UNIVERSITY OF CALIFORNIA
Santa Barbara

Understanding the Semantics of Networked Text

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

by

Gengxin Miao

Committee in Charge:

Professor L. E. Moser, Chair

Professor X. Yan, Co-Chair

Professor P. M. Melliar-Smith

Professor V. Rodoplu

Dr. J. Tatemura

June 2012

The Dissertation of
Gengxin Miao is approved:

Professor P. M. Melliar-Smith

Professor V. Rodoplu

Dr. J. Tatemura

Professor X. Yan, Committee Co-Chair

Professor L. E. Moser, Committee Chair

June 2012

Understanding the Semantics of Networked Text

by

Gengxin Miao

Curriculum Vitæ

Gengxin Miao

Education

2012 Ph.D. of Science in Electrical and Computer Engineering, University of California, Santa Barbara.

2008 Master of Science in Computer Engineering, University of California, Santa Barbara.

2006 Master of Science in Automation, Tsinghua University.

2003 Bachelor of Engineering in Automation, Tsinghua University.

Experience

Intern Researcher, IBM TJ Watson Research Center, Hawthorne, NY, June 2011 - September 2011.

Analyzed the static and dynamic properties of real-world collaborative networks.

Developed graph model and routing algorithm to simulate the human dynamics in collaborative networks.

Proposed the first technique to evaluate quantitatively the working efficiency of collaborative networks.

Graduate Research Assistant, UC Santa Barbara, Santa Barbara, CA, September 2009 - June 2012.

Developed probabilistic generative models to characterize information flow over a social network.

Analyzed roles of the individuals in a social routing task.

Developed topic models to analyze latent topics among multiple document corpuses simultaneously.

Intern Researcher, Google, Mountain View, CA, June 2009 - September 2009.

Recovered semantics and identified subjects of data tables from the deep Web.

Enhanced Web search results by leveraging the deep Web data tables.

Assistant Student Researcher, NEC Laboratories America, Cupertino, CA, June 2008 - September 2008.

Developed a domain-independent and fully automatic Web data records extraction algorithm.

The algorithm captures repetitive patterns rendered in Web pages by analyzing the HTML tag paths.

Both flat data records and nested data records can be extracted automatically.

No prior knowledge on how the Web page is designed is necessary to extract data records.

Intern Researcher, Google China, Beijing, China, April 2007 - December 2007.

Parallelized spectral clustering, co-clustering and kernel K-means.

Evaluated the effectiveness and efficiency of these parallel algorithms using large-scale text data and social network data.

Intern Researcher, Google China, Beijing, China, July 2006 - September 2006.

Surveyed existing clustering algorithms and implemented them.

Compared the performance of clustering algorithms using both synthetic data and real-world data.

Research Assistant, Tsinghua University, Beijing, China, September 2004 - July 2006.

Developed a real-time, vision-based driver assistance system.

The system analyzes the video taken in front of the vehicle and detects pedestrians.

Visiting Student, Microsoft Research Asia, Beijing, China, September 2004 - June 2005.

Classify Web search queries based on underlying information needs.

Information needs are defined as Navigational, Informational and Interactional.

The classifier takes input from user's click-through information, as well as the query terms.

Visiting Student, Microsoft Research Asia, Beijing, China, January 2004 - August 2004.

Developed a Web-page adaptation engine to render Web-pages on small handheld devices.

This work aims to enhance the mobile user's Web browsing experience.

Selected Publications

Z. Guan, G. Miao, X. Yan, R. McLoughlin, “Expertise ranking using co-occurrence relationships on the Web,” To appear in *IEEE Transactions on Knowledge and Data Engineering.*

G. Miao, L. E. Moser, X. Yan, S. Tao, Y. Chen, N. Anerousis, “Reliable ticket routing in expert networks,” *Reliable knowledge discovery, Springer.*

F. Kart, G. Miao, L. E. Moser, P. M. Melliar-Smith, “A distributed e-healthcare system,” *Handbook of Research on Distributed Medical Informatics and E-Health*, vol. 1, no. 7, 2008.

G. Miao, Z. Guan, L. E. Moser, X. Yan, S. Tao and N. Anerousis, “Latent association analysis of document pairs,” *Proceedings of the 18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Beijing, China, August 2012.

G. Miao, S. Tao, W. Cheng, R. Moulic, L. Moser, D. Lo, X. Yan, “Understanding task-driven information flow in collaborative networks,” *Proceedings of the 21st International Conference on the World Wide Web*, Lyon, France, April 2012.

P. Venetis, A. Halevy, J. Madhavan, M. Pasca, W. Shen, F. Wu, G. Miao, C. Wu “Recovering semantics of tables on the Web,” *Proceedings of the 37th International Conference on Very Large Data Bases*, Seattle, WA, August 2011, pp. 528-538.

G. Miao, L. E. Moser, X. Yan, S. Tao, Y. Chen and N. Anerousis, “Generative models for ticket resolution in expert networks,” *Proceedings of the 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Washington, D.C., July 2010, pp. 733-742.

G. Miao, F. Kart, L. E. Moser and P. M. Melliar-Smith, “Collaborative Web data record extraction as a Web Service for social networks,” *Proceedings of the 7th IEEE International Conference on Web Services*, Los Angeles, CA, July 2009, pp. 896-902.

G. Miao, J. Tatemura, W. Hsiung, A. Sawires L. E. Moser, “Extracting data records from the Web using tag path clustering,” *Proceedings of the 18th International Conference on World Wide Web*, Madrid, Spain, April 2009, pp. 981-990.

F. Kart, G. Miao, L. E. Moser and P. M. Melliar-Smith, “A distributed e-healthcare system,” *Handbook of Research on Distributed Medical Informatics and E-Health*, vol. 1, no. 7, 2008.

G. Miao, Y. Song, D. Zhang and H. Bai, “Parallel spectral clustering algorithm for large-scale community data mining,” *Proceedings of the World Wide Web Workshop on Social Web Search and Mining*, Beijing, China, April 2008.

F. Kart, G. Miao, L. E. Moser and P. M. Melliar-Smith, “A distributed e-healthcare system based on the Service Oriented Architecture,” *Proceedings of the IEEE*

International Conference on Services Computing, Salt Lake City, UT, July 2007, pp. 652-659 (Won First Prize in the IEEE Services Computing Contest).

G. Miao, Y. Luo, Q. Tian and J. Tang, "A filter module used in pedestrian detection system," In: *Proceedings of IFIP Artificial Intelligence Applications and Innovations*, vol. 204, 2006, Springer, Boston, MA, pp. 212-220.

X. Xie, G. Miao, R. Song, J. R. Wen and W. Y. Ma, "Efficient browsing of Web search results on mobile devices based on block importance model," *Proceedings of the Third IEEE International Conference on Pervasive Communication*, March 2005, pp. 17-26.

U.S. Patents

J. Madhavan, C. M. Wu, A. Halevy, G. Miao and M. Pasca, "Table search using recovered semantic information," U.S. Patent pending.

X. Xie, W. Y. Ma and G. Miao, "Block importance analysis to enhance browsing of Web page search results," U.S. Patent 20060123042.

X. Xie, G. Miao, G. Xin, R. Song, J. R. Wen and W. Y. Ma, "Categorizing page block functionality to improve document layout for browsing," U.S. Patent 20070074108.

Honors and Awards

IBM Ph.D. Fellowship Award, 2011-2012.

UC Santa Barbara Doctoral Student Travel Grant, 2010.

KDD Student Travel Award, 2010.

UCSB Fellowship Award, Summer 2009.

First prize in IEEE Services Computing Contest, July 2007.

UCSB Fellowship Award, September 2006.

Second place in Best Rank Contest, Microsoft Research Asia, September 2005.

Scholarship for Outstanding Academic Performance, Tsinghua University, October 2000.

Tangshi Fellowship, 1999 - 2003 (consecutive years).

Excellent Student in Sports Competition, 1999 - 2001 (consecutive years).

Professional Activities

Reviewer for International Conference on Data Engineering 2011.

Reviewer for UCSB Graduate Student Workshop, 2011.

Reviewer for ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2010.

Reviewer for SIAM Conference on Data Mining, 2010.

Reviewer for IEEE 7th International Conference on Web Service, 2009.

Reviewer for IEEE Transactions on Knowledge and Data Engineering, 2007.

Teaching Experience

Teaching Assistant, ECE 155B: Network Computing, UCSB, Winter 2008, Spring 2009

Teaching Assistant, ECE 155A: Computer Networks, UCSB, Fall 2007, Winter 2009

Teaching Assistant, ECE 152A: Digital Design Principles, UCSB, Winter 2007, Fall 2008

Teaching Assistant, ECE 154: Introduction to Computer Architecture, UCSB, Fall 2006

Teaching Assistant, Fundamentals of Analog Circuits, Tsinghua University, Fall 2001, Spring 2002

Abstract

Understanding the Semantics of Networked Text

Gengxin Miao

Social networks are a powerful means for information sharing. A large social network typically has hundreds of millions of users. These users are interconnected through social links to friends, colleagues, family members, etc. The frequent interaction and information exchange between users form a massive heterogeneous information network. Understanding the semantic information in the textual data and the topological information in the social network poses a grant challenge for data mining researchers. This Ph.D. dissertation tackles the problem of understanding the unstructured or semi-structured data in social networks. First, we describe a parallel spectral clustering algorithm that makes possible clustering analysis on large-scale social networks with hundreds of millions of users. Comprehensive analysis, extraction and integration of information from multiple sources are necessary. Next, we describe an information extraction engine that extracts data items from Web pages without knowing the data wrapping template. We also present an information integration approach to aggregate data tables collected from the Web and hence better serve general Web search. To make information routing in collaborative networks more efficient, we describe generative models to characterize expertise awareness relationships between agents in collab-

orative networks and provide efficient task routing recommendations. We also describe, in depth, the first quantitative analysis of the information flow efficiency in collaborative networks. To utilize the accumulated information, we developed a topic modeling approach that allows document retrieval across multiple document sets with possible semantic gaps and vocabulary gaps.

Professor L. E. Moser
Dissertation Committee Chair

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Web and social networks are powerful means for information sharing. A large social network typically has hundreds of millions of users. To date, Facebook has achieved 630 million users. LinkedIn and Twitter are also experiencing a stunning user growth rate. These users are interconnected through social links to friends, colleagues, family members, etc. Users with common interests form communities. Users interact with each other by writing posts, asking questions, sharing information, etc. These social activities create a tremendous amount of data.

Analyzing the data and information flow in social networks facilitates the recognition of major events with world-wide impact, the prediction of trends in public opinion, and more, in a timely and scalable manner. Often, it is the case that social media respond much more quickly than traditional public media. For example, Twitter had a

large burst of twits about the earthquake in Virginia in 2011 before the news media released the first formal news. Detecting bursts of activity in social media can enable public media to achieve faster responses and larger coverage. Social media also plays an important role in politics and business. For example, the Libyan revolution received tremendous support from social media. Online merchandisers gain ideas for their businesses from new hot topics discussed in social networks. Social networks also provide an important source for fundamental sociological research as traditional social interactions become more technology-based. Thus, data found in social networks offers great opportunities for researchers in many research domains.

However, analysis of data within social networks also presents great challenges. First of all, data within social networks are typically created in a large-scale, distributed manner. With the advance of technology, data storage capacity continues to increase. On the other hand, data analysis tools do not scale well to satisfy big data analytic needs, especially for dealing with incremental data. Existing data mining and machine learning techniques that work well with small datasets need to be re-invented to fit big data settings, where executions are typically performed in parallel or online.

Moreover, comprehensive data analysis needs to leverage data collected from multiple sources. Each data source publishes its own data in its own specific way. These distributed, independent data sources lack a uniform standard for data publication. Thus, data extraction and data integration are huge challenges. Even more challenging, the

Web postings in social networks are written by humans in natural languages. Different people use different terminology to express the same idea, and they use the same terminology with different meanings. Analyzing the semantics of natural language texts with proper consideration of the underlying network structures that connect the texts is yet another modeling challenge.

This Ph.D. Dissertation addresses large-scale unstructured or semi-structured data within social networks and contributes toward semantic understanding of the data with emphasis on parallel and distributed computing, data extraction and integration, information flow analysis, and topic modeling. The specific contributions of this Ph.D. Dissertation are highlighted below and are described in detail in subsequent chapters.

1.1 Parallel Spectral Clustering

Users of social networks connect with each other and form communities of interest. As the scale of the network increases to hundreds of millions of users, the edges that join users become very sparse. It is reported that Facebook users have an average of approximately 130 connections among the 630 million Facebook users. For this large user population size, it is almost impossible for a user to explore all of the other users or communities of users with potential common interests.

Clustering algorithms can be used to group together users into communities and, hence, they facilitate the users' exploration of the data in the network. Although the k-means algorithm can be parallelized to accommodate large-scale datasets on the MapReduce platform, its assumption that the data samples follow a Gaussian distribution inside each cluster does not hold for super-sparse datasets, not to mention the algorithm's sensitivity to the choice of the initial cluster centroid. Spectral clustering has proven to be effective in finding clusters with non-linear boundaries. Unfortunately, spectral clustering suffers from the scalability problem in both memory space and computing time.

This Ph.D. Dissertation contains the first study of parallelization of spectral clustering. The Parallel Spectral Clustering (PSC) algorithm is based on the MPICH2 platform, which provides distributed memory and distributed computation within a distributed computing system. The PSC algorithm finds clusters of communities in a large social network of users with similar interests. Experiments performed for the Orkut social network, with more than 10,000,000 users and 150,000 communities, demonstrate the effectiveness of the PSC algorithm. The PSC algorithm derives 100 clusters of communities for this dataset and finishes within 20 minutes when using 90 computers. The PSC algorithm makes possible online clustering of social networks with large user populations, such as Orkut. Clustering greatly enables the users in finding communities of users with interests that match their particular interests.

1.2 Extraction and Integration of Data from Distributed Sources

For many social networks, the data are stored in a database and, at query time, the contents are rendered in HTML code and are displayed on Web pages. The data scale is large, and the data schema differ from site to site. Automatic methods that extract lists of data items have been extensively studied. In existing data extraction algorithms, typically a wrapper is used to compare contiguous segments of HTML code. These methods suffice for simple search, but often fail to handle more complicated or noisy Web page structures due to a limitation: their greedy manner of identifying lists of records through pairwise comparison of consecutive segments.

The novel DataExtractor system, presented in this Ph.D. Dissertation, mimics the process of how a human finds data records on a Web page or screen. To the human eye, the data items on a Web page are rendered in visually repeating patterns. The distinct HTML tag paths, that correspond to these visual signals, are extracted and clustered, and the data records are then extracted based on the visual signals. The DataExtractor system yields higher extraction precision and recall than existing algorithms, especially when the Web pages contain nested data items or loosely formatted data items.

The data tables extracted from the Web pages offer a corpus of more than 100 million tables, and are difficult for a computer to process, because the semantics of

the data are typically not explicit in the tables. Table headers (record fields) exist in few cases and even when they do, the attribute names are often useless. Moreover, the ranking methods for searching document corpora for general Web search do not work well for table corpora.

The novel TableFinder system, presented in this Ph.D. Dissertation, attempts to recover the semantics of the extracted data in the tables by enriching the tables with additional annotations. The annotations facilitate operations such as searching for tables and finding related tables. To recover the semantics of the extracted data in the tables, the TableFinder system leverages a database of class labels and relationships automatically extracted from the Web pages. The database of classes and relationships has very wide coverage, but is also very noisy. The TableFinder system attaches a class label to a column if a sufficient number of values in the column are identified with that label in the database of class labels, and similarly for binary relationships.

This Ph.D. Dissertation further introduces a formal model for reasoning about when there exists sufficient evidence for a label. Experiments demonstrate the utility of the recovered semantics for table search and shows that the method performs substantially better than previous approaches, such as a simple majority scheme. In addition, this Ph.D. Dissertation characterizes what fraction of the tables on the Web can be annotated using this approach.

1.3 Modeling Information Flow in Collaborative Networks

In contrast to Web search engines that facilitate information retrieval in a library paradigm, social networks follow a village paradigm in which information flows from person to person. Unlike general Web search where an individual seeks to find a Web document that contains the target information, in a social network individuals desire to find an efficient social route that leads to a person who has the target information. Thus, information flow within social networks needs to be analyzed. The posts, notes, and comments conveyed in social networks contain valuable semantic information for analyzing information flow. They are usually unstructured and difficult for a computer to organize and analyze.

This Ph.D. Dissertation presents the ticket resolution process for expert networks, collaborative research conducted with researchers at IBM T.J. Watson. Problems and work requests are submitted to an expert network in the form of tickets. These tickets sometimes bounce among many expert groups before they are transferred to the correct resolver, particularly when the network size is large. Finding a methodology that reduces such bouncing and hence shortens the ticket resolution time is a long-standing challenge.

This Ph.D. Dissertation presents generative models that capture semantic-level information flow in expert networks. Based on these generative models, routing algorithms are developed. These routing algorithms provide suggestions that quickly route tickets to an appropriate expert within a large expert network. These models and algorithms apply to posts, notes, and comments found in many different kinds of social networks.

This Ph.D. Dissertation further studies the behavior of experts in expert networks. The typical roles of experts in expert networks are as resolvers and transferrers. The resolvers resolve many tickets by themselves. The transferrers have knowledge of what other experts are capable of doing and are essential for routing tickets. For a ticket that traverses extremely long paths before being resolved, there might exist experts who can neither resolve the ticket, nor make good routing decisions. Identifying such experts can help to provide targeted training and, hence, improve the efficiency of routing tickets through the network.

1.4 Quantitative Analysis of Task-Driven Information Flow

Collaborative networks are a special type of social network formed by members who collectively achieve particular goals, such as fixing software bugs and resolv-

ing customers' information technology problems. In such networks, information flow among the members of the network is driven by the tasks assigned to the network, and by the expertise of its members to complete those tasks.

This Ph.D. Dissertation analyzes real-life collaborative networks to understand their common characteristics and how information is routed in these networks. It shows that the topology of collaborative networks exhibits significantly different properties compared to other common complex networks. Collaborative networks have truncated power-law node degree distributions and other organizational constraints. Furthermore, the number of steps along which information is routed follows a truncated power-law distribution.

Based on these characterizations, this Ph.D. Dissertation presents a novel network model that can be used to generate synthetic collaborative networks subject to certain structural constraints. Moreover, it presents a novel routing model that emulates task-driven information routing conducted by human beings in collaborative networks. Together, these two models are used to study the efficiency of information routing for various topologies of a collaborative network - a problem that is important in practice yet difficult to solve without the methods presented in this Ph.D. Dissertation.

1.5 Modeling Networked Document Sets

Many social networks feature a question-answering process that allows individuals to ask questions or answer the questions of others. The collections of questions and answers form a pairwise document set. Among the many questions raised by individuals, the same questions are likely to be asked many times and presented in different ways. An individual who can answer a question is unlikely to have the energy to answer all of the variations of the question posed by other individuals.

Given a new question, automatically ranking the potential answers using the existing question-answer pairs can help boost the coverage of answered questions. Such ranking presents a challenge for information retrieval involving two or more document sets that is different from traditional information retrieval in a single document set. Relevance ranking based on keyword matching no longer fits the problem due to the multiple document sets involved.

Questions are typically asked by individuals who think from an application perspective. The answers are typically written by professionals who think from a technical perspective. For example, when a user asks a Microsoft Windows blue-screen question, the solutions can be related to multiple software components in the Windows system of which the customer might be unaware. Moreover, the pairs of documents can be written in different languages, such as the English and Chinese versions of articles on

the Wikipedia Website. Thus, there might be a vocabulary gap between the source documents (queries) and the target documents. This vocabulary gap identifies the problem settings for information retrieval with multiple document sets that are different from traditional information retrieval. There might also be a topic gap between the source documents and the target documents, considering that the questions and the answers might emphasize different topics.

This Ph.D. Dissertation describes a novel topic modeling approach – Latent Association Analysis (LAA) – that explicitly mines the correlation between a pair of documents. The generative process defined by the LAA model first draws a correlation factor that holds together a pair of documents, just as an underlying disease explains why a certain symptom leads to a specific treatment. Based on the correlation factor, two separate topic proportion vectors are drawn for the corresponding source and target documents. Given the topic proportion vector, the LAA method draws the topic assignment and the word from the topic-to-word distribution, similar to other topic modeling approaches.

Experiments demonstrate that the LAA method significantly outperforms other state-of-the-art methods in identifying the correct target document, when a source document is given. The LAA method roughly ranks the correct target document within the top 10 out of 100 candidates. Thus, the LAA method reduces the search space by an order of magnitude. If a user initially needs to search through 100 documents to find the correct

answer, with the help of the LAA model the user needs to search through only 10 documents to find the correct answer. The LAA method can greatly improve information consumption efficiency, especially when the document corpus is large.

1.6 Summary

In summary, this Ph.D. Dissertation addresses the general problem of unstructured or semi-structured data within social networks. It focuses more specifically on the following issues: (1) scalability for unstructured data within social networks that comprise millions of users, (2) unstructured data extraction and integration, (3) information flow modeling over social networks and topic analysis, (4) quantitative analysis of task-driven information flow on collaborative networks, and (5) topic modeling across multiple large-scale document sets within social networks. This Ph.D. Dissertation presents novel models, methods, algorithms, and systems that address these issues and that contribute toward the understanding of unstructured or semi-structured data within social networks.

Chapter 2

Parallel Spectral Clustering

The Web and social networks allow users to engage each other through both information and application sharing. For instance, users share data via Blog, Wiki, or BBS services. Users share applications on social platforms such as Facebook and OpenSocial. Communities are formed by users of similar interests. Being able to discover communities of common interests is of the paramount importance for maintaining high viral energy in social networks. Such discoveries can enable effective friend suggestions, topic recommendations, and advertisement matchings, just to name a few.

One approach to discover communities of common interests is through clustering. The biggest challenge that a clustering algorithm faces is scalability. An algorithm must be able to handle millions of data instances in a relatively short period of time. For example, Orkut [6] consists of more than 20 million communities and more than

50 million users[1]. Performing clustering on such a large dataset on a single computer is prohibitive in both memory use and computational time.

In this chapter, we present a parallel spectral clustering algorithm that runs on distributed computers. With the increasing popularity of distributed data centers and clouds that contain millions of computers, this parallel approach can scale up to solve large-scale clustering problems.

We select spectral clustering as our base algorithm because of its well-known effectiveness. The graph cut can be formulated as an eigenvalue decomposition problem of the graph Laplacian [33] by relaxing the labels to be real values. The graph Laplacian can be seen as an approximation of the Laplace-Beltrami operator on the manifold [15]. Representative spectral clustering methods include Min Cut [142], Normalized Cut [118], Radio Cut [60], Min-Max Cut [47] and Co-Clustering [40, 151]. Moreover, in a general relaxation view, graph cut, k-means, Principle Component Analysis (PCA) and Nonnegative Matrix Factorization (NMF) [76] (and their corresponding kernel versions) can be seen as unified frameworks [41, 45, 46]. Many practical applications, such as image segmentation [118] and text categorization [40, 151], have proven to be well-suited spectral clustering applications.

Unfortunately, eigenvalue decomposition and k-means calculations present bottlenecks for spectral clustering. The memory use of eigenvalue decomposition is $\mathcal{O}(n^2)$,

[1]The claim was based on statistics in year 2007.

where n is the number of data instances. The time complexity for eigenvalue decomposition is $\mathcal{O}(n^3)$ at the worst case. When n is very large, say beyond a million, traditional single-computer speedup schemes [42, 55, 77, 99] still suffer from either memory or CPU limitations.

Our parallel algorithm employs a parallel ARPACK algorithm (PARPACK) [89] to perform parallel eigenvalue decomposition. Although there exist other parallel eigenvalue or singular-value decomposition techniques [67, 71, 85], the PARPACK algorithm has the following advantages: (1) It can be computed on distributed computers as well as multi-core systems, and (2) it is fast when the matrix is sparse. Moreover, we implement a parallel k-means algorithm to cluster data in the eigenvector space. To reduce the memory use, our algorithm loads onto each computer only the necessary rows of data for conducting parallel computation. Empirical studies show that our parallel spectral clustering algorithm is both accurate and efficient.

Chu et al. [32] employed map reduce on multi-core computers and parallelized a variety of learning algorithms including k-means to obtain speedups. However, these solutions are implemented on a shared memory, multi-core system. The limit of memory space still exists. The closest work to our work is that of [43], which presents a parallel k-means clustering algorithm that is also based on distributed memory. However, using k-means alone, it is not possible to deal with non-linearly separable datasets. Moreover, the time complexity of the k-means algorithm grows linearly with the dimen-

sionality of the data, whereas spectral clustering does not suffer from this problem. The eigenvalue decomposition procedure has the virtue of reducing dimensionality for the k-means algorithm.

2.1 Spectral Clustering

In this section, we briefly review the eigenvalue decomposition problem involved in both spectral clustering and co-clustering. This review introduces notation that is used in the rest of this chapter.

2.1.1 Spectral Analysis of Graph Cuts

Consider $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as a weighted neighborhood graph that is constructed by the point cloud $X = (x_1, ..., x_n)$, where n is the point number, $\mathcal{V}$ is the vertex set of graph, and $\mathcal{E}$ is the edge set that contains the pairs of neighboring vertices (x_i, x_j). A typical similarity matrix S of a neighborhood graph can be defined as:

$$S_{ij} = \begin{cases} S(x_i, x_j) & \text{if } (x_i, x_j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

where $S(x_i, x_j)$ is a similarity score given by, *e.g.*, a Gaussian kernel function. The graph Laplacian of a neighborhood graph is $L = D - S$, and the normalized graph

Laplacian is $\bar{L} = I - D^{-\frac{1}{2}} S D^{-\frac{1}{2}}$, where the diagonal matrix D satisfies $D_{ii} = d_i$, and $d_i = \sum_{j=1}^{n} S_{ij}$ is the degree of vertex x_i [33].

Consider the normalized cut. We need to find subsets $\mathcal{A}$ and $\mathcal{B}$ such that the normalized cut criterion $\mathcal{J}_{NCut}(\mathcal{A}, \mathcal{B}) = \frac{cut(\mathcal{A},\mathcal{B})}{assoc(\mathcal{A},\mathcal{V})} + \frac{cut(\mathcal{B},\mathcal{A})}{assoc(\mathcal{B},\mathcal{V})}$ is minimized. It has been shown [118] that the solution is given by optimizing the following criterion:

$$f_L^* = \underset{f^T f_0 = 0}{\operatorname{argmin}} \frac{f^T L f}{f^T D f} \tag{2.2}$$

where $f = (f(x_1), f(x_2), ..., f(x_n))^T \in \mathcal{R}^{n \times 1}$. The solution iis given by the second smallest eigenvector of the generalized system $Lf = \lambda D f$, where $f_0 = \vec{1}$ is the eigenvector corresponding to the smallest eigenvalue $\lambda_0 = 0$. Note that, if we use the normalized graph Laplacian instead of the unnormalized one, the solution is $f_{\bar{L}}^* = \underset{f^T f_0 = 0}{\operatorname{argmin}} \frac{f^T \bar{L} f}{f^T f}$. This solution is further related to (2.2) because $f_{\bar{L}}^* = D^{\frac{1}{2}} f_L^*$.

Note the following fact:

$$\operatorname{argmin} \frac{f^T \bar{L} f}{f^T f} = \operatorname{argmin} \frac{f^T (I - D^{-\frac{1}{2}} S D^{-\frac{1}{2}}) f}{f^T f} = \operatorname{argmax} \frac{f^T \bar{S} f}{f^T f}$$

where $\bar{S} = D^{-\frac{1}{2}} S D^{-\frac{1}{2}}$. The spectral clustering problem can be solved in the scaled kernel PCA (KPCA) framework. The difference is that KPCA uses full connection graphs, while spectral clustering methods can use neighborhood graphs. The advantage of using neighborhood graphs is that their corresponding similarity matrices are sparse and, therefore, fast algorithms can be introduced.

2.1.2 Co-Clustering

For text categorization or community analysis problems, the word-by-document or user-by-community co-occurrence matrices can be used to generate a bipartite graph. Taking user-by-community co-occurrence as an example, the graph is defined as $\mathcal{G} = (\mathcal{U}, \mathcal{C}, \mathcal{E})$, where $\mathcal{U}$ denotes the set of user vertices, $\mathcal{C}$ denotes the set of community vertices and $\mathcal{E}$ denotes the edge set. We can make use of co-clustering techniques to cluster users and communities simultaneously [40, 151]. Unlike the edges of traditional graphs, the edges of a bipartite graph are related only to the co-occurrences, such that if a user i joins the community j, we introduce an edge connecting them.

It is not difficult to verify that the similarity matrix can be calculated from the adjacency matrix

$$S = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \tag{2.3}$$

where $A \in R^{n \times n'}$ is the adjacency matrix that indicates the co-occurrence of the users and communities, and n and n' are the number of communities and users, respectively.

Then the normalized graph Laplacian is

$$\bar{L} = \begin{bmatrix} I & -D_1^{-1/2} A D_2^{-1/2} \\ -D_2^{-1/2} A^T D_1^{-1/2} & I \end{bmatrix} \tag{2.4}$$

where D_1 and D_2 are diagonal matrices, calculated as $(D_1)_{ii} = \sum_{j=1}^{n'} A_{ij}$ and $(D_2)_{jj} = \sum_{i=1}^{n} A_{ij}$.

By using eigenvalue decomposition of the normalized graph Laplacian $\bar{L}f = \lambda f$ where $f = (f_1^T, f_2^T)^T \in \mathcal{R}^{(n+n')\times 1}$, we obtain

$$\begin{aligned} D_1^{-1/2} A D_2^{-1/2} f_1 &= (1-\lambda) f_2, \\ D_2^{-1/2} A^T D_1^{-1/2} f_2 &= (1-\lambda) f_1. \end{aligned} \tag{2.5}$$

Performing the SVD technique shows that f_1 and f_2 are the left and right singular vectors of the matrix $D_1^{-1/2} A D_2^{-1/2}$.

The above analysis pertains to the 2-way clustering problem. For the k-way (k is the number of clusters) clustering problem, many approaches have been proposed. For example, we can use the 2-way clustering algorithm to partition the data recursively $k-1$ times [118]. Other clustering algorithms, *e.g.*, k-means, can be used to cluster the embedded points in the eigenvector space [98]. Moreover, eigenvectors can be discretized into class indicators by means of matrix decomposition [150]. Because k-means is a fast way to cluster data and can be easily parallelized, we select this way to obtain the final k-way clustering results.

2.2 Parallel Spectral Clustering Algorithm

This section presents our parallel spectral clustering algorithm that can be used to cluster large-scale datasets.

Table 2.1: The traditional ARPACK algorithm.

1. Input: an $n \times n$ matrix S.

2. Start: Build a length m Arnoldi factorization

$$SV_m = V_m H_m + f_m e_m^T \tag{2.6}$$

with the starting vector v_1, where V_m is an $n \times m$ matrix, with normalized orthogonal columns derived from the Krylov subspace. H_m is the projection matrix (upper Hessenberg). $f_m e_m^T$ is the residual vector with length n.

3. Iteration: Until convergence.

3.1. Compute the eigenvalues $\{\lambda_j : j = 1, 2, ...m\}$ of H_m. Sort these eigenvalues according to the user selection criterion into a wanted set $\{\lambda_j : j = 1, 2, ...k\}$, and an unwanted set $\{\lambda_j : j = k+1, k+2, ..., m\}$.

