where the user is bookmarking a webpage regarding data mapper and the system recommends relevant

[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 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 tags, while the relationship between tags themselves cannot be observed directly (shown in Figure 5.1). Consequently, recommending relevant tags to new users or new documents can only be done indirectly from the user perspective or the document perspective.

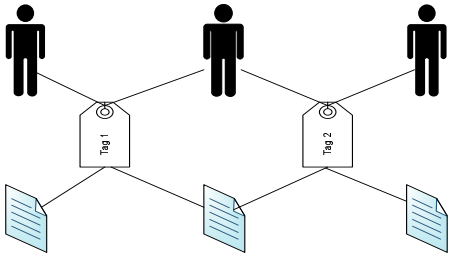


Figure 5.1. A connectivity graph of users, tags and documents. In the scenario of tagging, a user annotates a document by creating a personal tag. As it can be observed, tags are not directly connected to each other, but to the users and documents instead.

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 [15] 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 [12]) and supervised classification models (e.g., SVM [27]) are good candidates for categorizing document topic groups.

While both approaches seem to be plausible, it turns out that the user-centered ap-

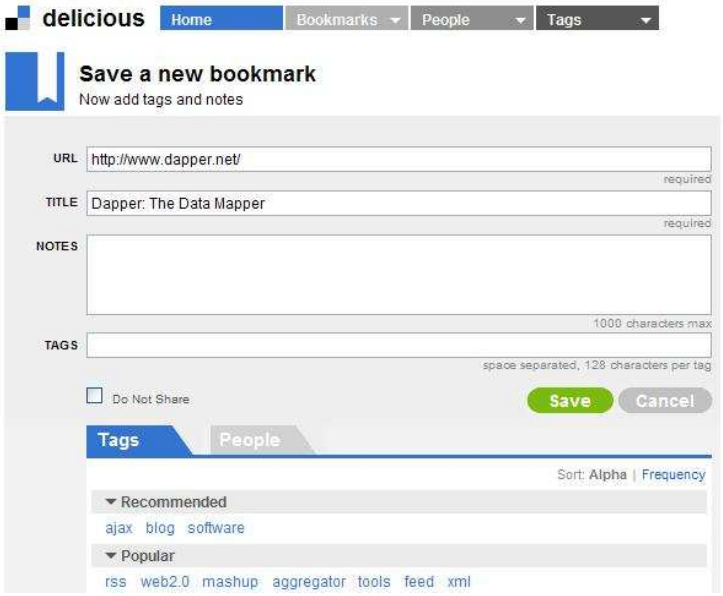
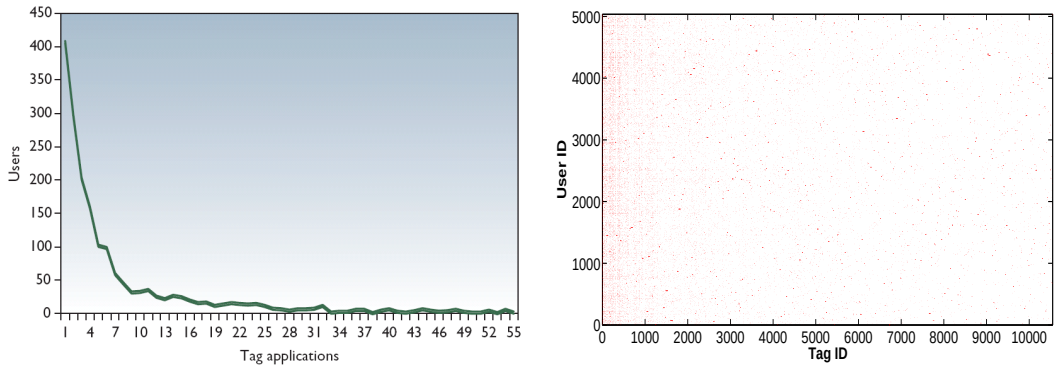


Figure 5.2. An example of recommended tags by the Del.icio.us recommender system.

proaches are not very effective due to several obvious reasons. First, according to research in [37], 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 5.3 (a)). Additionally, researchers have also shown that the reusability of tags are quite low, while the vocabulary of tags constantly grows [37] (see Figure 5.3 (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.

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



(a) (b)

Figure 5.3. 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.

effectiveness [7, 22, 45]. Efficiency, while not being totally ignored, has only been of recent interest [80].

5.1 Our Contributions

In this thesis, 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) [149] 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 [81] 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 Pois-

son 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 [101] 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) [27] usually do not consider the predictive variance of test cases[4]. For tag recommendation where

[4]Although Platt suggested an ad-hoc probabilistic SVM in [99], it does not consider the predictive

the tagged data (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 [37]. Therefore, the tagged data sets used for empirical analysis are usually of high-dimensionality and sparseness [119]. 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 [101].

5.2 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 [45, 7]. 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

variance of the function.

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 [22], 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 [7] 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 [80]. 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.

5.3 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.

5.3.1 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 5.4 depicts an undirected bipartite graph with 4 documents and 5 terms.

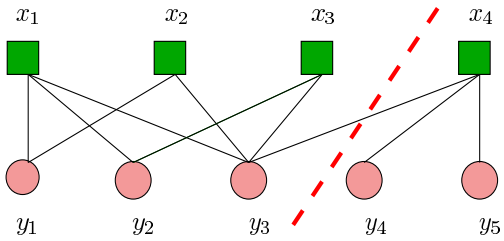


Figure 5.4. A bipartite graph of X (documents) and Y (terms). Dot line represents a potential (best) cut of the graph.

5.3.2 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 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

$$L(W) = D^{(-1/2)} W D^{(-1/2)}. \tag{5.1}$$

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

$$\lambda_1 \geq ... \geq \lambda_n > 0, \tag{5.2}$$

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:

$$\|A - T\|_F \leq \epsilon \|A\|_F, \tag{5.3}$$

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.

5.3.3 Bipartite Graph Partitioning

For multi-clustering on bipartite graphs, we apply the Spectral Recursive Embedding (SRE) algorithm [149]. 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:

$$Ncut(A, B) = \frac{cut(A, B)}{W(A, Y) + W(X, B)} + \frac{cut(A^c, B^c)}{W(A^c, Y) + W(X, B^c)}, \tag{5.4}$$

where

$$\begin{aligned} cut(A, B) &= W(A, B^c) + W(A^c, B) \\ &= \sum_{i \in A, j \in B^c} w_{ij} + \sum_{i \in A^c, j \in B} w_{ij}. \end{aligned} \tag{5.5}$$

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.

5.3.4 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 $C(i)$,

$$np_i = \frac{\sum_{j=1}^{n} w_{ij} \mathbb{I}[C(j) = C(i)]}{\sum_{j,k=1}^{n} w_{jk} \mathbb{I}[C(j) = C(k) = C(i)]}, j, k \neq i. \tag{5.6}$$

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 (5.6) 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

$$nr_i = \frac{|E_i|}{|E_i| - \sum_{j=1}^{n} \mathbb{I}[C(j) = C(i)]}, \tag{5.7}$$

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:

$$\begin{aligned} Rank_i &= \begin{cases} \exp\left(-\frac{1}{r(i)^2}\right) & r(i) \neq 0, \\ 0 & r(i) = 0, \end{cases} \\ \texttt{where}\ r(i) &= (np_i) * \log(nr_i). \end{aligned} \tag{5.8}$$

Depicted in Figure 5.5, 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 5.5 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.

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.

5.3.5 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 5.6.

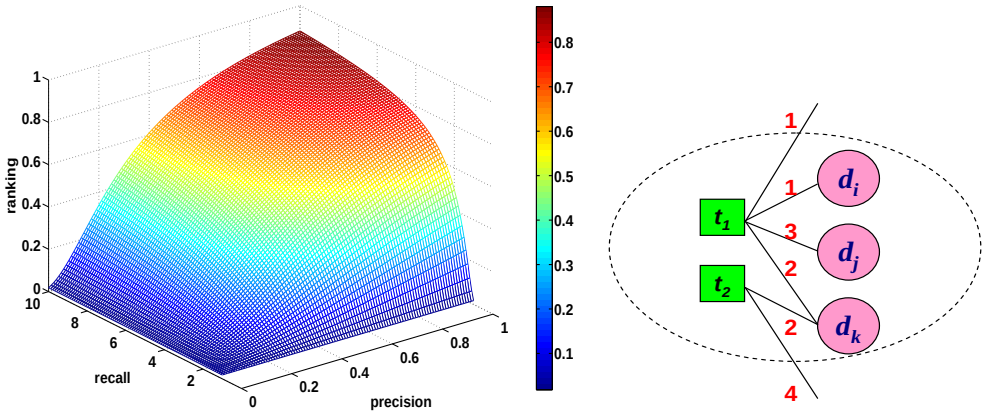


Figure 5.5. Smoothed Ranking Function (left) and an example of two-tag three-document cluster (right), with the numbers on the edges showing the frequencies of tags being annotated to specific documents.

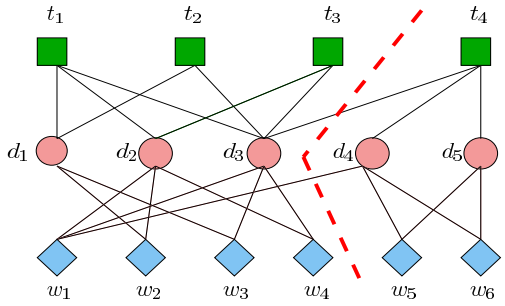


Figure 5.6. Two bipartite graphs of documents, words and tags.

The weighted graph can be written as

$$W = \begin{pmatrix} 0 & A & 0 \\ A^T & 0 & B \\ 0 & B^T & 0 \end{pmatrix}, \tag{5.9}$$

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 [15]. 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 8.

Algorithm 8 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. (5.9)
Normalize W using the normalized Laplacian
$L(W) = D^{(-1/2)} W D^{(-1/2)}$ (eq. (5.1))
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 [149],
$\tilde{W} = \{\tilde{W}_1, ..., \tilde{W}_K\}$
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. (5.8))
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. (5.9))
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. (5.16)))
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. (5.17))

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.

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

Nevertheless, traditional unsupervised learning approaches of mixture models [38, 107] 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 [81] in order to simultaneously cluster word features and classify documents.

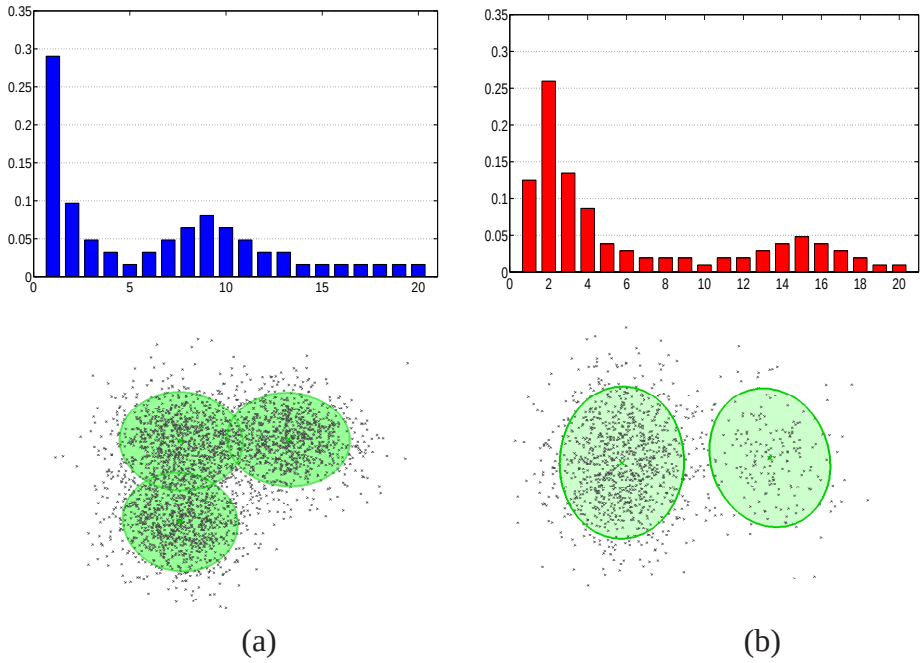
Given a document $D = \{D_1, ..., D_p\}$, where p is the dimension, the distribution of the document vector in each class can be estimated by using a parametric mixture model. Let the class label be $C = \{1, 2, ..., K\}$, then

$$P(D = d|C = k) = \sum_{m=1}^{M} \pi_m \mathbb{I}(F(m) = k) \prod_{j=1}^{p} \phi(d_j|\lambda_{j,m}), \tag{5.10}$$

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 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 5.7 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



(a) (b)

Figure 5.7. 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.

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

$$P(D = d|C = k) \propto \sum_{m=1}^{M} \pi_m \mathbb{I}(F(m) = k) \prod_{l=1}^{L} \phi(d_{k,l}|\tilde{\lambda}_{l,m}). \tag{5.11}$$

5.3.6.1 Parameter Estimation

With the classes determined, we apply EM algorithm [32] 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 $p_{i,m}$:

$$p_{i,m} \propto \pi_m^{(t)} \mathbb{I}(C(i)) \prod_{j=1}^{p} \theta(d(i,j)|\tilde{\lambda}_{m,i,j}^{(t)}). \tag{5.12}$$

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

$$\pi_m^{(t+1)} = \frac{\sum_{i=1}^{n} p_{i,m}}{\sum_{m'=1}^{M} \sum_{i=1}^{n} p_{i,m'}}, \tag{5.13}$$

$$\tilde{\lambda}_m^{(t+1)} = \frac{\sum_{i=1}^{n} p_{i,m} \sum_j d(i,j) \mathbb{I}(C(i))}{|d(i,j)| \sum_{i=1}^{n} p_{i,m}}, \tag{5.14}$$

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:

$$c^{(t+1)}(k,j) = \arg\max_l \sum_{i=1}^{n} \sum_{m \in \mathcal{R}_k} \log(d(i,j)|\tilde{\lambda}_{m,l}^{(t+1)}). \tag{5.15}$$

5.3.7 Tag Recommendation for New Documents

Normally, the class label $C(d_t)$ of a new document d_t is determined by the maximum likelihood $\hat{C}(x) = \arg\max_k P(C = k|D = d_t)$. 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 (5.11) and the Bayes rule,

$$\begin{aligned} P(C = k|D = d_t) &= \frac{P(D = d_t|C = k)P(C = k)}{P(D = d_t)} \\ &= \frac{\sum_{m=1}^{M} \pi_m \mathbb{I}(F(m) = k) \prod_{l=1}^{L} \phi(d_{k,l}|\tilde{\lambda}_{l,m}) P(C = k)}{P(D = d_t)}, \end{aligned} \tag{5.16}$$

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

$$R(T_i, d_t) = P(T = T_i | D = d_t) = Rank_{T_i} * P(C = x | D = d_t). \quad (5.17)$$

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

5.4 Approach 2: A Prototype-based method

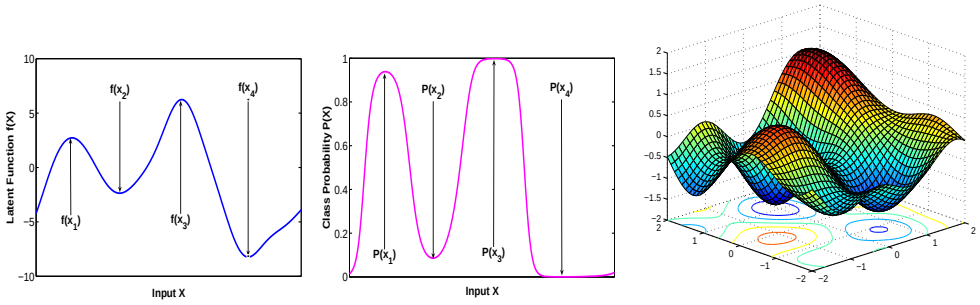
The second method we introduce here, a prototype-based method, is made up of three main parts: (1) train a multi-class multi-label Gaussian processes classifier, (2) find the most informative prototypes (i.e., representatives) for each class, (3) perform a multi-label classification for a new document by assigning it to one or more class, and recommend the highest-ranked tags to the document.

5.4.1 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 5.8 (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 $\mathbf{f}_*$.