3.2. Perform $m - k = l$ steps of the *QR* iteration with the unwanted eigenvalues $\{\lambda_j : j = k+1, k+2, ..., m\}$, as shifts to obtain $H_m Q_m = Q_m H_m^+$, where H_m^+ is the projection matrix in the next iteration.

3.3. Restart: Postmultiply the length m Arnoldi factorization with the matrix Q_k consisting of the leading k columns of Q_m to obtain the length k Arnoldi factorization $SV_m Q_k = V_m Q_k H_k^+ + f_k^+ e_k^T$ where is H_k^+ is the leading principal submatrix of order k for H_m^+. Set $V_k \leftarrow V_m Q_k$.

3.4. Extend the length K Arnoldi factorization to a length m factorization.

4. Calculate the eigenvalues and eigenvectors of the small matrix H_k: The eigenvalues of H_k, $\{\lambda_j : j = 1, 2, ..., k\}$, is the approximation of S's eigenvalues. The eigenvectors of H_k is $\{e_j : j = 1, 2, ..., k\}$, and E_k is the matrix formed by e_j.

5. Given $SV_k \approx V_k H_k$, we can derive the approximate eigenvectors of S, $\{u_j : j = 1, 2, ..., k\}$, where u_j is the j^{th} column of matrix $V_k \cdot E_k$.

2.2.1 Parallel Matrix Decomposition

Parallel matrix decomposition includes eigenvalue decomposition (EVD) and parallel singular value decomposition (SVD). First, we present the EVD problem, and then we show how the SVD problem can be converted into the EVD problem.

Parallel EigenValue Decomposition (EVD)

The traditional ARPACK algorithm (shown in Table 2.1) [77] calculates the approximated top k eigenvalues and the corresponding eigenvectors of a large matrix[2]. Given a matrix $S \in R^{n \times n}$, we build a length m Arnoldi factorization [9] as

$$SV_m = V_m H_m + f_m e_m^T \tag{2.7}$$

where $V_m \in R^{n \times m}$; $H_m \in R^{m \times m}$; $f_m e_m^T$ is the residual orthogonal to V_m and H_m is the projection of S in the space $Range(V_m)$. If $f_m e_m^T$ is small, H_m can be viewed as an approximation of S of dimension $m \times m$. Eigenvalues and eigenvectors of S can be calculated from H_m's eigenvalue decomposition:

$$\begin{aligned} SV_m &\approx V_m H_m \\ \lambda_j &\approx \delta_j, j \in \{1, 2, ..., m\} \\ u_j &\approx V_m e_j, j \in \{1, 2, ..., m\} \end{aligned} \tag{2.8}$$

[2] The traditional ARPACK algorithm, as used on a single computer to determine approximate eigenvectors for a large matrix S.

where the λ_j are the eigenvalues of matrix S, the δ_j are the eigenvalues of matrix H_m; the u_j are the eigenvectors of matrix S, and the e_j are the eigenvector of matrix H_m.

To parallelize the process, the data and work space are segmented and loaded onto multiple computers tha operate in parallel:

- S is distributed across the computers in a row-based, round-robin fashion.
- H_m is replicated on every computer.
- V_m is distributed across computers in a row-based, round-robin fashion.
- f_m and the workspace are distributed accordingly.

Distributed Matrix-Vector Multiplication

Compared to the single-computer algorithm, our parallel algorithm has the features that the local block of the set V_m^{local} is passed in place of V_m, and the dimension of the local block n_m^{local} is passed instead of n. Thus, we need to implement a matrix-vector multiplication to calculate the Krylov vectors. In our case, we divide the similarity matrix S into rows.

Figure 2.1 illustrates the matrix-vector multiplication on distributed computers. In each step, first we reduce each column of the Arnoldi vectors to a replicated vector using the standard message-passing interface. Although the rows of the similarity matrix are stored on different computers, the products of each local row by the replicate Arnoldi

vector can be locally computed. Therefore, the updated Arnoldi vectors are actually stored on different computers. The elements that correspond to the local rows of the similarity matrix are non-zero, whereas the other elements are still zero. By summing the results from all computers, matrix-vector multiplication is achieved.

In addition to matrix-vector multiplication, our algorithm requires two communications: Computing the L_2-norm of the distributed vector f_m, and orthogonalizing f_m to V_m. These can be performed by using the parallel computing summing interface.

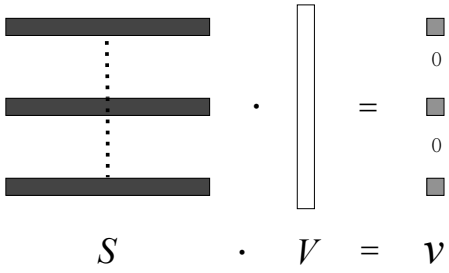


Figure 2.1: Illustration of the distributed matrix-vector multiplication.

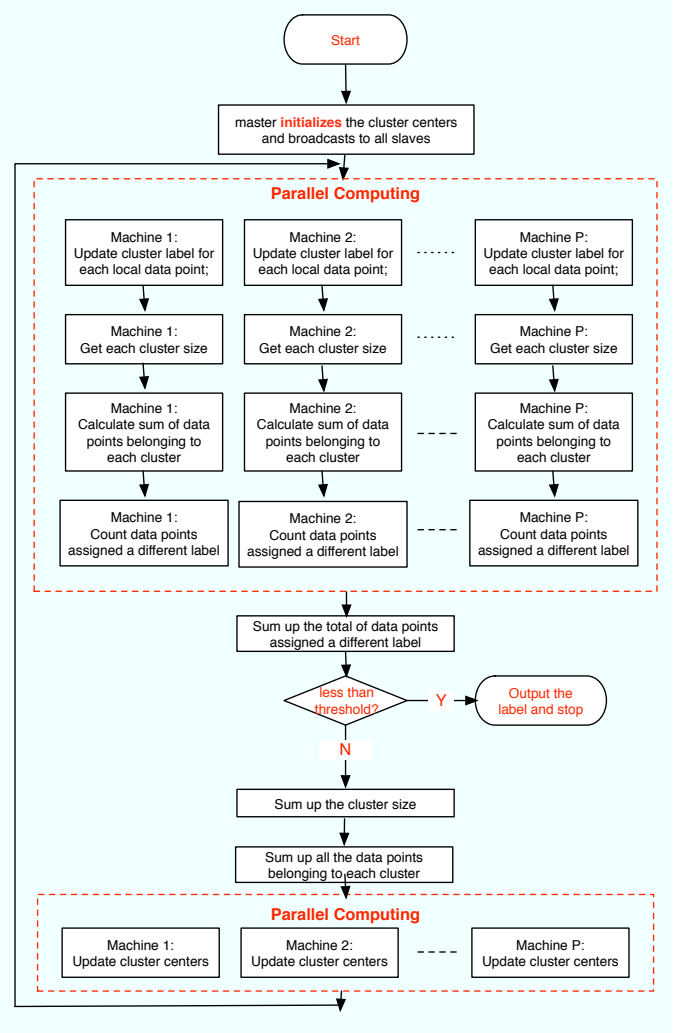


Figure 2.2: The parallel K-means clustering algorithm.

Parallel Singular Value Decomposition (SVD)

For each rectangular matrix $A \in R^{n \times n'}$, there exists a singular value decomposition:

$$A = USV^T, \tag{2.9}$$

where U (the left singular vectors) and V^T (the right singular vectors) are matrices with orthonormal columns and S is a diagonal matrix with singular values as the diagonal elements.

Given the Parallel EVD algorithm described in Section 2.2.1, we can calculate the SVD as follows:

$$A^T A = V S^2 V^T \tag{2.10}$$

$$U = AVS^{-1} \tag{2.11}$$

By calculating EVD on the matrix $A^T A$ using Equation (2.10), we can obtain the right singular vectors in the matrix V^T and the singular values in the matrix S. Equation (2.11) gives a solution of the left singular vectors U.

2.2.2 Parallel K-Means

The inputs to the k-means algorithm are the eigenvectors generated by the parallel EVD/SVD algorithm described in Section 2.2.1. The outputs of the k-means algorithm are the cluster labels of each data point in the original data space.

Here, the k-means algorithm aims to minimize the total intra-cluster variance, *i.e.*, the squared error function in the spectral space:

$$V = \sum_{i=1}^{k} \sum_{x_j \in C_i} ||x_j - \mu_i||^2 \tag{2.12}$$

where there are k clusters $C_i, \{i = 1, 2, ..., k\}$, and μ_i is the centroid or mean point of all the points $x_j \in C_i$.

We implemented the parallel k-means algorithm in such a way to minimize communication and maximize parallel computation. The flowchart of the algorithm is shown in Figure 2.2. In the parallel EVD algorithm, the output matrix U is formed by the eigenvectors and is distributed across all computers based on the rows. Each row of the matrix U is regarded as one data point for the k-means algorithm. These data points are naturally distributed on the computers, and don't need to be moved them for the k-means algorithm.

To initialize the process, the master computer chooses a set of initial cluster centers and broadcasts the coordinates of the centers to all of the computers. Each computer works on its local data independently. New labels are assigned and local sums of clusters are calculated without any inter-computer communication. Again, we make use of the message-passing interface to combine the local information after each local computer has finished the computation. By gathering the statistical information (including the sum of data points in each cluster, the cluster numbers and the local cost values), each computer can update the cluster center coordinates and start a new round of computation until the computation converges. The output cluster labels for data points in the spectral space are mapped to the original data space.

Table 2.2: The parallel spectral clustering algorithm.

1. Each computer loads a set of rows of the similarity matrix S into memory.
2. Multiply the matrix S with vector $\vec{1} = [1, 1, ..., 1]^T$. The product vector is the diagonal elements of the matrix D.
3. Calculate the scaled similarity matrix $\bar{S}$.
4. Compute the approximated eigenvalue decomposition of $\bar{S}$ using parallel matrix decomposition.
5. Use parallel k-means to cluster the rows of matrix U.
6. Map the cluster labels to original data points.

Table 2.3: Spectral clustering matrix comparison.

Form of $\bar{S}$	Method
$X^T X$	Relaxed k-means
Gram matrix G	Relaxed kernel k-means
Similarity matrix on graph	Min-cut
$D^{-\frac{1}{2}} S D^{-\frac{1}{2}}$	Normalized cut
$\bar{A}\bar{A}^T$ where $\bar{A} = D_1^{-\frac{1}{2}} A D_2^{-\frac{1}{2}}$	Co-clustering

2.2.3 Complexity Comparison

Our algorithm is shown in Table 2.2. Steps 4 and 5 are the key parallelization steps. For step 3, we do not constrain the form of the scaled similarity matrix $\bar{S}$. If we use the original similarity $\bar{S} = X^T X$, we obtain the relaxed version of k-means. If we use $\bar{S} = G$ where G is the Gram matrix computed by the kernel function, we obtain the relaxed kernel k-means algorithm. If the matrix $\bar{S}$ is constructed by a graph similarity

matrix, which can be either fully connected (can be the same as kernel k-means) or a neighborhood graph, we obtain the min-cut algorithm. If we use the normalized similarity matrix $\bar{S} = D^{-\frac{1}{2}}SD^{-\frac{1}{2}}$, we obtain the normalized cut algorithm. For the co-clustering problem, we input the matrix $\bar{A} = D_1^{-\frac{1}{2}}AD_2^{-\frac{1}{2}}$ and then compute $\bar{A}\bar{A}^T$ as $\bar{S}$. We summarize the above analysis in Table 2.3.

Now, we analyze the memory requirement and the computational complexity. We use n to denote the number of data points, d to denote the dimensionality, and k to denote the number of clusters. Here, we introduce a new variable z. Because we assume that the data similarity matrix is sparsely stored, we let z denote the mean number of rows in the similarity matrix. For the iterated algorithms, we let i_{iter} denote the iteration time. If we have p computers, the computational complexity of the key steps is determined as follows:

k-means. For the traditional k-means algorithm, the memory requirement is $\mathcal{O}(nd)$ and the computational complexity is $\mathcal{O}(ndk \cdot i_{iter})$, because we need to compute the Euclidean distance between every point and every cluster center.

Parallel k-means. For parallel k-means, the memory requirement is reduced to $\mathcal{O}(\frac{nd}{p})$ for each computer and the computational complexity is reduced to $\mathcal{O}(\frac{ndk}{p} \cdot i_{iter})$. Because the parallel algorithm also involves communication among computers, we need to estimate the communication time. Most of the calculation is done in paral-

lel. Only the summation is performed repeatedly on each computer. Therefore, the communication time is $\mathcal{O}(pkd \cdot i_{iter})$.

Spectral Clustering. For spectral clustering based on the Arnoldi method, the memory requirement of loading the similarity matrix and eigenvectors is $\mathcal{O}(n(z+k))$. The computational complexity of the eigenvalue decomposition of the similarity matrix is $\mathcal{O}(nzk \cdot i_{iter})$.

Parallel Spectral Clustering. For our parallel spectral clustering algorithm, the memory requirement for each computer is $\mathcal{O}(\frac{n(z+k)}{p})$ and the computational complexity is $\mathcal{O}(\frac{nzk \cdot i_{iter}}{p})$. Moreover, because we compute the Arnoldi vector using the message-passing interface, the communication cost is $\mathcal{O}(pnk \cdot i_{iter})$.

Those costs are summarized in Table 2.4.

Table 2.4: Computational cost comparison. P. k-means represents parallel k-means, S. C. represents spectral clustering and P. S. C. represents parallel spectral clustering.

Method	Memory	Comp. Time	Comm. Time
k-means	$\mathcal{O}(nd)$	$\mathcal{O}(ndk \cdot i_{iter})$	-
P. k-means	$\mathcal{O}(\frac{nd}{p})$	$\mathcal{O}(\frac{ndk}{p} \cdot i_{iter})$	$\mathcal{O}(pdk \cdot n_{iter})$
S. C.	$\mathcal{O}(n(n+k))$	$\mathcal{O}(nzk \cdot i_{iter})$	-
P. S. C.	$\mathcal{O}(\frac{n(z+k)}{p})$	$\mathcal{O}(\frac{nzk \cdot i_{iter}}{p})$	$\mathcal{O}(pnk \cdot i_{iter})$

2.3 Experiments

First, we conducted experiments on artificial datasets to investigate the accuracy and time cost of our parallel algorithm. Then, we performed scalability experiments on a large real-world dataset. We ran all of our experiments on Google's production data centers.

2.3.1 Accuracy Experiments

For the accuracy experiments, we collected nine datasets with different sizes and numbers of clusters. These nine datasets consist of 1k, 10k, and 100k data points distributed across 4, 9 and 16 non-overlapping circles, as shown in Table 2.5. We denote these datasets as C1 to C9.

Table 2.5: Description of datasets.

	4 clusters	9 clusters	16 clusters
1K data points	C1	C4	C7
10K data points	C2	C5	C8
100K data points	C3	C6	C9

(a) 4 classes. (b) 9 classes. (c) 16 classes.

Figure 2.3: Artificial test datasets.

Figure 2.3 shows three of the above nine datasets for the purposes of illustration. Pairwise similarity between two data points is calculated using an RBF kernel function. The width of the RBF kernel is tuned by the self-tuning technique of [149]. Then, the RBF is modified as

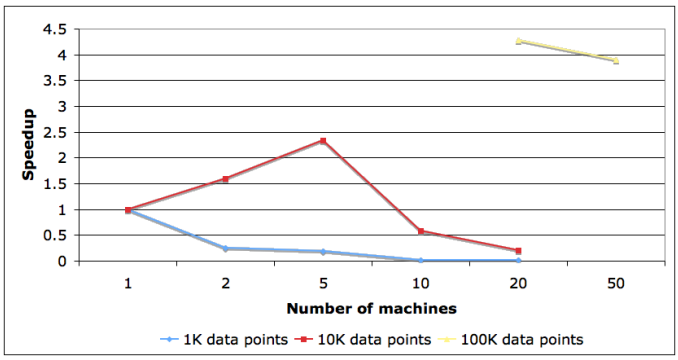
$$S_{ij} = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma_i\sigma_j}\right) \tag{2.13}$$

where $\sigma_i = ||x_i - x_{i_k}||$, the distance between x_i and k's neighborhood of x_i. For the neighborhood graphs, we set k equal to one-half of the neighborhood number.

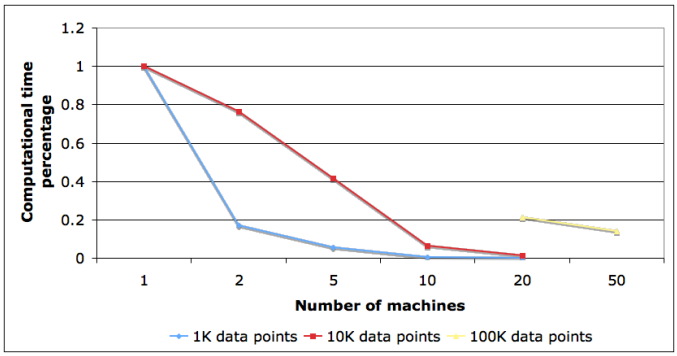
The Speedup Factor

Ideally, with p computers, we have a linear speedup, compared to a single computer. However, because of the communication overhead, the speedup is usually not linear. The speedup factor is defined as follows:

$$speedup = \frac{T_1}{T_p} \tag{2.14}$$



(a) Algorithm speedup of different scale of data.



(b) Ratio between computation time and communication time.

Figure 2.4: Time analysis of parallel spectral clustering.

where T_1 is the execution time using one computer, and T_p is the execution time using p computers.

Results

We applied parallel spectral clustering on all of the artificial datasets. The purpose of this experiment is to evaluate the accuracy of the clustering results. (Using multiple

computers on a small dataset does not yield much benefit, as we will see shortly.) We compared the clusters generated by the original spectral clustering algorithm and our parallel version, and they yield identical results.

We document the running time of these nine datasets in Table 2.6[3]. Each dataset was run on 1, 2, 5, 10, 20, and 50 computers, respectively. As predicted, when the dataset size is very small, the running time for the datasets C1, C4, and C7 shows that adding computers actually increases the total running time. The reason is that inter-computer communication results in greater time than parallelization can save. When the dataset size grows from $1k$ to $10k$, parallelization yields a benefit. When using up to 10 computers, C8 enjoys a speedup of about 2.2 times. When the dataset continues to grow beyond what the main memory of one computer can store, we have to employ enough computers to do the job. For the datasets C3, C6, and C9, we can complete the clustering task only when 20 or 50 computers are used.

[3]Because we conducted experiments on Google's production data centers, we could not ensure that all these computers are fully dedicated to our task. Therefore, the running time is partially dependent on the slowest computer being allocated for the task.

Table 2.6: Algorithm running time on different datasets using multiple computers.

Data	Number of computers					
	1	2	5	10	20	50
C1	2.952s	7.709s	21.70s	465.0s	503.2s	
C2	199.5s	139.8s	58.70s	62.13s	589.1s	
C3	NA	NA	NA	NA	NA	343.4s
C4	1.936s	5.548s	21.89s	120.2s	232.1s	
C5	140.96s	67.63s	51.71s	283.6s	91.72s	
C6	NA	NA	NA	NA	558.5s	348.8s
C7	1.570s	5.452s	20.43s	17.65s	52.36s	
C8	281.22s	255.80s	185.92s	132.77s	491.9s	
C9	NA	NA	NA	NA	757.3s	820.4s

Given the total time spent on each task, we can calculate the speedup using Equation (2.14). The results are shown in Figure 2.4(a). As the problem scale grows, the speedup can be more significant, which implies that our parallel spectral clustering algorithm is more efficient for large-scale problems than for small ones. Figure 2.4(b) shows the percentage of time spent on computation. The main factor that affects the percentage of computation time is the problem scale. Using a fixed number of computers, the percentage of computation time for $10k$ datasets is larger than that of the three $1k$ datasets. Again, this substantiates that our algorithm is more efficient for large-scale problems.

2.3.2 Experiments Using Text Data

In this experiment, we used the pre-processed 20 newsgroups dataset given in [160] to investigate the accuracy of our parallel spectral clustering algorithm. The dataset originally included $20,000$ messages within 20 different newsgroups. The data were pre-processed by the Bow toolkit [90]. We chopped off the headers, removed stop words and also words that occurred in fewer than three documents [160]. Thus, the document is represented by a feature which is a $43,586$ dimensional sparse vector. Several empty documents were also removed [160]. Finally we obtained $19,949$ examples.

For comparison of the results, we used the Normalized Mutual Information(NMI) method to evaluate the algorithms. NMI between two random variables Y_1 and Y_2 is defined as $NMI(Y_1;Y_2) = \frac{I(Y_1;Y_2)}{\sqrt{H(Y_1)H(Y_2)}}$, where $I(Y_1;Y_2)$ is the mutual information between Y_1 and Y_2. The entropies $H(Y_1)$ and $H(Y_2)$ are used for normalizing the mutual information to be in the range $[0,1]$. To estimate the NMI score, we used the following formulation [125, 160]:

$$NMI = \frac{\sum_{s=1}^{K}\sum_{t=1}^{K} n_{s,t}\log\left(\frac{nn_{s,t}}{n_s \cdot n_t}\right)}{\sqrt{\left(\sum_s n_s \log\frac{n_s}{n}\right)\left(\sum_t n_t \log\frac{n_t}{n}\right)}} \tag{2.15}$$

where n denotes the number of data points, n_s and n_t denote the number of data points in class s and cluster t, $n_{s,t}$ denotes the number of data points in class s and cluster t. The NMI score is 1 if the clustering results perfectly match the category labels; it is

0 if the clustering algorithm returns a random partition. Thus, the larger the score, the better are the clustering results.

Table 2.7: Comparison result for text categorization.

Method	NMI
E-k-means	0.10±7.0e-05
S-k-means	0.30±1.6e-06
Co-clustering	0.54±3.6e-06
Normalized cut	0.55±4.9e-05

We compared the following algorithms: relaxed k-means algorithm based on the Euclidean distance (E-k-means), the relaxed spherical k-means based on the cosine distance (S-k-means) [44], the co-clustering algorithm [40], and the normalized cut algorithm using the 30 neighborhood adjacency graph (without weights on graph edges) [118]. The results are shown in Table 2.7. We see that the normalized cut algorithm performs the best. The parallel normalized cut on the $20k$ documents using 5 computers took only about 10 seconds to complete.

2.3.3 Experiments Using Orkut Data

Social networks have become increasingly popular. The development of those social networks has enabled people to find new friends with common interests. User can create communities as well as join existing communities on the Web. Orkut is an Internet social network service run by Google. Since October 2006, Orkut has permitted

Table 2.8: Cluster examples.

Sample Cluster 1: *Cars*		**Sample Cluster 2:** *Food*	
Community ID	Community title	Community ID	Community title
22527	Honda CBR	622109	Seafood Lovers
287892	Mercedes-Benz	20876960	Gol gappe
35054	Valentino Rossi	948798	I LOVE ICECREAM
5557228	Pulsar Lovers	1614793	Bounty
2562120	Top Speed Drivers	1063561	Old Monk Rum
19680305	The Art of DriftIng	970273	Fast Food Lovers
3348657	I Love Driving	14378632	Maggi Lovers
726519	Luxury & Sports Cars	973612	Kerala Sadya
2806166	Hero Honda Karizma	16537390	Baskin-Robbins Ice Cream
1162256	Toyota Supra	1047220	Oreo Freax!!
Sample Cluster3: *Education*		**Sample Cluster4:** *Pets, animals, wildlife*	
Community ID	Community title	Community ID	Community title
15284191	Bhatia Commerce Classes	18341	Tigers
7349400	Inderprastha Engineering Cllge	245877	German shepherd
1255346	CCS University Meerut	40739	Naughty dogs
13922619	Visions - SIES college fest	11782689	We Love Street Dogs
2847251	Rizvi College of Engg., Bandra	29527	Animal welfare
6386593	Seedling public school, jaipur	370617	Lion
4154	Pennsylvania State University	11577	Arabian horses
15549415	N.M. College, Mumbai	2875608	Wildlife Conservation
1179183	Institute of Hotel Management	12522409	I Care For Animals
18963916	I Love Sleeping In Class	1527302	I hate cockroaches

users to create accounts without an invitation; now, Orkut has more than 50 million users and 20 million communities.

In our experiments, we used Orkut's user-by-community co-occurrence data. All of the users are anonymized, and each community is associated with a name and an optional description. To make the clustering results readable, first we filtered out the non-English-language communities. We also removed inactive communities that contain few users. We obtained $151,973$ communities with more than 10 million users.

We ran our parallel spectral clustering algorithm on 90 computers to group the communities into 100 clusters. The program finished within 20 minutes. Communities with similar topics are clustered together. We choose four clusters among the clustering results. Popular communities are listed in Table 2.8 as representative examples of the clusters.

2.4 Summary

This chapter presented a parallel approach for spectral graph analysis, including spectral clustering and co-clustering. By using multiple computers in a distributed system, we have increased the scalability of spectral methods in both computation time and memory use. This approach makes it possible to analyze Web-scale data using spectral methods. Experiments show that our parallel spectral clustering algorithm

performs accurately on artificial datasets and real text data. We also applied our parallel spectral clustering algorithm to a large Orkut dataset to demonstrate its scalability.

Chapter 3

Extraction and Integration of Data from Distributed Sources

Fully automatic methods that extract lists of objects from the Web have been studied extensively. Record extraction, the first step of this object extraction process, identifies a set of Web page segments, each of which represents an individual object (*e.g.*, a product). State-of-the-art methods suffice for simple search, but they often fail to handle more complicated or noisy Web page structures due to a key limitation – their greedy manner of identifying a list of records through pairwise comparison (*i.e.*, similarity match) of consecutive segments. This chapter introduces a novel method for record extraction that captures a list of objects in a more robust way based on a holistic analysis of a Web page. The method focuses on how a distinct *tag path* appears repeatedly in the

DOM tree of the Web document. Instead of comparing a pair of individual segments, it compares a pair of tag path occurrence patterns (called *visual signals*) to estimate how likely these two tag paths represent the same list of objects. The chapter introduces a similarity measure that captures how closely the visual signals appear and interleave. Clustering of tag paths is then performed based on this similarity measure, and sets of tag paths that form the structure of data records are extracted. Experiments show that this method achieves higher accuracy than previous methods.

3.1 Motivation

The Web contains a large amount of structured data, and serves as a good user interface for databases available over the Internet. A large amount of Web content is generated from databases in response to user queries. Such content is sometimes referred to as the *deep Web*. A deep Web page typically displays search results as a list of objects (*e.g.*, products) in the form of structured data rendered in HTML. A study in 2004 found 450,000 databases in the deep Web [31]. Structured data also plays a significant role on the *surface Web*. Google estimated that their crawled dataset contains 154 million *Web tables*, *i.e.*, relational data rendered as HTML tables [27]. In addition to relational tables, the Web contains a variety of lists of objects, such as conference programs and comment lists in blogs. It is an important and challenging

task to identify such object lists embedded in Web pages in a scalable manner, which enables not only better search engines but also various applications related to Web data integration (*i.e.*, data mashups) and Web data mining (*e.g.*, blog analysis).

There have been extensive studies of fully automatic methods to extract lists of objects from the Web [8, 35]. A typical process to extract objects from a Web page consists of three steps: record extraction, attribute alignment, and attribute labeling. Given a Web page, the first step is to identify a *Web record* [81], *i.e.*, a set of HTML regions, each of which represents an individual object (*e.g.*, a product). The second step is to extract object attributes (*e.g.*, product names, prices, and images) from a set of Web records. Corresponding attributes in different Web records are aligned, resulting in spreadsheet-like data [152, 159]. The final step is the optional task (which is very difficult in general) of interpreting aligned attributes and assigning appropriate labels [136, 163].

In this chapter we focus on Web record extraction. Our study is motivated by our experience in developing an automatic data extraction component of a data mashup system [130], where we scrape a set of objects from a *variety* of Web pages automatically. The extraction component, developed with existing state-of-the-art technologies, sometimes fails at the very first step, *i.e.*, record extraction, which significantly affects the entire mashup process.

Most state-of-the-art technologies for Web record extraction employ a particular similarity measure between Web page segments to identify a region in the page where a similar data object or record appears repeatedly. A representative example of this approach is MDR [81], which uses the edit distance between data segments (called *generalized nodes*). By traversing the DOM tree of a Web document, MDR discovers a set of consecutive sibling nodes that form a data region. More recent work [120, 152] extends this approach by introducing additional features such as the position of the rendered data. In our experience, an approach based on MDR is sufficient for simple search, but it starts to fail as the Web page structure becomes more complicated.

We observe that, on many Web pages, objects are rendered in a highly decorated manner, which affects the quality of extraction. For instance, an image that is inserted between objects as a separator makes objects no longer consecutive. As a work around, we employ a heuristic rule to exclude decorative images from the DOM tree. In fact, such visual information can be helpful or harmful. A heuristic rule might utilize such decorations to identify object boundaries. However, it is not easy to generalize such a heuristic rule so that it applies to a variety of Web pages. Thus, in general, the irregularity that decorative elements introduce is more harmful than helpful. Moreover, as [159] notes, the same HTML tag can sometimes work as a template token (that contributes to form an object structure) and can sometimes work as a decorative element

(that is used in an unstructured manner). Such tags can be very noisy but, if the algorithm ignores these tags, it can miss useful evidence of structured objects.

We also observe that objects are sometimes embedded in a complicated Web page structure with various context information. In such cases, objects are not necessarily rendered consecutively. Existing work tries to address such complex Web page structures [8, 158]. However, that work typically assumes availability of multiple Web page instances.

A key limitation that we have identified in the MDR approach is its greedy manner of identifying a *data region* (a region containing records) through pairwise comparison of consecutive segments. In many cases, one misjudgment due to noise causes separation of an object list into multiple lists. We can imagine an extended algorithm that employs more sophisticated search for data regions instead of the greedy approach, but its computational cost is very high.

We have developed an alternative approach to the Web record extraction problem, which captures a list of objects based on a holistic analysis of a Web page. Our method focuses on how a distinct *tag path* (*i.e.*, a path from the root to a leaf in the DOM tree) appears repeatedly in the document. Instead of comparing a pair of individual subtrees in the data, we compare a pair of tag path occurrence patterns (called *visual signals*) to estimate how likely these two tag paths represent the same list of objects. We introduce a similarity measure that captures how closely the tag paths appear and

how they interleave. We apply clustering of tag paths based on this similarity measure, and extract sets of tag paths that form the structure of the data records.

Compared to existing approaches, our method has the following advantages:

- Data records do not have to be consecutive. Based on the discovery of non-consecutive data records, our method can also detect nested data records.
- Template tags and decorative tags are distinguished naturally. When a tag (path) appears randomly in unstructured content, the corresponding visual signal will not be similar to other signals. A tag (path) is clustered based on the structure of the data records only when it repeats similarly to other tags.

3.2 Related Work

Extracting structured data from HTML pages has been studied extensively. Early work on wrapper induction utilizes manually labeled data to learn data extraction rules [74]. Such semi-automatic methods are not scalable enough for extraction of data on the scale of the Web. To address this limitation, more fully automatic methods have been studied recently. Fully automatic methods address two types of problems: (1) extraction of a set of objects (or data records) from a single page, and (2) extraction of underlying templates (or schema) from multiple pages [8, 35]. Our work focuses on the

former, which does not assume the availability of multiple instance pages containing similar data records.