5.4.2 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 un-



(a) (b) (c)

Figure 5.8. 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.

known 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 [101], we introduce a vector of latent function values of N training points for C classes, which has length CN

$$\mathbf{f} = (f_1^1, ..., f_N^1, ..., f_1^j, ..., f_N^j, ..., f_1^C, ..., f_N^C)^T, \tag{5.18}$$

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 $\mathbf{K}(\mathbf{x}_n, \mathbf{x}_{n'}) \triangleq [\mathbf{K}_N]_{nn'}$. 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 [101]. A commonly used function in the classification case is the *squared exponential* function, defined as:

$$[\mathbf{K}_N]_{nn'} = l \exp\left(-\frac{1}{2} \frac{\sum_{d'=1}^{D} \left(x_n^{(d')} - x_{n'}^{(d')} \right)^2}{\mathbf{\Sigma}^2} \right), \tag{5.19}$$

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,

$$p(\mathbf{f}|X,y) = \frac{p(\mathbf{f}|x)p(y|\mathbf{f})}{p(X,y)} \overset{i.i.d.}{=} \frac{\mathcal{N}(\mathbf{0},K)}{p(X,y)} \prod_{i=1}^{N} p(\mathbf{y}_i|\mathbf{f}_i), \tag{5.20}$$

which is non-Gaussian. In eq.(5.20), 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:

$$p(y_i^c|\mathbf{f}_i) = \frac{\exp(f_i^c)}{\sum_{c'} \exp(f_i^{c'})}. \tag{5.21}$$

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 $\mathbf{f}$:

$$p(\mathbf{f}_*|X,y,\mathbf{x}_*) = \int p(\mathbf{f}_*|\mathbf{f},X,\mathbf{x}_*) \underbrace{p(\mathbf{f}|X,y)}_{\text{eq.(5.20)}} d\mathbf{f}, \tag{5.22}$$

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

$$p(\mathbf{y}_*|X,y,\mathbf{x}_*) = \int p(\mathbf{y}_*|\mathbf{f}_*) \underbrace{p(\mathbf{f}_*|X,y,\mathbf{x}_*)}_{\text{eq.(5.22)}} d\mathbf{f}_*. \tag{5.23}$$

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 [74, 108]. 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.

5.4.3 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 5.9 for details.

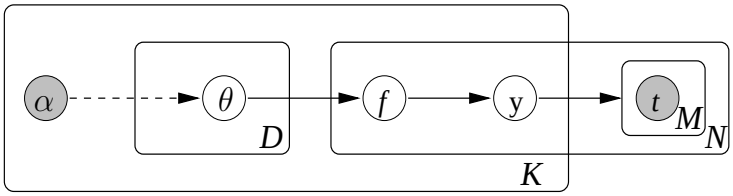


Figure 5.9. Graphical representation of our sparse multi-class GP model. θ is the hyper-parameter that define the latent function f. α denotes the extra parameter for placing a distribution over θ.

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. (5.19), such that these points have a similar distribution to the training data,

$$p(\bar{\mathbf{f}}|\bar{X}) = \mathcal{N}(\bar{\mathbf{f}}|\mathbf{0}, \mathbf{K}_M). \tag{5.24}$$

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:

$$p(\mathbf{f}_*|\mathbf{x}_*, X, \mathbf{y}, \bar{\mathbf{f}}, \bar{X}) = \int \underbrace{p(\mathbf{f}_*|\mathbf{x}_*, \bar{\mathbf{f}}, \bar{X})}_{A} \underbrace{p(\bar{\mathbf{f}}|X, \mathbf{y}, \bar{X})}_{B} d\bar{\mathbf{f}}, \tag{5.25}$$

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:

$$\mathbf{f}_*|\mathbf{x}_*, \bar{\mathbf{f}}, \bar{X} \sim \mathcal{N}(\mathbf{f}_*|\mathbf{k}_{\mathbf{x}_*}^T \mathbf{K}_M^{-1} \bar{\mathbf{f}}, K_{\mathbf{x}_*\mathbf{x}_*} - \mathbf{k}_{\mathbf{x}_*}^T \mathbf{K}_M^{-1} \mathbf{k}_{\mathbf{x}_*}), \tag{5.26}$$

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.

5.4.4 Laplace Approximation for the Posterior

Our method to approximate B in eq.(5.25) is based on the Laplace approximation, which were used in [101] for binary classification. Using the Bayes' rule,

$$\begin{aligned} p(\bar{\mathbf{f}}|X,\mathbf{y},\bar{X}) &= \frac{p(\bar{\mathbf{f}}|\bar{X})p(\mathbf{y}|\bar{\mathbf{f}},X,\bar{X})}{p(\mathbf{y}|X,\bar{X})} \\ &= \frac{p(\bar{\mathbf{f}}|\bar{X})\int \overbrace{p(\mathbf{f}|\bar{\mathbf{f}},X,\bar{X})p(\mathbf{y}|\mathbf{f})}^{C}\,d\mathbf{f}}{p(\mathbf{y}|X,\bar{X})}. \end{aligned} \tag{5.27}$$

Since the denominator $p(\mathbf{y}|X,\bar{X})$ in eq.(5.27) 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.(5.21) and is not Gaussian. Taking the logarithm of C in eq. (5.27), we have:

$$\mathcal{L}(\mathbf{f}) \triangleq \log \underbrace{p(\mathbf{f}|\bar{\mathbf{f}},X,\bar{X})}_{\mathcal{L}_1} + \log \underbrace{p(\mathbf{y}|\mathbf{f})}_{\mathcal{L}_2}, \tag{5.28}$$

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

$$p(\mathbf{f}|\bar{\mathbf{f}},X,\bar{X}) = \prod_{n=1}^{N} \underbrace{p(\mathbf{f}_n|\mathbf{x}_n,\bar{\mathbf{f}},\bar{X})}_{\text{eq.(5.26)}} = \mathcal{N}(\mathbf{f}|\mathbf{K}_{NM}\mathbf{K}_M^{-1}\bar{\mathbf{f}},\mathbf{\Lambda}), \tag{5.29}$$

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.(5.29) $\&$ (5.21), we can evaluate eq.(5.28) as follows:

$$\begin{aligned} \mathcal{L}(\mathbf{f}) &= \left(-\frac{CN}{2}\log 2\pi - \frac{1}{2}\log|\mathbf{\Lambda}| - \frac{1}{2}(\mathbf{W}^T\mathbf{\Lambda}^{-1}\mathbf{W})\right) \\ &+ \left(\mathbf{y}^T\mathbf{f} - \sum_{i=1}^{N}\log(\sum_{c=1}^{C}\exp f_i^c)\right), \end{aligned} \tag{5.30}$$

where $\mathbf{W} = \mathbf{f} - \mathbf{K}_{NM}\mathbf{K}_M^{-1}\bar{\mathbf{f}}$. By differentiating eq.(5.30) w.r.t. $\mathbf{f}$, we obtain

$$\nabla_{\mathbf{f}}\mathcal{L}(\mathbf{f}) = -\mathbf{\Lambda}^{-1}\mathbf{f} + \mathbf{\Lambda}^{-1}\mathbf{K}_{NM}\mathbf{K}_M^{-1}\bar{\mathbf{f}} + \mathbf{y} - \mathbf{m}, \tag{5.31}$$

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 $\mathbf{f}$:

$$\hat{\mathbf{f}} = \mathbf{K}_{NM}\mathbf{K}_M^{-1}\bar{\mathbf{f}} + \mathbf{\Lambda}(\mathbf{y} - \hat{\mathbf{m}}). \tag{5.32}$$

Differentiating eq.(5.31) again, we obtain

$$\nabla\nabla_{\mathbf{f}}\mathcal{L}(\mathbf{f}) = -\mathbf{\Lambda}^{-1} - \mathbf{M}, \;\; \mathbf{M} \stackrel{\triangle}{=} \text{diag}(\mathbf{m}) - \Pi\Pi^T. \tag{5.33}$$

According to [101], 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 $\mathbf{f}$:

$$\begin{aligned} \mathbf{f}' &= \mathbf{f} - (\nabla\nabla_{\mathbf{f}})^{-1}\nabla_{\mathbf{f}} \\ &= (\mathbf{\Lambda}^{-1} + \mathbf{M})^{-1}(\mathbf{M}\mathbf{f} + \Lambda^{-1}\mathbf{K}_{NM}\mathbf{K}_M^{-1}\bar{\mathbf{f}} + \mathbf{y} - \mathbf{m}). \end{aligned} \tag{5.34}$$

Applying the Taylor Expansion, we obtain

$$\mathcal{L}(\mathbf{f}) = \mathcal{L}(\hat{\mathbf{f}}) - \frac{1}{2}\nabla\nabla_{\mathbf{f}}\mathcal{L}(\mathbf{f})(\mathbf{f} - \hat{\mathbf{f}})^2. \tag{5.35}$$

Thus the integral part in eq.(5.27) can be estimated analytically:

$$\begin{aligned} \int (C)d\mathbf{f} &= \int \exp(\mathcal{L}(\mathbf{f}))d\mathbf{f} \\ &= \int \exp\left(\mathcal{L}(\hat{\mathbf{f}}) - \frac{1}{2}\nabla\nabla_{\mathbf{f}}\mathcal{L}(\mathbf{f})(\mathbf{f} - \hat{\mathbf{f}})^2\right) d\mathbf{f} \\ &= \exp\left(\mathcal{L}(\hat{\mathbf{f}})\right) \int \exp\left(-\frac{1}{2}\nabla\nabla_{\mathbf{f}}\mathcal{L}(\mathbf{f})(\mathbf{f} - \hat{\mathbf{f}})^2\right) d\mathbf{f} \\ &= \exp\left(\mathcal{L}(\hat{\mathbf{f}})\right) \sqrt{2\pi}\left|\nabla\nabla_{\mathbf{f}}\mathcal{L}(\mathbf{f})\right|^{-1}. \end{aligned} \tag{5.36}$$

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.(5.27), as $p(\bar{\mathbf{f}}|\bar{X})$ follows a normal distribution according to eq.(5.24), the posterior also forms a normal distribution. Consequently, we only need to calculate the mean and variance.

After some matrix manipulation, we have

$$\begin{aligned} \boldsymbol{\mu}_p = \frac{\mathbf{Q}^{-1}\mathbf{P}}{2}, \boldsymbol{\Sigma}_p = \mathbf{Q}^{-1}, \\ \text{where } \mathbf{Q} &= (\mathbf{K}_{NM}\mathbf{K}_M^{-1})^T \boldsymbol{\Lambda}^{-1}(\mathbf{K}_{NM}\mathbf{K}_M^{-1}) + \mathbf{K}_M, \\ \mathbf{P} &= \hat{\mathbf{f}}^T \boldsymbol{\Lambda}^{-1}(\mathbf{K}_{NM}\mathbf{K}_M^{-1}). \end{aligned} \tag{5.37}$$

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.(5.27). We can thus compute the latent values of the new $\mathbf{x}_*$ by plugging the result into eq.(5.25). 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

$$\boldsymbol{\mu}_* \simeq \boldsymbol{\mu}_p, \tag{5.38}$$

$$\boldsymbol{\Sigma}_* = \mathbf{K}_M + \boldsymbol{\Sigma}_p. \tag{5.39}$$

5.4.4.1 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_*:

$$p(\mathbf{y}_*|\mathbf{x}_*, X, \mathbf{y}, \bar{\mathbf{f}}, \bar{X}) = \int \tilde{p}(\mathbf{f}_*|\mathbf{x}_*, X, \mathbf{y}, \bar{\mathbf{f}}, \bar{X}) p(\mathbf{y}_*|\mathbf{f}_*) df_*, \tag{5.40}$$

which again cannot be solved analytically. One way to approximate is to use cumulative Gaussian likelihood. In [101], 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):

$$t(\mathbf{x}_*) = \arg\max_c p(y(\mathbf{x}_*)^c|\cdot), \quad c = 1, ..., C. \tag{5.41}$$

5.4.5 Informative Points Selection

It remains to optimize the parameters $\Theta = \{\theta, \bar{X}\}$, which contain the hyper-parameters $(l, \boldsymbol{\Sigma})$ for the covariance matrix $\mathbf{K}$ as well as finding the subset $\bar{X}$ of M points. Tradi-

tionally, they are optimized jointly by optimizing the marginal likelihood of the training data. In our approach, we instead treat them individually.

5.4.5.1 Parameter Inference for the Covariance Matrix

The marginal likelihood of $\mathbf{y}$ can be obtained by integrating out $\bar{\mathbf{f}}$,

$$p(\mathbf{y}|X, \bar{X}, \Theta) = \int p(\mathbf{y}|X, \bar{X}, \bar{\mathbf{f}}) p(\bar{\mathbf{f}}|\bar{X}) d\bar{\mathbf{f}} = \int \exp(\mathcal{L}(\bar{\mathbf{f}})) d\bar{\mathbf{f}}. \tag{5.42}$$

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

$$p(\mathbf{y}|X, \bar{X}, \Theta) = \exp(\mathcal{L}(\hat{\bar{\mathbf{f}}})) \int \exp\left(\frac{1}{2} (\bar{\mathbf{f}} - \hat{\bar{\mathbf{f}}})^T \nabla \nabla_{\bar{\mathbf{f}}} \mathcal{L}(\bar{\mathbf{f}}) (\bar{\mathbf{f}} - \hat{\bar{\mathbf{f}}}) \right) d\bar{\mathbf{f}}. \tag{5.43}$$

The log marginal likelihood can be obtained by taking logarithm on both sizes of the above equation,

$$\log p(\mathbf{y}|X, \bar{X}, \Theta) = \mathcal{L}(\hat{\bar{\mathbf{f}}}) - \frac{CN}{2} \log 2\pi - \frac{1}{2} \log |\nabla \nabla_{\bar{\mathbf{f}}} \mathcal{L}(\bar{\mathbf{f}})|, \tag{5.44}$$

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

5.4.5.2 Prototype selection for $\bar{X}$

The original gradient calculation in eq.(5.44) is very complicated. However, we can simplify it with the assumption made on the covariance matrix. Since each $\boldsymbol{\Sigma}_c$ 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 [110], which is used for K-nearest neighbor classification. We select a set of M prototypes by minimizing the misclassification rate of the training set,

$$\mathfrak{L}(X, \bar{X}) = \frac{1}{N} \sum_{n=1}^{N} \sum_{m=1}^{M} P(\bar{\mathbf{x}}_m|\mathbf{x}_n)(1 - \mathbb{I}(\bar{y}_m = y_n)), \tag{5.45}$$

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:

$$P(\bar{\mathbf{x}}_m|\mathbf{x}) = \frac{\mathbf{k}_{\bar{\mathbf{x}}_m \mathbf{x}}}{\sum_{m'=1}^{M} \mathbf{k}_{\bar{\mathbf{x}}_{m'} \mathbf{x}}}. \tag{5.46}$$

We can further rewrite the loss function in eq.(5.45) by removing the indicator function:

$$\mathfrak{L}(X, \bar{X}) = \frac{1}{N} \sum_{n} \underbrace{\sum_{\{m:\bar{y}_m \neq y_n\}} P(\bar{\mathbf{x}}_m|\mathbf{x}_n)}_{\mathfrak{l}_m}, \tag{5.47}$$

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 notice that only those points falling into a particular area of the input space can contribute to

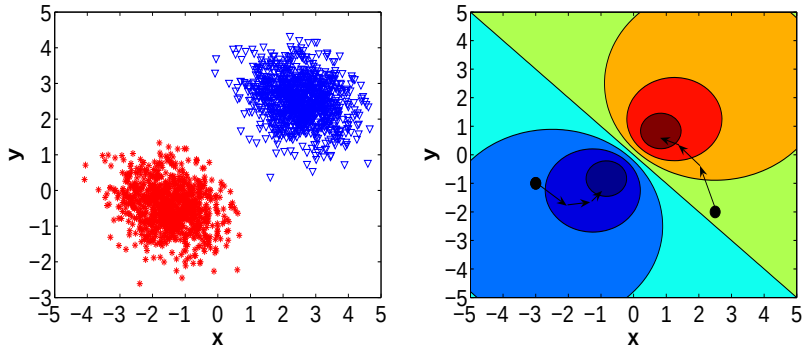


Figure 5.10. An example of prototype selection with $M = 2$. Left figure shows the original distribution; right figure, contour-plots the results of descent where black dots are the starting points.

the update of the prototypes. This fact is explained as the *window* rule in [67]. So we can speed up the prototype updates by searching over those points only. Figure 5.10 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).

5.4.6 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.(5.29), $\mathbf{f}'$ in eq.(5.35), as well as $\mathbf{Q}$ and $\mathbf{P}$ in eq.(5.37). For efficient inversion, Cholesky decomposition is often employed [101], 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.(5.46), no additional storage and computation are required. Therefore, eq.(5.48) can

be efficiently updated in at most $O(NC)$ time.

5.4.7 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 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[129].

As pointed out in [16], 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

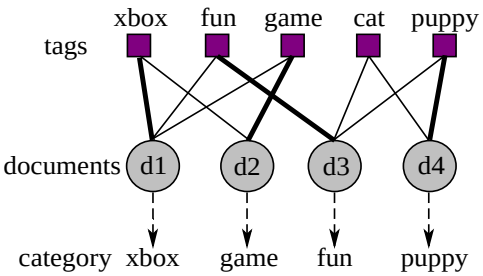


Figure 5.11. Example of document-tag graph. Each document is associated with multiple tags. Tag with the highest frequency is treated as the category of that document (shown in bold line).

Algorithm 9 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.(5.40))
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 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 5.11) 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 5.11 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 5.11, *fun* appears in two categories. The above two phenomenon can be roughly explained by the behavior of *polysemy* and *synonymy* in linguistics. Table 5.1 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 5.1. 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

$$Rank_{y_{i'}}^{(c)} = \frac{1}{Z^{(c)}} \sum_{i:\mathbf{x}_i \in c} \sum_j s(y_{ij}) \mathbb{I}(y_{ij} = y_{i'}), y_{i'} = \{y_{i1}, ..., y_{i\tilde{K}}\} \tag{5.48}$$

where $Z^{(c)}$ is a normalization factor for category c. We summarize this approach in Algorithm 9, $\tilde{K}$ 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 5.12 illustrates the process.