Techniques that address record extraction from a single page can be categorized into the following approaches, which evolved in this order: (a) early work based on heuristics [23], (b) mining repetitive patterns [30, 136], and (c) similarity-based extraction [81, 120, 159]. OMINI [23] applies a set of heuristics to discover separator tags between objects in a Web page, but is applicable to only simple cases. IEPAD [30] identifies substrings that appear multiple times in a document encoded as a token string. DeLa [136] extends that approach to support nested repetition, such as "(AB*C)*D". One limitation of such a pattern mining approach is that it is not robust against optional data inserted into records. The similarity-based approach tackles this limitation with approximate matching to identify repeating objects. MDR [81] is one such technique, which utilizes edit distance to assess whether two consecutive regions are a repetition of the same data type. It is reported that MDR out-performs both OMINI and IEPAD.

As discussed previously, even similarity-based extraction has limitations when the data are complex and noisy. MDR relies on a greedy approach based on a similarity match between two segments, with a pre-determined threshold. A limitation of MDR is that it does not handle nested data objects. The researchers who developed MDR proposed an extended algorithm, NET, to address this issue [82]. NET handles nested objects by traversing a DOM tree in post-order (bottom-up), whereas MDR traverses

the tree in pre-order (top-down). When a list of objects is discovered during traversal, the list is collapsed into a single object (pattern) so that the number of objects does not affect detection of higher-layer objects. However, NET still employs a greedy approach based on similarity match. Moreover, its bottom-up traversal with edit distance comparison is expensive. Whereas MDR's top-down traversal can stop as soon as it finds data records, NET's bottom-up traversal requires a full scan from the bottom up to the root. For each visit of a node in this traversal, NET executes all-pair tree comparisons within its children.

Other work extends the similarity approach by incorporating a variety of additional features such as visual layout information [157] and hyperlinks to detail pages [78]. However, without any assumptions about the target domain, it is difficult to identify such additional features. Moreover, such features are not always available or generally useful. In future work, we plan to extend our method to incorporate additional feature information.

Our method focuses on record extraction and does not extract detailed data in a record. There exist other techniques that address extraction and alignment of attributes in records [152, 159]. Our method can be combined with those techniques to realize the entire data extraction process.

Among existing approaches for template extraction from multiple pages, EXALG [8] is related to our method in its key idea. EXALG identifies a set of tokens that forms

a template based on the intuition that tokens that co-occur with the same frequency within multiple pages are likely to form the same template. Whereas EXALG utilizes occurrence patterns across multiple documents, our method utilizes occurrence patterns within a single document. Thus, the two algorithms are very different.

3.3 Methodology

Although automatically identifying and extracting data records from Web pages is considered a hard problem in the computing community, it is fairly easy for human beings to identify such records. The data records that constitute a Web page are typically represented using an HTML code template. Thus, they often have a similar appearance and are visually aligned. Such a visually repeating pattern can be easily captured by human eyes, and the data records in the visually repeating part can be accurately located. Inspired by this observation, our method comprises three steps: (1) detecting visually repeating information, (2) data record extraction, and (3) semantic-level nesting detection. The first step addresses the problem of what appears repeatedly on the Web page. The second step extracts the data records from the HTML blocks where the repeating patterns occur. The third step extracts the high-level data objects when there is a nested list. The method is fully automatic and does not involve human labeling or feedback. The three steps are described in more detail below.

3.3.1 Detecting Visually Repeating Information

A *data region* is part of a Web page that contains multiple data records of the same kind, which can be consecutive or non-consecutive. Instead of viewing the Web page as a DOM tree, we consider it as a string of HTML tags. A data region maps to one or more segments of the string with a repeating texture composed of HTML tags, which result in the visually repeating pattern rendered on a Web page. We aim to find the HTML tags that are elements of the data regions.

Visual Signal Extraction

The visual information rendered on a Web page, such as fonts and layout, is conveyed by HTML tags. A given hyperlink tag can have different appearances when it follows different paths in the DOM tree. For each tag occurrence, there is an HTML *tag path*, containing an ordered sequence of ancestor nodes in the DOM tree. Figure

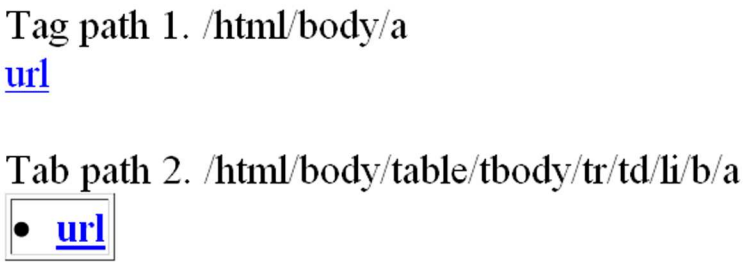


Figure 3.1: Hyperlinks following different tag paths.

Table 3.1: Finding tag paths for HTML tags.

HTML code	Pos	Tag path
<html>	1	html
<body>	2	html/body
A Web page	3	html/body/h1
<table>	4	html/body/table
<tr>	5	html/body/table/tr
<td> Cell #1 </td>	6	html/body/table/tr/td
</tr>	NA	NA
<tr>	7	html/body/table/tr
<td> Cell #2 </td>	8	html/body/table/tr/td
</tr></table></body> </html>	NA	NA

Table 3.2: Extracting visual signals from a Web page.

Unique tag path	Pos	Visual signal vector
html	1	$[1,0,0,0,0,0,0,0]$
html/body	2	$[0,1,0,0,0,0,0,0]$
html/body/h1	3	$[0,0,1,0,0,0,0,0]$
html/body/table	4	$[0,0,0,1,0,0,0,0]$
html/body/table/tr	5,7	$[0,0,0,0,1,0,1,0]$
html/body/table/tr/td	6,8	$[0,0,0,0,0,1,0,1]$

3.1 shows the different appearances of hyperlink tags defined by two different HTML tag paths.

A Web page can be viewed as a string of HTML tags, where only the opening position of each HTML tag is considered. Each HTML tag maps to an HTML tag path. An example is shown in Table 3.1. Roughly speaking, each tag path defines a unique visual pattern. Our goal is to mine the visually repeating information in the Web page using this simplified representation.

An inverted index characterizing the mappings from HTML tag paths to their locations in the HTML document can be built for each Web page, as shown in Table 3.2. Each indexed term in the inverted index, *i.e.*, one of the unique tag paths, is defined to be a visual signal.

Formally, a *visual signal* s_i is a triple $< p_i, S_i, O_i >$, where p_i is a tag path, S_i is a *visual signal vector* that represents occurrence positions of p_i in the document, and O_i represents individual occurrences (*i.e.*, DOM tree nodes). S_i is a binary vector where $S_i(j) = 1$ if p_i occurs in the HTML document at position j and $S_i(j) = 0$ otherwise. O_i is an ordered list of occurrences $(o_i^1, \cdots, o_i^m)$, where o_i^k corresponds to the kth occurrence of 1 in S_i.

Examples of visual signal vectors are shown in the third column of Table 3.2. All of the visual signal vectors extracted from a Web page have the same length, which is the total number of HTML tag occurrences in the Web page.

The vector representation $\{S_i\}$ of a Web page is much simpler than the DOM tree representation. It also captures how a Web page is organized. Figure 3.3(a) shows a snapshot of a DBLP [3] Web page containing lists of publication records and other data objects. The extracted visual signals and the visual signal vectors are shown in Figures 3.3(b) and 3.3(d). Each row in Figure 3.3(d) is a visual signal vector. We show here only the first part of each visual signal vector. The visual signal vectors represent how each atomic-level visual pattern repeats in the Web page. The visually repeating

patterns in a Web page involve multiple visual signals. These visual signals together form a certain repeating texture as shown in Figure 3.3(d). Each texture corresponds to a data region that contains multiple data records of the same kind.

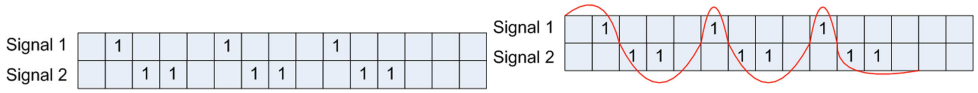
Detecting the visually repeating information is equivalent to identifying the set of visual signals with similar patterns that are elements of the same data region. In other words, detecting visually repeating information is a *clustering problem*. The visual signals in the same data region are grouped together, while the visual signals not in the same data region are split into different clusters. We use spectral clustering [98] to cluster the visual signals, because of its superior experimental performance and theoretical soundness.

Similarity Measurement

The spectral clustering algorithm produces clustering results based on the pairwise similarity matrix calculated from the data samples. A similarity function captures the likelihood that two data samples belong to the same cluster. A critical factor in determining clustering performance is the choice of similarity function.

In our case, the similarity function captures how likely two visual signals belong to the same data region. Figure 3.2(a) shows a pair of visual signals that are highly likely to belong to the same data region. Their positions are close to each other, and

they interleave with each other. Every occurrence of visual signal 1 is followed by two occurrences of visual signal 2.



(a) A pair of similar visual signal vectors. (b) Segmented visual signal vectors.

Figure 3.2: Example pair of visual signals that appear regularly.

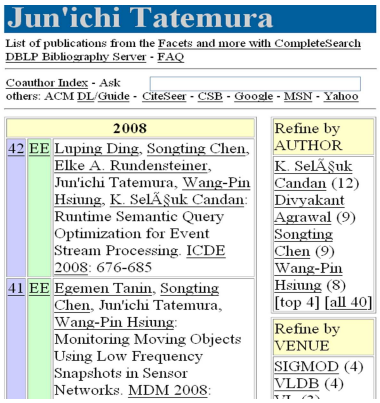
The distance between the centers of gravity of two visual signals characterizes how close they appear. We call this measure the *offset* ω and calculate it in Equation (3.1).

$$\omega(S_i, S_j) = \left| \frac{\sum_{S_i(k)=1} k}{\sum S_i(k)} - \frac{\sum_{S_j(k)=1} k}{\sum S_j(k)} \right| \tag{3.1}$$

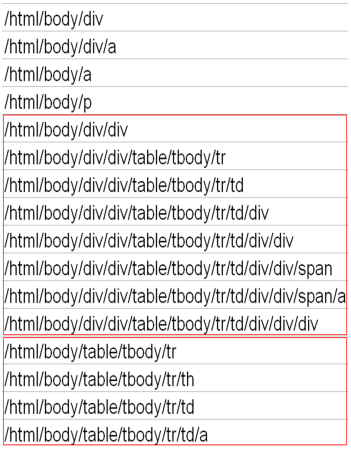
In Equation (3.1), S_i and S_j are two visual signal vectors and $k \in \{1, 2, ..., l\}$, where l is the length of the visual signal vectors, and $S_i(k)$ is the kth element of S_i.

To capture the interleaving characteristic, we estimate how *evenly* one signal is *divided by* the other. We define *a segment of* S_i *divided by* S_j as follows: a segment is a (non-empty) set of occurrences of visual signal s_i between any pair of which there is no occurrence of visual signal s_j. Figure 3.2(b) illustrates how two signals divide each ohter. Let D_{S_i/S_j} be the occurrence counts in the segments of S_i divided by S_j. In our example, $D_{S_1/S_2} = \{1, 1, 1\}$ and $D_{S_2/S_1} = \{2, 2, 2\}$. We define the *interleaving measure* ι in terms of the variances of counts in D_{S_i/S_j} and D_{S_j/S_i} in Equation (3.2).

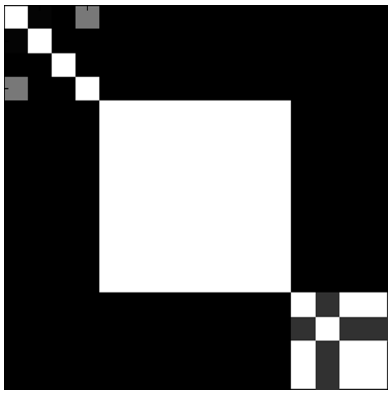
$$\iota(S_i, S_j) = \max\{Var(D_{S_i/S_j}), Var(D_{S_j/S_i})\} \tag{3.2}$$



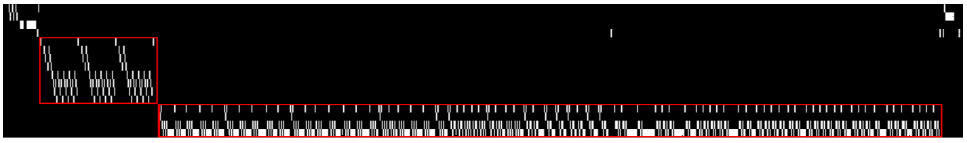
(a) Web page snapshot.



(b) Unique HTML tag paths.



(c) Pairwise similarity matrix.



(d) Visual signal vectors. Each row is a visual signal vector. Bright pixels correspond to 1s and dark pixels correspond to 0s.

Figure 3.3: Pairwise similarity matrix calculated from Equation (3.3).

Both the offset measure and the interleaving measure yield non-negative real numbers. A smaller value of either measure indicates a high probability that the two visual signals come from the same data region. The *similarity measure* $\sigma(s_i, s_j)$ between two visual signals is inversely proportional to the product of these two measures and is defined by Equation (3.3).

$$\sigma(s_i, s_j) = \frac{\varepsilon}{\omega(S_i, S_j) \times \iota(S_i, S_j) + \varepsilon} \tag{3.3}$$

In Equation (3.3), ε is a non-negative term that avoids dividing by 0 and that normalizes the similarity value so that it falls into the range $(0, 1]$. In our experiments, we chose $\varepsilon = 10$.

Given Equation (3.3), we can calculate the similarity value of any pair of visual signals. Example results are shown in Figure 3.3(c). The pixel in the ith row and jth column shows the similarity value for visual signal s_i and visual signal s_j. A bright pixel indicates a high similarity value, whereas a dark pixel indicates a low similarity value. Thus, the visual signals in Figure 3.3(b) aligned with the large bright blocks in Figure 3.3(c) are likely to be from the same data region. The visual signals actually involved in the data regions (*i.e.*, the *ground truth*) are highlighted in rectangles. Each box includes one data region. As expected, the similarity measure captures the likelihood of two visual signals being from one data region.

Visual Signal Clustering

The pairwise similarity matrix can be fed into a spectral clustering algorithm directly. We employ the normalized cut spectral clustering algorithm developed by Shi *et al.* [118] to produce the groups of visual signals with similar patterns. A cluster containing n visual signals indicates that those n visual signals are from the same data region with high probability.

A data region contains multiple data records that use the same HTML code template, and a template typically has multiple HTML tags that differentiate the data attributes. The *size of a template* is defined to be the number of unique HTML tag paths involved in the template. Thus, a template with size greater than n should correspond to a cluster containing more than n visual signals. Given the fact that most HTML code templates contain more than three HTML tags that differentiate different data attributes, we assume that the smallest size of a template is three. Thus, we need to examine only the clusters containing three or more visual signals. We call these clusters the *essential clusters* of the document, and denote them by $\mathcal{C} = \{C_1, \cdots, C_m\}$.

In the example shown in Figure 3.3, there are two clusters of size greater than three produced by the spectral clustering algorithm. These clusters correspond to the *ground truth*, *i.e.*, match the visual signals involved in the data regions exactly, as shown in Figure 3.3(b), where each cluster corresponds to one data region and contains a set of homogenous data records, as shown in Figure 3.3(a).

3.3.2 Data Record Extraction

Visual signals that are grouped together in an essential cluster $C \in \mathcal{C}$ should represent the same data region. Each occurrence o_i^k of visual signal s_i in C represents part of a data record, an entire data record, or a set of data records. The goal of data record extraction is to identify occurrences that represent individual data records.

To find such occurrences, we introduce ancestor and descendant relationships between visual signals. We say that s_i is an *ancestor* of s_j, denoted by $s_i//s_j$, iff p_i is a prefix of p_j. For example, $/html/body/p$ is an ancestor of $/html/body/p/a$. We also employ the standard relationships between occurrences o_i and o_j by viewing them as DOM nodes: $o_i//o_j$ (o_i is an ancestor of o_j in a DOM tree), and $o_i < o_j$ (o_i is a predecessor of o_j in the document order).

If $s_i//s_j$ then, for each $o_j \in O_j$, there exists $o_i \in O_i$ such that $o_i//o_j$, meaning that the HTML region represented by s_i contains the region represented by s_j. Recall that, if s_i and s_j are clustered together, they are in the same data region. Thus, an ancestor visual signal s_i is more likely to represent a set of entire data records while a descendant visual signal s_j is more likely to represent a set of data attributes.

Among the visual signals in an essential cluster C, there is at least one visual signal that has no ancestor in C. We call these visual signals the *maximal ancestor visual signals*. The occurrences of maximal ancestor visual signals are considered first in data

record extraction because they are more likely to be individual data records. We discuss below how to find the exact data record boundaries in two different scenarios.

Single Maximal Ancestor Visual Signal

If there is only one maximal ancestor (say s_m) in an essential cluster C, the occurrences o_m^i are likely to be individual data records. However, note that:

1. *Not all of the occurrences are data records.* Recall that o_m^i is one of the occurrences of tag path p_m (*e.g.*, $/html/body/p$). This path may be used for representing not only data records but also different regions. Thus, we need to exclude occurrences that are used for different purposes based on the following intuition: A data record should consist of not only an occurrence of s_m but also occurrences of other visual signals in C (that are descendants of s_m).

2. *An occurrence can contain multiple data records.* For example, product information on an e-commerce Web page might be organized in multiple columns (*e.g.*, Figure 3.5(a)). Let s_R and s_P be visual signals that represent rows of the product list and individual product records, respectively. They are likely grouped together into C. Because $s_R//s_P$, s_R is the maximal ancestor and, thus, we identify occurrences of s_P as data records.

To address the above issues, we introduce the techniques of record candidate filtering and record separation.

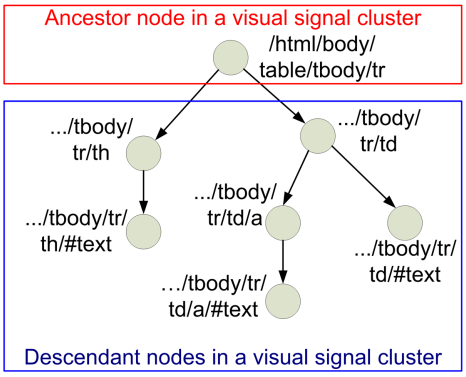
Record candidate filtering. Record candidate filtering selects occurrences from O_m that contain data records. The intuition is as follows: If o_m^i has many descendants that are occurrences of other visual signals in C, o_m^i is likely to contain data records. Let $D_m^i (\subset C)$ be a set of visual signals that have occurrences in the descendants of o_m^i. A greater value of $|D_m^i|$ indicates that o_m^i is a record candidate. We assume further that not all of the visual signals in C are equally important. If a visual signal appears in every data record, it has high similarity to other visual signals in C. Thus, we introduce a weighting factor for each visual signal s_j in D_m^i based on its intra-cluster similarity in C and define the record candidate score ρ of an occurrence o_m^i by:

$$\rho(o_m^i) = \sum_{s_j \in D_m^i} \sum_{s_k \in C} \sigma(s_j, s_k) \tag{3.4}$$

where $\sigma(s_j, s_k)$ is be calculated using Equation (3.3).

We filter out o_m^i iff $\rho(o_m^i) < \rho_{max} \times \alpha$, where ρ_{max} is the maximum ρ score of occurrences of all the visual signals in C. In our experiments, we chose $\alpha = 0.5$.

To identify D_m^i, we need to check if there is an occurrence o_j of a visual signal s_j such that $o_m^i // o_j$ for each s_j in C. Note that $s_m // s_j$ because s_m is the only maximal ancestor in C. Thus, we only need to check if $o_m^i < o_j < o_m^{(i+1)}$ to write $o_m^i // o_j$, which is done efficiently using the visual signal vectors S_m and S_j.

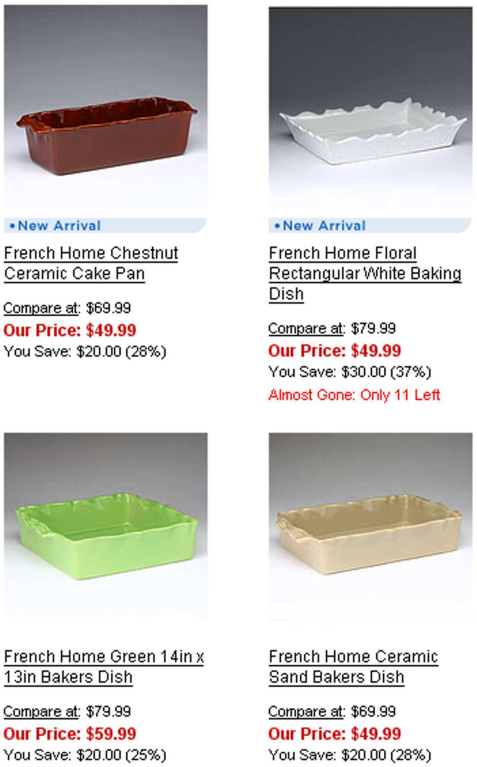


(a) Recovered ancestor and descendant relationships within one cluster.

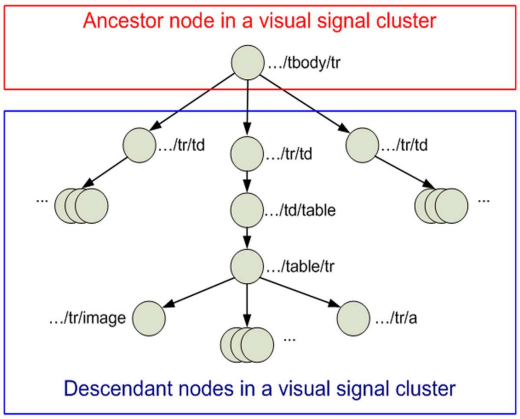
Record #1	42	EE	Luping Ding, Songting Chen, Elke A. Rundensteiner, Jun'ichi Tatemura, Wang-Pin Hsiung, K. SelÃ§uk Candan: Runtime Semantic Query Optimization for Event Stream Processing. ICDE 2008: 676-685
Record #2	41	EE	Egemen Tanin, Songting Chen, Jun'ichi Tatemura, Wang-Pin Hsiung: Monitoring Moving Objects Using Low Frequency Snapshots in Sensor Networks. MDM 2008: 25-32
Record #3	40	EE	Jun'ichi Tatemura, Songting Chen, Fenglin Liao, Oliver Po, K. SelÃ§uk Candan, Divyakant Agrawal: UQBE: uncertain query by example for web service mashup. SIGMOD Conference 2008: 1275-1280
Record #4	39	EE	Yan Qi, K. SelÃ§uk Candan, Jun'ichi Tatemura, Songting Chen, Fenglin Liao: Supporting OLAP operations over imperfectly integrated taxonomies. SIGMOD Conference 2008: 875-888

(b) Data record extraction results after filtering out the occurrences of the common ancestor visual signal.

Figure 3.4: Maximal ancestor visual signal containing one data record.



(a) Web page snapshot.



(b) Recovered ancestor/descendant relationships within one cluster.

(c) Data record extraction results.

Figure 3.5: Maximal ancestor visual signal containing multiple data records.

Nested objects set #2

Description	Repeating Item Sets
2008	[4]
2007	[5]
2006	[6]
2005	[7]
2003	[8]
2000	[9]
1999	[10]
1997	[11]
1994	[12]
1993	[13]
1992	[14]
1991	[15]
1989	[16]

(a) Nested objects.

Repeating item set #4 2008

Record #1 42 EE Luping Ding, Songting Chen, Elke A. Rundensteiner, Jun'ichi Tatemura, Wang-Pin Hsiung, K. SelÃ§uk Candan: Runtime Semantic Query Optimization for Event Stream Processing. ICDE 2008: 676-685

Record #2 41 EE Egemen Tanin, Songting Chen, Jun'ichi Tatemura, Wang-Pin Hsiung: Monitoring Moving Objects Using Low Frequency Snapshots in Sensor Networks. MDM 2008: 25-32

Record #3 40 EE Jun'ichi Tatemura, Songting Chen, Fenglin Liao, Oliver Po, K. SelÃ§uk Candan, Divyakant Agrawal: UQBE: uncertain query by example for web service mashup. SIGMOD Conference 2008: 1275-1280

Record #4 39 EE Yan Qi, K. SelÃ§uk Candan, Jun'ichi Tatemura, Songting Chen, Fenglin Liao: Supporting OLAP operations over imperfectly integrated taxonomies. SIGMOD Conference 2008: 875-888

Record #5 38 EE Yu-Ru Lin, Hari Sundaram, Yun Chi, Jun'ichi Tatemura, Belle L. Tseng: Detecting splogs via temporal dynamics using self-similarity analysis. TWEB 2(1): (2008)

Repeating item set #5 2007

Record #1 37 EE Yu-Ru Lin, Hari Sundaram, Yun Chi, Jun'ichi Tatemura, Belle L. Tseng: Splog Detection Using Self-similarity Analysis on Blog Temporal Dynamics. AIRWeb 2007

Record #2 36 EE Yun Chi, Shenghuo Zhu, Xiaodan Song, Jun'ichi Tatemura, Belle L. Tseng: Structural and temporal analysis of the blogosphere through community factorization. KDD 2007: 163-172

Record #3 35 EE Jun'ichi Tatemura, Arsany Sawires, Oliver Po, Songting Chen, K. SelÃ§uk Candan, Divyakant Agrawal, Maria Goveas: Mashup Feeds: : continuous queries over web services. SIGMOD Conference 2007: 1128-1130

(b) Atomic-level objects.

Figure 3.6: Data record extraction result for nested lists.

Record separation. If the occurrences of the maximal ancestors contain multiple data records, their direct descendant should be able to better separate the data records. We examine the DOM subtrees of the occurrences to determine whether the child nodes are more likely to be individual data records. First, they must be occurrences of the same visual signal. Next, they must have a similar visual pattern so that together they comprise a large visually repeating block. This idea is similar to one employed in MDR [81] that checks if a single row contains multiple data records. Whereas MDR utilizes edit distance of tag structures, our method takes a simpler approach that performs well in experiments. From the rendered Web page, we retrieve the width and height of all of the descendant visual signal occurrences. We calculate their variances to determine whether the descendant node is a better data record separator.

The record filtering and separation are performed repeatedly until no better separator is found. The results are the atomic-level data records in a Web page.

Figure 3.4 shows an example where the maximal ancestor represents the data record and no record separation is required. In the DBLP page, the publications of a researcher are listed in a table and all of them are extracted correctly. An example that requires record separation is shown in Figure 3.5. In this example, each row contains two product records. Our algorithm extracts the visual signal corresponding to a row as the maximal ancestor and then determines whether its direct descendant visual signal is a better record separator.

Multiple Maximal Ancestor Visual Signals

When there are multiple maximal ancestors, there is no single visual signal that captures the entire data record. Typically, occurrences of these different maximal ancestors are consecutive siblings that together represent a data record.

Our problem now is to identify a repeating pattern from a sequence of occurrences from different signals. Our current implementation uses a simple heuristic: The visual signal, say s_B, that occurs first is chosen as the record boundary. The intuition is that the first component of a record is typically a mandatory part of the data (*e.g.*, a title). An occurrence o of other maximal ancestor visual signals is a part of the ith data record if $o_B^i < o < o_B^{(i+1)}$. After forming the data record candidates, we filter them as in Section 3.3.2.

3.3.3 Semantic-Level Nesting Detection

Nested lists are common on the Web. Usually, data records are organized into semantic categories with an arbitrary number of data records in each category. A description might be attached to each category.

Our approach can capture such nesting through discovery of non-consecutive lists of atomic-level data records. The semantic categories are usually explicitly marked by HTML tags, and data records inside one semantic category are consecutive in the HTML document. Thus, if the data records are not consecutive, they might belong to

different semantic categories. Based on this intuition, we extract the nesting structure as follows: If a visual signal occurs at each point where the same set of data records is partitioned, the visual signal corresponds to a visual pattern that separates two semantic categories. The text lying between the sets of extracted data records is the description of the semantic category. Using this rule, we extract both the "year" objects and the "publication" objects in the DBLP page example, as shown in Figure 3.6.

3.4 Experiments

3.4.1 Experimental Setup

We evaluated both the effectiveness and the efficiency of our algorithm using two datasets. We compare the performance of our algorithm with that of MDR, an implementation of which was available on the Web. Implementations of NET and EXALG were not available, so we do not compare the performance of our algorithm with that of those algorithms.

Dataset #1 was chosen from the testbed for information extraction from the deep Web, collected by Yamada *et al.* [144]. The testbed data has 253 Web pages from 51 Web sites randomly drawn from 114,540 Web pages with search forms. The data records in these Web pages are manually labeled; the results are available online together with the testbed dataset. To provide a fair comparison between our algorithm

and the MDR algorithm [81], which is designed for flat data records, we filtered out the Web pages with nested structures in the testbed. The resulting dataset #1 contains 213 Web pages from 43 Web sites.

Dataset #2 was introduced mainly for the purpose of evaluating our algorithm on nested list structures. Lacking an existing test data set, we collected the Web pages ourselves. Dataset #2 contains 45 Web pages, each from one Web site, randomly chosen from the domains of business, education, and government. Each Web page contains a two-level nested list structure. Both the atomic-level data records and the nested data records are manually labeled.

Our experiments were carried out on a Pentium 4 computer with a 3.2GHz CPU and 2G of RAM. Our Java implementation of the algorithm utilizes the open source Web renderer Cobra [2], which resolves ill-formed HTML and executes JavaScript for dynamic HTML pages.

3.4.2 Accuracy Analysis

The experimental results for our algorithm compared with MDR [81] are shown in Figure 3.7. We ran both algorithms for all of the Web pages in dataset #1. The results are aggregated based on the Web sites. The *ground truth* is the set of data records in all Web pages from one Web site. *True positives* are the set of data records correctly extracted by the algorithms from that Web site. The perfect case is that the true positives

match the ground truth exactly. *False positives* are the set of data records that the algorithm incorrectly includes in the same list with the true positives. To distinguish the false positives from the true positives, we flip the sign of the false positives and show them in the same figure. Generally speaking, our algorithm has more true positives and fewer false positives compared with the MDR algorithm. We also calculated the *precision* and *recall* as given in Equations (3.5) and (3.6) for all of the Web sites. The results are shown in Table 3.3.

$$Precision = \frac{|true\ positives|}{|true\ positives| + |false\ positives|} \tag{3.5}$$

$$Recall = \frac{|true\ positives|}{|ground\ truth|} \tag{3.6}$$

When none of the records is detected, both $|true\ positives|$ and $|false\ positives|$ are zero, hence Equation (3.5) is ill-formed. We define the precision to be zero in such a case.

Table 3.3: Accuracy comparison for dataset #1.