5.5 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

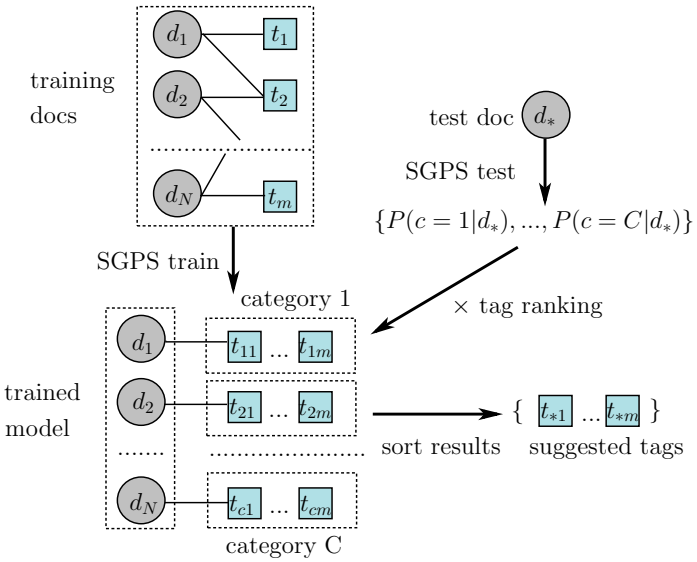


Figure 5.12. The training and test processes of MMSG. Each d_i is a document and each t_i is a tag.

tagging results as well as the efficiency of the tagging algorithms[5].

5.5.1 Evaluation Metrics

In addition to the standard precision, recall, F-score and Kendall τ rank correlation metric [64] 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.
- *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]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 [117].

5.5.2 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 5.2 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%.

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

5.5.3 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 [15]. 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, $Sim(Q, D_i) = \frac{\sum_i n(Q,j)n(i,j)}{\sqrt{\sum_j n(Q,j)^2}\sqrt{\sum_i n(i,j)^2}}$, 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 [72] (VS+IG) to select roughly 5% of the total features.

The second method we compare to is the famous topic model by Blei [12], namely Latent Dirichlet Allocation (LDA). For tag recommendation, we first trained a n-topic LDA model [12], 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 [27]. 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.(5.48) to return top ranked tags for recommendation.

5.5.4 Quality of the Tagging Performance

Table 5.7 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 paper "A Tutorial on Learning With Bayesian Networks", which has a Kendall τ rank of 0.78.

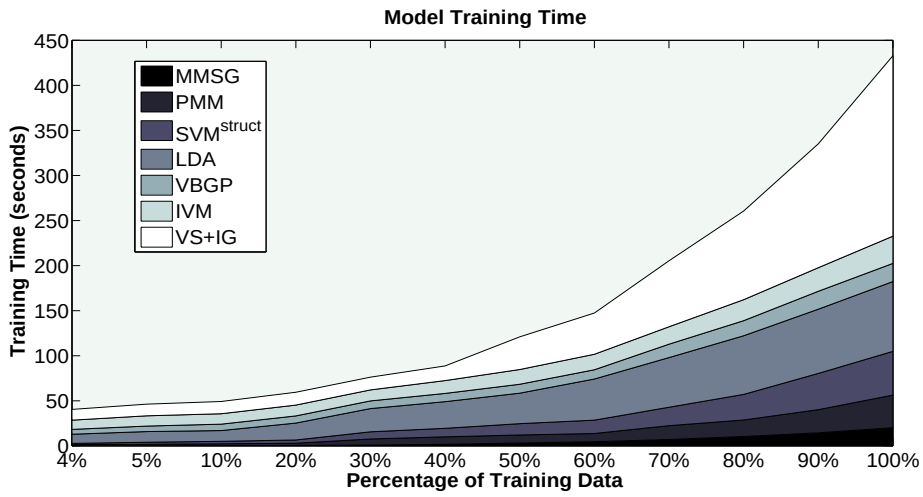


Figure 5.13. Average tagging time on the CiteULike data set. Our models require the least time for making recommendations.

We present a summary of the experimental results in Table 5.6. Overall, our models PMM and MMSG exhibit better performance for all three data sets. On average, PMM

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

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 5.3. Tagging performance.

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 5.16 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.

5.5.4.1 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.

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 used for classification is the squared exponential function (SE) in eq.(5.19). An alternative function takes the form of neural network (NN):

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = \frac{2}{\pi} \sin^{-1} \left(\frac{2\tilde{\mathbf{x}}^T \Sigma \tilde{\mathbf{x}}'}{\sqrt{(1 + 2\tilde{\mathbf{x}}^T \Sigma \tilde{\mathbf{x}})(1 + 2\tilde{\mathbf{x}}'^T \Sigma \tilde{\mathbf{x}}')}} \right), \tag{5.49}$$

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 5.14 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.

5.5.4.2 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 [109] which efficiently includes points into the active set based on information gain (IG). We also include a random selection

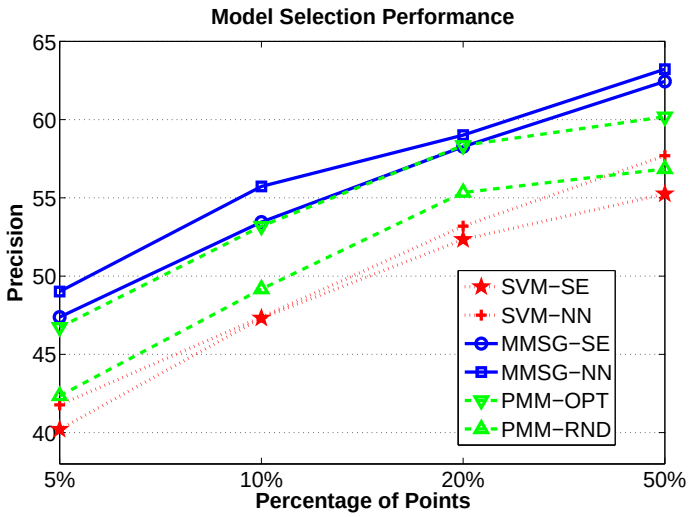


Figure 5.14. Comparison of tagging performance of SVM, PMM and MMSG. Two covariance functions used: SE = squared exponential, NN = neural network.

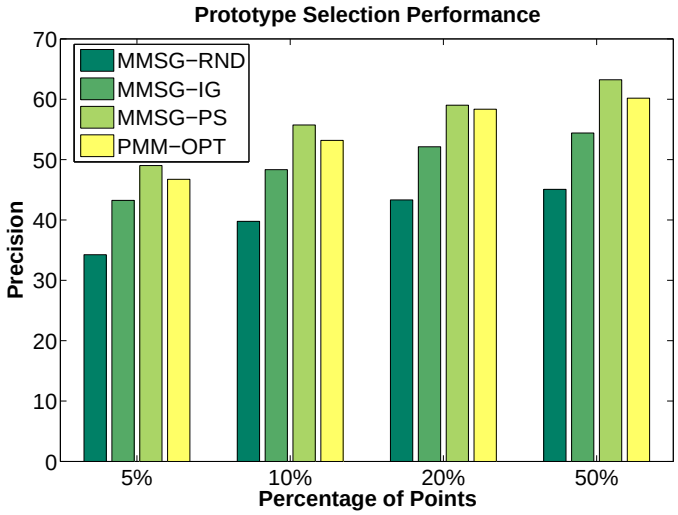


Figure 5.15. Tagging performance of three selection algorithms and PMM-OPT. RND = random selection, IG = information gain, PS = prototype selection.

(RS) method as the baseline. Figure 5.15 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 $M = 50\%$.

5.5.5 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 [45] 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.

5.5.6 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 5.13 and Table 5.4 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

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

respectively, expected to grow exponentially with the increase of the features.

% **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 5.4. Average tagging time (seconds) for the three data sets.

5.6 Tag Recommendation for Rich Media Data

The amount of digital interactive media has been growing at a phenomenal rate since the emergence of the Web 2.0. Web sites that populate rich media such like Flickr[10] (image) and Youtube[11] (video) have attracted a significant amount of Internet traffic, as well as millions of Internet users. These web sites also allow users to specify keywords or tags for resources which are of interest.

However, making tag suggestions to rich media data is not as straightforward as suggesting tags for a collection of text data [49](e.g., del.icio.us[12]). The reason is multifold. Tags are usually in the format of text while the content of the objects is not. Therefore, domain knowledge is often required for content-based object retrieval, which is not universally applicable for applications across domains. In addition, scalability needs to be addressed when making tag suggestion, since storing/retrieval rich media data is not cheap due to the size of the object (usually much larger than text data). In [80], the authors proposed a real-time annotation method for images. A generative model is trained by exploiting statistical relationships between words and images. The model is capable of annotating an individual image in approximately 2 seconds.

[10]http://www.flickr.com/
[11]http://youtube.com/
[12]http://del.icio.us/

Nevertheless, the state-of-the-art for the training time for digital data leaves much to be desired. In [80], the reported training time is roughly 16.6 hours for 599 categories (classes), each of which contains 80 training images. The cost to re-train the model for other data sets or domains is obviously substantial.

Therefore, we propose a universal framework for tag rich media data by using our MMSG tagging algorithms. To be exact, we leverage the side (textual) information of the data as features for training MMSG. We claim that our tagging approach is especially suitable for large-scale digital data in the sense that:

- Our tagging algorithm only leverages the side information, meaning that only textual information needs to be stored and retrieved, which is usually cheaper to store/retrieve than the digital data itself. Furthermore, the sparse framework only requires a small portion of the training data maintained in main memory for predicting new instances. Thus, the program could be easily fit into real-world systems and necessitates no out-of-core treatment. As new labeled instances becoming available, real-time updates or online learning should be possible as well.

- In general, tag suggestion is still a complex problem, which can be addressed in many aspects. It should be noticed that our approach does not rely on the actual content of the data, and thus could be considered as a potential complement of the content-based method. In practice, our approach can serve as a component of a large commercial system and boost performance.

5.6.1 Flickr and Youtube Data

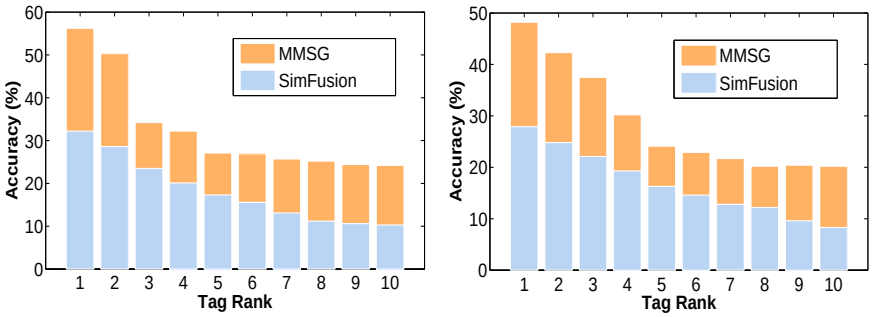
	Side information for an object
Flickr	*title, description, user_comments, category, additional_information*
Youtube	*title, description, comments, category, name_of_related_videos, videos_from_same_person*

Table 5.5. Side information for training the model.

We consider an application of our MMSG algorithm on tag suggestion for real-world rich media data. For this purpose, we collected data from Flickr and Youtube between Sep 15 2007 and Oct 21 2007, using their image and video data respectively. We subscribed to their RSS feeds of the top 30 most popular tags from both sites, including

art, birthday, movie, people, travel and so on. An individual feed contains the basic information of the data such like *title, category, url, description, tags* and etc. We then re-crawled each individual URL in the feeds to get the needed side information. Typical side information of an image/video used in the experiment can be found in Table 5.5. We further eliminate instances that contain too little side information from our experiment. Stemming and stop-words removal were performed to reduce the dimensionality. Overall, the Flickr data contains 22,186 unique items with 68,215 words, whereas Youtube has 2,489 items with 9,761 words. The total number unique tags is 10,341 and 6,724 for Flickr and Youtube respectively.

Algorithm	**Precision**	**Recall**	**F-Score**
Flickr			
SimFusion	31.6%	56.2%	43.1%
MMSG	43.6%	68.4%	57.2%
Youtube			
SimFusion	27.8%	48.0%	36.4%
MMSG	38.2%	54.9%	49.4%



(a) Flickr data (b) Youtube data

Table 5.6. Results on Flickr and Youtube data. The accuracy corresponds to the percentage of objects correctly tagged by the ith tag.

For training, the data is organized into 30 classes using their top-ranked tags. Note that due to the temporal characteristics of tags, we think it is more reasonable to order the data chronically. As such we use the first half for training, and the second half for testing. Overall, the training time is 98 minutes for Flickr and 24 minutes for Youtube. During testing, the top-ranked tags are returned for evaluation against the ground-truth

tags from users. Unfortunately, the previously used LIBSVM cannot handle this problem. Instead, we compared with a recently developed method SimFusion [142] which outperforms other ranking algorithms in several data sets. In additional to the standard metrics *precision*, *recall* and *F-score*, we present the performance for the top-10 suggested tags as suggested in [80].

Figure 5.17 lists several examples with good tagging results. We also present a summary of the experimental results in Table 5.6. Overall, our model is able to boost the tagging performance significantly by comparing with SimFusion. This is efficiently achieved by using only 5% of the training instances. For individual tags, the top-most suggested tag for Flickr is able to achieve a 56.2$\%$ accuracy, compared with a 32.3$\%$ accuracy for SimFusion. Likewise, the top-most tags have 48.2$\%$ accuracy for Youtube, whereas the value is 27.9$\%$ for SimFusion. However, it costs less than one second for our algorithm to make a prediction per case, where SimFusion takes more than 4 seconds.

5.6.2 Limitations of Our Approach

Since our algorithm for tag suggestion only leverages side (textual) information, the limitation is evident. For an image/video without any supporting textual information, our algorithm performs no better than a random guess. However, since textual information is usually cheap and abundant, our algorithm can serve as a good complement for the content-based approach, or an individual component for large-scale commercial systems. In fact, many online image search systems still rely on the surrounding textual information of the objects, including Google Image Search[13].

[13]http://images.google.com/

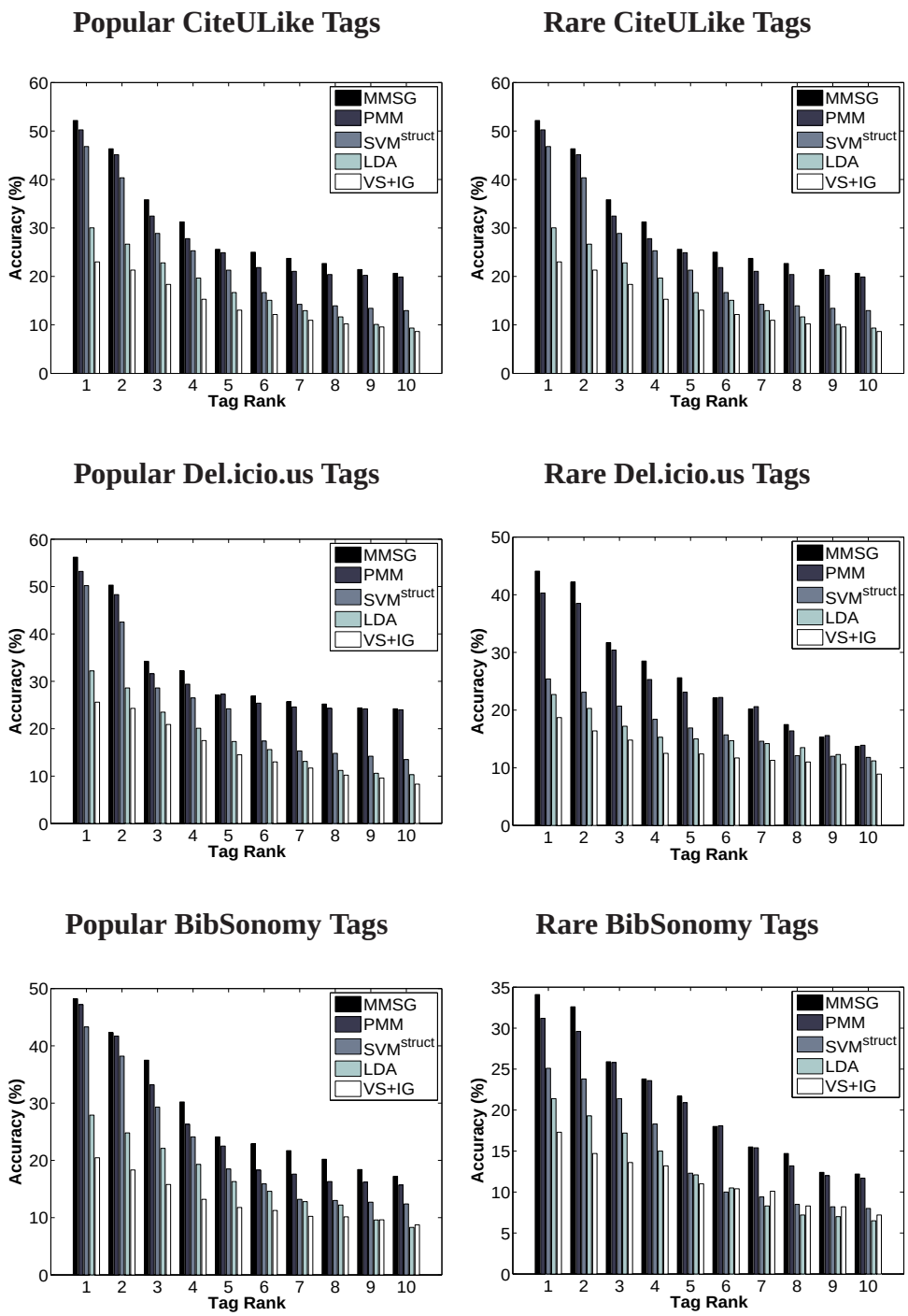


Figure 5.16. Tag suggestion results on popular and rare tags for CiteULike, Delicious and BibSonomy.