Algorithm	Average Precision	Average Recall
Our algorithm	90.4%	93.1%
MDR	59.8%	61.8%

The experimental results for dataset #2 show the performance of our algorithm for nested list structures. We compare each atomic-level data record and nested data record extracted by our algorithm with the manually labeled ground truth. The results of the

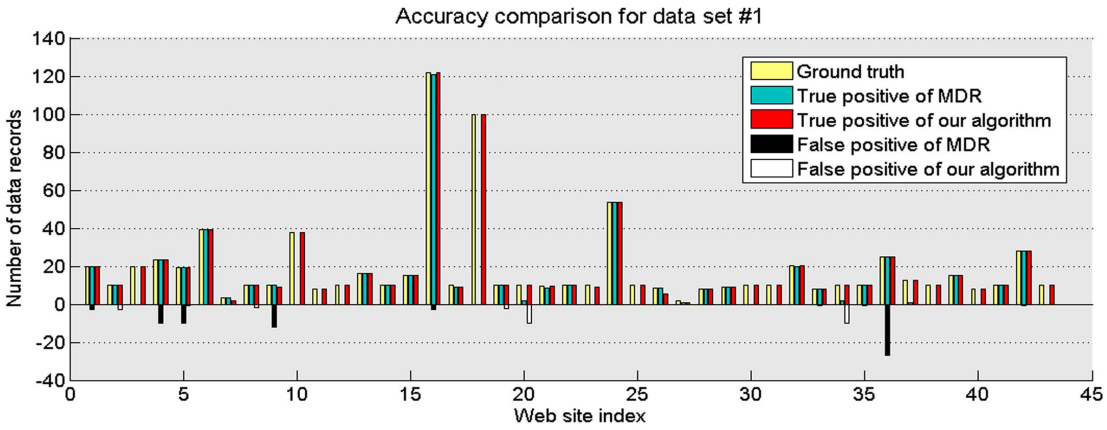


Figure 3.7: Accuracy comparison between our algorithm and MDR for dataset #1.

Table 3.4: Experimental results for dataset #2.

Domain	Ground Truth		Our Results	
	Nested	Atomic	Nested	Atomic
Business	46	415	46	415(1)
Education	215	1672	208(2)	1672(17)
Government	104	955	104(1)	954(1)

Overall Accuracy Measure				
Nested Records	Precision	98.9%	Recall	98.1%
Atomic Records	Precision	99.4%	Recall	99.9%

comparison are shown in Table 3.4. The ground truth numbers of data records for the Web pages are listed in columns 2 and 3. The numbers of true positives are listed in columns 4 and 5. The numbers of false positives are listed in parentheses if they are greater than zero. There are 15 Web pages from the business domain, 15 Web pages from the education domain, and 15 Web pages from the government domain.

3.4.3 Time Complexity Analysis

The algorithm consists of three steps. We analyze the time complexity for each step individually.

Detecting visually repeating information. In this step, first we scan the Web page and extract the visual signals, which takes $\mathcal{O}(L)$ time, where L is the total number of HTML tag occurrences in the Web page. Calculating the pairwise visual signal similarity matrix and performing spectral clustering on it takes $\mathcal{O}(M \times L) + \mathcal{O}(M^3)$, where M is the number of unique HTML tag paths in the Web page. Thus, the step of visual repeating information detection takes $\mathcal{O}(M \times L) + \mathcal{O}(M^3)$ time in total.

Data record extraction. In this step, first we retrieve all of the occurrences of the common ancestors in the Web page for each essential cluster. When filtering these occurrences, the algorithm visits all of the descendants. The total number of HTML nodes visited is less than L. Thus, the time complexity of this step is $\mathcal{O}(L)$.

Semantic-level nesting detection. In this step, we examine the visual signals that appear at each point where the data records are not consecutive. The number of HTML tags visited is still less than L. Thus, the time complexity of this step is $\mathcal{O}(L)$.

In total, the time complexity of the algorithm is $\mathcal{O}(M \times L) + \mathcal{O}(M^3)$, where L is the total number of tag occurrences and M is the number of unique tag paths in the Web page.

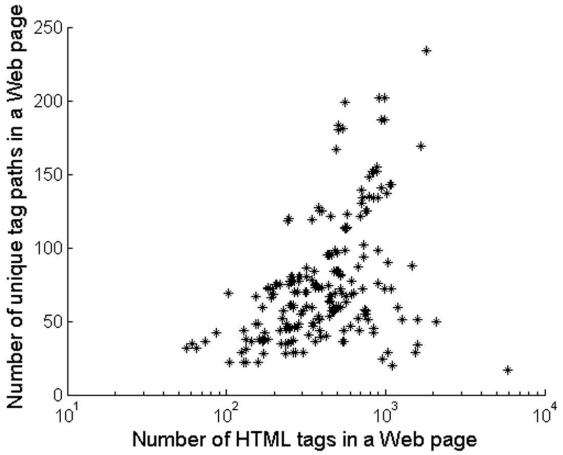


Figure 3.8: Number of unique tag paths vs. number of HTML tags. The number of unique tag paths does not increase as the number of HTML tags increases.

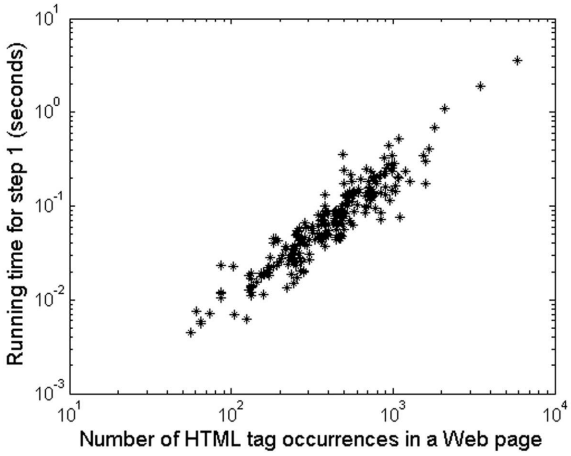


Figure 3.9: Step 1 is linear in the document length.

For comparison purposes, we also analyze the time complexity of existing similarity-based approaches, MDR [81] and NET [82]. These algorithms traverse a DOM tree and apply edit distance computation between sibling subtrees. Let N be the number of children of each node. At the root, the algorithms compute the edit distance between its children with size L/N, taking $\mathcal{O}((L/N)^2)$ time. MDR computes the edit distance N times, and NET computes it N^2 times in the worst case. At depth d, there are N^d trees, each of which has N children of size L/N^{d+1}. The total cost is $\sum_d (L/N^{d+1})^2 N^k N^d = L^2 N^{k-2} \sum_d (1/N)^d < L^2 N^{k-2} \times N/(N-1)$ where $k = 1$ for MDR and $k = 2$ for NET. Thus, the time complexity of MDR and NET are $\mathcal{O}(L^2/N)$ and $\mathcal{O}(L^2)$, respectively. From this analysis, we conclude that MDR is efficient ($\mathcal{O}(L)$) when the document structure is simple (and N is as large as L). However, if the document structure is complex, MDR is not as scalable.

The key question then is how the number M of unique tag paths, grows as L becomes large. If M does not scale up, our algorithm is more scalable than NET, and even MDR when the document is complex. Recall that our algorithm and NET can detect nested structures whereas MDR cannot. For the experimental dataset, M stays small as L grows, as shown in Figure 3.8. Thus, the complexity of our algorithm is $\mathcal{O}(L)$ for practical datasets, *i.e.*, it is linear in the document length. Figure 3.9 shows that the completion time of Step 1 is linear in L.

Table 3.5: Execution time analysis.

Function	Average Time (ms)	Percentage
Rendering	208.90	NA
Total execution time	328.73	100%
Step 1	157.63	47.95%
Step 2	72.99	22.20%
Step 3	98.11	29.84%

On average, the total execution time of our algorithm for one Web page is similar to the rendering time. We divide the execution time into three parts based on the three steps as presented in Table 3.5. Step 1 takes 47.95% of the total time, and Steps 2 and 3 together take 52.05% of the total time. Because Steps 2 and 3 are conducted for each essential cluster, and there is no interaction between clusters, this part of the algorithm can be parallelized.

3.5 Summary

This chapter presented a novel approach to data record extraction from Web pages. A data record list corresponds to a set of visual signals that appear regularly on the Web page. The method first detects visual signals that repeat in a similar pattern. Page segmentation is performed based on clusters of similar visual signals. Experimental results on flat data record lists are compared with a state-of-the-art algorithm. Our algorithm shows significantly higher accuracy than existing algorithms. For data record lists with a nested structure, we collected Web pages from the domains of business,

education, and government. Our algorithm demonstrates high accuracy in extracting both atomic-level and nested-level data records. The execution time of the algorithm is linear in the document length for practical datasets.

Chapter 4

Recovering the Semantics of Tables to Enable Table Search

The Web offers a corpus of more than 100 million high-quality tables [25], but the meaning of each table is rarely explicit in the table itself. Header rows exist in few cases and even when they do, the attribute names are typically useless. In this chapter, we describe a system that attempts to recover the semantics of tables by enriching the table with additional annotations. Our annotations facilitate operations such as searching for tables and finding related tables.

To recover the semantics of tables, we leverage a database of class labels and relationships automatically extracted from the Web. The database of classes and relationships has a very wide coverage, but also is very noisy. We attach a class label

to a column if a sufficiently many values in the column are identified with that label in the database of class labels, and analogously for binary relationships. We describe a method for reasoning about when we have seen sufficient evidence for a label, and show that the method performs substantially better than a simple majority scheme. We describe a set of experiments that illustrate the utility of the recovered semantics for table search and show that the method performs substantially better than previous approaches. In addition, we characterize what fraction of tables on the Web can be annotated using our method.

4.1 Overview

The corpus of more than 100 million tables on the Web cover a wide variety of topics [25]. These tables are embedded in HTML and, therefore, their meaning is described only bin the text surrounding them. Header rows exist in few cases, and even when they do, the attribute names in the headers are typically useless.

Without knowledge of the semantics of the tables, it is very difficult to leverage their content, either in isolation or in combination with other tables. The challenge arises in particular for *table search* (for queries such as *countries population*, or *dog breeds life span*), which is the first step in exploring a large collection of tables. Search engines typically treat tables like any other text fragment, but signals that work well for text

do not apply as well to table corpora. In particular, document search often considers the proximity of search terms on the Web page to be an important signal, but in tables the column headers apply to every row in the table even if they are textually far away. Furthermore, unlike text documents, where small changes in the document structure or wording do not correspond to vastly different content, variations in table layout or terminology can change the semantics significantly. In addition to table search, knowledge of the semantics of the tables is necessary for higher-level operations such as combining tables via join or union.

In principle, we would like to associate semantics with each table in the corpus, and use the semantics to guide retrieval, ranking and combining tables. However, given the scale, breadth and heterogeneity of the tables on the Web, we cannot rely on hand-coded domain knowledge. Thus, this chapter presents techniques for automatically recovering the semantics of tables on the Web. Specifically, we add annotations to a table that describe the sets of entities represented in the table, and the binary relationships represented by the columns in the table. For example, in the table of Figure 4.1, we would add the annotations `tree species`, `tree`, and `plant` to the first column, and the annotation `is known as` to describe the binary relation represented by the table[1].

The key insight underlying our approach is that we can use facts extracted from text on the Web to interpret the tables. Specifically, we leverage two databases that are

[1] The complete table can be found at `http://www.hcforest.sailorsite.net/Elkhorn.html`.

	Alnus serrulata
	Fraxinus pennsylvanica
	Fraxinus americana
	Taxodium distichum
	Fagus grandifolia
	Betula nigra
	Acer negundo
Red Cedar	*Juniperus virginiana*
	Prunus serotina
Sweet Cherry	*Prunus avium*
	Malus coronaria
	Lagerstroemia indica
	Cornus florida

Figure 4.1: An example table on the Web, associating common names of trees with their scientific names.

extracted from the Web: (1) an isA database that contains a set of pairs of the form (instance, class), and (2) a relations database of triples of the form (argument1, predicate, argument2). Because they are extracted from the Web, both of these databases have very broad coverage of topics and instances, but they are very noisy. We use them to annotate the columns in the table as follows. We label a column A with class C in the isA database if a substantial fraction of the cells in a column A are labeled with class C in the isA database. We label the relationship between columns A and B with R if a substantial number of pairs of values from A and B occur in extractions of the form (a, R, b) in the relations database. We describe a method that lets us determine how much evidence we need to find in the extracted databases to deem a label appropriate for a column or a pair of columns. In particular, the method addresses the challenge that the extracted databases are not a precise description of the real world or even of the Web, because some entities occur more frequently on the Web than others.

We show experimentally that the labels we generate describe the contents of the table well and are rarely explicit in the table itself. We show that the labels are even more accurate when we consider only the labels that are associated with a column in the table that is the *subject* of the table. Based on this, we build a table search engine with much higher precision than previous approaches.

4.2 Related Work

Similar to our work, the work of Limaye *et al.* [79] annotates tables on the Web with column and relationship labels. However, unlike our work, their work aims to choose a *single* label from an ontology (YAGO [127]). They propose a graphical model for labeling table columns with types, pair of columns with binary relations, and table cells with entity IDs, and use YAGO as a source for their labels. The key idea of their work is to use joint inference about each of the individual components to boost the overall quality of the labels. As we show in our experiments, YAGO includes only a small fraction of the labels we find. In particular, YAGO includes fewer than 100 binary relationships. Our work is the first that tries to detect binary relationships at any serious scale. In principle, we can also apply joint inferences with our approach, but we leave that for future work.

Cafarella *et al.* [25] considered the problem of table search, but approached it as a modification to document search. They added new signals to ranking documents, such as hits on the schema elements and subject columns. The weights of the new signals were determined by machine learning techniques. As we show subsequently, table search aided by our annotations offers significantly higher precision than that of [25].

Several works have considered how to extract and manage data tables found on the Web (*e.g.*, [24, 50, 64]), but they do not consider annotation or search problems. Gupta and Sarawagi considered how to answer fact queries from lists on the Web [59]. In addition, there is a significant body of work that considers how to rank tuples within a single database in response to a keyword query [63]. The distinguishing challenge in our context is the vast breadth of the data and the fact that it is formatted on Web pages in very different ways.

Downey *et al.* [49] proposed a theoretical model for measuring the confidence of extractions from the Web. They proposed a combinatorial "urns" model that computes the probability that *a single extraction* is correct based on sample size, redundancy, and corroboration from multiple extraction patterns. In contrast, we compute the probabilistic distribution of semantic labels for columns in Web tables based on *a set of cell values/pairs*. Hence, one of the challenges in our model is to provide smaller weights for missing extractions when the entities involved do not appear frequently on the Web.

The output of the "urns" model can be used as one of the inputs to our method to infer label distributions.

Existing methods for extracting classes of instances from the Web require sets of instances that are each either unlabeled [80, 103, 137], or associated with a class label [12, 61, 101, 102, 138]. When associated with a class label, the sets of instances can be organized as flat sets or hierarchically, relative to existing hierarchies such as WordNet [121, 127] or the category network within Wikipedia [106, 140]. To the best of our knowledge, the isA database described in this chapter is larger than similar databases extracted from unstructured text. In particular, the number of useful extracted class labels (*e.g.*, class labels associated with 10 instances or more) is at least one order of magnitude larger than that for the isA databases described in [128], although those databases are extracted from document collections of similar size, and using the same initial sets of extraction patterns as in our experiments.

Previous work on automatically generating relevant labels, given sets of items, focuses on scenarios where the items within the sets to be labeled are descriptions, or full-length documents within document collections [29, 36, 133]. Relying on semi-structured content assembled and organized manually as part of the structure of Wikipedia articles, such as article titles or categories, the method introduced in [29] derives labels for clusters each containing 100 full-length documents. In contrast, our method relies on isA relations and binary relations automatically extracted from unstructured text

within arbitrary Web documents, and computes labels given textual input that are orders of magnitude smaller, *i.e.*, table columns.

4.3 Problem Description

We begin by describing the Web table corpus and the problems of annotating tables and table search.

Table corpus: Each table in our corpus is a set of rows, and each row is a sequence of cells with data values (see Figure 4.1 for an example). Tables may be semi-structured and might have very little metadata. Specifically:

- We do not have a name for the table (*i.e.*, the relationship or entities that it is representing).
- The attributes might not have names, and we might not know whether the first row(s) of the table are attribute names or data values (as in Figure 4.1).
- The values in a particular row of a column will typically be of a single data type, but there might be exceptions. Values might be taken from different domains and different data types. Often, we see sub-header rows of the type one would see in a spreadsheet.

• The quality of tables in the corpus varies significantly, and it is hard to determine whether HTML table tags are used for high-quality tabular content or as a formatting convenience.

Annotating tables: Our goal is to add annotations to tables to expose their semantics more explicitly. Specifically, we add two kinds of annotations. The first, *column labels* are annotations that represent the set of entities in a particular column. For example, in the table of Figure 4.1 possible column labels are `tree`, `tree species` and `plant`. The second, *relationship labels* represent the binary relationship that is expressed by a pair of columns in the table. For example, a possible relationship label in the table of Figure 4.1 is `is known as`. We note that our goal is to produce *any* relevant label that appears on the Web and, therefore, to match more keyword queries that users might pose. In contrast, previous work [79] focused on finding a *single* label from an ontology (YAGO [127]).

Typically, tables on the Web have a column that is the subject of the table. The subject column contains the set of entities that the table represents, and the other columns represent binary relationships or properties of those entities. We have observed that more than 75% of the tables in our corpus exhibit this structure. Furthermore, the subject column need not be a key — it may contain duplicate values.

Identifying a subject column is important in our context because the column label we associate with it offers an accurate description of what the table represents, and the

binary relationships between the subject column and other columns reflect the properties that the table is representing. Hence, although our techniques do not require the presence of a subject column, we show that the accuracy of our annotations and resulting table search are higher when a subject column is identified.

Table search: We investigate the quality of our annotations by their ability to improve table search, the most important application that the annotations enable. We assume that queries to table search can be posed using any keyword because it is unreasonable to expect users to know the schemata of such large collections of heterogeneous tables in such vast arrays of topics.

In this work, we consider returning a ranked list of tables in response to a table search query. However, the ability to retrieve tables based on their semantics lays the foundation for more sophisticated query answering. In particular, we might want to answer queries that require combining data from multiple tables through join or union. For example, consider a query that asks for the relationship between the incidence of malaria and the availability of fresh water. There might be a table on the Web for describing the incidence of malaria, and another for access to fresh water, but the relationship can only be gleaned by joining the two tables.

We analyzed Google's query stream and found that there are two kindes of queries that can be answered by table search: (1) find a property of a set of instances or entities (*e.g.*, *wheat production of African countries*), and (2) find a property of an individual

instance (*e.g., birth date of Albert Einstein*). This chapter focuses on queries of the first kind. We assume that they are of the form (C, P), where C is a string denoting a class of instances and P denotes some property associated with those instances. Both C and P can be *any* string rather than being drawn from a particular ontology, but we do not consider the problem of transforming an arbitrary keyword query into a pair (C, P) in this chapter. Also, we note that there are millions of queries of both kinds being posed every day.

Our techniques can be used to help answering the second kind of queries, but there are many other techniques that come into play [12,59]. In particular, answers to queries about an instance and a property can often be extracted from free text and corroborated against multiple occurrences on the Web.

Finally, we note that we do not consider the problem of blending results of table search with other Web results.

4.4 Annotating Tables

Given the size and breadth of the table corpus we are considering, manually annotating the semantics of tables does not scale. The key idea underlying our approach is to annotate tables automatically by leveraging resources that are already on the Web and, hence, have similar breadth to the table corpus. In particular, we use two different data

resources: (1) an isA database consisting of pairs (instance, class) that are extracted by examining specific linguistic patterns on the Web, and (2) a relations database consisting of triplets of the form (argument1, predicate, argument2) extracted without supervision from the Web. In both databases, each extraction has a score associated with it describing our confidence in the extraction. The isA database is used to produce column labels, and the relations database is used to annotate relationships expressed by pairs of columns. Importantly, our goal is not necessarily to recover a single most precise semantic description (*i.e.*, we cannot compete with manual labeling), but just enough to provide useful signals for search and other higher-level operations.

The isA database and relations database are described in Section 4.4.1 and Section 4.4.2, respectively. In Section 4.4.3, we consider the problem of how evidence from the extracted databases should be used to choose labels for tables. Because the Web is not a precise representation of the real world and the algorithms used to extract the databases from the Web are not perfect, the model weights some evidence more heavily than other evidence in determining possible labels. As described earlier, when labels are associated with a subject column, they are even more indicative of the table's semantics.

4.4.1 The isA Database

The goal of column labels is to describe the class of entities that appear in that column. In Figure 4.1, the labels `tree`, `tree species` and `plant` describe the entities in the first column and might correspond to terms used in searches that should retrieve this table. Recall that the isA database is a set of pairs of the form (instance, class). We refer to the second part of the pair as a *class label*. We assign column labels to tables from the class labels in the isA database. Intuitively, if the pairs (I, C) occur in the isA database for a substantial number of values in a column A, then we attach C as a column label to A. We now describe how the isA database is constructed.

We begin with techniques such as those presented in [101] to create the isA database. We extract pairs from the Web by mining for patterns of the form:

$$\langle \text{[..] } C \text{ [such as|including] } I \text{ [and|,|.]} \rangle,$$

where I is a potential instance and C is a potential class label for the instance (*e.g.*, cities such as Berlin, Paris and London).

To apply such patterns, special attention needs to be paid to determining the boundaries of C and I. Boundaries of potential class labels C in the text are approximated from the part-of-speech tags (obtained using the TnT tagger [22]) of the sentence words. We consider noun phrases whose last component is a plural-form noun and that are not contained in and do not contain another noun phrase. For example, the class label

`michigan counties` is identified in the sentence [..] michigan counties such as van buren, cass and kalamazoo [..]. The boundaries of instances I are identified by checking that I occurs as an entire query in query logs. Because users type many queries in lower case, the collected data is converted to lower case. These types of rules have also been widely used in the literature on extracting conceptual hierarchies from text [61, 121].

To construct the isA database, we applied patterns to 100 million documents in English using 50 million anonymized queries. The extractor found around 60,000 classes that were associated with 10 or more instances. The class labels often cover closely-related concepts within various domains. For example, `asian countries`, `east asian countries`, `south asian countries`, and `southeast asian countries` are all present in the extracted data. Thus, the extracted class labels correspond to a broad and relatively deep conceptualization of the potential classes of interest to Web search users, on the one hand, and to human creators of Web tables, on the other hand. The reported accuracy for class labels in [101] is greater than 90% and the accuracy for class instances is almost 80%.

To improve the coverage of the database beyond the techniques described in [101], we use the extracted instances of a particular class as seeds for expansion by considering additional matches in Web documents. We look for other patterns on the Web that match more than one instance of a particular class, effectively inducing document-

specific extraction wrappers [73]. For example, we might find the pattern ⟨headquartered in I⟩ and, thus, be able to mine more instances I of the class label `cities`. The candidate instances are scored across all documents, and added to the list of instances extracted for the class label [137]. Doing so increases the coverage with respect to instances, although not with respect to class labels.

Given the candidate matches, we then compute a score for every pair (I, C) using the following formula [100]:

$$Score(I, C) = Size(\{Pattern(I, C)\})^2 \times Freq(I, C). \quad (4.1)$$

In the formula, $Pattern(I, C)$ is the set of different patterns in which (I, C) was found and $Freq(I, C)$ is the frequency count of the pair. However, because high frequency counts are often indicative of near-duplicate sentences appearing on many Web pages, we perform the following computation. We compute a sentence fingerprint for each source sentence, by applying a hash function to at most 250 characters from the sentence. Occurrences of (I, C) with the same sentence fingerprint are counted only once in $Freq(I, C)$.

4.4.2 The Relations Database

We also want to annotate a table with the set of relationships that the table represents between pairs of entities. For example, the table in Figure 4.1 represents the relation-

ship `is known as` between trees and their scientific names. In general, two types of relationships are common in tables on the Web: symbolic (*e.g.*, `capital of`) and numeric (*e.g.*, `population`). In what follows, we use the relations database to obtain labels for the symbolic relationships.

Intuitively, given two columns, A and B, we look at corresponding pairs of values in the columns. If we find that the relation (a, R, b) is extracted for many rows of the table, then R is a likely label for the relationship represented by A and B.

We use the Open Information Extraction (OIE) [52] method to extract triples for the relations database. Unlike traditional information extraction that outputs instances of a *given* relation, OIE extracts any relation using a set of relations-independent heuristics. In our implementation, we use the TextRunner open extraction system, which has precision around 73.9% and recall around 58.4%, according to [13].

4.4.3 Evaluating Candidate Annotations

The databases described above provide evidence from the Web that a particular label applies to a column, or that a particular binary relationship is represented by a pair of columns. However, the immediate question that arises is how much evidence is enough to assign a label to a column or pair of columns, or alternatively, how to rank the candidate labels.

If the isA and relations databases were a precise description of the real world, then we would require a label to apply to *all* rows of a table before it is assigned. However, the databases have two kinds of imprecision: (1) the Web is *not* an accurate representation of the real world, and (2) no matter how good the extractors are, they will miss some facts that are mentioned on the Web. Consider the effect of the first kind of imprecision. Paris and France are mentioned very frequently on the Web and, thus, we expect to find sentences on the Web that state that Paris is the capital of France. However, Lilongwe and Malawi are not mentioned as often, and therefore there is a smaller likelihood of finding sentences that say that Lilongwe is the capital of Malawi. Hence, if we have a table that includes a row for Paris, France and one for Lilongwe, Malawi, but we do not find (Lilongwe, capital of, Malawi) in the relations database, that should not be taken as strong evidence against assigning the label `capital of` to that pair of columns.

The second kind of imprecision stems from the fact that, ultimately, the extractors are based on rules that might not extract everything that is said on the Web. For example, to extract cities, we look for patterns of the form $\langle$cities such as $I\rangle$, which might not be found for rare entities such as Lilongwe. In addition, some entities are simply not mentioned in such patterns at all. For example, there are many tables on the Web that describes the meaning of common acronyms, but there are very few sentences of the form $\langle$acronyms such as $I\rangle$.

The method we describe below lets us reason about how to interpret the different kind of positive and negative evidence we find in our extracted database. We use a maximum-likelihood method based on the following intuition. A person constructing a table in a Web page has a particular intent ("schema") in mind. The intent is to describe properties of instances of an entity class. The maximum-likelihood method attempts to assign class labels to a column given the contents the person has used to populate the column. The best label is therefore the one that, if chosen as part of the underlying intent, is most likely to have resulted in the observed values in the column. Consequently, we try to infer the intent of the table designer based on the evidence they have given us.

We begin by considering the problem of assigning class labels to a column. Let $V = \{v_1, \ldots, v_n\}$ be the set of values in a column A. Let $l_1, \ldots, l_m$ be all possible class labels.

To find the best class label, we use the maximum likelihood hypothesis [95], *i.e.*, the best class label $l(A)$ is the one that maximizes the probability of the values, given the class label for the column:

$$l(A) = \arg\max_{l_i} \{\Pr[v_1, \ldots, v_n \mid l_i]\}.$$

We assume that each row in the table is generated independently, given the class label for the column and, thus, $\Pr[v_1, \ldots, v_n \mid l_i] = \prod_j \Pr[v_j \mid l_i]$. This is a reasonable assumption in our context, because tables that are relational in nature are likely to

have dependencies between column values in the same row, rather than across rows. Furthermore, from Bayes rule, we have $\Pr[v_j \mid l_i] = \frac{\Pr[l_i|v_j] \times \Pr[v_j]}{\Pr[l_i]}$. It follows that:

$$\Pr[v_1, \ldots, v_n \mid l_i] = \prod_j \frac{\Pr[l_i \mid v_j] \times \Pr[v_j]}{\Pr[l_i]} \propto \prod_j \frac{\Pr[l_i \mid v_j]}{\Pr[l_i]}.$$

The product term $\prod_j \Pr[v_j]$ applies identically to each of the labels. Hence, it follows that $l(A) = \arg\max_{l_i} \prod_j \frac{\Pr[l_i|v_j]}{\Pr[l_i]}$.

We assign a score $U(l_i, V)$ to each class that is proportional to the expression in the above equation and normalize them so that they sum up to 1, *i.e.*,

$$U(l_i, V) = K_s \prod_j \frac{\Pr[l_i \mid v_j]}{\Pr[l_i]}, \tag{4.2}$$

where the normalization constant K_s is such that $\sum_i U(l_i, V) = 1$.

The probability $\Pr[l_i]$ can be estimated from the scores in the isA database (see Equation (4.1)). However, estimating the conditional probability $\Pr[l_i \mid v_j]$ is more challenging. A simple estimator such as $\frac{Score(v_j,l_i)}{\sum_k Score(v_j,l_k)}$ has two problems. First, when computing the maximum likelihood hypothesis, because we are multiplying the $\Pr[l_i \mid v_j]$, none of these probabilities can be 0. Second, because information extracted from the Web is inherently incomplete, it is likely that there are values for which there is an incomplete set of labels in the isA database.

To address the incompleteness, we *smooth* the estimates of the conditional probabilities:

$$\Pr[l_i \mid v_j] = \frac{K_p \times \Pr[l_i] + Score(v_j, l_i)}{K_p + \sum_k Score(v_j, l_k)},$$

where K_p is a smoothing parameter.

The above formula ensures that in the absence of any isA extractions for v_j, the probability distributions of the labels tends to be the same as the prior $\Pr[l_i]$. As a result, values with no known labels are not taken as negative evidence and do not contribute to changing the ordering among best hypotheses. On the other hand, if there are many known class-label extractions for v_j, the conditional probabilities tend towards their true values and hence such v_j contribute significantly (positively or negatively) toward selecting the best class labels. As the score in the isA database increases (with increased extractions from the Web), the conditional probability estimator depends more on the scores. The parameter K_p controls how sensitive the probabilities are to low extraction scores. If we assume that extractions from the Web are mostly true (but incomplete), then we can set K_p to be very low (say 0.01).

Finally, we need to account for the fact that certain expressions are inherently more popular on the Web and can skew the scores in the isA database. For example, for a value v with two labels $Score(v, l_1) = 100$ and $Score(v, l_2) =$ 10,000, a fraction will result in $\Pr[l_1 \mid v] \ll \Pr[l_2 \mid v]$. We refine our estimator further to instead use the logarithm of the scores, *i.e.*,

$$\Pr[l_i \mid v_j] = \frac{K_p \times \Pr[l_i] + \ln(Score(v_j, l_i) + 1)}{K_p + \sum_k \ln(Score(v_j, l_k) + 1)}. \tag{4.3}$$

The $+1$ in the logarithm prevents $\ln 0$. As before, the probabilities are normalized to sum to 1.

To determine the prior probabilities of the class labels $\Pr[l_i]$, we add the scores across all values for that label, *i.e.*,

$$\Pr[l_i] \propto \sum_j \ln(Score(v_j, l_i) + 1 + \delta). \tag{4.4}$$

We use $1 + \delta$ to ensure that $\Pr[l_i] \neq 0$. The probabilities are normalized such that $\sum_i \Pr[l_i] = 1$.

Given the set of values in a column, we estimate the likelihood score U for each possible label (Equation 4.2). We consider only the labels that have a normalized likelihood score greater than a threshold t_l and rank the labels in decreasing order of their scores.

4.5 Experiments

We evaluate the quality of the table annotations and their impact on table search in Section 4.5.1 and Section 4.5.2, respectively.

Table corpus: Following [25], we constructed a corpus of HTML tables extracted from a subset of the crawl of the Web. We considered pages in English with high page rank. From these, we extracted tables that were clearly not HTML layout tables, and then filtered out empty tables, form tables, calendar tables, tiny tables (with only 1 column or with less than 5 rows). We were left with about 12.3 million tables. We estimate

that this corpus represents about a tenth of the high-quality tables on the Web as of late 2010.