Paper Name	Tags	Top User Tags	Our Matched Tags
The PageRank Citation Ranking: Bringing Order to the Web (Larry Page et al.)	135	google, pagerank, search, ranking, web, networks, ir social-networks	*PMM:* **search**, **web**, **pagerank**, **ir** *MMSG:* **google**, **ir** **pagerank**, **ranking** **web**,
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**, **www**,**engine**, **ir** *MMSG:* **google**, **www** **ir**, **web**
ReferralWeb: Combining Social Networks and Collaborative Filtering (Henry Kautz et al.)	88	folksonomy, collaboration, social-networks, networks, filtering, recommender, tagging, social, network	*PMM:* **networks**, **filtering**, **tagging**, **social** *MMSG:* **networks**, **social**, **recommender**, **tagging**,
A Tutorial on Learning With Bayesian Networks (David Heckerman)	78	bayesian, networks, learning, network, statistics, bayes, modeling, graphs, algorithms tutorial,	*PMM:* **bayesian**, **network**, **bayes**, **modeling**, **graphical** *MMSG:* **network**, **bayes**, **networks**, **algorithms**,
Maximizing the Spread of Influence through a Social Network (David Kempe et al.)	73	social, influence, network, socialnetworks, diffusion, research, spread, networking	*PMM:* **network**, **social**, **socialnetworks**, *MMSG:* **network** **social**,**research**,
Authoritative Sources in a Hyperlinked Environment (Jon M. Kleinberg)	47	ranking, hyperlink, web, www, ir, graphs, clustering, hub, authority, hyperlinks search	*PMM:* **web**, **search**, **hyperlink**, *MMSG:* **ir**, **web**, **www**, **ranking**, **search**
Indexing by Latent Semantic Analysis (Scott Deerwester et. al.)	45	lsi, indexing, ir, lsa, semantics, semantic, information-retrieval, latent, language, index	*PMM:* **index**, *MMSG:* **ir**, **indexing**, **index**
The Small-World Phenomenon: An Algorithmic Perspective (Jon M. Kleinberg)	43	small-world, networks, web, social, webgraph, ir, algorithm, graphs, graph power-law, network	*PMM:* **networks**, **web**, **algorithm**, **graphs**, **ir**,**network**, *MMSG:* **network**, **ir**, **web**, **algorithm**, **social**, **graph**

Table 5.7. 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.

japan tokyo wood forrest **architecture acient** monks

japan architecture tokyo scene building buildings photo **acient**

birthday birthdaycake **cakes** candles **people** women **photo**

birthday cakes people dinner **photo** party birthparty

flower tulip tulips red **sky** netherlands **blue** rood

flower flowers colors color nature **red** rose **blue**

newyork skycrapers **usa sky** america **blue buildings glass**

newyork usa blue white **buildings glass sky** street

dana wedding bride **portrait** roads lady **woman** beautiful

portrait woman people photo photography family black tree

fun cat lazy town **summer** kitty gatto **hot** 2005

summer hot cat fun portrait funny cute beauty

pets animals comedy **fun funny** cute kitten kitty pet

animals cats **pets fun** animal **funny** fight music

halo2 halo mod **game weapon** weapons **xbox** mods

game halo2 xbox live 360 **halo weapon** fun

sports sport tennis roger federer top 10 best game

sports sport roger tennis raphi fun match **federer**

travel place **england london** bus tour castle thornbury

travel london england world pictures scotland tour building

auto vehicle drift **bmw** m3 track race roadster

auto car **vehicle bmw** coupe ford fun movie

games gadgets **movie** south **park songs** team mom

comedy **movie park song** fun cartoon fight pet

Figure 5.17. Examples of good tag suggestions. The first row is the object (image/video). The second row corresponds to the user tags. The third row is the recommended tags by our algorithm.

Chapter 6

Topic Discovery: Dynamic Topic Correlation Detection

Topic models have been powerful tools for statistical analysis of text documents [31, 54, 12]. As an example, the latent Dirichlet allocation (LDA) model [12] assumes that documents are mixtures of topics, and topics are probability distribution of words, where topics are shared by all documents. The LDA model further assumes the *exchangeability* of words, i.e., words from each document are drawn independently from a mixture of multinomials. The model uses a Dirichlet prior to draw the topic proportions, so that each document may exhibit different topic distributions. LDA is capable of modeling the semantic relations between words and topics, and using multiple topics to describe document collections. Essentially, LDA, as well as other topic models, can be treated as statistical dimension reduction techniques that reduce the original word representation of documents into topic representation, which is usually of much lower dimension. Successful applications of topic models include discovering author-topic relations in scientific papers [122], disambiguating author names in large collections of documents [115], as well as extensions to image analysis [112].

Since most topic models are *generative models*, scalability is always an issue. With a large number of model parameters, the time for the models to converge is prohibitively long. As one example, we applied LDA to over 700,000 full-text scientific documents. The program took more than one week to finish for a 200-component model. Additionally, these models inevitably suffer from the problem of overfitting. As stated in [88], the variational inference for parameter estimation in LDA is problematic, which failed

to achieve accurate inference for large data sets.

Moreover, since the LDA model treats words exchangeably, it is not suitable to capture the evolution of documents over time. LDA is also unable to model the topic correlations since it assumes topics are drawn from unique priors. These two issues have been addressed by two extensions of LDA, the dynamic topic models [11] and the correlated topic models (CTM) [10], respectively. Nevertheless, neither of these two models is immune to the aforementioned issues.

In this thesis, we present the *dynamic correlated topic models* (DCTM) for analyzing document topics over time. Our model is inspired by the hierarchical Gaussian process latent variable model (HGP-LVM) [76] which has been used for human motion capture. Similar to HGP-LVM, DCTM maps the high-dimensional observed space (words) into low-dimensional latent space (topics), which models the dynamic topic evolution within a corpus. A document corpus considered here is either a conference proceedings or a collection of journal articles. Furthermore, the topic latent space is mapped into a lower-dimensional space which captures the correlations between document corpora. The dynamics of the topics and correlations are captured by a temporal prior, which constructs a hierarchy over the correlation latent space. Unlike generative models, DCTM makes no assumption on word exchangebility. All variables (words, topics and correlations) exhibit dynamics at different time point. Meanwhile, by marginalizing out the mapping parameters rather than the latent variables, DCTM becomes a *non-parametric* model, which shows a much faster model convergency rate than the generative processes. The posterior inference of topic and correlation distributions in DCTM is helpful for discovering the dynamic changes of topic-specific word probabilities, and predicting the evolutions of topics and correlations. The reduced topic space is also helpful for improving the performance of document classification.

6.1 Related Work

(Topic Models) The first well-known topic model was introduced by Deerwester in 1990 [31], the *Latent Semantic Analysis* (LSA). LSA maps high-dimensional data to a lower dimensional representation in a *latent semantic space* that reflects semantic relations between words. The model makes an assumption that K underlying latent topics exist for a specific data set, where the documents can be generated according to these topics

based on probability distributions. Those latent topics are assumed to be approximately the same as document categories, resulting in a significant compression of data in large collections.

Hofmann [54] later presented an alternative to LSA from a statistical perspective, namely Probabilistic Latent Semantic Analysis (PLSA). The model is capable of discovering latent variables with a more solid statistical foundation. PLSA is described as an *aspect model* which can be viewed as a statistical mixture model for documents and words, assuming the existence of hidden factors underlying the co-occurrences among two sets of objects. Specifically, a single word is generated from a single topic while different words may belong to different topics within a document. It is evident that PLSA has a number of parameters that grow linearly with the size of the corpus, resulting a potential for overfitting.

Another generative topic model was introduced by Blei et al. as a Bayesian hierarchical model, which is well-known as the *Latent Dirichlet Allocation* (LDA) [12]. In LDA, each document has its own topic distribution, drawn from a conjugate Dirichlet prior that remains the same for all documents in a collection. The words within that document are then generated by choosing a topic from this distribution. A word is picked from that topic according to the posterior probability of the topic, which is determined by another Dirichlet prior. Inference of parameters and model learning are performed efficiently via variational EM algorithm, since exact inference is intractable in LDA due to the coupling of parameters. Experimental results indicate that LDA has better generalization performance than PLSA. However, as pointed out by Minka [88], variational inference can lead to serious bias and inaccurate learning especially when the data set is large. Thus, Expectation-Propagation was proposed for better inference and learning.

(Correlated Topic Models) An evident limitation of the LDA model attributes to the fact that the topics generated by the multinomial distribution are mutually exclusive. This assumption can be seriously violated in practice. To address this issue, Blei proposed a correlated topic model (CTM) [10], in which the topic proportions are correlated through logistic normal distribution. Mean-field variational methods were employed for parameter estimation. The model was empirically studied by 16,351 *Science* documents over 10 years. A 100-topic CTM shows superior over the traditional LDA model in terms of the complete data likelihood and the predictive perplexity.

(Dynamic Topic Models) For modeling topic trends over time, Blei developed a

time series model, or the dynamic topic models [11], to capture the time evolution of topics in document collections. Rather than using a Dirichlet prior, the dynamic topic model uses a more reasonable Gaussian prior for the topic parameters β, which can capture the evolutions of the topics over the time slices. The topic proportions are drawn from a logistic normal distribution α whose mean values also follow a Gaussian distribution. Two approximate inference methods are developed, namely variational Kalman filtering and wavelet regression. Experiments were performed on a large collection of 30,000 *Science* documents, ranging from 1881 to 1999.

6.2 Gaussian Processes

A Gaussian process (GP) is a *stochastic* process that 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 $k(\mathbf{X}, \mathbf{X}')$. GP models have been used as powerful non-parametric tools for approximate Bayesian learning, with two successful applications on *regression* and *classification* [101]. In regression, the objective is to determine the value of Y_* for a new observation $\mathbf{X}_*$. The GP prior is placed over $y(\mathbf{X})$ of a training set. One first determine the predictive distribution $p(Y_*|\mathbf{Y})$, where $\mathbf{Y}|\mathbf{X}_{train} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$, with $\mathbf{K}$ denoting the multivariate covariance matrix. The value of Y_* is then inferred by using Gaussian prediction methods. Figure 6.1 illustrates an 2-D example of GP. With the uncertainty addressed, GP has shown better performance than other learning methods in the context of classification, including the Support Vector Machines (SVMs) and K-NN [101, 65, 44].

6.2.1 Gaussian Process Latent Variable Models

In Gaussian process latent variable models (GP-LVM), given a set of n observations $\mathbf{Y} \in \mathbb{R}^{n \times d}$, it seeks a probabilistic approach to non-linear dimension reduction by introducing the latent variables $\mathbf{X} \in \mathbb{R}^{n \times q}$, where $q \ll d$, via a parameterized function

$$Y_{ij} = f(\mathbf{X}_i; \mathbf{W}) + \epsilon_i, \tag{6.1}$$

where Y_{ij} corresponds to the entry from the i^{th} row and j^{th} column of the matrix $\mathbf{Y}$, $\mathbf{X}_i$ is the i^{th} row of $\mathbf{X}$ with the noise ϵ_i, and $\mathbf{W}$ is the matrix of parameters to be estimated.

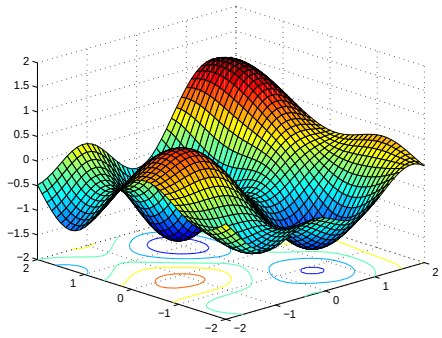


Figure 6.1. An example of two-dimensional input GP framework with an independent noise-free covariance function of each input. For the output latent function f, both dimensions are equally important.

Traditional non-linear probabilistic approach seeks to maximize the likelihood of the model w.r.t. $\mathbf{W}$ by placing prior distribution $p(\mathbf{X})$ over the latent variables $\mathbf{X}$ [127]. Nevertheless, from the Bayesian perspective of view, the parameters $\mathbf{W}$ are trivial and should be marginalized out. Therefore, in GP-LVM, a Gaussian prior is placed on the parameters, i.e., $p(\mathbf{W}) = \prod_{ij} p(w_{ij}) = \prod_{ij} N(w_{ij}|0,1)$. The marginal likelihood can then be optimized w.r.t. the latent variables ($\mathbf{f}$ being the latent functions)

$$p(\mathbf{Y}|\mathbf{X}) = \int p(\mathbf{Y}|\mathbf{f})p(\mathbf{f}|\mathbf{X})d\mathbf{f}. \tag{6.2}$$

It has been shown [75] that this model leads to principal component analysis (PCA) given a *linear* covariance function, or a probabilistic non-linear latent variable model given a *non-linear* covariance function. Consequently, the optimized latent variables $\mathbf{X}$ are capable of reducing the original data into a much lower representation.

6.3 Dynamic Correlated Topic Models

Assume that a set of n document corpora is given, i.e., $\mathbf{D} = \{\mathbf{D}_1, ...\mathbf{D}_n\}$, in which each corpus $\mathbf{D}_i$ contains documents divided into several sets by their timestamps, e.g., the year of publication for scientific documents. We assume that all corpora in our setting share the same timescale, denoted as $[1, ..., T]$, so that each $\mathbf{D}_i = \{\mathbf{D}_{i,1}, ...\mathbf{D}_{i,T}\}$, where $\mathbf{D}_{i,t}$ denotes the set of documents appeared in corpus $\mathbf{D}_i$ at time t. We further assume

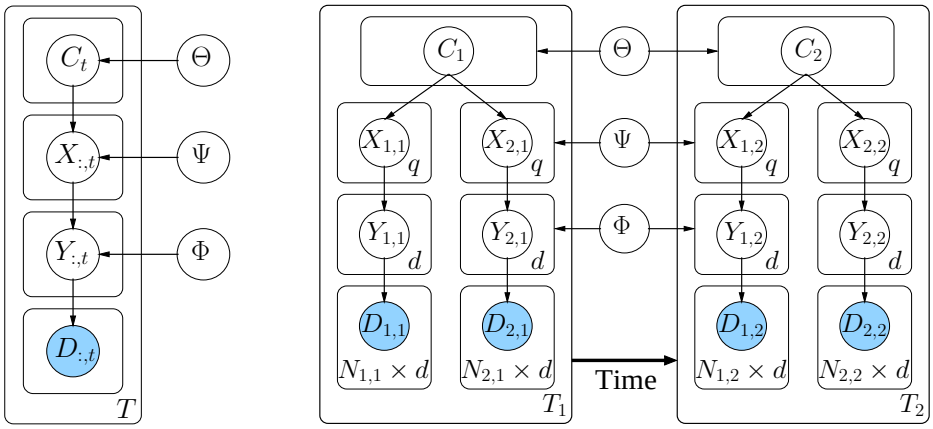
Notations	Explanations
$\mathbf{D}$	a set of document corpora
$\mathbf{D}_i$	a document corpus in $\mathbf{D}$
$\mathbf{D}_{i,t}$	documents appeared in corpus $\mathbf{D}_i$ at time t
$\mathbf{D}_{i,t}^{j,k}$	original count of word k in document corpus i at time t of the j's document
$\mathbf{T}$	number of time points
$\mathbf{Y}$	vector representation of $\mathbf{D}$
$\mathbf{Y}_{i,t}$	word vectors for document corpus i at time t
$\mathbf{Y}_{i,t}^{k}$	normalized count of word k in document corpus i at time t
$\mathbf{X}$	topic vectors
$\tilde{\mathbf{X}}$	normalized topic vectors
$\mathbf{C}$	correlation matrix
$\mathbf{K}$	covariance matrix
$\{\Theta, \Phi, \Psi\}$	parameters for the covariance matrices
$N_{i,t}$	number of documents in $\mathbf{D}_i$ during time t
n	number of document corpora
d	number of words (vocabulary)

Table 6.1. Notations used in this chapter.

that a controlled vocabulary with size d is shared across all $\mathbf{D}_i$ over time, so that each $\mathbf{D}_{i,t}$ can be represented into a matrix, $\mathbf{D}_{i,t} \in \mathbb{R}^{N_{i,t} \times d}$, with $N_{i,t}$ denoting the number of document in $\mathbf{D}_i$ at time t. Note that the value of $N_{i,t}$ may vary for different i and t. Table 6.1 summarizes the notations used in this paper.

As in most topic models, we also assume that a set of q underlying latent topics exist for each $\mathbf{D}_i$, where the number of topics remain the same over time. In order to model the correlations of those topics over time, we need to first discover the latent topics at time t for each corpus $\mathbf{D}_i$, and specify a proper function for calculating the correlations between topics and corpora. Furthermore, we wish to capture the dynamics of the latent spaces. In what follows, we extend the hierarchical Gaussian process latent variable model (HGP-LVM) [76] for dynamic topic correlation detection.

We first represent each $\mathbf{D}_{i,t}$ into a vector form $\mathbf{Y}_{i,t} \in \mathbb{R}^d$ by aggregating the corre-



(a) The DCTM model. (b) An example of two corpora over two time frames.

Figure 6.2. Graphical representation of the DCTM model. Shaded nodes represent observed values. Although looks alike, DCTM differs from generative aspect models (e.g., LDA) fundamentally.

sponding features in all instances

$$Y_{i,t}^{k} = \frac{\sum_{j=1}^{N_{i,t}} (D_{i,t}^{j,k} - \overline{\mathbf{D}_{i,t}^{:,k}})}{\text{var}(\mathbf{D}_{i,t}^{:,k})}, \text{for } k = 1, ..., d, \tag{6.3}$$

where $Y_{i,t}^{k}$ is the summarized value of feature k in $\mathbf{D}_{i,t}$, $D_{i,t}^{j,k}$ is the number of times feature k occurred in the j's document of $\mathbf{D}_{i,t}$, $\overline{\mathbf{D}_{i,t}^{:,k}}$ denotes the mean value of feature k and the denominator computes the variance of feature k. In this way we summarize the contributions of individual documents at a certain time and leave only the relationship between words and time.

In the context of textual documents, each $\mathbf{Y}_i = \{\mathbf{Y}_{i,1}, ..., \mathbf{Y}_{i,T}\}$ has the dimensionality of $T \times d$, with each $Y_{i,t}^{k}$ corresponding to the latent position of word k at time t in $\mathbf{D}_i$, i.e., the position that k appears most probably according to the maximum likelihood estimation. To find q latent topics given $\mathbf{Y} = \{\mathbf{Y}_1, ..., \mathbf{Y}_n\}$, we define n sets of q-dimensional latent variables, with $\mathbf{X}_i = \{\mathbf{X}_{i,1}, ..., \mathbf{X}_{i,T}\} \in \mathbb{R}^{T \times q}, i = 1, ..., n$. We use GP-LVM to model the relations between each pair of $\mathbf{Y}_i$ and $\mathbf{X}_i$,

$$P(\mathbf{Y}_i | \mathbf{X}_i) = \prod_{j=1}^{d} N(\mathbf{Y}_{i,:}^{j} | \mathbf{0}, \mathbf{K}_x^{(i)}). \tag{6.4}$$

Each $\mathbf{Y}_{i,:}^{j}$ is a size T column vector of $\mathbf{Y}_i$, with each element representing the latent position of word j at different time point. $\mathbf{K}_x^{(i)}$ is a kernel covariance matrix of size $T \times T$, where each element is defined by a kernel function, $[\mathbf{K}_x^{(i)}]_{m,n} = k_x(\mathbf{X}_{i,m}, \mathbf{X}_{i,n})$. In this paper, we use the radial basis function (RBF) kernel

$$k_x(\mathbf{X}_{i,m}, \mathbf{X}_{i,n}) = \phi_1 \exp\left(-\frac{\|\mathbf{X}_{i,m} - \mathbf{X}_{i,n}\|^2}{2\phi_2}\right) + \phi_3 \delta_{mn}, \tag{6.5}$$

with $\Phi = \{\phi_1, \phi_2, \phi_3\}$ being the kernel parameters, where δ_{mn} is the delta function that has the value 1 if $m = n$ and 0 otherwise. Our assumption is that given a topic, words follow a zero-mean Gaussian distribution, where the highest probability occurs when a word appears most in a topic. Note that this zero-mean assumption is valid here since the mean values of word frequency have been extracted from $\mathbf{D}$ during the initialization in eq.(6.3). To ensure a well-defined probability distribution of topics at each t, we seek to transform the original $\mathbf{X}_i$ using the multiple logistic function

$$P(\tilde{\mathbf{X}}_i|\mathbf{X}_i) = \frac{\exp(\mathbf{X}_{i,:}^{j})}{\sum_{j'} \exp(\mathbf{X}_{i,:}^{j'})}, \text{ so that } \sum_j P(\tilde{\mathbf{X}}_{i,t}^{j}) = 1. \tag{6.6}$$

In this way the relations between $\mathbf{Y}_i$ and $\mathbf{X}_i$ can be rewritten as the product of two probabilities: $P(\mathbf{Y}_i|\mathbf{X}_i) = P(\mathbf{Y}_i|\tilde{\mathbf{X}}_i)P(\tilde{\mathbf{X}}_i|\mathbf{X}_i)$, with $P(\mathbf{Y}_i|\tilde{\mathbf{X}}_i)$ computed using eq.(6.4).

We then construct a hierarchy by placing a latent variable $\mathbf{C}$ over $\mathbf{X}$, which captures the correlation between each pair of topic sets $\mathbf{X}_i$ and $\mathbf{X}_j$. A proper approach is the Gaussian process where topics that are highly correlated are also close in geometrical interpretation. One approach used in [76] is to construct the concatenation of latent variables $[\mathbf{X}_i \ \mathbf{X}_j]$ and find $\mathbf{C}$ by principal component analysis (PCA). This method works well for high-dimensional problems such as video tracking. An alternative is to use singular value decomposition (SVD) where features (words) are usually of equal importance such as in text analysis. In this paper, we choose SVD for our algorithms, which deal mainly with textual documents.

Furthermore, to capture the correlation dynamically, we place a temporal prior over the element of $\mathbf{C}$,

$$P(\mathbf{C}|\mathbf{t}) = \prod_{i=1}^{n} N(\mathbf{c}_{:,i}|\mathbf{0}, \mathbf{K}_t), \tag{6.7}$$

Algorithm 10 Parameter Optimization for DCTM

1: **Input:** a set of document corpora $\mathbf{D} = \{\mathbf{D}_1, ..., \mathbf{D}_n\}$, number of estimated topics q, number of time frames T, the size of the vocabulary d, initial kernel parameters $\{\Phi, \Theta, \Psi\}$, number of iterations I.
2: Initialize each $\mathbf{Y}_i \in \mathbb{R}^{T\times d}$ for the corresponding $\mathbf{D}_i$ by eq.(6.3),
3: Initialize each latent topic variable set $\mathbf{X}_i \in \mathbb{R}^{T\times q}$ through SVD from each $\mathbf{Y}_i$,
4: Initialize each latent correlation variable set $\mathbf{C}$ through SVD for each pair of $[\mathbf{X}_i\ \mathbf{X}_j]$.
5: **for** $i = 1$ to I
6: **for** $j = 1$ to n
7: optimize each $\{\mathbf{X}_i, \Phi, \Theta\}$ using gradient method
8: **end for**
9: **for** $j = 1$ to n
10: optimize $\{\mathbf{C}, \Psi\}$ using the optimized $\mathbf{X}$
11: **end for**
12: **end for**

where $\mathbf{K}_t$ is the covariance matrix for $\mathbf{t} = \{1, ..., T\}$, which takes the exact form as eq.(6.5) except for the input of $\mathbf{t}$ with a different parameter set $\Theta = \{\theta_1, \theta_2, \theta_3\}$. Fig.6.2 shows the graphical representation of the general DCTM model.

The temporal prior can be combined with equations to marginalize out latent variables $\mathbf{Y}, \mathbf{X}$ and $\mathbf{C}$. The joint probability distribution of the hierarchy can be written as

$$\begin{aligned} P(\mathbf{D}_1, ..., \mathbf{D}_n|\mathbf{t}) &= \int P(\mathbf{D}_1|\mathbf{Y}_1)P(\mathbf{Y}_1|\tilde{\mathbf{X}}_1)P(\tilde{\mathbf{X}}_1|\mathbf{X}_1)\cdots \\ &\times \int P(\mathbf{D}_n|\mathbf{Y}_n)P(\mathbf{Y}_n|\tilde{\mathbf{X}}_n)P(\tilde{\mathbf{X}}_n|\mathbf{X}_n)\cdots \\ &\times \int P(\mathbf{X}_1, ..., \mathbf{X}_n|\mathbf{C})P(\mathbf{C}|\mathbf{t}) \\ &\quad \mathrm{d}\mathbf{C}\mathrm{d}\mathbf{X}_1\cdots\mathrm{d}\mathbf{X}_n\mathrm{d}\tilde{\mathbf{X}}_1\cdots\mathrm{d}\tilde{\mathbf{X}}_n. \end{aligned}$$

However, this marginalization is intractable so that we instead attempt to use a maximum a posterior (MAP) approach to approximating the integration, i.e., to maximize the aggregated Gaussian process log likelihoods [76]

$$\begin{aligned} & \mathcal{L}(\mathbf{D}) \\ \triangleq\ & \log P(\mathbf{D}_1, ..., \mathbf{D}_n|\mathbf{t}) \end{aligned}$$

$$= \sum_{m=1}^{n}(\log P(\mathbf{D}_m|\mathbf{Y}_m) + \log P(\mathbf{Y}_m|\tilde{\mathbf{X}}_m) + \log P(\tilde{\mathbf{X}}_m|\mathbf{X}_m)) + \log P(\mathbf{X}_1, ..., \mathbf{X}_n|\mathbf{C}) + \log P(\mathbf{C}|\mathbf{t}) \tag{6.8}$$

w.r.t. each $\mathbf{X}_m$ and $\mathbf{C}$. The solution of eq.(6.8) can be easily found by gradient search methods.

Practically, when optimizing the latent variables and parameters, we seek a fast converging algorithm which also avoids local minimum. To this point, we initialize each latent variable $\mathbf{X}_i$ and $\mathbf{C}$ by using LSA (which uses singular value decomposition (SVD)) as described in Alg.10. We then minimize $\mathcal{L}$ by optimizing each set of latent variables and their correlations alternatively. A maximum of 100 steps have been fixed in advanced for the optimization. The step of each gradient search was empirically set to be 10^{-6}.

6.3.1 Smoothing

In eq.(6.8), $\mathcal{L}_1$ corresponds to the estimation of the learned latent positions, while all terms in $\mathcal{L}_2$ sum up to the MAP estimation of the dynamic correlations. It can be observed that unsmooth correlations usually result in high values which are not desirable. However, due to the effect of summation of $\mathcal{L}_1$ which involves a large number of instances, the value of $\mathcal{L}_2$ is usually underestimated in practice.

Therefore, to encourage smoothness of $\mathcal{L}(\mathbf{D})$ by penalizing the correlations and the positions on the same granularity, we seek to balance the contribution of both terms by raising the dynamics density function to the ratio of their dimensions, $i.e., \pi = d/q$. Thus the terms corresponding to the dynamics are rescaled in eq.(6.8) [131]:

$$\pi \left(\frac{q}{2} \log |\mathbf{K}_c| + \log |\mathbf{K}_t| - \frac{1}{2} \sum_{i=1}^{q} \mathbf{X}_{:,i}^T \mathbf{K}_c \mathbf{X}_{:,i} - \frac{1}{2} \mathbf{C}_{:,i}^T \mathbf{K}_t \mathbf{C}_{:,i} \right), \tag{6.9}$$

which leads to a simple and balanced learning function for the model. Empirically, this has shown to be effective for Gaussian process-based 3D people tracking [131]. However, the theoretical best choice of the scaling factor is still subject to future work.

6.3.2 Inference and Predictions

(Posterior Inference) Since we made an assumption on the conditional distribution of $P(\mathbf{Y}_i|\mathbf{X}_i)$ by eq.(6.4), the topic-specific word distributions $P(\mathbf{Y}_i|\mathbf{X}_i)$ can not be straightforwardly inferred from the model. Instead, we can make inference on the word-specific topic probabilities, to monitor the change of words over time. First, inference can be made for $P(\mathbf{X}_i|\mathbf{Y}_i)$ by using the Bayes rule,

$$P(\mathbf{X}_i|\mathbf{Y}_i) \propto P(\mathbf{Y}_i|\mathbf{X}_i)P(\mathbf{X}_i), \tag{6.10}$$

so that we can get the word-specific topic probabilities at a certain time t, $\mathbf{X}_{i,t}$, by marginalizing out all latent variables $\mathbf{X}_i$ except for $\mathbf{X}_{i,t}$ (denoted as $\mathbf{X}_{-i,t}$):

$$\begin{aligned} P(\mathbf{X}_{i,t}|\mathbf{Y}_{i,t}) &= \int P(\mathbf{X}_i|\mathbf{Y}_{i,t})d\mathbf{X}_{-i,t} \\ &\propto \int P(\mathbf{Y}_{i,t}|\mathbf{X}_i)P(\mathbf{X}_i)d\mathbf{X}_{-i,t}, \end{aligned} \tag{6.11}$$

We use importance sampling [121] to estimate the integral.

(Document Classification) Meanwhile, remember that our model (as well as other topic models) is essentially a method for dimensionality reduction, it would then be interesting to observe how much performance gain/loss will be fulfilled by using the topic feature representation comparing to the original word features. One way to study this is by analyzing the performance of document classification.

For traditional binary classification tasks, it is required to have a vector of features representing each class. Here we treat each document corpus as one class. Since the topic distributions are different at different time points for a specific corpus (class), one reasonable approach to summarizing topic features is by marginalizing out the temporal prior,

$$p(\mathbf{X}^k_{i,:}) = \sum_{t=1}^{T} p(\mathbf{X}^k_{i,t}) * p(t), \text{ for } k = 1, ..., q. \tag{6.12}$$

By assuming that the document-level observations are equally spaced, we are actually taking the MAP values of $p(\mathbf{X}^k_{i,:})$ as the normalized features.

A more challenging classification task can also be carried out by classifying documents into a specific time point (e.g., year). For n corpora with T timescales, this

problem requires an $n \times T$ multi-class classification algorithm.

(Topic $\&$ Correlation Predictions) We show the predictive power of DCTM by proposing two prediction methods, using regression analysis and Gaussian processes.

Besides between-topic correlations, the autocorrelations (AC) within each topic can also be computed. Specifically, we can model the autocorrelations of a set of topic distributions over time $\mathbf{X}_i = \{\mathbf{X}_{i,1}, ..., \mathbf{X}_{i,T}\}$ by

$$\begin{aligned} P(AC(l)|\mathbf{X}_i) &= \frac{\sum_{j=1}^{T-l}(\mathbf{X}_{i,j} - \bar{\mathbf{X}}_i) * (\mathbf{X}_{i,j+l} - \bar{\mathbf{X}}_i)}{\sum_{j=1}^{T}(\mathbf{X}_{i,j} - \bar{\mathbf{X}}_i)^2}, \\ \text{for } l &= 1, ..., T-1, \end{aligned} \tag{6.13}$$

where $AC(l)$ corresponds to the lag-l autocorrelation function and $\bar{\mathbf{X}}_i$ takes the mean value of $\mathbf{X}_i$. A typical autocorrelation generally decreases with the increase of lag, indicating that only the first few lags demonstrate significantly non-zero. The values of the lags are often used to discover repeating patterns in the data such as the topic distributions during a certain period of time. Mathematically, the values can be used as the coefficient for the regression function.

Meanwhile, due to conjugation, the posterior probabilities of topics and correlations are also Gaussian. We thus propose a simple Gaussian dynamic prediction model [11, 134] for the next time point $t+1$:

$$\begin{aligned} \mathbf{X}_{i,t+1}|\mathbf{X}_{i,t} &\sim N(\mu(\mathbf{X}_{i,t}), \sigma^2(\mathbf{X}_{i,t})\mathbf{I}), \text{ where} \\ \mu(\mathbf{X}_{i,t}) &= \mathbf{K}(\mathbf{X}_i, \mathbf{X}_{i,t})^T \mathbf{K}_X^{-1} \mathbf{X}_i, \\ \sigma^2(\mathbf{X}_{i,t}) &= \mathbf{K}_X(\mathbf{X}_i, \mathbf{X}_i) - \mathbf{K}(\mathbf{X}_i, \mathbf{X}_{i,t})^T \\ &\quad \mathbf{K}_X^{-1} \mathbf{K}(\mathbf{X}_i, \mathbf{X}_{i,t}). \end{aligned} \tag{6.14}$$

From a standard Gaussian process perspective, making predictions require averaging all parameter values, with their associated posterior weights. However, this approach is computationally demanding which involves expensive Monte Carlo sampling methods. Thus, what we suggested here can be considered as a shortcut of achieving roughly the same predictive power, with much less computational complexity.