4.5.1 Column and Relation Labels

We discuss the quality of the labels assigned with the isA and relations databases. We show that our method labels an order of magnitude more tables than is possible with Wikipedia labels and many more tables than Freebase. We show that the vast majority of the remaining tables can either be labeled using a few domain specific rules or do not contain useful content.

Label quality

We compare three methods for assigning labels. The first, denoted *Model*, is the maximum likelihood method described in Section 4.4.3. The second, denoted *Majority*, requires that at least $t\%$ of the cells of the column have a particular label. Of these, the algorithm ranks the labels according to a $MergedScore(C) = \sum_L \frac{1}{Rank(C,L)}$ (if C has not been assigned to cell content L, then $Rank(C, L) = \infty$). After experimenting with different values, we observed that the Majority algorithm performs best when $t = 50$. We also examined a Hybrid method that uses the ranked list of the Majority method concatenated with the ranked list of the Model method (after removing labels output by

the Majority method. The Hybrid method performs better than both the Model method and the Majority method, as explained below.

Gold standard: To create a gold standard, we considered a random sample of tables and removed those that did not have any class or relations labels assigned by run R_{10}, the Majority algorithm with $t = 10\%$ (*i.e.*, a very permissive labeling). We then manually removed tables whose subject columns were incorrectly identified or do not correspond to any meaningful concept. For each of the remaining 168 tables, we presented to human annotators the result of R_{10}. The annotators mark each label as *vital*, *okay*, or *incorrect*. For example, given a table column containing the cells {Allegan, Barry, Berrien, Calhoun, Cass, Eaton, Kalamazoo, Kent, Muskegon, Saint Joseph, Van Buren}, the assigned class labels `southwest michigan counties` and `michigan counties` are marked as *vital*; labels `counties` and `communities` as *okay*; and `illinois counties` and `michigan cities` as *incorrect*. In addition, the annotators can manually enter any additional labels that apply to the table column, but are missing from those returned by any of the experimental runs. The resulting gold standard associates the 168 tables with an average of 2.6 *vital* and 3.6 *okay* class labels, with 1.3 added manually by the annotators. For the relations labels, we had an average of 2.1 *vital* and 1.4 *okay* labels. Because there were many options for our relations database, we did not add any new (manual) annotations to the binary relations labels.

Evaluation methodology: For a given table, the evaluation consists of automatically comparing the ranked lists of labels produced by an experimental run to the labels available in the gold standard. To compute precision, a retrieved label is assigned a score of 1 if it was marked as *vital* or manually added to the gold standard; 0.5 if it was marked as *okay*, and 0 otherwise [101]. Similarly, recall is computed by considering a label as relevant (score 1) if it was marked as *vital* or *okay* or was manually added to the gold standard, and irrelevant (score 0) otherwise.

Results: Figure 4.2 summarizes the performance results for the three algorithms. We varied the precision and recall by considering the top k labels for values of k between 1 and 10; k increases from left to right in the graph.

We observed that Majority (with $t = 50$) has a relatively high precision but low recall (it labeled only 30% of the 168 tables). The reason is the requirement that a label must be given to 50% of the rows. In addition, Majority tends to output general labels (*e.g.*, `compound chemical` vs. `antibiotic`), because they are more common on the Web and more rows are likely to agree on them. Nonetheless, its labels are generally of high quality. On the other hand, Model tends to do well in the cases where there are good labels, but they do not appear for a majority of rows in the table, in a sense, where more subtle reasoning is required. Consequently, Hybrid is the best of both methods.

We obtained similar results for binary relationships except that Majority did not perform well. The reason is that our extractions are more sparse than in the unary case;

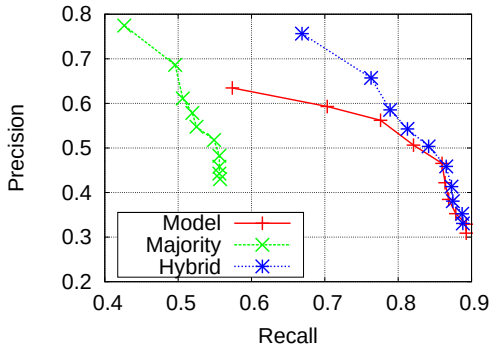


Figure 4.2: Precision/recall for class labels for various algorithms and top k values.

thus, it is harder to find labels that occur for 50% of the rows. Even so, we obtained a precision of 0.45 and a recall of 0.7.

One may wonder if the class labels are not redundant with information that is already on the Web page of the table. In fact, there are only about 60,000 tables in our corpus (4%) where all class labels already appear in the table header, and only about 120,000 tables (8%) where a label appears anywhere in the body of the Web page. Hence, assigning class labels adds important new information to the table.

Labels from ontologies

Next, we compare the coverage of our labeling to what can be obtained by using a manually created ontology. Currently, the state-of-the-art, precision-oriented isA database is YAGO [127], which is based on Wikipedia. Table 4.1 compares the labeling of tables using YAGO vs. the isA database extracted from the Web. Our Web-extracted isA database is able to assign labels to the subject columns of almost 1.5 million tables

(out of 12.3 million tables at hand), while YAGO assigns labels to ∼185 thousand tables (an order of magnitude difference). This is explained by the very large coverage that our Web-extracted repository has in terms of instances (two orders of magnitude larger than YAGO).

Table 4.1: Comparing the isA database and YAGO.

	Web-extracted	**YAGO**	**Freebase**
Labeled subject columns	1,496,550	185,013	577,811
Instances in ontology	155,831,855	1,940,797	16,252,633

For the binary relations, we were able to assign about three times as many labels for pairs of columns than Freebase (2.1M compared to 800K). We also examined the quality of the binary labels on our gold standard that included 128 binary relations involving a subject column. Our method found 83 of them (64.8%) correctly (assigning *vital* or *average* binary labels), whereas Freebase only managed to find 37 of them (28.9%) correctly.

We also compared our labeling on the same datasets (wiki manual and Web manual datasets) used in [79], where the authors proposed using YAGO to label columns in tables. These datasets have tables from Wikipedia and tables that are very related to Wikipedia tables; hence, we expected YAGO to do relatively well. Nonetheless, we achieved an F1 measure of 0.67 (compared to the 0.56 reported in [79]) on the wiki manual dataset, and 0.65 for the Web manual dataset (compared to 0.43), both for the top 10 class labels returned by the majority-based algorithm. We note that using YAGO

will result in higher precision. For the goal of table search though, coverage (and hence recall) is key.

The unlabeled tables

Our methods assigned class labels to approximately 1.5 million tables out of the 12.3 in our corpus when only subject columns are considered, and 4.3 million tables otherwise. We investigated why the other tables were not labeled, and most importantly, whether we are missing good tables in the unlabeled set. We discovered that the vast majority of these tables were either not useful for answering (C, P) queries, or can be labeled using a handful of domain-specific methods. Table 4.2 summarizes the main categories of the unlabeled tables.

Table 4.2: Class label assignment to various categories of tables.

Category	Sub-category	# tables (M)	% of corpus
Labeled	Subject column	1.5	12.20
	All columns	4.3	34.96
Vertical		1.6	13.01
Extractable	Scientific Publications	1.6	13.01
	Acronyms	0.043	0.35
Not useful		4	32.52

First, we found that many of the unlabeled tables are *vertical* tables. These tables contain (attribute name, value) pairs in a long two-column table. We developed an algorithm for identifying such tables by considering tables that had at least two known attribute names in the subject columns (the known attribute names were mined from Wikipedia and Freebase). This process identified around 1.6 million tables. After look-

ing at a random sample of more than 500 of these tables, we found that less than 1% of them would be useful for table-search queries.

Next, we found a few categories where the entities are too specific to be in the isA database. In particular, the most voluminous category comprises tables about publications or patents (1.45 million tables). It turns out that these tables can be identified using simple heuristics from very few sites. Another, much smaller category comprises 43,000 tables of acronyms on a single site. Thus, extending our work to build a few domain-specific extractors for these tables could significantly increase the recall of our class assignment.

Among the remaining 4 million tables, we found that (based on a random sample of 1,000) very few of them are useful for (C, P) queries. In particular, we found that many of these tables have enough text in them to be retrieved by traditional search techniques. Examples of such categories include course description tables (with the course number and university on the page) and comments on social networks, bug reports and job postings.

Thus, although we have annotated about a sixth of our tables, our results indicate that these are the *useful* contents for table-search queries.

4.5.2 Table Search

We now describe the impact of our annotations on table search. We built a table search engine which we refer to as TABLE, that leverages the annotations on tables. Given a query of the form (C, P), where C is a class name and P is a property, TABLE proceeds as follows:

Step 1: Consider tables in the corpus that have the class label C in the top k class labels according to Section 4.4.3.[2] Note that tables that are labeled with C might also contain only a subset of C or a named subclass of C.

Step 2: We rank the tables found in Step 1 based on a weighted sum of the following signals: occurrences of P on the tokens of the schema row, occurrences of P on the assigned binary relations of the table, page rank, incoming anchor text, and number of rows and tokens found in the body of table and the surrounding text. The weights were determined by training on a set of examples. In our current implementation we require that there be an occurrence of P in the schema row (which exist in 71% of the tables [26]) or in the assigned binary relations of the table.

Table 4.3 shows the results of our study. The columns under All Ratings present the number of results (totalled over the 3 users) that were rated to be (a) *right on*, (b) *right on* or *relevant*, and (c) *right on* or *relevant* and *in a table*. The Ratings by Queries

[2] As an extension, when C is not in the isA database, TABLE could search for other class names that are either the correct spelling of C or could be considered related — these extensions are currently not supported in TABLE.

Table 4.3: Results of our user study

Method	All Ratings				Ratings by Queries				Query Precision			Query Recall		
	Total	(a)	(b)	(c)	Some Result	(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)
TABLE	175	69	98	93	49	24	41	40	**0.63**	**0.77**	**0.79**	0.52	0.51	**0.62**
DOCUMENT	399	24	58	47	93	13	36	32	0.20	0.37	0.34	0.31	0.44	0.50
GOOG	493	63	116	52	100	32	52	35	0.42	0.58	0.37	**0.71**	**0.75**	0.59
GOOGR	156	43	67	59	65	17	32	29	0.35	0.50	0.46	0.39	0.42	0.48

columns aggregate ratings by queries: the sub-columns indicate the number of queries for which at least two users rated a result similarly (with (a), (b) and (c) as before). The Precision and Recall are as defined in Section 4.5.2.

In principle, it would be possible to estimate the size of the class C (from our isA database) and to try to find a table in the result whose size is close to C. However, this heuristic has several disadvantages. First, the isA database might have only partial knowledge of the class, and therefore the size estimate may be off. Second, it is very common that the answer is not in a table that is precisely about C. For example, the answer to (*african countries*, *GDP*) is likely to be in a table that includes all of the countries in the world, not only the African countries. Hence, we find that, in general, longer tables tend to provide better answers.

We compare TABLE with three other methods: (1) GOOG: the results returned by `www.google.com`, (2) GOOGR: the intersection of the table corpus with the top 1,000 results returned by GOOG, and (3) DOCUMENT: the document-based approach proposed in [25]. The document-based approach considers several signals extracted from the document in the ranking, including hits on the first two columns, hits anywhere in the table (with a different weight), and hits on the header of the subject column.

Query set: To construct a realistic set of user queries of the form (C, P), we analyzed the query logs from Google Squared, a service in which users search for structured data. We compiled a list of 100 queries (*i.e.*, class names) submitted by users to the Web site. For each class name, each of the authors identified potential relevant property names. Then, we randomly selected two properties for each class name to create a test set of 200 class-property queries. We chose a random subset of 100 out of the 200 queries.

Evaluation methodology: We performed a user study to compare the results of each algorithm. For the purpose of this experiment, each algorithm returns Web pages (if an algorithm originally returned Web tables, we now modified it to return the Web pages containing those Web tables). For each of the 100 queries, we retrieved the top five results using each of TABLE, DOCUMENT, GOOG, and GOOGR. We combine and randomly shuffle these results, and present to the user this list of at most 20 search results (only GOOG is always guaranteed to return five results). For each result, the user had to rate whether it was *right on* (has all information about a large number of instances of the class and values for the property), *relevant* (has information about only some of the instances, or of properties that were closely related to the queried property), or *irrelevant*. In addition, the user marked if the result, when *right on* or *relevant*, was contained *in a table*. The results for each query were rated independently by three separate users.

Note that, by presenting a combined shuffled list of search results, and asking the user to rate the resulting Web documents, we can determine which algorithm produced each result. We cannot present the extracted tables directly to the users, because GOOG does not always retrieve results with tables. Furthermore, we do not ask users to compare directly the ranked lists of results listed separately by each algorithm, because it might be possible for a rater to work out which algorithm produced each list. Thus, we are able to achieve a fair comparison to determine which algorithm can retrieve information (not just tables) that is relevant to a user query.

Precision and recall: The results of our user evaluation are summarized in Table 4.3. We compare the different methods using measures similar to the traditional notions of precision and recall. Let $N_q(m)$ denote the number of queries for which the method m retrieved some result, $N_q^a(m)$ denote the number of queries for which m retrieved some result that was rated *right on* by at least two users, and $N_q^a(*)$ denote the number of queries for which some method retrieved a result that was rated *right on*. We define $P^a(m)$ and $R^a(m)$ to be:

$$P^a(m) = \frac{N_q^a(m)}{N_q(m)}, \qquad R^a(m) = \frac{N_q^a(m)}{N_q^a(*)}.$$

Note that we can likewise define $P^b(m)$ and $R^b(m)$ by considering results that were rated *right on* or *relevant* and $P^c(m)$ and $R^c(m)$ by considering results that were rated

in a table (*right on* or *relevant*). Note that each $P(m)$ and $R(m)$ roughly correspond to the traditional notions of precision and recall.

In our experiments, we found $N_q^a(*) = 45$ (*right on*), $N_q^b(*) = 75$ (*right on* or *relevant*), and $N_q^c(*) = 63$ (*in a table*). The resulting values for precision and recall are listed in Table 4.3. Note that we could likewise define these measures in terms of the number of results (the patterns are similar).

Results: As shown in Table 4.3, TABLE has the highest precision (0.79 when considering *right on* and *relevant* results). These results show that even modest recovery of table semantics leads to very high precision. GOOG on the other hand, has a much higher recall, but a lower precision.

We note that the recall performance of GOOG is based on retrieving Web pages that are relevant Web pages (not necessarily tables that are *right on*). In fact, the precision of GOOG is lower, if we consider only the *right on* ratings (0.42). If we consider only the queries for which the relevant information was eventually found in a table, TABLE has both the highest precision (0.79) and highest recall (0.62) and clearly outperforms GOOG. These results show that not only does TABLE have high precision, but it does not miss many tables that are in the corpus. Hence, we can use TABLE to build a search service for tables, and when it returns too few answers, we can fall back on general Web search.

Observe that DOCUMENT does not perform well in comparison to either TABLE or GOOG. The probable reason is that DOCUMENT (as described in [25]) was designed to perform well for instance queries. DOCUMENT does not have the benefit of class labels, which are no doubt important for class-property queries. DOCUMENT is like GOOG, but with a far smaller corpus (only our $\sim$4.3 million extracted tables), and hence has poor performance.

GOOGR in general has a higher precision and lower recall than GOOG. GOOGR filters the results from GOOG to include only Web pages that have tables with class labels. Thus, GOOGR will retrieve information present in the tables (higher precision and excellent at answering class-property queries), but omits relevant Web pages without tables.

Our results clearly demonstrate that, whenever there is a table that satisfies a class-property query, our table search algorithm is likely to retrieve it. At the same time, it rarely retrieves irrelevant tables.

The importance of subject columns: In our experiments we considered labels on any columns in the tables, but we observe the importance of subject columns in two ways. First, in 80.16% of the results returned by TABLE, the class label was found in the subject column. For the other approximately 20%, we typically observed tables that had more than one possible subject column. Second, in our collection of 168 tables for which we know the subject column and the binary relations, we observed the following.

Of the pairs of columns that involved a subject, our algorithms found labels in 43.3% of the cases, compared to only 19.1% for pairs of arbitrary columns.

4.6 Summary

In this chapter, we have described algorithms for partially recovering the semantics of tables on the Web. We explored an intriguing interplay between structured and unstructured data on the Web, where we used text on the Web to recover the semantics of structured data on the Web. Because the breadth of the Web matches the breadth of structured data on the Web, we are able to recover the semantics effectively. In addition, we have provided a detailed analysis of when our techniques will not work and how these limitations can be addressed.

Chapter 5

Modeling Information Flow in Collaborative Networks

Ticket resolution is a crucial aspect of the delivery of Information Technology (IT) services. A large service provider needs to handle, on a daily basis, thousands of tickets that report various types of problems. Many of those tickets bounce among multiple expert groups before being transferred to the group with the right expertise to solve the problem. Finding a methodology that reduces such bouncing and hence shortens ticket resolution time is a long-standing challenge. In this chapter, we present a unified generative model, the Optimized Network Model (ONM), that characterizes the lifecycle of a ticket, using both the content and the routing sequence of the ticket. ONM uses maximum likelihood estimation, to represent how the information contained in a ticket

is used by human experts to make ticket routing decisions. Based on ONM, we develop a probabilistic algorithm that generates ticket routing recommendations for new tickets in a network of expert groups. Our algorithm calculates all possible routes to potential resolvers and makes globally optimal recommendations, in contrast to existing classification methods that make static and locally optimal recommendations.

5.1 Motivation

Problem ticket resolution is critical to the IT services business. A service provider might need to handle, on a daily basis, thousands of tickets that report various types of problems from its customers. The service provider's ability to resolve the tickets in a timely manner determines, to a large extent, its competitive advantage. To manage ticket resolution effectively, human experts are often organized into expert groups, each of which has the expertise to solve certain types of problems. As IT systems become more complex, the types of reported problems become more diverse. Finding an expert group to solve the problem specified in a ticket is a long-standing challenge for IT service providers.

In practice, a typical ticket processing system works as follows. A ticket is initiated by a customer or by internal staff, and is subsequently routed through a network of expert groups for resolution. The ticket is closed when it reaches a *resolver group*

that provides the solution to the problem reported in the ticket. Figure 5.1 shows an interaction network between groups with ticket routing examples. Ticket t_1 starts at group A and ends at group D, and ticket t_2 starts at group G and ends at group C (note that we omit the dispatching step in which a ticket is first assigned to the initial group). The sequences $A \rightarrow B \rightarrow C \rightarrow D$ and $G \rightarrow E \rightarrow C$ are called *ticket routing sequences*.

In a large network of expert groups, being able to quickly route a new ticket to its resolver is essential to reduce labor cost and to improve customer satisfaction. Today, ticket routing decisions are often made manually and, thus, can be quite subjective and error-prone. Misinterpretation of the problem, inexperience of human individuals, and lack of communication between groups can lead to routing inefficiency. These difficulties call for models that can accurately represent the collaborative relationship between groups in solving different kinds of problems. Such models ought to provide fine-grain information not only to help experts reduce ticket routing errors, but also

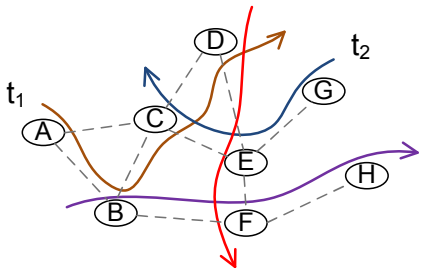


Figure 5.1: Ticket routing.

to help service enterprises better understand group interactions and identify potential performance bottlenecks.

In [117] Shao *et al.* proposed a Markov model-based approach to predict the resolver of a ticket, based on the expert groups that processed the ticket previously. In essence, their approach is a rule-based method, *i.e.*, if group A processed a ticket and did not have a solution, it calculates the likelihood that group B can resolve it. A drawback of that approach is that it is locally optimized and, thus, might not be able to find the best ticket routing sequences. Moreover, it does not consider the contents of the tickets. That is, it uses a "black-box" approach that can neither explain, nor fully leverage, the information related to why group A transfers a ticket to group B, when it cannot solve the problem itself.

In this work, we aim to address these issues by deriving a more comprehensive model that incorporates ticket content. Rather than simply calculating the transfer probability $P(B|A)$ between two groups A and B, we build a generative model that captures why tickets are transferred between two groups A and B, *i.e.*, $P(w|A \rightarrow B)$ where w is a word in the ticket. In addition, we build a model that captures why a certain ticket can be resolved by a group B, *i.e.*, $P(w|B)$. Finally, we combine the local generative models into a global model, the Optimized Network Model (ONM), which represents the entire ticket resolution process in a network of expert groups.

The Optimized Network Model has three major applications. First, it can be trained using historical ticket data and then used as a recommendation engine to guide the routing of new tickets. Second, it provides a mechanism to analyze the role of expert groups, to assess their expertise level, and to study the expertise awareness among them. Third, it can be used to simulate the ticket routing process, and help analyze the performance of an expert network under various ticket workloads. We focus on the first application and demonstrate the superior performance of ONM compared to previous models. We briefly discuss the other two applications, but leave the detailed studies of those applications for future work.

5.2 Related Work

Ticket routing can be considered an extension of the text classification problem, which has been extensively studied in the literature [16, 28, 68, 84, 114, 145, 146, 162]. For instance, Yang and Liu [145] studied the robustness of different text categorization methods. Calado *et al.* [28], Lu and Getoor [84], and Sen *et al.* [114] proposed methods to combine content and link information for document classification.

Ticket routing is also related to the multi-class classification problem [105]. Compared to multi-class classification, ticket routing has distinct properties. First, ticket routing involves multiple predictions if the current prediction is not correct, which leads

to different evaluation criteria. Second, ticket routing takes place in a network, which is also different from the traditional classification problem. Third, instead of relying on a single classifier, ticket routing requires leveraging the interactions between multiple local classifiers to find a globally optimized solution.

Belkin *et al.* [16] and Zhou *et al.* [162] introduced text classification using graph-based methods. Collective classification, such as loopy belief propagation [148], mean field relaxation labeling [147], interactive classification [97] and stacked models [72], are popular techniques for classifying nodes in a partially labeled graph. The problems studied in those papers are quite different from our problem, as we assume one resolver in the network for a given ticket, and the classification needs to be repeatedly applied until the resolver is found.

Generative models and maximum likelihood estimation are standard approaches. Generative models seek the joint probability distribution over the observed data. Classification decisions are typically made based on conditional probabilities formed using Bayesian rules. One example is the Naive Bayes classifier [66, 154], which assumes conditional independence between variables. Another example is the Gaussian Mixture Model [104], which estimates the probability distribution using a convex combination of several Gaussian distributions. These models are good for analyzing sparse data. We chose the generative model because the transition probabilities in the ticket resolution sequences can be seamlessly embedded in the probabilistic framework. Our

contribution is the combination of multiple local generative models to yield a globally optimized solution.

Besides the generative models, discriminative models, such as the Support Vector Machine (SVM), have been shown to be effective for text classification [68]. One can potentially build a support vector classifier for each resolver and each transfer relationship. However, they are locally optimized for individual resolvers and transfer relationships; once trained, the SVM classifiers remain stationary. In our approach, the resolver predictions can be dynamically adjusted if previous predictions are incorrect.

The ticket routing problem is also related to the expert finding problem, *i.e.*, given a keyword query, find the most knowledgeable persons regarding that query. The expert finding algorithms proposed by Balog *et al.* [11] and Fang and Zhai [53] use a language model to calculate the probability of an expert candidate to generate the query terms. Serdyukov *et al.* [115] enhanced those models by allowing the candidates' expertise to be propagated within a network, *e.g.*, via email. Deng *et al.* [39] explored the links in documents such as those listed in DBLP [3]. Expert recommendation systems also use text categorization techniques to characterize bugs [7] and documents [122]. Because most expert finding algorithms are content-based, they share the same weakness of the Resolver Model (RM) given in Section 5.4.1.

Our study demonstrates that better routing performance can be achieved by combining together ticket contents and routing sequences. Nevertheless, considering existing

sophisticated text classification methods and language models, it is an open research problem to investigate how to embed these models in a collaborative network and learn their parameters in a holistic way for ticket processing, a challenging problem in the IT service industry.

5.3 Preliminaries

We use the following notation: $\mathcal{G} = \{g_1, g_2, ..., g_L\}$ is a set of expert groups in a collaborative network; $\mathcal{T} = \{t_1, t_2, ..., t_m\}$ is a set of tickets; and $\mathcal{W} = \{w_1, w_2, ..., w_n\}$ is a set of words that describe the problems in the tickets. A ticket consists of three components: (1) a problem category to which the ticket belongs, *e.g.*, a WINDOWS problem or a DB2 problem, that is identified when the ticket is generated, (2) the ticket content, *i.e.*, a textual description of the problem symptoms, and (3) a routing sequence from the initial group to the final resolver group of the ticket. Although some complex tickets can be associated with multiple problem categories or can involve multiple resolvers, most tickets are associated with one problem category and can be resolved by one expert group. Our model focuses on ticket routing in these common cases.

In the first step of routing, each ticket t is assigned to an initial expert group $g_{init}(t)$. If the initial group cannot solve the problem, it transfers the ticket to another group that it considers the right candidate to solve the problem. After one or more transfer steps,

the ticket eventually reaches the resolver group $g_{res}(t)$. The route that the ticket takes in the expert network is denoted $R(t)$. Table 5.1 shows a ticket example, which is first assigned to group HDBTOIGA, and is finally resolved by group NUS_N_DSCTS.

Table 5.1: A WINDOWS ticket example.

ID	Description	Initial Group
8805	User received an error R=12 when installing Hyperion. When tried to install again, got success msg, but unable to open the application in Excel	HDBTOIGA

ID	Time	Entry
8805	9/29/2006	... (multi transfer steps) ...
8805	10/2/2006	Ticket 8805 transferred to Group NUS_N_DSCTS
8805	10/2/2006	Resolution: Enabled Essbase in Excel

To model the interactions between groups in an expert network, we need to understand how and why the tickets are transferred and resolved. Specifically, we aim to develop a modeling framework that consists of (1) a Resolution Model $M_g(t)$ that captures the probability that group g resolves ticket t, and (2) a Transfer Model $M_{g_i \to g_j}(t)$ that captures the probability that group g_i transfers ticket t to group g_j, if g_i cannot resolve t. Our goal is to develop these two models, and then combine them into a unified network model, that represents the ticket lifecycle in the expert network, as shown in Figure 5.2.

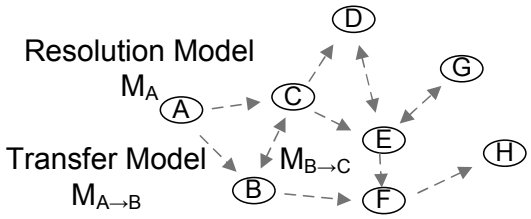


Figure 5.2: Unified network model.

5.4 Generative Models

The ticket contents and routing sequences of the historical tickets provide clues as to how tickets are routed by expert groups. In our expert network, each group has its own special expertise. Thus, if an expert group is capable of resolving one ticket, chances are it can also resolve other tickets with similar problem descriptions. Likewise, similar tickets typically have similar routing paths through the network. In this section, we characterize these properties using generative models.

5.4.1 Resolution Model (RM)

First, we build a generative model for each expert group using the textual descriptions of the problems the group has solved previously. Given a set $\mathcal{T}_i$ of tickets resolved by group g_i and $\mathcal{W}$ the set of words in the tickets in $\mathcal{T}_i$ we build a resolver profile P_{g_i} defined as the following column vector:

$$P_{g_i} = [P(w_1|g_i), P(w_2|g_i), ..., P(w_n|g_i)]^T \tag{5.1}$$

Equation (5.1) represents the word distribution among the tickets resolved by g_i. Here, $P(w_k|g_i)$ is the probability of choosing w_k if we randomly draw a word from the descriptions of all tickets resolved by g_i. Thus, $\sum_{k=1}^{n} P(w_k|g_i) = 1$.

Assuming that different words appear independently in the ticket content, the probability that g_i can resolve a ticket $t \in \mathcal{T}_i$ can be calculated from the resolver profile vector P_{g_i} as follows:

$$P(t|g_i) \propto \prod_{w_k \in t} P(w_k|g_i)^{f(w_k,t)} \tag{5.2}$$

where w_k is a word contained in the content of ticket t and $f(w_k, t)$ is the frequency of w_k in the content of t.

To find a set of most probable parameters $P(w_k|g_i)$, we use the maximum likelihood method. The likelihood that group g_i resolves all of the tickets in $\mathcal{T}_i$ is:

$$\mathcal{L}(\mathcal{T}_i, g_i) = \prod_{t \in \mathcal{T}_i} P(t|g_i) \tag{5.3}$$

We maximize the log likelihood:

$$\begin{aligned} P_{g_i} &= \arg\max_{P(\mathcal{W}|g_i)} (\log(\mathcal{L}(\mathcal{T}_i, g_i))) \\ &= \arg\max_{P(\mathcal{W}|g_i)} (\sum_{w_k} n(w_k, \mathcal{T}_i) \log(P(w_k|g_i))) \\ s.t. \quad & \sum_{w_k \in \mathcal{W}} P(w_k|g_i) = 1 \end{aligned}$$

where $n(w_k, \mathcal{T}_i) = \sum_{t \in \mathcal{T}_i} f(w_k, t)$ is the total frequency of the word w_k in the ticket set $\mathcal{T}_i$. Hence, the maximum likelihood solution for the resolver profile vector P_{g_i} is:

$$P(w_k|g_i) = \frac{n(w_k, \mathcal{T}_i)}{\sum_{w_j \in \mathcal{W}} n(w_j, \mathcal{T}_i)} \tag{5.4}$$

The Resolution Model is a standard multi-class text classifier, which considers only ticket content. Embedded in the ticket routing sequences are the transfer relations between groups, which can be used to improve the accuracy of our model, as described below.

5.4.2 Transfer Model (TM)

As Shao *et al.* [117] pointed out, not only the resolver group, but also the intermediate groups in the ticket routing sequences, contribute to the resolution of a ticket. The reason is that, even if an expert group cannot solve a problem directly, it might have knowledge of which other group is capable of solving it. To capture this effect, we use both the ticket content and the routing sequence to model the transfer behavior between expert groups.

Considering an edge $e_{ij} = g_i \rightarrow g_j$ in the expert network, we let $\mathcal{T}_{ij}$ denote the set of tickets that are transferred along the edge e_{ij} and let $\mathcal{W}$ denote the set of words in the tickets in $\mathcal{T}_{ij}$. Using the same technique as described in Section 5.4.1, we build the

transfer profile of an edge between two expert groups as the column vector:

$$P_{e_{ij}} = [P(w_1|e_{ij}), P(w_2|e_{ij}), ..., P(w_n|e_{ij})]^T \tag{5.5}$$

where $P_{e_{ij}}$ characterizes the word distribution among the tickets routed along edge e_{ij} and $P(w_k|e_{ij})$ is the probability of choosing word w_k if we randomly draw a word from the tickets transferred along edge e_{ij}. Similarly, we derive the maximum likelihood solution for the transfer profile of e_{ij} as follows:

$$P(w_k|e_{ij}) = \frac{n(w_k, \mathcal{T}_{ij})}{\sum_{w_\ell \in \mathcal{W}} n(w_\ell, \mathcal{T}_{ij})} \tag{5.6}$$

The Transfer Model for the edges can be combined with the Resolution Model for the nodes to form the network model shown in Figure 5.2. However, the parameters of these models are learned independently and, thus, might not achieve the best modeling accuracy. To address this problem, we study how to optimize the network model by learning these parameters globally.