6.4 Experiments

We assess our model with both synthetic data and a real-world large corpus. For the synthetic data, we compare the correlations and the likelihood with a simple SVD method to give a quick snapshot of the model. We then compare our model to LDA by using a large-scale real-world data set.

6.4.1 Simulated Data

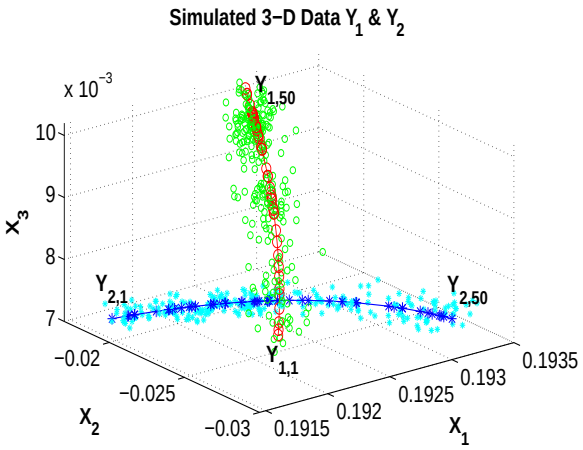
The simulated data set we created comprises of two groups $(\mathbf{D}_1, \mathbf{D}_2)$ of 3-D data, divided into 50 time frames. At each time t, we randomly create several points that follows a normal distribution $N(\boldsymbol{\mu}|\mathbf{Y}_{it}, \sigma^2 I)$, for each $\mathbf{D}_i$ respectively. The vector $\mathbf{Y}_i$ is a randomly chosen polynomial function. The geometric distance between $\mathbf{Y}_1$ and $\mathbf{Y}_2$ reaches the minimum and the maximum at the 18^{th} and 50^{th} time frame, respectively. As we want to use a lower dimension to capture and visualize the correlation between these two random variables $\mathbf{Y}$, we choose the dimension of $\mathbf{X}$ to be 2, e.g., $q = 2$.

Figure 6.3 illustrates the results after 14 iterations, where (a) shows the original data points as well as the polynomial functions $\mathbf{Y}_1$ and $\mathbf{Y}_2$. It can be seen that the original correlation is almost random after simple SVD. We expect to see a strong correlation around the 18^{th} time frame, which gradually decreases until the end. This was indeed observed after the model converges. The correlation can well interpret the geometric distance between $\mathbf{Y}_1$ and $\mathbf{Y}_2$. Meanwhile, DCTM reaches the optimal result quickly in this case, where the log likelihood converges to -175.2326 after the just 14 iterations.

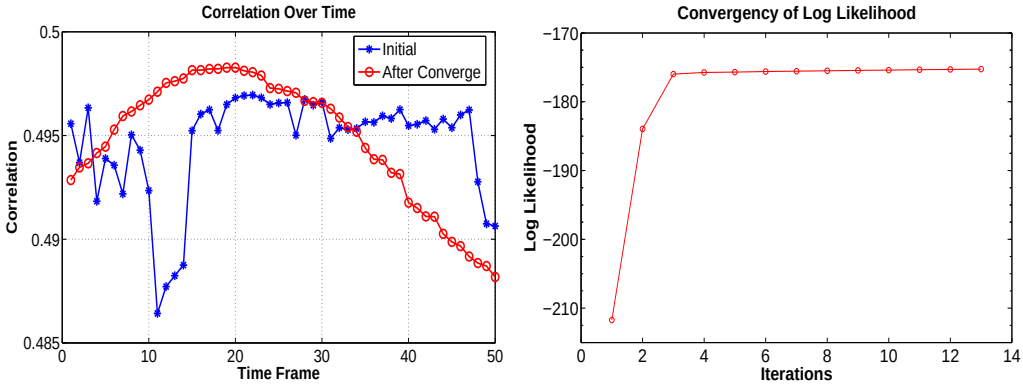
6.4.2 CiteSeer Scientific Documents

We further analyzed a subset of scientific documents from the CiteSeer[1] digital library. We crawled top 80 most prolific venues according to the CiteSeer impact factor, divided them into 18 time frames by year from 1988 to 2005. The total number of documents in our experiment is 268,231. For efficiency consideration, we used metadata (title, abstract and keywords) and introductions as the document contents. The data set consists of 6,530,000 unique words. We applied information gain to reduce the dimensionality and resulted in top 24,351 words. We ran a series of experiments on different numbers

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



(a) Simulated Data.



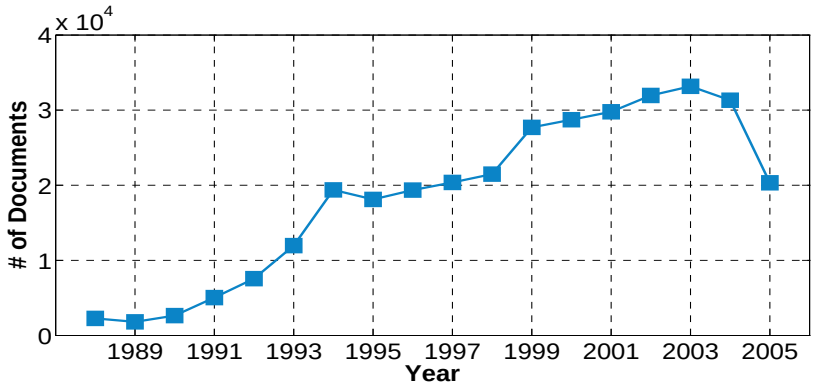
(b) The change of the correlation. (c) The change of the log likelihood.

Figure 6.3. Simulated Results.

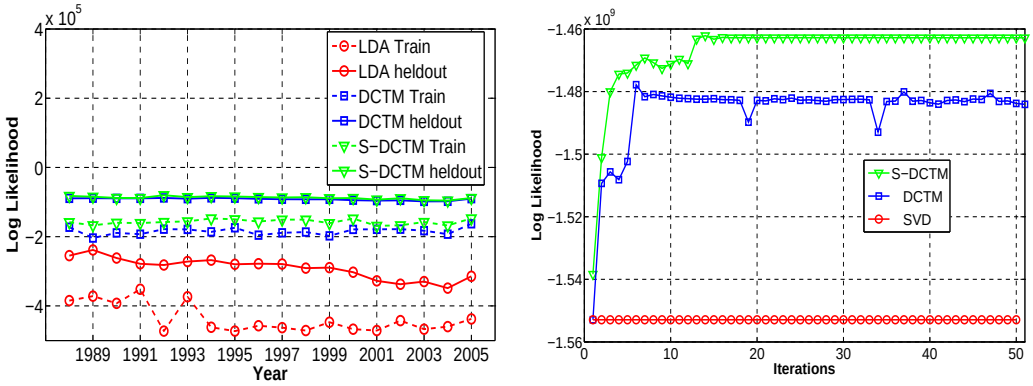
of topics from 10 to 200. Due to space consideration, we only show the result with 25 topics. To investigate the change of the log likelihood in eq.(6.8), we split the data into 90% for modeling (training), and use the rest 10% for testing the model with optimized parameters. Figure 6.4 (b) demonstrates the log likelihood of these two data sets. It is clearly that DCTM shows better fit than LDA for documents across all years. Meanwhile, the smoothing method we used for DCTM (S-DCTM) does show a positive effect on refining the model, by showing higher likelihood than DCTM. It can also be observed that with the increased number of documents by year (Figure 6.4 (a)), LDA generally shows worse performance with lower likelihood. However, this has minor effect on our models, which supports our argument that DCTM does not suffer from overfitting of

	LDA	DCTM	S-DCTM
Training log likelihood (10^6)	-2.6482	-1.8447	**-1.5736**
Held-out log likelihood (10^5)	-1.9073	-0.9166	**-0.8740**

Table 6.2. Results of the CiteSeer data set.



(a) Number of documents per year.



(b) Training and held-out log likelihood. (c) Convergency of the log likelihood.

Figure 6.4. Results of log likelihood on the CiteSeer data set.

large data sets. It can also been seen from Figure 6.4 (c) that the convergency of DCTM is fast. The log likelihood converges after merely 10 iterations(cf Table 6.2). By comparison, DCTM uses merely 4 hours to train and optimize the model, which takes LDA more than four days to finish.

Figure 6.5 presents some results for the SIGMOD corpus. The top figure shows the top 6 venues which have the highest correlations with SIGMOD for each year. It can be observed from the list that most top-ranked venues from the posterior infer-

ence are database-related venues, which share lots of interests with SIGMOD. Meanwhile, the research trends of SIGMOD can also be observed. While maintaining a steady and strong correlations with traditional database-related venues like ICDE, PODS and VLDB, the correlations of SIGMOD with application-oriented venues are decreasing gradually, e.g., DEXA. Instead, SIGMOD correlates more with data-mining and information-retrieval venues like WWW, AAAI and ICDM (cf middle figure).

Furthermore, we can monitor the trends of specific keywords. Using eq.(6.11), we further marginalize out all the topics at the same time frame to get a mean probability of these keywords. The middle-right figure demonstrates several example keywords. It can be discovered that topics like *query* and *security* have retained their popularity during the last two decades. Meanwhile, database research has shifted its focus from traditional topics like *storage* and *recovery* to more promising areas like *mining* and *relational databases*.

The bottom figure depicts two highly-correlated topics in SIGMOD at three different years. The words are sampled from the distribution with probabilities directly computed from the prior. Based on our knowledge, the first topic focuses on *algebra* and *association rules*, with *mining* gradually gets more attention. The second topic addresses *users* and *programming*, which has shifted to *web* applications recently.

6.4.2.1 Classification Performance

We performed both binary and multi-class classifications on a subset of the CiteSeer data. For binary classification, we use two venues, SIGIR and ICML, as two classes. For multi-class classification, we simply divide SIGIR into 18 classes, according to the publication timestamp by year (1988 – 2005). The topic features generated by DCTM are compared with original word features for evaluation. The SIGIR venue contains a total of 572 documents, while ICML contains 854 documents. The total vocabulary in this case is 4,523. We trained a 40-topic model using DCTM, which reduces the feature space by 99%. The original word features are processed using the TF-IDF representation and L-1 normalization. We employed the support vector machine (SVM)[2] as our classifier. The parameters are estimated by 10-fold cross validation. The linear kernel is chosen. We evaluated the classification performance by using $10\%, 20\%$ and 50% data for training.

[2] SVM^{light} is used. http://svmlight.joachims.org/

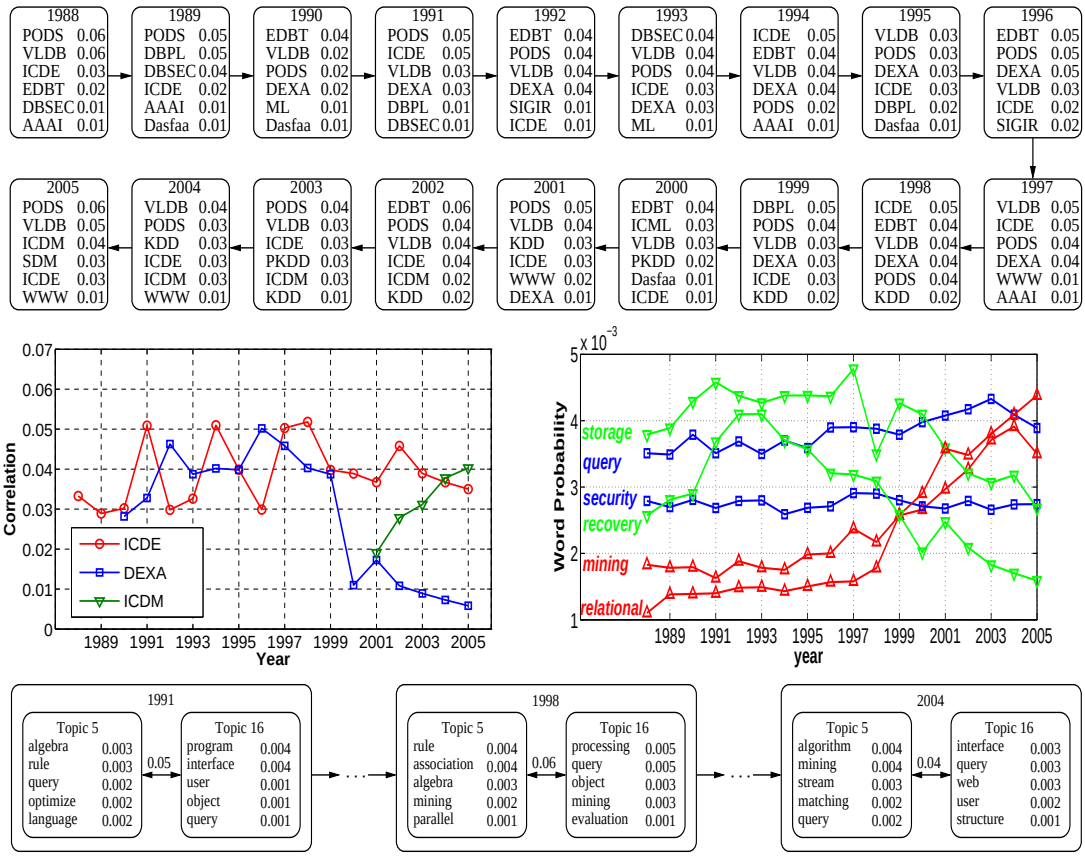
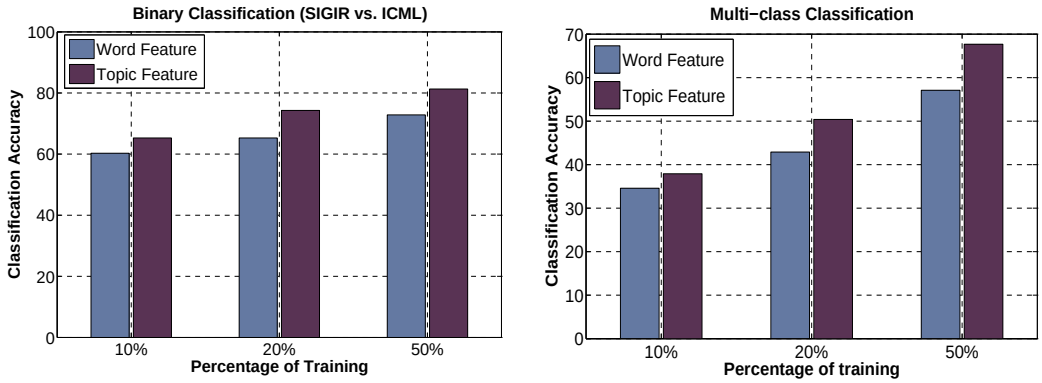


Figure 6.5. Results of SIGMOD corpus results. (**Top**) Top-ranked correlated venues with SIGMOD, from the year 1988 to 2005. (Middle left) The change of correlation as a function of time of three example venues. (Middle right) The posterior probability of words as a function of time by marginalizing out the topics. (Bottom) Two correlated topics with associated word probabilities at different time. Note that 0 correlations are removed from this graph. The data of ICDM is only available after 2001.

Figure 6.6 illustrates the classification performance. Clearly, topic features improve the accuracy for both cases. Especially in the multi-class case, the features generated by DCTM gain a 20% improvement over the word features when using 50% training data.

6.4.2.2 Prediction Performance

Finally, we assessed the predictive powerful of our model. The objective is to predict the correlations between SIGMOD and 10 other venues. We trained our model using data containing the first 16 years (1988–2003), and tested on the year of 2004 and 2005. Both autocorrelation regression (ACR) and mean prediction (MP) are tested. Least square



(a) Binary classification. (b) Multi-class classification.

Figure 6.6. Classification performance on a subsect of 1,326 CiteSeer documents. S-DCTM (topic features) outperforms TF-IDF (word features).

error is applied to measure the performance of the prediction. We also made a simple comparison to the dynamic LDA models [11] by using the variational wavelet regression (VWR). Table 6.3 lists the results. Both of our methods outperform VWR on all venues. Meanwhile, the MP method clearly has an advantage over ACR in most scenarios. As an example of the correlation between SIGMOD and AAAI (black solid line) shown in Fig. 6.7, MP shows the predictive distributions by a mean (blue solid line) and two standard deviations (dash green lines), which can well fit into the dynamic correlations and make reasonable predictions on the trends. Note that the standard deviations of the last two test points are slightly larger than previous ones, showing possible divergency from the data.

6.4.3 Discussion

As it can be seen, the comparison of DCTM and LDA did not go through *perplexity* as well as other metrics. This is because these two models differ from each other fundamentally, it is difficult to find a common metric for evaluation. As explained, our model is able to make inferences on corpus-level correlations, which is a clear advantage over LDA. Nevertheless, DCTM aggregated the contributions of individual documents so that the document-topic relationship can not be retrieved, which is achievable in LDA.