5.4.3 Optimized Network Model (ONM)

Both the Resolution Model and the Transfer Model are local models. They are not optimized for end-to-end ticket routing in the expert network. In this section, we present an optimized model that accounts for the profiles of the nodes and edges together in a global setting. Instead of considering only the tickets resolved by a certain expert group or transferred along a certain edge, the model learns its parameters based on the entire

set of tickets, using both their contents and their routing sequences. As we will see, this global model outperforms the local models.

Routing Likelihood

When a set $\mathcal{T}_i$ of tickets is routed to a group g_i, some of the tickets will be resolved if g_i has the right expertise, while the rest of the tickets will be transferred to other groups. If g_i resolves a ticket, we assume that g_i transfers the ticket to itself. We let $\mathcal{T}_{ij}$ be the set of tickets that are transferred from group g_i to group g_j. Thus, $\mathcal{T}_i = \bigcup_{j=1}^{L} \mathcal{T}_{ij}$, where $\mathcal{T}_{ii}$ is the set of tickets resolved by group g_i itself, and L is the number of expert groups.

Given a ticket t and the expert group g_i that currently holds the ticket t, the probability that t is transferred from group g_i to group g_j is:

$$\begin{aligned} P(g_j|t, g_i) &= \frac{P(t|e_{ij})P(g_j|g_i)}{Z(t, g_i)} \\ &= \frac{(\prod_{w_k \in t} P(w_k|e_{ij})^{f(w_k,t)})P(g_j|g_i)}{Z(t, g_i)} \end{aligned} \tag{5.7}$$

where $Z(t, g_i) = \sum_{g_j \in \mathcal{G}} P(t|e_{ij})P(g_j|g_i)$ and $P(g_j|g_i)$ is the prior probability that g_i transfers a ticket to g_j. $P(g_j|g_i)$ can be estimated by $|\mathcal{T}_{ij}|/|\mathcal{T}_i|$. To simplify the notation, we let $P(g_i|t, g_i)$ represent the probability that group g_i is able to resolve ticket t if t is routed to g_i. Hence, $P(w|e_{ii})$ is the resolution model of g_i. Because a ticket description is often succinct with few redundant words, we assume $f(w_k, t) = 1$ if w_k occurs in t

and $f(w_k, t) = 0$ otherwise. This assumption significantly simplifies the derivation of the model.

Each historical ticket t has a routing sequence $R(t)$. For example, $R(t) = g_1 \rightarrow g_2 \rightarrow g_3$, with initial group $g_{init}(t) = g_1$ and resolver group $g_{res}(t) = g_3$. We assume that an initial group g_1 is given for each ticket t, *i.e.*, $P(g_1|t) = 1$ and that each expert group makes its transfer decisions independently. In this case, the probability that the routing sequence $g_1 \rightarrow g_2 \rightarrow g_3$ occurs is:

$$\begin{aligned} P(R(t)|t) &= P(g_1|t)P(g_2|t, g_1)P(g_3|t, g_2)P(g_3|t, g_3) \\ &= P(g_2|g_1)P(g_3|g_2)P(g_3|g_3) \\ &\quad \times \frac{P(t|e_{1,2})P(t|e_{2,3})P(t|e_{3,3})}{Z(t, g_1)Z(t, g_2)Z(t, g_3)} \end{aligned}$$

We assume further that the tickets are independent of each other. Thus, the likelihood of observing the routing sequences in a ticket set $\mathcal{T}$ is:

$$\mathcal{L} = \prod_{t \in \mathcal{T}} P(R(t)|t) \tag{5.8}$$

Parameter Optimization

To find a set of globally optimal parameters $P(w_k|e_{ij})$, we use maximum likelihood estimation to maximize the log likelihood:

$$\begin{aligned}\log \mathcal{L} &= \sum_{t\in\mathcal{T}} \log P(R(t)|t) \\ &= \sum_{t\in\mathcal{T}} \sum_{e_{ij}\in R(t)} \log \frac{P(t|e_{ij}) \times P(g_j|g_i)}{Z(t, g_i)} \\ &= \sum_{e_{ij}\in\mathcal{E}} \sum_{t\in\mathcal{T}_{ij}} (\log(P(t|e_{ij})) + \log(P(g_j|g_i))) \\ &\quad - \sum_{g_i\in\mathcal{G}} \sum_{t'\in\mathcal{T}_i} \log(Z(t', g_i))\end{aligned} \tag{5.9}$$

where $\mathcal{E} = \{e_{ij}|1 \le i, j \le L\}$ and $P(t|e_{ij}) = \prod_{w_k\in t} P(w_k|e_{ij})$. The optimal transfer profile is given by the following constrained optimization problem:

$$\begin{aligned}P(\mathcal{W}|\mathcal{E})^* = \arg \max_{P(\mathcal{W}|\mathcal{E})} (\log \mathcal{L}) \\ \text{s.t.} \sum_{w_k\in\mathcal{W}} P(w_k|e_{ij}) = 1; \\ P(w_k|e_{ij}) \ge 0\end{aligned} \tag{5.10}$$

where $\mathcal{W}$ is the set of words and $\mathcal{E}$ is the set of edges.

This optimization problem is not convex, and it involves many free dimensions (the degree of freedom is $(|\mathcal{W}| - 1) \times |\mathcal{G}|^2$). It cannot be solved efficiently with existing tools.

Thus, we seek solutions that are near-optimal but easier to calculate. Our approach is to update the parameters $P(w_k|e_{ij})$ iteratively to improve the likelihood. Specifically,

we use the steepest descent method to maximize the lower bound of the log likelihood. By Jensen's inequality, we have

$$Z(t, g_i) \leq \prod_{w_k \in t} \sum_{g_\ell \in \mathcal{G}} P(g_\ell|g_i) P(w_k|e_{i\ell}) \tag{5.11}$$

Combining Equation (5.9) and Equation (5.11), we have:

$$\begin{aligned} \log \mathcal{L} \quad \geq \quad \lfloor \log \mathcal{L} \rfloor = & \sum_{e_{ij}} \sum_{t \in \mathcal{T}_{ij}} (\log(P(t|e_{ij})) + \log(P(g_j|g_i))) \\ & - \sum_{g_i \in \mathcal{G}} \sum_{t' \in \mathcal{T}_i} \sum_{w_k \in t'} \log(\sum_{g_\ell \in \mathcal{G}} (P(g_\ell|g_i) \times P(w_k|e_{i\ell}))) \end{aligned}$$

The gradient is given by:

$$\begin{aligned} \nabla \lfloor \log(\mathcal{L}) \rfloor \quad = \quad & \frac{\partial \lfloor \log \mathcal{L} \rfloor}{\partial P(w_k|e_{ij})} \\ = \quad & \frac{\sum_{t \in \mathcal{T}_{ij}} n(w_k, t)}{P(w_k|e_{ij})} \\ & - \frac{P(g_j|g_i) \times \sum_{t' \in \mathcal{T}_i} n(w_k, t')}{\sum_{g_\ell \in \mathcal{G}} P(g_\ell|g_i) \times P(w_k|e_{i\ell})} \end{aligned}$$

Using the values of $P(w_k|e_{ij})$ calculated in Equation (5.6) as the starting point, we iteratively improve the solution along the gradient. To satisfy the constraints, we calculate the projection of the gradient in the hyperplane defined by $\sum_{w_k \in \mathcal{W}} P(w_k|e_{ij}) = 1$ to ensure that the solution stays in the feasible region. The profiles of the edges in the network are updated one at a time, until they converge. Although the gradient-based method might produce a local optimum solution, it estimates the model parameters all together from a global perspective and provides a better estimation than the TM locally-optimized solution.

5.5 Ticket Routing

We now study the application of the generative models presented in Section 5.4 to ticket routing.

Given a new ticket t and its initial group $g_{init}(t)$, a routing algorithm uses a model $\mathcal{M}$ to predict the resolver group $g_{res}(t)$. If the predicted group is not the right resolver, the algorithm keeps on predicting, until the resolver group is found. The performance of a routing algorithm can be evaluated in terms of the number of expert groups it tried until reaching the resolver. Specifically, we let the predicted routing sequence for ticket t_i be $R(t_i)$ and let $|R(t_i)|$ be the number of groups tried for ticket t_i. For a set of testing tickets $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$, we evaluate the performance of a routing algorithm using the Mean Number of Steps To Resolve (MSTR) [117] given by:

$$S = \frac{\sum_{i=1}^{m} |R(t_i)|}{m} \tag{5.12}$$

The ticket routing problem is related to the multi-class classification problem in that we are seeking a resolver (class label) for each ticket. Different from a classification problem, our goal here is not to maximize the classification precision, but to minimize the expected number of steps before the algorithm reaches the right resolver.

Nevertheless, we can adapt a multi-class classifier to fit our problem. We assume that a classifier C predicts group g as the resolver of ticket t, with probability $P(g|t)$. A simple approach is to rank the potential resolver groups in descending order of $P(g|t)$

and then transfer the ticket t to them one by one, until the right resolver is found. In this approach, the ranking of groups does not change, even if the current prediction is incorrect. We take the Resolution Model as an example, and as the baseline method, for building a classifier. Then, we develop two dynamic ranking methods, using the Transfer Model and the Optimized Network Model, to achieve better performance.

5.5.1 Ranked Resolver

The Ranked Resolver algorithm is designed exclusively for the Resolution Model (RM). Expert groups are ranked based on the probability that they can resolve the ticket according to the ticket content.

Given a new ticket t, the probability that expert group g_i can resolve the ticket is:

$$\begin{aligned} P(g_i|t) &= \frac{P(g_i)P(t|g_i)}{P(t)} \\ &\propto P(g_i) \prod_{w_k \in t} P(w_k|g_i)^{f(w_k,t)} \end{aligned} \tag{5.13}$$

Here, $P(g_i)$ is the prior probability of group g_i being a resolver group, which is estimated by $|\mathcal{T}_i|/|\mathcal{T}|$, where $\mathcal{T}_i$ is the set of tickets resolved by g_i and $\mathcal{T}$ is the ticket training set.

A routing algorithm for this model is to try different candidate resolver groups in descending order of $P(g_i|t)$. The algorithm works fine unless the new ticket t contains a word that has not appeared in the training ticket set $\mathcal{T}$. In that case, $P(g_i|t)$ is zero

for all i. To avoid this problem, we introduce a smoothing factor λ to calculate the probability, *i.e.*,

$$P(w|g_i)^* = \lambda \times P(w|g_i) + (1 - \lambda)/|\mathcal{W}| \tag{5.14}$$

Using the smoothed value $P(w|g_i)^*$ guarantees a positive value of $P(g_i|t)$ for all i.

5.5.2 Greedy Transfer

The Greedy Transfer algorithm makes one step transfer predictions and selects the most probable resolver as the next step.

When a new ticket t first enters the expert network, it is assigned to an initial group g_{init}. Instead of calculating which group is likely to solve the problem, we determine the group to which the ticket should be transferred, because tickets should be transferred to the group that can solve the problem or the group that knows which group can solve the problem. The probability that a ticket t is routed through the edge $e_{init,j} = g_{init} \rightarrow g_j$, where $g_j \in \mathcal{G} \setminus \{g_{init}\}$, is:

$$\begin{aligned} P(g_j|t, g_{init}) &= \frac{P(g_j|g_{init})P(t|e_{init,j})}{\sum_{g_l \in \mathcal{G}} P(g_l|g_{init})P(t|e_{init,l})} \\ &= \frac{P(g_j|g_{init}) \prod_{w_k \in t} P(w_k|e_{init,j})^{f(w_k,t)}}{\sum_{g_l \in \mathcal{G}} P(g_l|g_{init}) \prod_{w_k \in t} P(w_k|e_{init,l})^{f(w_k,t)}} \end{aligned} \tag{5.15}$$

Note that smoothing is applied as in Equation (5.14).

The expert group $g^* = \arg\max_{g_j \in \mathcal{G}} P(g_j|t, g_{init})$ is selected to be the next expert group to handle ticket t. If g^* is the resolver, the algorithm terminates. If not, the

algorithm gathers the information of all previously visited expert groups to make the next step routing decision. If a ticket t has gone through the expert groups in $R(t)$ and has not yet been solved, the rank of the remaining expert groups in $\mathcal{G} \setminus R(t)$ is:

$$Rank(g_j) \propto \max_{g_i \in R(t)} P(g_j|t, g_i) \tag{5.16}$$

and the ticket is routed to the group with the highest rank. The rank of g_j is determined by the maximum probability of $P(g_j|t, g_i)$ for all the groups g_i that have been tried in the route. The ranked order of the candidate resolvers might change during routing.

5.5.3 Holistic Routing

Th Holistic Routing algorithm recognizes the most probable resolver that can be reached within K transfer steps, and selects the next group from a global perspective. Based on our experiments, we set K equal to 3. Instead of predicting only one step as do the Ranked Resolver and Greedy Transfer algorithms, the Holistic Routing algorithm calculates the probability that a candidate group can be reached and can solve the ticket in multiple steps.

For a new ticket t, the one step transition probability $P(g_j|t, g_i)$ between two expert groups g_i and g_j is calculated using Equation (5.15). Thus, we perform a breadth-first search to calculate the probability that a ticket t is transferred by g_i to g_j in exactly K

steps. This probability can be estimated iteratively, using the following equations:

$$\begin{aligned} P(g_j, 1|t, g_i) &= \begin{cases} P(g_j|t, g_i) & \text{if } i \neq j \\ 0 & \text{otherwise} \end{cases} \\ P(g_j, K|t, g_i) &= \sum_{g_k \in \mathcal{G}; k \neq j} P(g_k, K-1|t, g_i) P(g_j|t, g_k) \\ & \qquad \text{if } K > 1. \end{aligned}$$

If $g_l = g_{init}$ the initial group for ticket t, the above equation can be written as:

$$P(g_j, K|t, g_l) = vM^K \tag{5.17}$$

where v is the unit vector whose lth component is 1 and other components are 0. The one step transfer probability matrix M is a $|\mathcal{G}| \times |\mathcal{G}|$ matrix, where an entry of M is the one step transition probability between the expert groups g_i and g_j given by:

$$M(i,j) = \begin{cases} P(g_j|t, g_i) & \text{if } i \neq j \\ 0 & \text{otherwise} \end{cases}$$

The probability that g_j can resolve the ticket t in K or fewer steps starting from the initial group g_{init} (which is used to rank the candidate resolver groups) is:

$$Rank(g_j|g_{init}) \equiv \sum_{k=1}^{K} P(g_j, k|t, g_{init}) \times P(g_j|t, g_j) \tag{5.18}$$

where $P(g_j|t, g_j)$ is the probability that g_j resolves t if t reaches g_j (see Equation (5.7)). Starting with g_{init}, we route t to the group $g^* = \arg\max_{g_j \in \mathcal{G}; j \neq init} Rank(g_j|g_{init})$.

Theoretically, we can derive the rank in closed form for an infinite number of transfer steps. In practice, M^K decays quickly as K increases, due to the probability of solving the ticket at each step. A small value of K suffices to rank the expert groups.

Given the predicted expert group g_k, if ticket t remains unresolved and needs to be transferred, the posterior probability of g_k being the resolver for t is zero and the one step transfer matrix M needs to be updated accordingly. Thus, if g_k is not the resolver, the elements in the kth row of M are updated by:

$$M(k,j) = \frac{P(g_j|t,g_k)}{\sum_{i,i\neq k} P(g_i|t,g_k)} \text{ for } j \neq k$$

Once M is updated, the algorithm reranks the groups according to Equation (5.18) for each visited group in $R(t)$. That is, $Rank(g_j) \propto max_{g_i \in R(t)} Rank(g_j|g_i)$. The group with the highest rank is selected as the next possible resolver.

For a given new ticket, the Holistic Routing algorithm is equivalent to enumerating all of the possible routes from the initial group to any candidate group. For each route $r = \{g_1, g_2, \ldots, g_m\}$ for a ticket t, we calculate the probability of the route as:

$$P(r|t) = P(g_m|t,g_m) \prod_{1\leq j\leq m-1} P(g_{j+1}|t,g_j)$$

The probability that group g_j resolves ticket t is:

$$Rank(g_j) \equiv \sum_r P(r|t) \text{ for all r ending at } g_j$$

Figure 5.3 shows an example where a ticket t enters the expert network at group A. We enumerate all of the routes that start at A and end at D to calculate how likely

D resolves the ticket. Note that loops in the routes are allowed in the calculation in Equation (5.17). It is also possible to calculate the resolution probability without loops. However, because the intermediate groups for each route must be remembered, the calculation might take a long time.

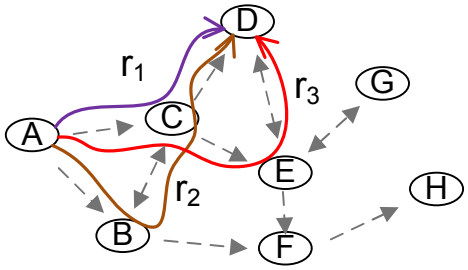


Figure 5.3: Holistic routing.

5.6 Experimental Results

To validate the effectiveness of our models and the corresponding routing algorithms,[1] we use real-world ticket data. The evaluation is based on problem tickets collected from IBM's problem ticketing system throughout 2006. When a ticket enters the system, the help desk assigns a category indicating a problem category for the ticket. For each problem category, a number of expert groups (ranging from 50 to 1,000) are involved in resolving the tickets.

[1] The source code is available at http://www.uweb.ucsb.edu/ ∼miao/resources.html.

For each problem category, we partition the dataset into the training dataset and the testing dataset. Using the training dataset, first we build the generative models introduced in Section 5.4. Then, we evaluate the effectiveness of the routing algorithms by calculating the number of routing steps (*i.e.*, MSTR) for the testing tickets. In particular, we compare our generative models with the Variable-Order Markov Model (VMS) proposed in [117]. Our experiments demonstrate:

- **Model Effectiveness:** The *Optimized Network Model* significantly outperforms the other models.
- **Routing Effectiveness:** Among the Ranked Resolver, Greedy Transfer and Holistic Routing algorithms, Holistic Routing achieves the best performance.
- **Robustness:** With respect to the size of the training dataset, the time variability of the tickets, and the different problem categories, our solution that combines *ONM* and *Holistic Routing* consistently achieves good performance.

We obtained our experimental results using an Intel Core2 Duo 2.4GHz CPU with 4GB memory.

5.6.1 Datasets

We present the results obtained from tickets in three major problem categories: AIX (operating system), WINDOWS (operating system), and ADSM (storage management),

as shown in Table 5.2. Tickets in these three categories have quite different characteristics. The problem descriptions for WINDOWS and ADSM tickets tend to be more diverse and, hence, more challenging for our models.

Table 5.2: Ticket resolution datasets.

Category	# of tickets	# of words	# of groups
AIX	18,426	16,065	847
WINDOWS	16,441	8,521	638
ADSM	3,563	1,815	301

These three datasets involve approximately 300 to 850 expert groups. For a new ticket, finding a resolver group among so many candidates can be challenging.

Table 5.3: Resolution steps distribution.

Steps	Percentage
2	68%
3	25%
4	6%
$\geq$5	1%

Table 5.3 shows the distribution of resolution steps for tickets in the WINDOWS category. We are more interested in solving tickets with long resolution sequences, because these tickets received most of the complaints.

5.6.2 Model Effectiveness

In this section, we compare the effectiveness of the three generative models, Resolution Model (RM), Transfer Model (TM), and Optimized Network Model (ONM) developed in Section 5.4, against the Variable-Order Markov Model (VMS) introduced in [117]. VMS considers only ticket routing sequences in the training data.

Each of the above models has its corresponding routing algorithm. VMS uses the conditional transfer probability learned from routing sequences to predict the resolver group. For RM, we use the Ranked Resolver algorithm. For TM and ONM, we can use either the Greedy Transfer algorithm or the Holistic Routing algorithm. In these experiments, we use the Holistic Routing algorithm to evaluate both models. For comparison, we also include the result of ONM using the Greedy Transfer algorithm. More details for the comparison between the Greedy Transfer algorithm and the Holistic Routing algorithm are shown in Section 5.6.3.

Because a routing algorithm might generate an extremely long routing sequence to resolve one ticket (considering that we have more than 300 expert groups in each problem category), we apply a cut-off value of 10. That is, if an algorithm cannot resolve a ticket within 10 transfer steps, it is regarded as unresolvable. Using this cut-off value, we define the *resolution rate* of a ticket routing algorithm to be the proportion of tickets that are resolvable within 10 steps.

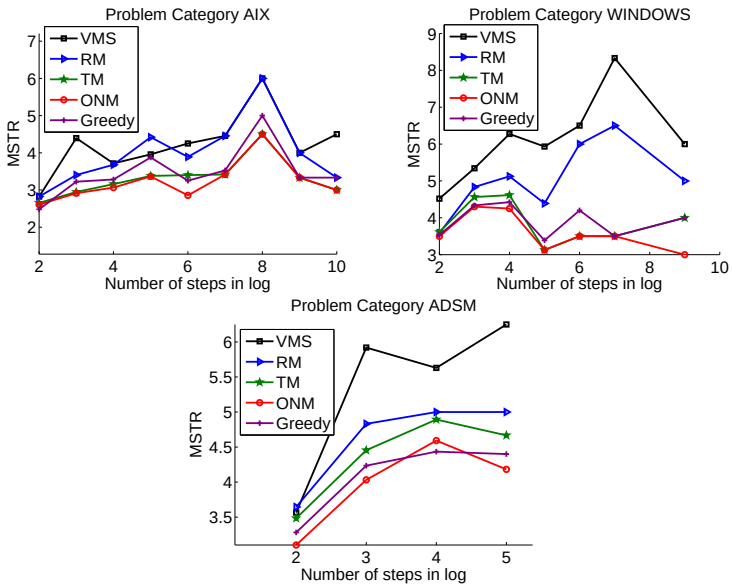


Figure 5.4: Prediction accuracy of different models.

We randomly divide the tickets in each problem category into two subsets: the training dataset and the testing dataset, where the former contains 75% of the tickets, and the latter contains 25% of the tickets. The four models are trained based on the training set, and the performance of the algorithms is compared.

Figure 7.5 compares the prediction accuracy of the four models. The x-axis represents the number of expert groups involved in the testing dataset, where the routing decisions are made by a human. The y-axis represents the resulting MSTR when the testing tickets are routed automatically using a model. Obviously, smaller MSTR means better prediction accuracy. As shown in the figure, TM and ONM (which combine the ticket contents and the routing sequences) result in better prediction accuracy than ei-

ther the sequence-only VMS model or the content-only RM. Moreover, ONM achieves better performance than TM, which indicates that the globally optimized model is more accurate in predicting a ticket resolver than the locally optimized model.

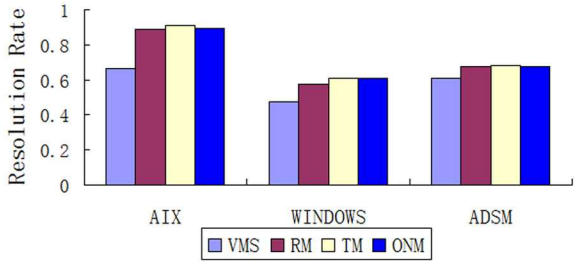


Figure 5.5: Resolution rate.

Combining together the ticket contents and the routing sequences not only boosts prediction accuracy, but also increases the resolution rate of the routing algorithm. Figure 5.5 shows that TM and ONM can resolve more tickets than either VMS or RM.

For RM and TM, the training time is mainly spent on counting word frequencies on transfer edges and at resolvers. For all three data sets, the time is less than 5 minutes. For ONM, the transfer profiles are updated one at a time and the optimization process repeats for multiple rounds until the transfer profiles converge. The training process takes less than 3 hours for all three datasets.

5.6.3 Routing Effectiveness

Using the same experimental setup as in Section 5.6.2, we compare the effectiveness of the Greedy Transfer and Holistic Routing algorithms.

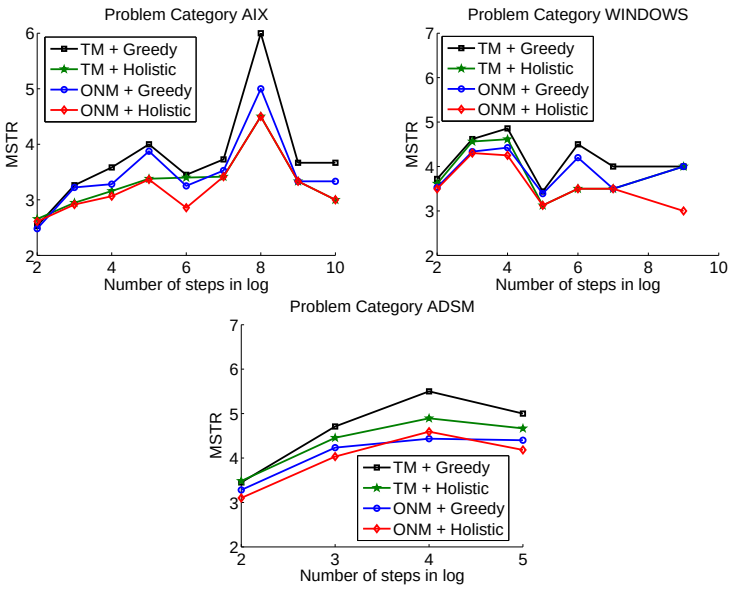


Figure 5.6: Routing efficiency: Greedy transfer vs. holistic routing.

Both of these algorithms can be executed on the TM and ONM generative models. We consider all four combinations: TM+Greedy, TM+Holistic, ONM+Greedy, and ONM+Holistic.

Figure 5.6 shows that, for each generative model, the Holistic Routing algorithm consistently outperforms the Greedy Transfer algorithm. These results validate our hypothesis that, even if an expert group is not the resolver for a problem ticket, it might have appropriate knowledge of which group can resolve the ticket. Therefore, besides the information about which groups resolve which tickets, the intermediate transfer groups can be instrumental in routing tickets to the right resolver, which is why the Holistic Routing algorithm has better performance.

The computational time for both routing algorithms to make a routing decision is less than 1 second, which is negligible compared to the time spent by the selected expert group to read and handle the ticket.

5.6.4 Robustness

For our generative models and routing algorithms to be useful in practice, they must apply to different problem categories and training samples. To confirm this, we divided the data in different ways with respect to the size of the training dataset, the time variability of the tickets, and the different problem categories, as presented in Table 5.4. For each training set, we rebuilt the models and applied the routing algorithms to measure the resulting MSTR for the corresponding testing set. Given the previous analysis, we focus on ONM and Holistic Routing.

Table 5.4: Datasets for robustness.

Training Set	Testing Set
Jan 1 - Mar 31, 2006	Apr 1 - Apr 30, 2006
Jan 1 - Apr 30, 2006	May 1 - May 31, 2006
Jan 1 - May 31, 2006	Jun 1 - Jun 30, 2006

As shown in Figure 5.7, with larger training data sets, the resulting MSTR tends to become smaller. Despite the variations in the size of the training set, our approach yields consistent performance. The problem descriptions in these ticket data sets are

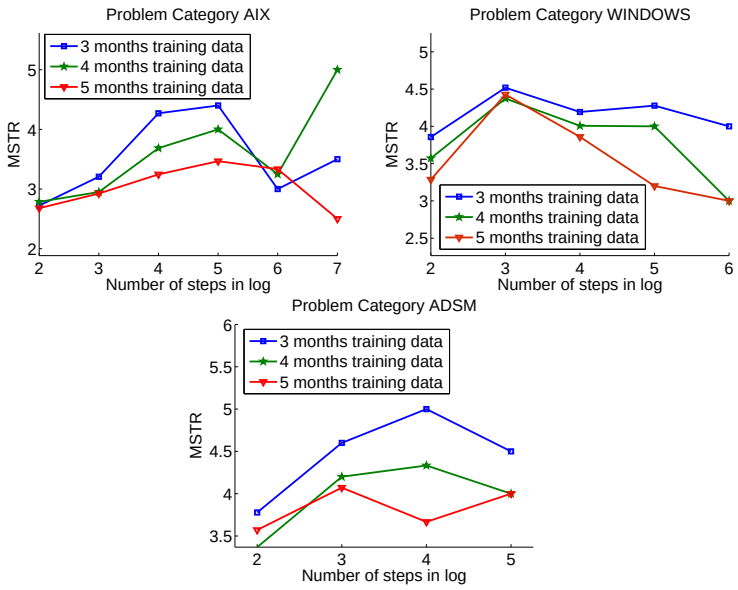


Figure 5.7: Robustness of ONM and holistic routing with variable training data.

typically short and sparse. The results demonstrate that generative modeling is particularly effective for this type of data.

5.7 Discussion

We have focused on using the model to make effective ticket routing decisions. However, the model has other significant applications, namely, expertise assessment in an expert network and ticket routing simulation for performance analysis and workforce/resource optimization. We briefly discuss these applications below.

5.7.1 Expertise Assessment

In essence, our model represents the interactions between experts in an enterprise collaborative network. By analyzing ticket transfer activities at the edges of the network, we can identify different roles of individual expert groups, *i.e.*, whether a group is more effective as a ticket resolver or a ticket transferrer. We can also analyze the expertise awareness between groups.

For instance, Figure 5.8 shows the most prominent words derived from ONM in the context of tickets transferred from group A to group B (List 1), as well as those resolved by group B itself (List 2). List 1 is related to system boot failures (bluescreen, freeze), while List 2 is related to data loading issues in hard drives. The mismatch between the two lists, indicates that either A is not well aware of B's expertise, or A thinks that B can better identify the resolvers for tickets described by words in List 1. Further analysis is needed to understand these interactions and implications. Our model can facilitate such analysis.

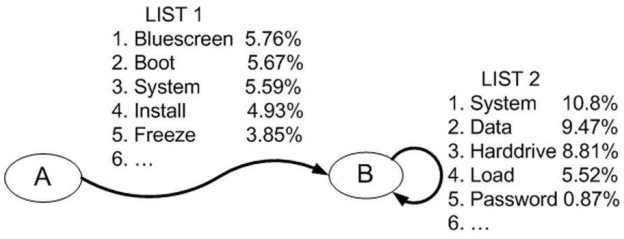


Figure 5.8: Expertise awareness example.

5.7.2 Ticket Routing Simulation

Our model can be used to simulate the routing of a given set of tickets. The simulation can help an enterprise analyze its existing ticket routing process to identify performance bottlenecks and optimize workforce/resources. Moreover, the simulation can be used to assess the "criticality" of expert groups, *e.g.*, whether the routing performance is improved or degraded, if a group is removed from the network. Such a knockout experiment is infeasible in practice, but can be conducted by simulation.

5.8 Summary

In this chapter, we have presented generative models that characterize ticket routing in a network of expert groups, using both ticket content and routing sequences. These models capture the capability of expert groups either in resolving the tickets or in transferring the tickets along a path to a resolver. The Resolution Model, introduced in this chapter, considers only ticket resolvers and builds a resolution profile for each expert group. The Transfer Model considers ticket routing sequences and establishes a locally optimized profile for each edge that represents possible ticket transfers between two groups. The Optimized Network Model (ONM) considers the end-to-end ticket routing sequence, and provides a globally optimized solution in the collaborative network. For

ONM, we presented a numerical method to approximate the optimal solution which, in general, is difficult to compute.