The inferences of our model and LDA are also quite different. In LDA, top-ranked words for each topic can be discovered by the posterior inference of topic-specific word

Venue Name	ACR	MP	VWR
AAAI	13.203	10.557	15.625
DEXA	25.445	17.883	25.982
ICDM	24.892	20.186	28.005
ICDE	17.241	15.936	24.175
ICML	45.209	45.317	47.194
KDD	33.004	27.508	34.175
PKDD	20.705	18.335	23.825
PODS	27.854	24.692	34.215
SIGIR	27.252	28.406	34.112
VLDB	37.225	36.901	45.229
mean	**27.203**	**24.572**	**31.254**

Table 6.3. Correlation prediction results of the SIGMOD venue. Lower least square errors indicate better performance.

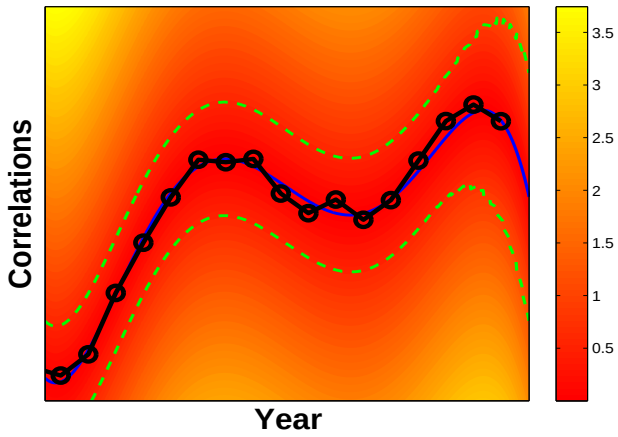


Figure 6.7. Example of prediction performance of the mean prediction method on the correlations between SIGMOD and AAAI. Lower values indicate better predictive performance, shown in darker colors.

probabilities. This is usually used for *naming* topics. However, this approach is very subjective and often requires a good domain knowledge for judgment. Comparatively, our model monitors topic probabilities given a specific word, by marginalizing out the topics at the same time, we can directly observe the popularity of that word at a certain time.

The most controversial part of our model is the initialization step. To minimize the computational cost, we initialized our model by SVD, which is a linear dimensionality reduction method. This method is known to have issues when applied to LSA [54],

though it seems to work well for Gaussian-based models when applied to human motion caption [76]. Besides, due to the restriction of matrix decomposition in SVD, monitoring a large number of topics in a fixed timescale becomes unachievable. The model needs to be re-trained once we change the number of topics.

Chapter 7

Conclusions

In this thesis we addressed four research topics in text mining, i.e., text classification, text retrieval, text recommendation and topic discovery. We approached the research issues within these topics by using both theoretical analysis as well as empirical studies.

For text classification, we proposed the use of entity extraction for reducing the dimensionality of feature spaces. We used noun phrases as features rather than the traditional bag-of-words representation. We then introduced a novel use of collaborative filtering technique for augmenting the feature spaces with relevant information. The collaborative filtering algorithm successfully augmented the feature space for classification, resulting in an accuracy improvement of the baseline, non-CF approach and the Information Gain feature selection method.

Moreover, we also seek to improve the performance of the traditional K-Nearest Neighbor classifier. We presented two approaches namely locally informative KNN (LI-KNN) and globally informative KNN (GI-KNN) to extending KNN method. Informativeness was introduced as a new concept that is useful as a query-based distance metric. LI-KNN applied this to select the most informative points and predict the label of a query point based on the most numerous class with the neighbors; GI-KNN found the globally informative points by learning a weight vector from the training points. Experiments that compared the performance between our methods and KNN, DANN, LMNN, SVM and Boosting indicated that our approaches were less sensitive to the change of parameters than KNN and DANN, meanwhile yielded comparable results to SVM and Boosting. Classification performance on UCI benchmark corpus, CiteSeer text data, and images suggests that our algorithms were application-independent and

could possibly be improved and extended to diverse machine learning areas.

For text retrieval, we aimed at accurately extracting named entities from webpages and scientific documents. We proposed a novel framework for unsupervised name disambiguation by leveraging graphical Bayesian models and a hierarchical clustering method. We presented an effective two-stage approach to disambiguate names. In the first stage, two novel topic-based models are proposed by extending two hierarchical Bayesian text models, namely Probabilistic Latent Semantic Analysis (PLSA) and Latent Dirichlet Allocation (LDA). Our models explicitly introduce a new variable for persons and learn the distribution of topics with regard to persons and words. After learning an initial model, the topic distributions are treated as feature sets and names are disambiguated by leveraging a hierarchical agglomerative clustering method. Our approach was demonstrated to be more effective than other unsupervised learning methods including spectral clustering and DBSCAN. A series of experiments were performed that verified the advantages of our approach on both web data and scientific documents. Although our primary focus of this framework is on person name disambiguation, our general approach should be equally applicable to other entity disambiguation domains. Potential applications include noun phrases disambiguation, e.g., “tiger” as an animal, “tiger” as a golf player, “tiger” the baseball team, “tiger” the operating system or “tiger” for the new Java version. And of course, it would be interesting to see whether our framework can be applied to automatic image annotation and other fields.

On the text recommendation research, we mainly applied machine learning methods for recommending tags to social bookmark websites. From our empirical observation of two large-scale data sets, we first argued that the user-centered approach for tag recommendation is not very effective in practice. Consequently, we proposed two novel document-centered approaches that are capable of making effective and efficient tag recommendations in real scenarios. The first graph-based method represented the tagged data into two bipartite graphs of (document, tag) and (document, word), then found document topics by leveraging graph partitioning algorithms. The second prototype-based method aimed 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 were ranked within each topic cluster/class by a novel ranking method. Recommendations were 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 examined the quality of tag recommendation as well as the efficiency of our recommendation algorithms. The results suggested 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.

Finally, we performed topic discovery on scientific documents. We introduced dynamic correlated topic models (DCTM) for analyzing discrete data over time. This model was inspired by the hierarchical Gaussian process latent variable models (GP-LVM). DCTM is essentially a non-linear dimension reduction technique which is capable of (1) detecting topic evolution within a document corpus by associating the original word feature space with a low-dimensional latent topic space, (2) discovering topic correlations between document corpora by constructing a hierarchy over the latent topic space and (3) monitoring topic and correlation trends dynamically by placing a temporal prior over the correlations, where the inputs are discrete time frames. By marginalizing model parameters rather than the latent variables, DCTM exhibited a non-parametric characteristic which is often desirable for large-scale text data. Unlike generative aspect models such like LDA, DCTM demonstrated a much faster converging rate with better model fitting to the data. We empirically assessed our approach using 268,231 scientific documents, from the year 1988 to 2005. Posterior inferences suggested that DCTM is useful for capturing topic and correlation dynamics, predicting their trends, and improving classification performance using the reduced feature space.

As a conclusion of this thesis, we will now try to answer the research questions we proposed in the first chapter:

1. **Can dimension reduction techniques boost the performance of text classification?**

   The short answer is: **YES**. Because unlike image or video classification, the feature space of text representation is usually very sparse. Many features, i.e., words, only appear very few times in the corpus. Or, on the other hand, appear many times in every single document. Statistically speaking, including these features in the feature space will not gain more information for the classifier but increase the computational cost instead. Moreover, different words in same documents may have identical or similar meanings, e.g., *increase* and *raise*. Thus, including

these redundant features is unlikely to increase the performance of the classifier either. In statistics, dimension reduction usually involves transforming the feature space to a lower coordinate space according to the directions of the first few principal components. However, for text classification, due to the extreme high dimensionality, feature selection methods are often more preferable to principal component analysis. Many empirical and theoretical analysis have shown that feature selection can greatly reduce the computational complexity while increase the classification performance [47, 13]. Recent research also shows that document-specific feature selection usually outperforms general feature selection methods for specific classifiers [68, 69]. Therefore, we believe that in general, dimension reduction can boost the performance of text classification.

2. **Is it possible to overcome the "curse of dimensionality" for the K-nearest neighbor classifier?** The short answer is: **YES**. Because KNN is a local-search classifier that only looks at its nearest neighbors, it then inevitably suffers from the curse of dimensionality problem where in very high-dimensional space, all features look the same. This is because most KNN classifiers use Euclidean distance as the distance metric which treats each dimension equally well. To address this issue, dynamic programming and approximate search have been applied in literature. We also proposed a new metric to measure the informativeness of the neighbors [116], which selects the best candidates that are close to the points within the same class while far away from points in other classes. More recently, the idea of learning a distance metric from a training set has been introduced [140], which aims at learning an optimal Mahanobolis distance that minimizes the classification error of a training set. A similar idea is also incorporated into our approach that learns the best weights of all neighbors by using a boosting like iterative learning algorithm [116]. Overall, the supervised approaches often outperform unsupervised methods during KNN classification.

3. **Are unsupervised learning algorithms comparable with supervised learning methods for retrieving correct name entities (i.e., name disambiguation)?**

   Our answer is: **DEFINITELY YES**. The ambiguity of names is very common on both webpages and scientific documents. Different persons may share the same name (or name initials) while one person may be referred to by different

name variants. Previous supervised learning approaches extract as features the side-information besides the names themselves [57]. These side-information includes addresses, phone-numbers, affiliations, co-author information and so on, which often requires careful human-labeling and sophisticated extracting algorithms. Unsupervised approaches, on the other hand, do not need these steps. Instead, they leverage the underlying relationships between each person, or between persons and documents. e.g., comparing the similarity between a person's social network [84], finding the topic distribution based on a person's publication record [114] and so on. These approaches, while successfully minimizing the cost and error of human-labeling, do not always result in decreased performance. Surprisingly, some of them even outperform the supervised approaches when leveraging sophisticated clustering algorithms [115].

4. **Are computerized text recommendation algorithms suitable for Web2.0 applications in recommending social bookmarks to users?**

   Our experience is: **YES**. Though some still question the use of computer algorithms to generate tags for social bookmarking services [49], a number of machine learning frameworks have been proposed to address the problem of automatic tag recommendation for both text and digital data on the web [22, 7, 80, 119]. Recent work has also shown the effectiveness of leveraging user tags to improve language models [144]. Our research shows that users indeed re-use the tags generated by other users while also takes the recommendations from computerized suggestions [37]. It should be noted, however, text recommendation for social bookmarks is more challenging than other applications. Since the user-generated tags are involving while new tags generated every day, the algorithms are required to adjust dynamically by learning new tags and user interests. Furthermore, with the increase of the data everyday, efficiency should also be an important factor when designing a recommendation algorithm for web-size applications.

5. **Instead of simply breaking a document collection into static topics, is it possible to model the dynamic change of topics within the document collection during a range of time? Moreover, can we monitor the topic correlations between several document collections dynamically?**

   Two graphical topic models can solve this question. Dynamic topic model (DTM)

[11] and Correlated topic model (CTM) [10]. Both models are extended from the latent Dirichlet allocation (LDA) model that represents documents into a mixture of topics. To model the topic trends *and* the correlations together, we proposed a dynamic correlated topic model (DCTM) which is extended from the hierarchical Gaussian process latent variable models (GP-LVM). However, since modeling of trends and correlations is generally a hard task, the framework we proposed also has flaws. Comparing to LDA, CTM and DTM, our model is unable to make document-level inference at all, i.e., words are directly connected with topics rather than documents. Thus, the prediction methods require optimization to prevent overfitting from large-scale of data.

In my personal opinion, future work of text mining should mainly focus on the scalability issue. The speed of the Internet growth is already beyond the control of human beings. Concequently, the amount of text data that is potentially available for computers to use is virtually unlimited and cost-free. A rule of thumb in machine learning is, the more training data is presented, the better performance the algorithm can achieve. A naive Bayes model learnt from 10 million labeled training data will most likely outperform a support vector machine classifier built on 1 thousand training data, for the same classification task. As a result, rather than searching for more sophisticated models for text mining, researchers should really pay more attention on improving the scalability of the learning algorithms during both training and prediction stages. Because a lot of sophisticated models used by today's researchers don't really scale well enough. e.g., the LDA model [12] for topic discovery, the SVM [27] and Boosting [105] classifiers for text classification, the conditional random fields (CRF) framework [73] for text segmentation and so on. Fortunately, some researchers have already envisioned this situation so that some promising methods have been recently introduced to the text mining field, to particularly address the scalability issue [128, 61, 132].

Bibliography

[1] Trec. text retrieval conference. http://trec.nist.gov/.

[2] R. Arora and B. Ravindran. Latent dirichlet allocation based multi-document summarization. In *AND '08: Proceedings of the second workshop on Analytics for noisy unstructured text data*, pages 91–97, New York, NY, USA, 2008. ACM.

[3] V. Athitsos, J. Alon, and S. Sclaroff. Efficient nearest neighbor classification using a cascade of approximate similarity measures. In *CVPR '05*, pages 486–493, Washington, DC, USA, 2005. IEEE Computer Society.

[4] V. Athitsos and S. Sclaroff. Boosting nearest neighbor classi.ers for multiclass recognition. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, page 45, Washington, DC, USA, 2005. IEEE Computer Society.

[5] L. Baoli, L. Qin, and Y. Shiwen. An adaptive k-nearest neighbor text categorization strategy. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(4):215–226, 2004.

[6] J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 9, New York, NY, USA, 2004. ACM Press.

[7] 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.

[8] R. Bekkerman and A. McCallum. Disambiguating web appearances of people in a social network. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 463–470, New York, NY, USA, 2005.

[9] I. Bhattacharya and L. Getoor. A latent dirichlet model for unsupervised entity resolution. In *SDM*, 2006.

[10] D. Blei and J. Lafferty. Correlated topic models. *Advances in Neural Information Processing Systems 18*, pages 147–154, 2006.

[11] D. M. Blei and J. D. Lafferty. Dynamic topic models. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 113–120, New York, NY, USA, 2006. ACM.

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

[13] A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artif. Intell.*, 97(1-2):245–271, 1997.

[14] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, New York, NY, USA, 1992. ACM Press.

[15] 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.

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

[17] A. Z. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust classification of rare queries using web knowledge. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 231–238, New York, NY, USA, 2007. ACM.

[18] C. H. Brooks and N. Montanez. Improved annotation of the blogosphere via autotagging and hierarchical clustering. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 625–632, New York, NY, USA, 2006. ACM Press.

[19] S. Büttcher and C. L. A. Clarke. A document-centric approach to static index pruning in text retrieval systems. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 182–189, New York, NY, USA, 2006. ACM.

[20] L. Cai and T. Hofmann. Text categorization by boosting automatically extracted concepts. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 182–189, New York, NY, USA, 2003. ACM Press.

[21] C. D. Charalambous. Maximum likelihood parameter estimation from incomplete data via the sensitivity equations: The continuous-time case. *IEEETAC*, 45:928–934, 2000.

[22] 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.

[23] M. Collins, R. E. Schapire, and Y. Singer. Logistic regression, adaboost and bregman distances. *Mach. Learn.*, 48(1-3):253–285, 2002.

[24] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[25] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2:265–292, 2002.

[26] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. In *AAAI '98/IAAI '98*, pages 509–516, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence.

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

[28] H. Cui, P. B. Heidorn, and H. Zhang. An approach to automatic classification of text for information retrieval. In *JCDL '02: Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 96–97, New York, NY, USA, 2002. ACM.

[29] A. Dasgupta, P. Drineas, B. Harb, V. Josifovski, and M. W. Mahoney. Feature selection methods for text classification. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 230–239, New York, NY, USA, 2007. ACM.

[30] A. David and B. Lerner. Support vector machine-based image classification for genetic syndrome diagnosis. *Pattern Recogn. Lett.*, 26(8):1029–1038, 2005.

[31] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[32] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[33] I. S. Dhillon, S. Mallela, and R. Kumar. Enhanced word clustering for hierarchical text classification. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 191–200, New York, NY, USA, 2002. ACM Press.

[34] N. Dimitrova and F. Golshani. Motion recovery for video content classification. *ACM Trans. Inf. Syst.*, 13(4):408–439, 1995.

[35] C. Domeniconi, J. Peng, and D. Gunopulos. Locally adaptive metric nearest-neighbor classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(9):1281–1285, 2002.

[36] T. Eliassi-Rad and J. W. Shavlik. A theory-refinement approach to information extraction. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 130–137, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[37] 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.

[38] 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.

[39] G. Forman. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3:1289–1305, 2003.

[40] D. Fragoudis, D. Meretakis, and S. Likothanassis. Integrating feature and instance selection for text classification. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 501–506, New York, NY, USA, 2002. ACM.

[41] J. Friedman. Flexible metric nearest neighbor classification. technical report 113, stanford university statistics department, 1994.

[42] R. Ghani. Combining labeled and unlabeled data for multiclass text categorization. In *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, pages 187–194, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[43] C. L. Giles, K. D. Bollacker, and S. Lawrence. Citeseer: an automatic citation indexing system. In *DL '98: Proceedings of the third ACM conference on Digital libraries*, pages 89–98, New York, NY, USA, 1998. ACM Press.

[44] M. Girolami and S. Rogers. Variational bayesian multinomial probit regression with gaussian process priors. *Neural Comput.*, 18(8):1790–1817, 2006.

[45] S. A. Golder and B. A. Huberman. Usage patterns of collaborative tagging systems. *J. Inf. Sci.*, 32(2):198–208, 2006.

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

[47] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. volume 3, pages 1157–1182, Cambridge, MA, USA, 2003. MIT Press.

[48] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Mach. Learn.*, 46(1-3):389–422, 2002.

[49] H. Halpin, V. Robu, and H. Shepherd. The complex dynamics of collaborative tagging. In *WWW '07*, 2007.

[50] E.-H. S. Han, G. Karypis, and V. Kumar. Text categorization using weight adjusted k -nearest neighbor classification. In *5th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 53–65, 2001.

[51] H. Han, L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis. Two supervised learning approaches for name disambiguation in author citations. In *JCDL '04: Proceedings of the 4th ACM/IEEE joint conference on Digital libraries*, pages 296–305, New York, 2004.

[52] H. Han, H. Zha, and C. L. Giles. Name disambiguation in author citations using a k-way spectral clustering method. In *JCDL '05: Proceedings of the 5th ACM/IEEE joint conference on Digital libraries*, pages 334–343, New York, NY, USA, 2005. ACM Press.

[53] T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(6):607–616, 1996.

[54] T. Hofmann. Probabilistic Latent Semantic Indexing. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, Berkeley, California, 1999.

[55] T. Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 259–266, New York, NY, USA, 2003.

[56] W.-C. Hu, K.-H. Chang, and G. X. Ritter. Web document classification using modified decision trees. In *ACM-SE 38: Proceedings of the 38th annual on Southeast regional conference*, pages 262–263, New York, NY, USA, 2000. ACM Press.

[57] J. Huang, S. Ertekin, and C. L. Giles. Efficient name disambiguation for large-scale databases. In *the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 536–544. Springer-Verlag Berlin Heidelberg, 2006.

[58] M. Iwayama and T. Tokunaga. Hierarchical Bayesian clustering for automatic text classification. In C. E. Mellish, editor, *Proceedings of IJCAI-95, 14th International Joint Conference on Artificial Intelligence*, pages 1322–1327, Montreal, CA, 1995. Morgan Kaufmann Publishers, San Francisco, US.

[59] X. Jin, Y. Zhou, and B. Mobasher. Web usage mining based on probabilistic latent semantic analysis. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 197–205, New York, NY, USA, 2004. ACM Press.

[60] T. Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *ECML '98: Proceedings of the 10th European Conference on Machine Learning*, pages 137–142, London, UK, 1998. Springer-Verlag.

[61] T. Joachims. Training linear svms in linear time. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226, New York, NY, USA, 2006. ACM.

[62] K. S. Jones. *A statistical interpretation of term specificity and its application in retrieval*. Taylor Graham Publishing, London, UK, UK, 1988.

[63] I.-H. Kang and G. Kim. Query type classification for web document retrieval. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 64–71, New York, NY, USA, 2003. ACM.

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

[65] H.-C. Kim and Z. Ghahramani. Bayesian gaussian process classification with the EM-EP algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(12):1948–1959, 2006.

[66] B. King. Step-wise clustering procedures. *Journal of the American Statistical Association*, 69:86–101, 1967.

[67] T. Kohonen. *Self Organization Maps*. Springer, 2001.

[68] A. Kolcz. Local sparsity control for naive bayes with extreme misclassification costs. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 128–137, New York, NY, USA, 2005. ACM.

[69] A. Kolcz and W. tau Yih. Raising the baseline for high-precision text classifiers. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 400–409, New York, NY, USA, 2007. ACM.

[70] T. Kudo and Y. Matsumoto. Use of support vector learning for chunk identification. In *Proceedings of the 4th Conference on CoNLL-2000 and LLL-2000*, pages 142–144, 2000.

[71] T. Kudo and Y. Matsumoto. Chunking with support vector machines. In *NAACL '01: Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies 2001*, pages 1–8, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

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

[73] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001.

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

[75] N. D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[76] N. D. Lawrence and A. J. Moore. Hierarchical gaussian process latent variable models. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 481–488, New York, NY, USA, 2007. ACM.

[77] D. Lee, B.-W. On, J. Kang, and S. Park. Effective and scalable solutions for mixed and split citation problems in digital libraries. In *IQIS '05: Proceedings of the 2nd international workshop on Information quality in information systems*, pages 69–76, New York, 2005.

[78] V. I. Levenshtein. Binary codes capable of correcting spurious insertions and deletions of ones. *Probl. Inform. Transmiss.*, 1:8–17, 1965.

[79] F.-F. Li and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 524–531, Washington, DC, USA, 2005. IEEE Computer Society.

[80] J. Li and J. Z. Wang. Real-time computerized annotation of pictures. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 911–920, New York, NY, USA, 2006. ACM.

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

[82] B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu. Building text classifiers using positive and unlabeled examples. In *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*, page 179, Washington, DC, USA, 2003. IEEE Computer Society.

[83] S. A. Macskassy and H. Hirsh. Adding numbers to text classification. In *CIKM '03: Proceedings of the twelfth international conference on Information*

and knowledge management, pages 240–246, New York, NY, USA, 2003. ACM Press.

[84] B. Malin. Unsupervised name disambiguation via social network similarity. In *Workshop on Link Analysis, Counterterrorism, and Security*, 2005.

[85] G. S. Mann and D. Yarowsky. Unsupervised personal name disambiguation. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 33–40, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[86] R. Marée, P. Geurts, J. Piater, and L. Wehenkel. A generic approach for image classification based on decision tree ensembles and local sub-windows. In K.-S. Hong and Z. Zhang, editors, *Proceedings of the 6th Asian Conference on Computer Vision*, volume 2, pages 860–865. Asian Federation of Computer Vision Societies (AFCV), 2004.

[87] G. Maskeri, S. Sarkar, and K. Heafield. Mining business topics in source code using latent dirichlet allocation. In *ISEC '08: Proceedings of the 1st conference on India software engineering conference*, pages 113–120, New York, NY, USA, 2008. ACM.

[88] T. Minka and J. Lafferty. Expectation-propagation for the generative aspect model. In *the 18th Conference on Uncertainty in Artificial Intelligence*, pages 352–359, 2002.

[89] A. Nakamura and N. Abe. Collaborative filtering using weighted majority prediction algorithms. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 395–403, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[90] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999.

[91] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Mach. Learn.*, 39(2-3):103–134, 2000.

[92] K. M. Ozonat and R. M. Gray. Image classification using adaptive-boosting and tree-structured discriminant vector quantization. In *DCC '04: Proceedings of the Conference on Data Compression*, page 556, Washington, DC, USA, 2004. IEEE Computer Society.

[93] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[94] S. B. Park, J. W. Lee, and S. K. Kim. Content-based image classification using a neural network. *Pattern Recogn. Lett.*, 25(3):287–300, 2004.

[95] D. Pavlov, R. Balasubramanyan, B. Dom, S. Kapur, and J. Parikh. Document preprocessing for naive bayes classification and clustering with mixture of multinomials. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 829–834, New York, NY, USA, 2004. ACM Press.

[96] J. Peng, D. R. Heisterkamp, and H. K. Dai. Lda/svm driven nearest neighbor classification. In *CVPR '01*, page 58, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

[97] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach. In *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 473–480, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[98] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin dags for multiclass classification. In S. Solla, T. Leen, and K.-R. Mueller, editors, *Advances in Neural Information Processing Systems 12*, pages 547–553, 2000.

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

[100] J. C. Platt, C. J. C. Burges, S. Swenson, C. Weare, and A. Zheng. Learning a gaussian process prior for automatically generating music playlists. In *Advances in Neural Information Processing Systems 14*, pages 1425–1432, 2002.

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

[102] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA, 1994. ACM Press.

[103] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. The author-topic model for authors and documents. In *AUAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 487–494, Arlington, Virginia, United States, 2004. AUAI Press.

[104] M. E. Ruiz and P. Srinivasan. Hierarchical text categorization using neural networks. *Inf. Retr.*, 5(1):87–118, 2002.

[105] R. E. Schapire. Msri workshop on nonlinear estimation and classification, 2002. the boosting approach to machine learning an overview, 2002.

[106] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. In *COLT' 98*, pages 80–91, New York, NY, USA, 1998. ACM Press.

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

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

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

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

[111] K. Seymore, A. McCallum, and R. Rosenfeld. Learning hidden Markov model structure for information extraction. In *AAAI 99 Workshop on Machine Learning for Information Extraction*, pages 37–42, 1999.

[112] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering objects and their localization in images. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pages 370–377, Washington, DC, USA, 2005.

[113] Y. Song and C. L. Giles. Efficient user preference predictions using collaborative filtering. In *the 19th International Conference on Pattern Recognition (ICPR 2008)*, 2008.

[114] Y. Song, J. Huang, I. G. Councill, J. Li, and C. L. Giles. Efficient topic-based unsupervised name disambiguation. In *JCDL '07: Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, pages 342–351, New York, NY, USA, 2007. ACM.

[115] Y. Song, J. Huang, I. G. Councill, J. Li, and C. L. Giles. Generative models for name disambiguation. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 1163–1164, New York, NY, USA, 2007.

[116] Y. Song, J. Huang, D. Zhou, H. Zha, and C. L. Giles. IKNN: Informative k-nearest neighbor pattern classification. In *PKDD 2007*, pages 248–264, 2007.

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

[118] Y. Song, D. Zhou, J. Huang, I. G. Councill, H. Zha, and C. L. Giles. Boosting the feature space: Text classification for unstructured data on the web. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pages 1064–1069, Washington, DC, USA, 2006. IEEE Computer Society.

[119] Y. Song, Z. Zhuang, H. Li, Q. Zhao, J. Li, W.-C. Lee, and C. L. Giles. Real-time automatic tag recommendation. In *SIGIR '08: Proceedings of the 31st annual in-*

ternational ACM SIGIR conference on Research and development in information retrieval, pages 515–522, New York, NY, USA, 2008. ACM.

[120] N. Srebro, J. D. M. Rennie, and T. S. Jaakola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1329–1336, 2005.

[121] R. Srinivasan. *Importance sampling - Applications in communications and detection*. Springer-Verlag, 2002.

[122] M. Steyvers, P. Smyth, M. Rosen-Zvi, and T. Griffiths. Probabilistic author-topic models for information discovery. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 306–315, New York, NY, USA, 2004. ACM.

[123] E. B. Sudderth, A. Torralba, W. T. Freeman, and A. S. Willsky. Learning hierarchical models of scenes, objects, and parts. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 1331–1338, Washington, DC, USA, 2005.

[124] A. Sun, E.-P. Lim, and W. K. Ng. Personalized classification for keyword-based category profiles. In *ECDL '02: Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries*, pages 61–74, London, UK, 2002. Springer-Verlag.

[125] H. Taira and M. Haruno. Feature selection in svm text categorization. In *AAAI '99/IAAI '99*, pages 480–486, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

[126] L. Tang, J. Zhang, and H. Liu. Acclimatizing taxonomic semantics for hierarchical content classification from semantics to data-driven taxonomy. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 384–393, New York, NY, USA, 2006. ACM Press.

[127] M. Tipping and C. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society*, 6:611–622, 1999.

[128] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast random walk with restart and its applications. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pages 613–622, Washington, DC, USA, 2006. IEEE Computer Society.

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

[130] L. Ungar and D. Foster. Clustering methods for collaborative filtering. In *Proceedings of the Workshop on Recommendation Systems*. AAAI Press, Menlo Park California, 1998.

[131] R. Urtasun, D. J. Fleet, and P. Fua. 3d people tracking with gaussian process dynamical models. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA*, pages 238–245, 2006.

[132] S. V. N. Vishwanathan, N. N. Schraudolph, M. W. Schmidt, and K. P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 969–976, New York, NY, USA, 2006. ACM.

[133] G. Wang and F. H. Lochovsky. Feature selection with conditional mutual information maximin in text categorization. In *CIKM '04: Proceedings of the thirteenth ACM conference on Information and knowledge management*, pages 342–349, New York, NY, USA, 2004. ACM Press.

[134] J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *Transactions on Pattern Analysis and Machine Intelligence*, 30(2):283–298, Feb. 2008.

[135] L. Wang, J. Zhu, and H. Zou. Hybrid huberized support vector machines for microarray classification. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 983–990, New York, NY, USA, 2007. ACM.

[136] X. Wang and A. McCallum. Topics over time: a non-markov continuous-time model of topical trends. In *KDD '06: Proceedings of the 12th ACM SIGKDD*

international conference on Knowledge discovery and data mining, pages 424–433, New York, NY, USA, 2006.

[137] Y. Wang, J. Hodges, and B. Tang. Classification of web documents using a naive bayes method. In *ICTAI '03: Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, page 560, Washington, DC, USA, 2003. IEEE Computer Society.

[138] J. H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58:236–244, 1963.

[139] X. Wei and W. B. Croft. Lda-based document models for ad-hoc retrieval. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 178–185, New York, NY, USA, 2006. ACM Press.

[140] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. In *NIPS*, pages 1473–1480, 2005.

[141] B. Wellner, A. McCallum, F. Peng, and M. Hay. An integrated, conditional model of information extraction and coreference with application to citation matching. In *AUAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 593–601, Arlington, Virginia, United States, 2004. AUAI Press.

[142] 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: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 130–137, New York, NY, USA, 2005. ACM.

[143] G. Xu, Y. Zhang, J. Ma, and X. Zhou. Discovering user access pattern based on probabilistic latent factor model. In *ADC '05: Proceedings of the sixteenth Australasian database conference*, pages 27–35, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[144] S. Xu, S. Bao, Y. Cao, and Y. Yu. Using social annotations to improve language model for information retrieval. In *CIKM '07: Proceedings of the sixteenth ACM*

conference on Conference on information and knowledge management, pages 1003–1006, New York, NY, USA, 2007. ACM.

[145] J. Yang, X. Yang, and J. Zhang. A parallel multi-class classification support vector machine based on sequential minimal optimization. In *IMSCCS '06: Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences - Volume 1 (IMSCCS'06)*, pages 443–446, Washington, DC, USA, 2006. IEEE Computer Society.

[146] Y. Yang, J. Zhang, and B. Kisiel. A scalability analysis of classifiers in text categorization. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 96–103, New York, NY, USA, 2003. ACM Press.

[147] H. Yu, C. Zhai, and J. Han. Text classification from positive and unlabeled documents. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 232–239, New York, NY, USA, 2003. ACM Press.

[148] L. Yu and H. Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of The Twentieth International Conference on Machine Leaning (ICML 2003)*, pages 856–863, 2003.

[149] 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.

[150] H. Zhang, A. C. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2126–2136, Washington, DC, USA, 2006. IEEE Computer Society.

[151] M.-L. Zhang and Z.-H. Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.

[152] X. Zhou, X. Hu, and X. Zhang. Topic signature language models for ad hoc retrieval. *IEEE Trans. Knowl. Data Eng.*, 19(9):1276–1287, 2007.

[153] D. Zhuang, B. Zhang, Q. Yang, J. Yan, Z. Chen, and Y. Chen. Efficient text classification by weighted proximal svm. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 538–545, Washington, DC, USA, 2005. IEEE Computer Society.

Vita

Yang Song

RESEARCH INTERESTS

Machine learning, data mining, information retrieval, text analysis, collaborative filtering

EDUCATION

- The Pennsylvania State University, University Park, PA — May 09
  Ph.D Candidate of Computer Science and Engineering — GPA: 3.90/4.0
- Zhejiang University, Zhejiang, China — Jul 03
  Bachelor of Science of Computer Science and Engineering — GPA (major): 3.92/4.0

RESEARCH EXPERIENCE

IBM Research, TJ Watson Center, Hawthorne, NY, Summer 08
Research Intern in Services Research & Platform Group
- Designed a novel hierarchical classification framework for problem classification in IT services
- Proposed an online learning algorithm for real-time efficient model learning
- Introduced a feature selection method that combines system log data and performance metrics

Microsoft Research, Microsoft, Redmond, WA, Summer 07
Research Intern in Live Labs
- Proposed a novel machine learning algorithm for large-scale binary text classification
- Applied this algorithm to content based spam filtering by testing over 1 million emails
- Improved the performance of the classifier by using a collaborative filtering technique

Ask.com Inc, Piscataway, NJ, Summer 05
Research Intern in the Web Research Group
- Designed an online name entity extraction algorithm with application to real-time news articles
- Co-built a component for query suggestions for the Ask.com search engine
- Built a demo of event visualization for Wikipedia topics

SELECTED FIRST-AUTHOR PUBLICATIONS

- A Non-parametric Approach to Pair-wise Dynamic Topic Correlation Detection (**ICDM 2008**)
- Real-time Automatic Tag Recommendation (**SIGIR 2008**)
- A Sparse Gaussian Processes Classification Framework for Fast Tag Suggestions (**CIKM 2008**)
- IKNN: Informative K-Nearest Neighbor Pattern Classification (**PKDD 2007**)
- Generative models for name disambiguation (**WWW 2007**)
- Boosting the feature space: Text classification for unstructured data on the web (**ICDM 2006**)

US PATENTS

- Problem Classification Method to Enhance the Incident Management Process for the IT Services
- Business Partner Selection algorithms for Remote Managed Services