Our generative models can be used to make routing predictions for a new ticket and minimize the number of transfer steps before it reaches a resolver. For the generative models, we presented three routing algorithms to predict the next expert group to which to route a ticket, given its content and routing history. Experimental results show that the proposed algorithms can achieve better performance than existing ticket resolution methods.

Chapter 6

Quantitative Analysis of Task-Driven Information Flow

Collaborative networks are a special type of social network formed by members who collectively achieve specific goals, such as fixing software bugs and resolving customers' problems. In such networks, information flow among members is driven by the tasks assigned to the network, and by the expertise of its members to complete those tasks. In this chapter, we analyze real-life collaborative networks to understand their common characteristics and how information is routed in these networks. Our work shows that the topology of collaborative networks exhibits significantly different properties compared with other common complex networks. Collaborative networks have truncated power-law node degree distributions and other organizational constraints.

Furthermore, the number of steps along which information is routed follows a truncated power-law distribution. Based on these observations, we developed a network model that can generate synthetic collaborative networks subject to certain structure constraints. Moreover, we developed a routing model that emulates task-driven information routing conducted by human beings in a collaborative network. Together, these two models can be used to investigate the efficiency of information routing for different topologies of a collaborative network – a problem that is important in practice yet difficult to solve without the method proposed in this chapter.

6.1 Motivation

Social networks as a means of communication have attracted much attention from both industry and academia. The studies so far have focused predominantly on public social networks, such as Facebook, Twitter, *etc.*, which support social interactions and information exchange among users. In this chapter, we address another type of social network, *collaborative networks*, that are formed by members who collaborate with each other to achieve specific goals. Such collaborative networks often exist on the Web, such as open source software development sites, *e.g.*, Eclipse [4] and Mozilla [5] supported by Bugzilla [1], and in the private sector such as customer service centers [92].

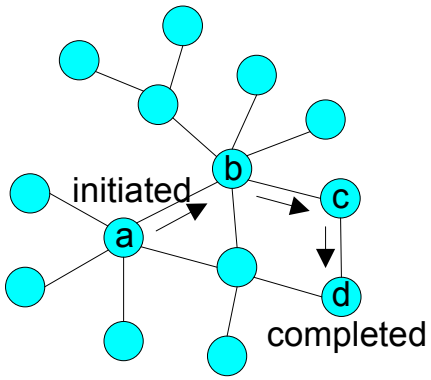


Figure 6.1: Task-driven information flow.

Information flow in collaborative networks is drastically different from that in public social networks [135]. In public social networks, information generated at a source spreads through the network with its members' forwarding activities [48, 69, 110, 113, 141]. The forwarding activities fade away as the information loses its value. In collaborative networks, information flow is driven by certain tasks. As illustrated in Figure 6.1, a task is initiated by or assigned to a source, and then routed through the network by its members until it reaches the person who can handle it. The purpose of routing is to find the right person(s) for the task, not to influence others. The routing conducted by a member is based on (1) understanding of the expertise required to complete the task, and (2) awareness of other members' expertise. For example, in fixing software bugs, the bug report is the information routed in a developer network. If a developer cannot fix the bug, he/she will attempt to forward the bug report to another developer

Table 6.1: Eclipse bug activity record.

Bug Description: NullPointerException referencing non-existing plugins.		
Who	**When**	**Description**
dean	2001-11-01 07:17:38 EST	Added component Core. Reassigned.
rodrigo	2001-11-20 18:53:40 EST	Added component UI. Reassigned.
dejan	2002-01-09 20:46:27 EST	Converted the unresolved plugin to a link. Fixed.

`https://bugs.eclipse.org/bugs/show_activity.cgi?id=325`

who he/she thinks is capable of fixing it. Table 6.1 shows one of the bug activity records extracted from the Eclipse development Web site.

The structure of collaborative networks usually evolves to facilitate the execution of tasks. It is desirable to determine whether the efficiency of the process can be improved. Efficiency can be measured by the number of steps it takes to navigate a task through a network to reach its resolver. For instance, a service provider might want to optimize the staffing structure of a call center, based on the expertise of its agents and the interactions between different agents. Such optimization might shorten the response time; however, it presents a unique challenge — one has to come up with recommendations without actually altering the network, an experiment that is not affordable in practice.

To address this challenge, we provide in this chapter an understanding of how collaborative networks are structured, and how their structures affect the efficiency of task execution. More importantly, we present a simulation-based approach with which vari-

ous hypotheses can be tested with low cost. In general, a collaborative network can be characterized in terms of two aspects: (1) structure of the network, and (2) information routing driven by the tasks. Correspondingly, we develop the following models in this study.

- *Network Model:* A model that captures the key topological characteristics of a collaborative network and that can be used to simulate networks, given specific structural constraints.
- *Routing Model:* A model that simulates human behavior in routing task-related information in a collaborative network.

Models to generate social networks have been studied extensively with consistent improvement in recent years, *e.g.*, [14, 51, 116, 132, 139]. In our problem setting, the model must be consistent with the routing algorithm so that the routing length satisfies the distribution observed in real networks. This two-body modeling requirement is new and not easy to satisfy.

To develop these two models, we investigate three real-world collaborative networks collected from different sources. The first two were extracted from the Eclipse and Netbeans software development communities. The third one comes from an IT service management system, in which service agents collaborate to solve problems reported by customers. For all three networks, we analyze their structure, as well as information flows, using the routing history (*i.e.*, bug reports or problem tickets). We observe

that the topology of collaborative networks exhibits not only the scale-free property in the node degree distribution, but also other organizational constraints. Furthermore, information routing in collaborative networks is different from routing tasks in conventional complex networks such as IP packet routing in computer networks and itinerary planning in airline networks. The number of routing steps for each task follows a heavy-tail distribution, indicating that a considerable number of tasks travel along long routes before reaching the resolvers. The three collaborative networks, collected independently from different sources, exhibit astonishingly similar characteristics, which validates the need to study them together. These observations contribute toward understanding the complicated behavior of human collaboration in these networks.

Based on our observations from real-world data, we develop a graph model to generate networks similar to real collaborative networks and a stochastic routing algorithm to simulate the human dynamics of collaboration. The models are independently validated using real-world data and simulation-based studies. We demonstrate that the proposed models can be used to answer real-world questions, such as *"How can one alter a collaborative network to achieve higher efficiency?"* To the best of our knowledge, our work is the first attempt to understand human dynamics in collaborative networks and to evaluate analytically the efficiency of real collaborative networks.

6.2 Related Work

Previous studies related to our work mainly belong to two categories: (1) Those that focus on network generation models, and (2) Those that analyze information flows in networks.

Network generation models. Generating synthetic networks that reflect statistics similar to real social networks has been of great interest to researchers in various fields. The Erdös-Rényi random network [51] is a classic random network, where any two nodes are connected according to a fixed probability. A regular lattice network is created with nodes placed on one or more dimensional lattices, *i.e.*, circle or grid, and each node is connected to its n nearest neighbors. Watts and Strogatz [139] added random rewiring to the regular lattice network such that the generated network has a small diameter as observed in a sample of the real social network [132]. Barabasi *et al.* [14] focused on the fact that many complex networks have degrees that follow a heavy-tail distribution and captured this phenomena by incrementally creating a random network, with new edges preferentially attached to already well-connected nodes.

To comply with both the small-world effect and the power-law degree distribution, Makowiec [86] and Ree [109] proposed rewiring processes in a constant-size network based on the preferential attachment principle. Serrano *et al.* [116] developed a network generation model to reproduce self-similarity and scale invariance properties observed

in real complex networks, by utilizing a hidden metric space with distance measurements. Sala *et al.* [112] studied how well the generated graphs match real social graphs extracted from Facebook.

Different from existing graph generation models, our method contributes toward understanding how links are established and how members with different expertise interact with each other in real collaborative networks. Both the expertise awareness and expertise exposure of each member are taken into consideration in our model. It not only generates a network topology with statistical characteristics similar to real-world collaborative networks, but also can be seamlessly combined with our routing model to simulate human dynamics in these networks.

Information flow analysis. The spreading of information has been extensively studied under different network settings, *e.g.*, social networks, the World Wide Web, the e-mail network, biological networks, *etc.* Examples include the spread of innovations [58, 110, 124, 134], opinions, rumors and gossip [56, 57, 87], computer/biological viruses [83, 113] and marketing [48, 69]. More recently, Wang *et al.* [135] have studied how information propagates from person to person using e-mail forwarding, and Wu *et al.* [141] analyzed the information spreading pattern on Twitter. This type of information flow aims to reach and influence more people and, hence, to achieve a large impact. Most of the work has focused on analyzing patterns of the information spreading pro-

cess. Kempe *et al.* [69] have addressed the question of how to choose a subset of nodes to initiate information spreading to maximize influence in a network.

In our work, we focus on another type of information flow: task-driven information flow, where the goal is to reach a user who can accomplish a task with a minimal number of transfer steps. Related to our problem, Milgram [93] demonstrated that short paths exist between any pair of nodes in a social network (*a.k.a.*, the small world phenomena). Kleinberg [70] investigated why decentralized navigation is efficient using a synthetic network lattice. Boguna *et al.* [20] studied the navigability of complex networks by running a greedy routing algorithm on synthetic networks generated by a model described in [116]. In the collaborative networks we studied, we observe that these networks exhibit degree distributions quite different from commonly-studied complex networks. Furthermore, the simple greedy algorithm does not provide a good approximation of information flow dynamics in collaborative networks. Thus, we developed the Stochastic Greedy Routing (SGR) model to evaluate the efficiency of task-driven information flow in such networks.

6.3 Observations

Frist, we illustrate the key characteristics of real-world collaborative networks and the information routing behavior in these networks. Our study is based on three datasets

collected from two different domains: software development (public) and IT service center (private).

The Eclipse and Netbeans[1] networks are extracted from the MSR 2011 Challenge[2], where each node represents a program developer. Both datasets contain a history of bug reports, user online interactions, and final resolutions. The Eclipse network has approximately $7,800$ developers who worked together on $272,000$ bugs. The Netbeans network contains around $156,000$ bug reports that involved $7,400$ developers. The third network, labeled "Enterprise network," is obtained from an IT service department, where each node represents a service agent. It contains around $2,000,000$ problem tickets submitted by customers. Similar to bug resolution in a programmer network, a ticket is transferred in a service agent network for resolution. The service agent network has around $19,000$ service agents. When one member in a collaboration network routes a bug report or a service ticket to another member, we construct a directed edge. Thus, the three collaborative networks are represented by directed graphs.

Although developer networks and service agent networks appear to be quite different, we were amazed by the similarity exhibited in their topologies and dynamic routing structures, indicating that commonality exists in human collaboration behaviors.

[1] Eclipse and Netbeans are Java development environments.

[2] `http://2011.msrconf.org/msr-challenge.html`

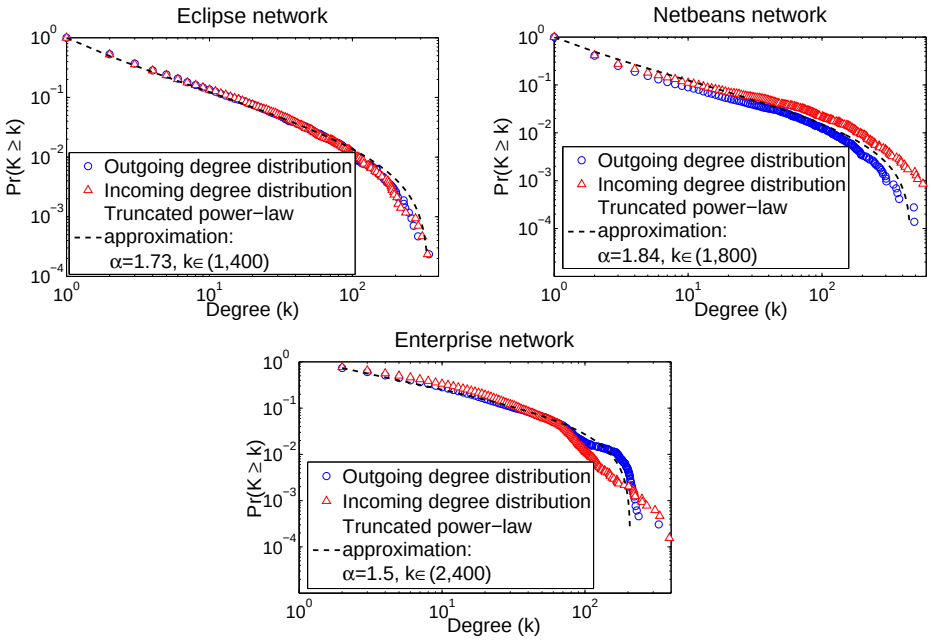


Figure 6.2: Degree distributions of collaborative networks.

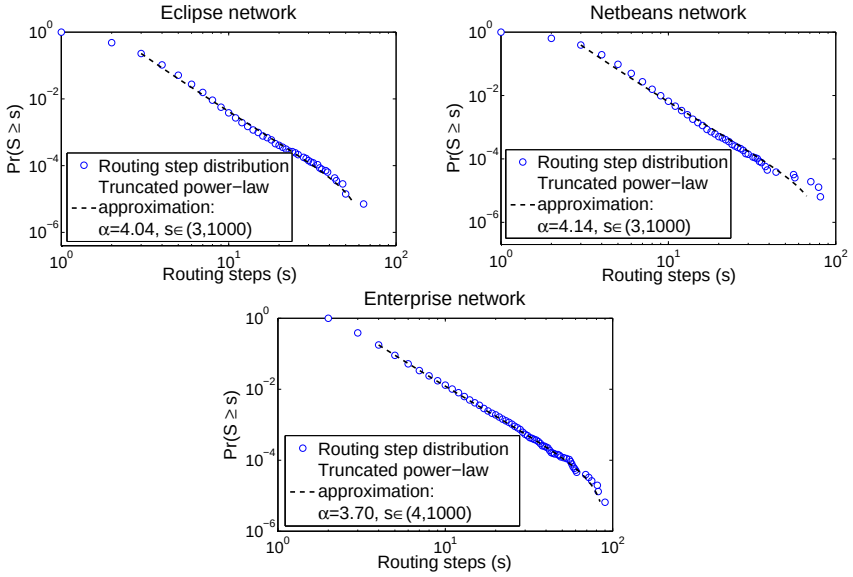


Figure 6.3: Routing steps distribution of problem solving in collaborative networks.

6.3.1 Degree Distribution

Figure 6.2 shows the incoming and outgoing degree distributions of the three collaborative networks. Different from common observations in other complex networks like the Internet, the Web, and social networks, which exhibit the scale-free property, these collaborative networks have truncated power-law node degree distributions.

We tested the power-law hypothesis on the degree distributions of the collaborative networks using a principled statistical framework proposed by Clauset *et al.* [34]. The power-law model was not accurate enough to characterize the node degree distribution in collaborative networks using the p test [34]. However, we observed that the node degree of these networks follow a truncated power-law distribution (Equation (6.1)) when the node degree k lies within a finite range. We applied a maximum likelihood approach, similar to [34], to fit the truncated power-law distributions. Inspired by [34], we further evaluated the goodness of fit using the p test based on the Kolmogorov-Smirnov statistic [107]. The truncated power-law model is a plausible fit to the node degrees because the statistical tests generate a value of p that is large enough ($p > 0.1$).

$$P(k) \propto k^{-\alpha} \text{ where } k \in (k_{min}, k_{max}) \tag{6.1}$$

The distributions in Figure 6.2 further differ from other complex networks in two aspects: (1) The power-law scaling parameter of the distribution falls in the range $\alpha \in$

$(1, 2)$, in contrast to the commonly reported range $\alpha \in (2, 4)$, and (2) The incoming degree and the outgoing degree follow roughly the same power-law distribution.

The smaller value of the power-law scaling parameter indicates that, in a collaborative network, the probability $P(k)$ decreases more slowly as k increases. This distinctive property leads to the consequent effect that the node degrees are bounded. The distribution $P(k) \propto k^{-\alpha}$, where $\alpha \in (1, 2)$, does not have a converged mean $E(k) = \sum_{k=1}^{\infty} kP(k)$. However, in reality, the degrees of the nodes do have a mean value. This mismatch implies that the degree distribution is bounded: $P(k) \propto k^{-\alpha}$, where $k \in [k_{min}, k_{max}]$. The reason for this distinctive property is that human interactions in a collaborative network have more realistic constraints than those in an ordinary social network or the Web or other complex networks. In a collaborative problem solving environment, it takes a significant amount of time for a person to establish close interactions with other persons.

6.3.2 Routing Steps

The number of routing steps to complete a task is a critical measure of efficiency in collaborative networks. Figure 6.3 depicts the routing steps distribution for the three collaborative networks that we studied. The routing steps follow a truncated power-law distribution with a very similar scaling parameter $\alpha \in (3.5, 4.5)$ in all three collaborative networks. Unlike [132], which discovered that short paths exist between any pair

of members in a collaborative network and that individual members are very adept at finding those short paths, the heavy-tail distribution for routing steps indicates that a considerable proportion of tasks travel along long sequences before reaching a resolver. We conjecture that the heavy tails in these distributions are largely due to the varying complexities of the tasks assigned to the network. Namely, when a task is fairly complex and the expertise required to complete the task is concealed in the task description, the members in a collaborative network have to try different directions before the task is routed to the correct destination.

6.3.3 Clustering Coefficient

The clustering coefficient measures how closely the neighbors of a node are connected, by calculating the number of connected triplets in a network that are closed triplets. In an undirected graph, the *local clustering coefficient of node* i is defined as follows:

$$c_i = 2t_i/(k_i(k_i - 1)), \tag{6.2}$$

where k_i is the degree of node i and t_i is the number of edges between i's neighbors. The *global clustering coefficient* is the average of the local clustering coefficients over all nodes in the network. To calculate the clustering coefficients in collaborative networks, we ignore the directions of edges. The clustering coefficients of the three networks studied are shown in Table 6.2. Note that the members in the enterprise network

interact more closely in local teams than those in the public developer networks. This observation is not surprising, because enterprise networks typically have more rigid hierarchical structures.

Table 6.2: Clustering coefficients.

Eclipse network	Netbeans network	Enterprise network
0.19	0.21	0.35

6.4 Network Model

As it is expensive, if not impossible, to alter real-world collaborative networks for hypothesis testing, *e.g.*, changing their structure for better performance, it is important to develop a network model for which various hypotheses can be examined with low cost. The network model must take into account the structural constraints discussed in Section 6.3, *i.e.*, the degree distribution and the clustering coefficient. The network model must be consistent with the routing algorithm so that the routing steps satisfy the power-law distribution. This coupled modeling requirement is new and not easy to satisfy, especially when there is no way to generate simulated bugs or problem tickets. In this section, we present a network model for collaborative networks. In Section 6.5, we discuss the corresponding routing model.

In the network model, first we determine the location of each node in the network, which corresponds to a member's expertise. Next, we add edges between pairs of nodes, representing the interactions among members. Then, we tune the network model to capture the interactions among nodes with similar expertise, using the clustering coefficient.

6.4.1 Node Generation

To model a collaborative network with N nodes, first we randomly assign coordinates (x_i, y_i), where $x_i, y_i \in [0, L]$, to each node $i \in \{1, 2, ..., N\}$ in a two-dimensional rectangular area, simulating the *expertise space*.

The coordinates of a node represent the specific expertise of a network member. Thus, two members with similar expertise tend to be close to each other. Different collaborative networks can have different expertise distributions. To make the model general, we take a simplified representation of the expertise space and the node distribution. We assume that the nodes are uniformly distributed in the rectangular expertise space. That is, different expertise areas have the same representation in the generated nodes. However, this simplified representation in the general model can be substituted with specific network configurations of real collaborative networks. The routing algorithm that we introduce in Section 6.5 applies to these specific network configurations,

as demonstrated by a direct embedding of real-world collaborative networks in two-dimensional space in Section 6.6.2.

Because the expertise space is limited to a rectangular area, nodes located at the center of the area are likely to have more neighbors than those located close to the boundary. To model the relationship between different expertise areas, we apply a periodic boundary condition that replicates the expertise area around the areas of interest, as shown in Figure 6.4. The distance $d_{i,j}$ between any pair of nodes i and j is defined as the minimum Euclidean distance between copies of i and j. In this way, each node is given a roughly equal-sized neighborhood.

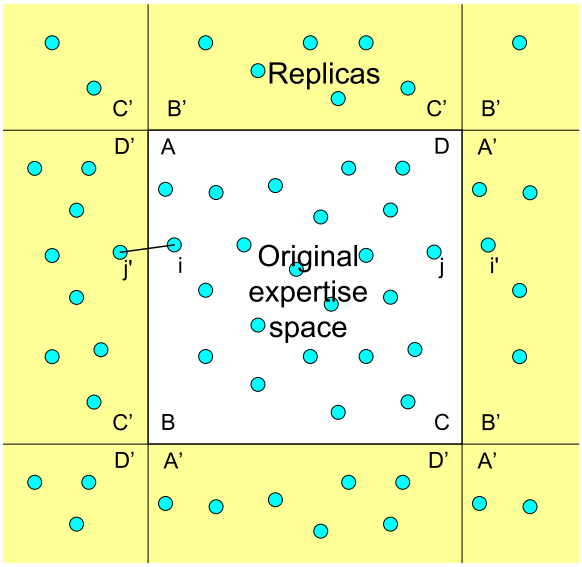


Figure 6.4: Periodic boundary condition in an expertise space.

6.4.2 Edge Generation

In a collaborative network, an edge from member i to member j exists when member i can transfer a task to member j. The establishment of an edge requires member j to expose his/her expertise sufficiently to the others, and member i to be aware of j's exposed expertise. Only with these conditions will member i transfer a task to member j, when i believes j has the right expertise to complete the task. Based on this intuition, we define two metrics for each node that guide edge generation in our network model: an expertise awareness coefficient and an expertise exposure coefficient.

For each node i in the network, its *expertise awareness coefficient* a_i and its *expertise exposure coefficient* e_i are random variables that follow probability distributions $a_i \sim P(a)$ and $e_i \sim P(e)$, respectively. An edge from node i to node j exists if and only if their awareness and exposure coefficients are large enough to cover the distance between i and j, *i.e.*, $a_i \times e_j > d_{i,j}$.

To simulate a network with certain incoming and outgoing node degree distributions, we need to tune the probabilities $P(a)$ and $P(e)$. Given that the incoming and outgoing degree distributions are identical in all collaborative networks studied in Section 6.3, we assume that the awareness and exposure coefficients have the same distribution. Therefore, if we know the form of one distribution, we can solve for the other symmetrically.

First, we assume that the distribution of the exposure coefficient is $P(e) = \beta \times e^{-\gamma}$, where $e \in [e_{min}, e_{max}]$. For any node i, when the awareness coefficient is chosen to be a_i, we calculate the probability that $edge_{i,j}$ exists, given the distance between node i and node j, as follows:

$$P(edge_{i,j}) = \begin{cases} 1 & d_{i,j} \leq a_i \times e_{min} \\ P(e_j > d_{i,j}/a_i) & e_{min} < d_{i,j}/a_i \leq e_{max} \\ 0 & \text{otherwise}. \end{cases} \tag{6.3}$$

Note that, when the nodes are uniformly distributed over the rectangular area, the node density ρ is a constant. Therefore, given the awareness coefficient a_i, we can estimate the outgoing degree $\widehat{k^i_{out}}$ of node i as follows:

$$\begin{aligned} \widehat{k^i_{out}} &= \int_{d_0=0}^{\inf} \rho \times 2\pi d_0 P(edge_{i,j}) d(d_0) \\ &= \rho \times \pi (a_i e_{min})^2 \\ &+ \int_{e_0=e_{min}}^{e_{max}} \rho \times 2\pi a_i^2 e_0 P(e_j > e_0) d(e_0) \end{aligned} \tag{6.4}$$

Thus, $\widehat{k^i_{out}}$ can be expressed as ba_i^2, where b is a constant. To guarantee that the outgoing degrees of the nodes follow the desired power-law distribution $P(k_{out}) = c \times (k_{out})^{-\alpha}$, where $k_{out} \in [k_{min}, k_{max}]$, the awareness coefficient must have the following probabil-

ity distribution:

$$\begin{aligned} P(a) &= \lim_{\Delta a \to 0} \frac{P(a \leq a_i \leq a + \Delta a)}{\Delta a} \\ &= \lim_{\Delta a \to 0} \frac{P(ba^2 \leq k_{out} \leq b(a + \Delta a)^2)}{\Delta a} \\ &= \lim_{\Delta a \to 0} \frac{cb^{-\alpha+1}((a + \Delta a)^{-2\alpha+2} - a^{-2\alpha+2})}{(-\alpha + 1)\Delta a} \\ &= 2cb^{-\alpha+1}a^{-2\alpha+1} \end{aligned} \tag{6.5}$$

That is, the awareness coefficient also follows a power-law distribution with coefficient $-2\alpha + 1$. According to the symmetric assumption between the exposure and awareness coefficients, we conclude that the exposure coefficient follows the same power-law distribution with coefficient $-2\alpha + 1$.

The range of the two coefficients should be set such that the degrees are restricted to the desired range. In Equation (6.5), a node with minimum awareness coefficient a_{min} is expected to have the minimum outgoing degree k_{min}; a node with the maximum awareness coefficient a_{max} is expected to have the maximum outgoing degree k_{max}. Thus,

$$a_{min} = e_{min} = \sqrt{\frac{k_{min}}{\rho \times \pi \langle e^2 \rangle}} \tag{6.6}$$

$$a_{max} = e_{max} = \sqrt{\frac{k_{max}}{\rho \times 2\pi \langle e^2 \rangle}} \tag{6.7}$$

where $\langle e^2 \rangle$ is the expected value of the squared exposure coefficient.

Given the power-law coefficient and the range of the awareness and exposure coefficients, their distributions are properly normalized. Using the normalized distribu-

tions, we then generate edges in the network model with the probability given in Equation (6.3), so that the incoming and outgoing degrees of the nodes follow the desired power-law distribution.

6.4.3 Modeling Expertise Domains

In a real collaborative network, the clustering coefficient indicates how closely its members work together in expertise domains. A higher clustering coefficient means that there are more collaborations between members within local expertise domains. To model collaborative networks with different expertise domains, the network model needs to form local teams that represent specific expertise domains required for certain tasks. Intuitively, members with expertise in similar domains tend to interact more with each other when working on these tasks. Consequently, the network should have more links between nodes inside the same expertise domain, and fewer links between nodes in different or unrelated expertise domains. Even though it is less likely for members from unrelated expertise domains to interact with each other, such connections still exist in real collaborative networks and a member who reaches beyond his/her own expertise domain is usually one with high connectivity.

To model this behavior, first we associate nodes in the network with different domains. Then, for any two different domains, as illustrated in Figure 6.5, we break inter-domain links and replace them with intra-domain links, using an *edge swapping*

process inspired by [131]. At each step of the edge swapping process, we choose a pair of inter-domain edges, pointing in opposite directions, and assign a swapping probability according to the degrees of the nodes to which they connect. If the connected nodes have high incoming or outgoing degrees, we swap the edges with low probabilities; otherwise, we swap the edges with high probabilities. Specifically, we deal with two inter-domain edges $u_1 \rightarrow v_2$ and $u_2 \rightarrow v_1$, with users u_1 and v_1 from one domain, and users u_2 and v_2 from the other domain. We assign the edge swapping probability $p = 1 - \max(k_{out}^{u_1}, k_{in}^{v_2}, k_{out}^{u_2}, k_{in}^{v_1})/k_{max}$, where k_{max} is the maximum outgoing/incoming degree among all of the nodes in the network. With probability p, we break the edges $u_1 \rightarrow v_2$ and $u_2 \rightarrow v_1$, and connect the edges $u_1 \rightarrow v_1$ and $u_2 \rightarrow v_2$. We repeat the edge swapping process until a certain fraction of the inter-domain edges have been swapped to intra-domain edges. The edge swapping process prefers to break inter-domain connections from nodes with low degree and to maintain the edges connecting well-connected nodes. Thus, we avoid isolated subgraphs during the edge swapping process, and the resulting network matches real collaborative networks.

With these adjustments, the node degree distribution still fits the desired power-law distribution achieved in Section 6.4.2. The more edge swapping one performs, the higher the local connectivity the network has within each domain. The result is higher clustering coefficients.

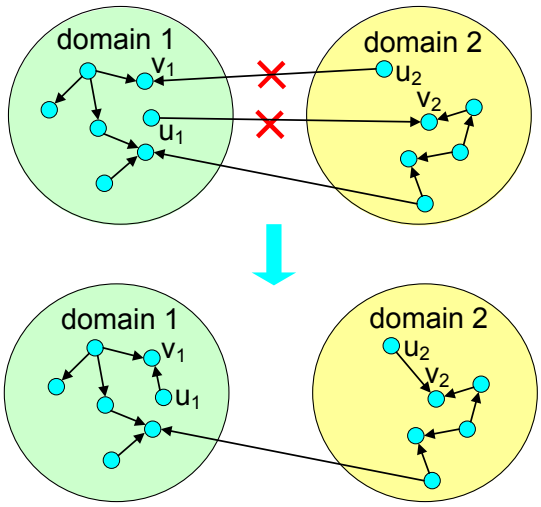


Figure 6.5: Inter-domains edge swapping.

For a network with a fixed number of nodes, when we increase the number of domains, the average size of a domain decreases. Consequently, the edge density inside each domain increases, and the clustering coefficient increases. After forming local domains, the generated network has the desired incoming/outgoing degree distribution, and approximates the clustering coefficients of real collaborative networks.

6.5 Routing Model

The task-driven routing model must capture the behavior of humans in routing tasks to appropriate experts. Although the small-world phenomena [70, 132] is also observed in collaborative networks, *i.e.*, a relatively short path typically exists between any pair of nodes in the three studied networks, there is no guarantee that the members in a

collaborative network are able to route tasks through these short paths. In fact, our analysis in Section 6.3 has shown that the number of routing steps for a task typically follows a truncated power-law or heavy-tail distribution. Thus, many tasks are routed along a long sequence of steps before they reach the resolvers. A commonly used routing algorithm in the Internet [20] and in social networks [70] is greedy routing. The greedy routing algorithm assumes that there exists a distance between any pair of nodes. A node has access to the distance from itself and its neighbors to the destination node. If there exists one or more neighbors closer to the destination than the current node, it routes the task (packet) to the neighbor node closest to the destination. Otherwise, the node does not have a better routing choice than itself. In this case, the task (packet) fails to reach the destination.

Unfortunately the greedy algorithm is not adequate for simulating human task routing behavior. First of all, the greedy algorithm is deterministic, and often fails to navigate a task if the current task holder does not have a better choice. In the three networks we studied, the greedy algorithm fails to route approximately 14% of the tasks. In contrast, most of these tasks were successfully routed by humans. Secondly, the routing steps generated by the greedy algorithm follow an exponential distribution. As the number of routing steps increases, the probability drops much more quickly than the power-law distribution. In real decision-making scenarios, a human tends to make different routing decisions when the situations (*e.g.*, availability of neighbors, priority

of tasks, *etc.*) are changing, even given similar tasks. Therefore, a different model is needed to incorporate the stochastic process of task routing, which is essential for modeling human behavior.

In a collaborative network, people make their task routing decisions based on many factors, including the availability of neighbors, priority of tasks, *etc.* A member of the network often makes a decision based on available local information, rather than on global information that can be used to optimize the end-to-end routing efficiency. Thus, the same task can be transferred by a member along various sub-optimal paths in different situations. Therefore, information routing in collaborative networks is a stochastic process, rather than a deterministic process.

We construct a Stochastic Greedy Routing (SGR) model based on the following intuition. When a member in a collaborative network cannot finish a task, he/she tends to transfer the task to a neighbor who has expertise closer to that of the resolver, similar to a greedy approach. The member also evaluates the connectivity of his/her neighbors, and tends to select a neighbor who has more outgoing connections, assuming that a better-connected neighbor is more likely to route the task along a shorter path to the resolver.

The SGR model assumes that each node relies on only local information to route tasks to one of its neighbors, following a stochastic process. Considering a task that is initially assigned to node u and has a resolver v, the SGR model guides each node to

navigate the task through the network, from the initiator u to the resolver v. At each step, when a non-resolver node holds a task, it evaluates the candidate set $\mathcal{C}$, consisting of its neighbors that have not yet been visited, and transfers the task to one of them. In some rare cases, the candidate set becomes empty and all the neighbors are marked as unvisited. As mentioned above, the task should be transferred to a node with expertise similar to that of the resolver and with a higher outgoing degree. Therefore, for each candidate i, we define the following utility function:

$$F(i) = d(i, v)^{-1} \times k_{out}^{i} \tag{6.8}$$

Note that this utility function is inversely proportional to $d(i, v)$, the geometric distance between a candidate and the resolver in our network model, which represents the similarity in their expertise. The holder of a task transfers the task to one of the candidates $i \in \mathcal{C}$ with a probability proportional to i's utility, *i.e.*, $P(i) = F(i)/\sum_{j\in\mathcal{C}} F(j)$. This process is repeated until the task reaches the resolver. To perform routing, the SGR method does not rely on the nature of the tasks; thus, it avoids the issue of generating synthetic tasks. Instead, it needs only a pair of initiators and resolvers to simulate a task, which significantly simplifies the model.

The SGR model assumes that each node can evaluate the geometric distance between its neighbors and the resolver, without knowing the topology of the network. This assumption is very close to real-life situations. In our network model, geometric distances between nodes represent similarity in the expertise of the nodes. Although

the current holder of a task does not know the shortest path to the resolver, he/she has knowledge of what expertise is required to complete the task, as well as the expertise of the neighbors. Hence, he/she can make a judgement as to which one of the neighbors is a better fit toward completing the task.

6.6 Evaluations

In this section, we evaluate the network model and the routing model presented earlier. First, we evaluate the network model by comparing the key characteristics of the synthetic networks generated from this model and those of real collaborative networks. Then, we evaluate the effectiveness of the routing model by applying it to synthetic networks, as well as to real collaborative networks. Finally, we present a case study that demonstrates how to combine the two models to optimize the structure of collaborative networks.

6.6.1 Evaluating the Network Model

To evaluate the network model, first we use it to generate synthetic networks that have similar incoming and outgoing degree distributions as observed in real collaborative networks. For example, the Eclipse network has a power-law degree distribution $P(k) \sim k^{-1.73}$, where $k \in [1, 400]$. For each node in the synthetic network, we

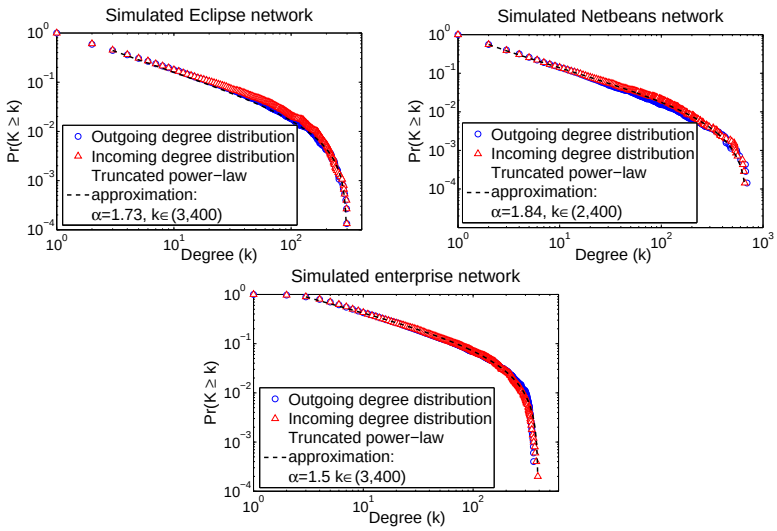


Figure 6.6: Degree distribution of simulated networks.

randomly select its awareness coefficient and exposure coefficient following the same power-law distribution $P(a) \sim a^{-2.92}$, $P(e) \sim e^{-2.92}$, where $a, e \in [0.047, 0.94]$, calculated from Eqs.(6.5)-(6.7). Similarly, for simulating the Netbeans network, we calculate the probability distribution for the awareness coefficient and the exposure coefficient as $P(a) \sim a^{-3.36}$, $P(e) \sim e^{-3.36}$, where $a, e \in [0.05, 1.6]$. For the Enterprise network, the awareness coefficient and the exposure coefficient follow the probability distribution $P(a) \sim a^{-2}$, $P(e) \sim e^{-2}$, where $a, e \in [0.036, 0.72]$. Figure 6.6 shows that the degree distributions in synthetic networks are very close to those observed in the three real collaborative networks (*i.e.*, Eclipse, Netbeans, and Enterprise), shown in Figure 6.2.

Besides degree distributions, we need to evaluate the capability of our network model in generating networks with various clustering coefficients. Recall that the clustering coefficient of a collaborative network reflects the existence of expertise domains

and the difference between inter- and intra-domain links. Here, we study the same three synthetic networks as shown in Figure 6.6. In each network, we divide the nodes into K expertise domains and then vary the clustering coefficient through edge swapping. As we vary the value of K, we expect different clustering coefficients. We select the clustering coefficient closest to that of the real network as an approximation.

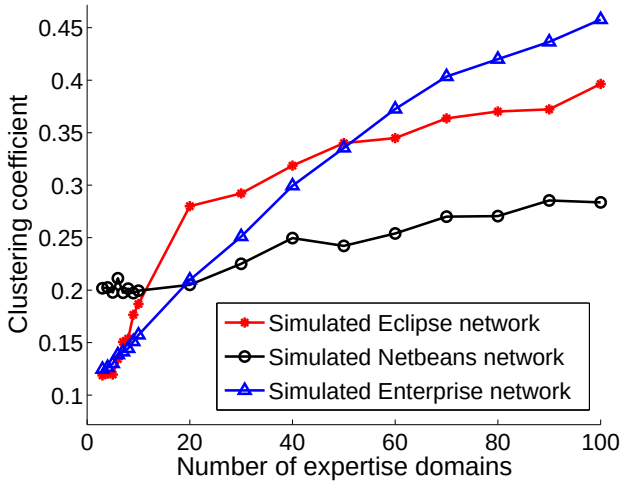


Figure 6.7: Tuning the clustering coefficient.

Figure 6.7 shows the variations of clustering coefficients of the synthetic networks for different values of K. By increasing the value of K, we observe that the clustering coefficient increases. Hence, by choosing a proper value of K, our network model can approximate a real collaborative network in both the degree distribution and the clustering coefficient. In our study, the Eclipse network is best approximated with 9 domains. The Netbeans network is best approximated with 10 domains. The Enterprise network is best approximated with about 60 expertise domains. We do not have information

regarding the number of expertise domains in the Eclipse network or the Netbeans network. However, we were able to confirm that, indeed, the Enterprise network had about 60 expertise domains.

It can also be observed in Figure 6.7 that, when the network has a power-law degree distribution with a large scaling parameter (*e.g.*, the Netbeans network), the clustering coefficient curve tends to be flatter than for the other networks. The reason is that, in such a network, most nodes have very few connections. Correspondingly, in our network model, most nodes have small awareness and exposure coefficients. Hence, the network is not very heavily connected. After dividing the nodes into different domains, the edge swapping process can affect only a small number of cross-domain edges; otherwise, the network will become disconnected. As a result, increasing the value of K has a small effect on changing the network clustering coefficient.

6.6.2 Evaluating the Routing Model

To evaluate the routing model, first we ran task routing simulations guided by the SGR model on a synthetic network generated by the network model and we demonstrated that the result is consistent with real observations.

We generated a collaborative network with $5,000$ nodes to simulate the Enterprise network. The incoming/outgoing degree of the generated network follows a power-law distribution $P(k) \sim k^{-1.5}$, where $k \in [1, 400]$. We divided the network into 60

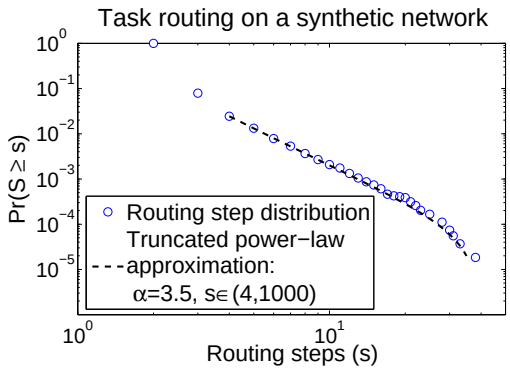


Figure 6.8: Routing steps distribution in a simulated Enterprise network.

expertise domains, which leads to a clustering coefficient of 0.37. We generated a set of $100,000$ tasks by choosing the initiators and the resolvers. For each task we choose an initiator node with probability proportional to its outgoing degree, and a resolver node with probability proportional to its incoming degree. As shown in Figure 6.8, the resulting routing steps distribution again follows a power-law distribution. Its power law factor $\alpha = 3.5$ is very close to the real value $\alpha = 3.53$, which indicates that we can seamlessly combine the two models without inconsistency.

We further ran task routing directly on a two-dimensional representation of real collaborative networks to illustrate that it can stand alone for routing simulations. To map a real collaborative network into a two-dimensional space, while preserving the local neighborhood relationships, we adopt the spectral embedding method [111]. The embedding process guarantees that, if two nodes are close to each other in the original space, they are likely to be close to each other in the embedding space. The closeness

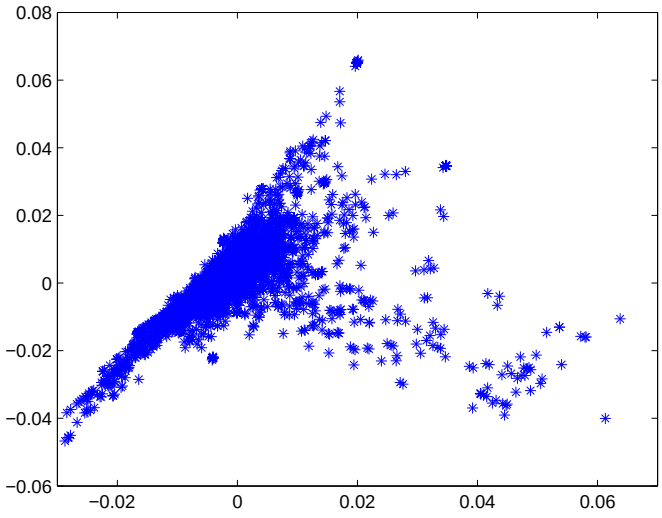


Figure 6.9: Two-dimensional spectral embedding of the Netbeans network.

between two nodes can be defined by the number of task transfers between them: the more frequent the task transfer, the closer are the two nodes.

Figure 6.9 shows the two-dimensional embedding of the Netbeans network, using the spectral embedding method. The embedding can be regarded as a non-uniform distribution of nodes in an expertise space. Given the embedding, we assign a two-dimensional coordinate to each node in the network, which enables distance measurement between pairs of nodes, a required input to the SGR model. Because we know the initiator and the resolver of each task, we then apply the SGR model to simulate the full path of each task routing. The routing steps distributions of the simulation for all three networks are shown in Figure 6.10. The simulated results match the observations well, as is evident in Figure 6.3.

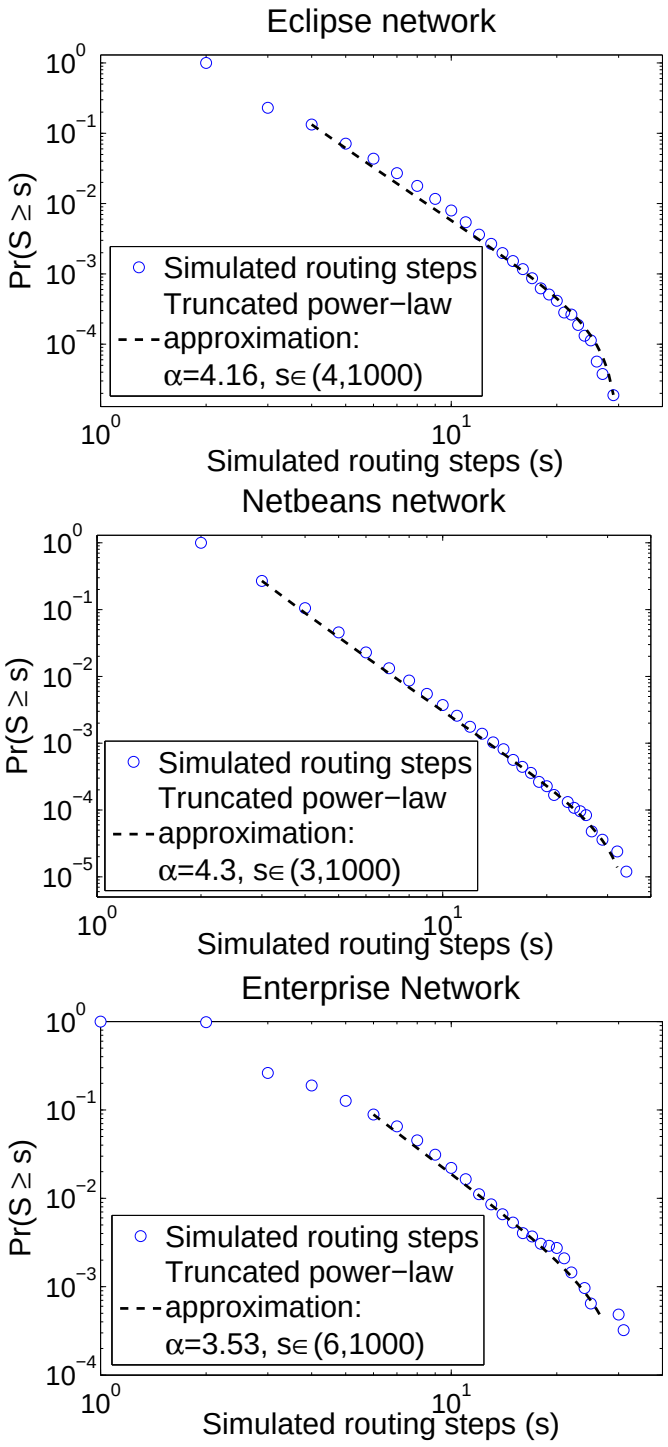


Figure 6.10: Simulated routing steps distributions.

6.6.3 Combining the Two Models: A Case Study

Our network model simulates the static connectivity of a collaborative network, whereas our SGR model simulates the dynamic user behavior in information routing in a collaborative network. Combined together, these two models provide an unprecedented means of studying real collaborative networks. It is particularly important to study how the structure changes of a collaborative network can affect the efficiency of task execution, without changing the real-world network structure. This case study demonstrates the simulation method for our network and information routing models.

The environment studied is the problem management organization of a large IT service provider. To accommodate the evolving workload and human resources, the IT service provider needs to restructure the service agent network to deliver the optimal performance in resolving the problems reported by its clients. Currently, these restructuring decisions are made manually by experienced managers or consultants, without quantitative analysis as to how the resulting network will perform after the restructuring of the service agent network.

Our models can be used to provide analytical insights to the decision makers. First, one can use our network model to generate new network topologies with different structural constraints that need to be imposed in practice. Then, given a set of tasks, the efficiency of different networks can be evaluated through the task routing simulation guided by the SGR model. Here, we assume that a collaborative network of $5,000$

service agents needs to be restructured. These service agents are divided into K pools (expertise domains) based on their expertise. A simple question to ask is: "How does one select the optimal number K of pools, to provide the best efficiency in task execution?" Intuitively, a smaller value of K indicates that the service agents are more generalized in their domain expertise, whereas a larger value of K suggests that the service agents are more specialized in their domain expertise. Furthermore, with more domains, a task is less likely to be assigned initially to the correct agent pool, which might lead to longer routing paths, because intra-domain routing is more likely to occur than inter-domain routing.

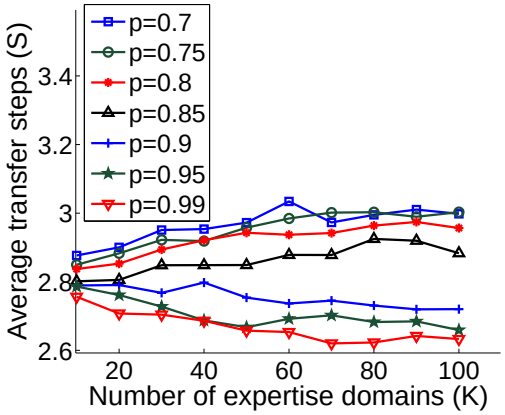


Figure 6.11: Evaluating the network structures.

For our analysis, we generate 10 collaborative networks, with 10 to 100 domains. In each network configuration, we simulate the routing of the same set of $100,000$ tasks. The probability p of correctly assigning a task to the correct domain is also taken into account in the simulation. For each task, first we select the resolver node with

probability proportional to its incoming degree. Then, with probability p, the initiator of the task is selected from within the same domain as the resolver; otherwise, the initiator is selected from outside the resolver's domain. We vary the "correct assignment probability" p from 0.7 to 0.99. For each value of p, we route the entire set of tasks in the 10 networks. The results of all simulations are shown in Figure 6.11. The y-axis shows the average number of transfer steps to the resolver for the entire set of tasks. Each curve shows the routing simulation results for a particular choice of p. Obviously, a lower average number of steps indicates a higher routing efficiency, because it usually takes less time when the tasks are routed to the resolver in fewer steps. As shown in the figure, when more tasks are initially assigned to the correct domain, increasing the number of domains leads to better performance. When fewer tasks are initially assigned to the correct domain, a smaller number of domains is more favorable.

Achieving a certain value of p, given various numbers of agent pools, has different implications in terms of training the initial assigner of the task. For the same p, the training cost typically increases as the number of service agent pools increases, because the assigner must have stronger knowledge in matching the task with the correct expertise domain. Configuring the collaborative network into different numbers of expertise domains also has implications on the training cost for the service agents. Given these implications, the decision maker can use our simulations to select the optimal number of service agent pools that suits the enterprise's budget or other constraints.

6.7 Summary

This chapter examined a special type of social networks – collaborative networks. Detailed observations of three real-world collaborative networks were presented along with the static network topology and dynamic information routing for each network. The collaborative networks exhibit not only the truncated power-law node degree distributions but also organizational constraints. Information routing in collaborative networks is different from routing in conventional complex networks, such as computer networks and airline networks, because of the random factors in human decision making. The routing steps also follow a truncated power-law distribution, which implies that a considerable number of tasks travel along long sequences of steps before they are completed. Our results and observations for several independent sources are mutually consistent, and can be generalized to other real-world collaborative networks. They help in understanding the complicated behavior in human collaboration.

Based on real-world data, we developed a graph model to generate networks similar to real collaborative networks, and a stochastic routing algorithm to simulate the human dynamics of collaboration. The models are independently validated using real-world data. We demonstrated that the two models can be used to answer real-world questions, such as: *"How can one design a collaborative network to achieve higher efficiency?"* To the best of our knowledge, our work is the first attempt to understand

human dynamics in collaborative networks and to estimate analytically the efficiency of real collaborative networks.

Chapter 7

Modeling Networked Document Sets

This chapter presents the novel Latent Association Analysis (LAA) framework, a generative model that analyzes the topics within two document sets simultaneously, as well as the correlations between the two topic structures, by considering the semantic associations among document pairs. LAA defines a correlation factor that represents the connection between two documents, and considers the topic proportion of paired documents based on this correlation factor. Words in the documents are assumed to be randomly generated by particular topic assignments and topic-to-word probability distributions.

The chapter also presents a nove ranking algorithm, based on LAA, that can be used to retrieve target documents that are potentially associated with a given source document. The ranking algorithm uses the latent correlation factor in LAA to rank

target documents by the strength of their semantic associations with the source document. We evaluate the algorithm with real datasets, specifically, the change-problem and the problem-solution paired document sets collected from an operational IT service environment.

7.1 Motivation

The vast number of documents generated in business and society presents both challenges and opportunities for data mining research. One of the common, yet relatively unexplored, types of documents are documents that appear in pairs. Examples of such document pairs include questions and answers, changes to IT systems and consequent problems, disease symptoms and diagnoses, *etc*. Such document pairs can be used to build valuable knowledge bases that help improve business decisions or generate more effective recommendations.

Table 7.1: Sample change and problem pairs.

Change (Source)	**Problem (Target)**
Set the schedule of weekly out of region backup on CARS: 3am on Sundays.	The backup is running for a long time, which is impacting the start of daytime BMP processing.
Replication of new data is loaded for all customer centers.	Server outage: User can ping the server but failed to access the database.
Back up authentication server.	User reported can access E-Pricer without inputting password.

Table 7.1 shows an example of document pairs that contain changes to IT systems (source documents) and the resulting problems (target documents). Given such document pairs, we seek to address two fundamental problems:

1. What is the underlying principle that makes the connection between a pair of documents? (*Modeling*)

2. Given a source document, how do we use this principle to rank the target documents based on how strongly they are related to the source document? (*Ranking*)

The solutions to the *Modeling* and *Ranking* problems can help us understand the semantic connection (*i.e.*, *latent association*) between paired documents and provide tremendous value in real-world applications. For instance, in the IT service industry, changes are frequently made to an operational IT environment. It is extremely valuable to enable service consultants to evaluate the potential problems caused by a proposed change, so that they can make plans accordingly. Another example is in IT problem management, where the service agents often need to search through a repository of solution documents to find the one that solves a reported problem. Both applications call for a model that captures not only the individual semantic information of two documents, but also the connections between the documents.

The modeling and the ranking problems present great challenges that cannot be readily addressed using existing approaches. For instance, topic models, such as CTM [18],

LDA [19] and PLSI [62], are designed to model only single document sets. In our problem, we need not only to model individual documents correctly, but also to capture the connection between the documents accurately. Furthermore, the existence of one-to-many or many-to-one mappings in a bipartite graph suggests possibly different interpretations of the topics of a document. For example, a question might refer to different topics if the answers emphasize different aspects of the question. What we need is a model that puts a document in the context of a document pair and allows its topic proportion to be interpreted differently in different contexts. None of the existing topic models supports flexible topic proportions in the same document. The ranking problem is also non-trivial. Given a source document, the number of potentially related target documents can be huge. The model needs to be able to identify the correct target document from a large number of candidate documents accurately.

In this chapter, we introduce the Latent Association Analysis (LAA) framework to address these challenges. The LAA framework models the topic structures and their correlations together. In the LAA model, each document pair is considered as a randomly drawn correlation factor that initiates the connection between the two documents. The topic proportions of the two documents are drawn conditionally depending on the correlation factor. Each word in the documents is assumed to be generated based on a topic assignment and the topic-to-word probability distribution.

For LAA, we adopt concepts from two well-known models, the Correlated Topic Model (CTM) [18] and Canonical Correlation Analysis (CCA) [10]. We then develop a novel ranking method to retrieve target documents based on their latent associations with the given source document. We evaluate this method using the change-problem and the problem-solution paired document sets collected from a real IT service environment. Experimental results show that the LAA-based algorithm significantly outperforms existing algorithms, which confirms that LAA successfully captures the semantic-level connections among document pairs.

7.2 Related Work

Topic models have been extensively studied and have become a powerful tool to explore the semantic content of large-scale document corpora. Most topic models deal with a single document corpus. LSI [38] uses SVD to approximate high-dimensional document-to-word co-occurrence matrix using a lower-dimensional document-to-topic co-occurrence matrix and a topic-to-word co-occurrence matrix. PLSI [62] introduces a probabilistic explanation of LSI. Both LSI and PLSI are not naturally generalizable to new documents. To overcome this problem, Blei *et al.* proposed LDA [19], in which the topic proportions of documents are randomly drawn from a Dirichlet distribution. The Dirichlet prior is used to guide the generation of topic proportions for new documents.

The CTM method [18] introduces a covariance matrix over the topic proportions and allows the topics to be correlated with each other. IFTM [108] combines CTM with PCA [119] to allow the exploration of a very large number of topics.

Besides the text information in a document corpus, a number of topic models consider additional structural information. Steyvers *et al.* [123] use the authorship graph between authors and articles to explore the author-to-topic relationships. Nallapati *et al.* [96] consider the citation graph for a document set to perform link predictions. Zhou *et al.* [161] study Web pages and tag graphs to explore user interests. Mei *et al.* [91] propose topic models with network regularization. Different from these models, our LAA model focuses on document-to-document associations, and explores topics of the two document sets simultaneously; therefore, it is better suited for ranking document pairs.

Researchers have studied topic structures of cross-lingual corpora. Zhao *et al.* [155, 156] explored probabilistic word alignments across languages using an aligned bilingual document pairs, *i.e.*, the same set of articles written in two different languages. Mimno *et al.* [94] studied the shared topic structure of an aligned document corpora over possibly many languages. Jagaralamudi *et al.* [65], assuming a dictionary exists between words in two languages, analyzed a single topic structure over a bilingual unaligned document sets. MuTo [21] also utilizes the word matchings in a dictionary to analyze the topics as distributions over the word pairs.

PTM [153] analyzes the topic structures of two linked document sets simultaneously. However, the topic structure of the target document set is assumed to be conditionally dependent on that of the source document set. In contrast, in LAA, both topic structures are drawn based on the same correlation factor simultaneously.

Other than topic models, our work is also related to link prediction and question answering [37, 54]. Several researchers [126, 129] have studied the citation graph or the hyperlink graph to predict links between documents within a single document set. Xue *et al.* [143] modeled the probabilistic mappings at the word level to facilitate question answering tasks. Although we evaluated LAA using a task similar to document retrieval, LAA can also be used for question answering, in which question understanding plays a key role for performance improvement.

7.3 Problem Formulation

The problem we address involves a source document set $\mathcal{D}_s$ and a target document set $\mathcal{D}_t$. Each source document $d_s \in \mathcal{D}_s$ is paired with at least one target document $d_t \in \mathcal{D}_t$, and vice versa. The pairing between the source document set and the target document set can be represented by a bipartite graph $\mathcal{G}$, with its two sets of vertices being the source document set and the target document set, and its set of edges corresponding to the source and target document pairs. Specifically,

- $\mathcal{G} = \{\mathcal{D}_s \bigcup \mathcal{D}_t, \mathcal{E}\}$ is a bipartite graph with its vertices defined by a set $\mathcal{D}_s$ of source documents, a set $\mathcal{D}_t$ of target documents, and a set $\mathcal{E}$ of edges between documents in $\mathcal{D}_s$ and documents in $\mathcal{D}_t$.

- Each edge $e_i = (d_{is}, d_{it})$ represents a document pair, where $d_{is} \in \mathcal{D}_s$, $d_{it} \in \mathcal{D}_t$ and $e_i \in \mathcal{E}$.

- The vocabulary set of $\mathcal{D}_s$ is $\mathcal{W}_s = \{w_{s1}, ..., w_{sN_s}\}$, and the vocabulary set of $\mathcal{D}_t$ is $\mathcal{W}_t = \{w_{t1}, ..., w_{tN_t}\}$.

In the example in Table 7.1, there is a one-to-one mapping between the source documents and the target documents. However, one-to-many or many-to-one mappings are not uncommon in other paired document sets. In this study, we consider the other mappings as special cases of one-to-one mappings and convert them to multiple one-to-one document pairs.

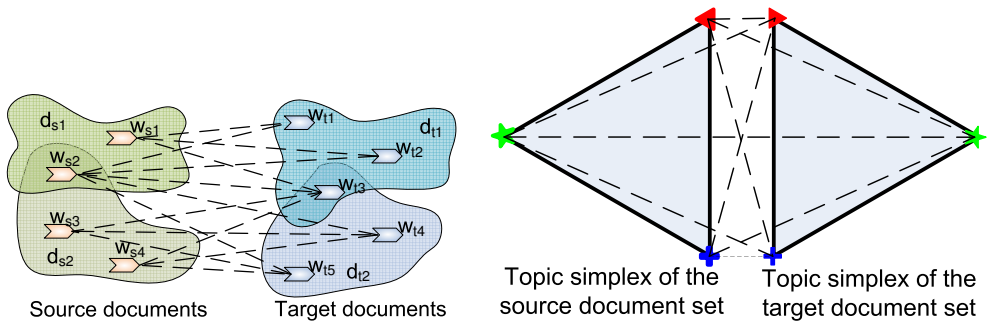
Given the above data as the training dataset, we aim to solve two problems: (1) *Modeling*: Model the associations between the source documents in $\mathcal{D}_s$ and the target documents in $\mathcal{D}_t$, and (2) *Ranking*: For a new source document d_s, rank and retrieve the target document d_t, that is most likely to be associated with d_s, from a repository of target documents.

7.4 Latent Association Analysis

The objective of our modeling problem is different from that of existing works [18, 19, 62]. Our main concern is to model the association between a pair of documents. The document retrieval task that we address is also different from traditional information retrieval tasks in two aspects: (1) Our query involves a document, which is much noisier than a keyword query in traditional information retrieval tasks, and (2) The source document (query document) and the target documents to be retrieved arise from two separate document sets, between which we do not assume any vocabulary overlap. Therefore, similarity-based relevance scores do not apply to this problem. These differences motivated us to develop a new model to capture the latent association existing among document pairs.

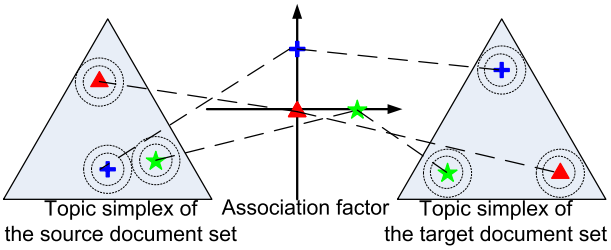
Conceptually, the association between the source and target documents can be considered at three different levels of granularity, yielding three possible solutions:

Word-level correlation (Figure 7.1(a)): Given individual words in the source documents, we can directly model whether and how they are correlated with the words in the target documents using a training dataset. Unfortunately, synonyms and polysemy in free text make the correlation at the word level noisy. It is better to first consider topics built from word co-occurrence patterns and then analyze topic-level correlations.



(a) Word-level correlation.

(b) Topic-level correlation.



(c) Document-level correlation.

Figure 7.1: Analyzing the associations at different levels of granularity.

Topic-level correlation (Figure 7.1(b)): Topics, usually considered a probabilistic distribution over words, are understood as a reduced-dimension representation of the semantic elements exposed in a document set. Topics are more stable than words. Topic-level correlation can be analyzed by first learning two topic structures from the two document sets separately and then discovering their correlations. A problem with this approach is that topics learned separately might not reflect the associations in document pairs. For instance, in question-answer document pairs, the topics of a question

(source) can be understood differently when the answers (target) emphasize different aspects of the question.

Document-level correlation (Figure 7.1(c)): Instead of generating topics separately, we can learn the topics for the source and target documents simultaneously. We define a correlation factor for a document pair. The topic proportions of the two documents are drawn based on this correlation factor. In this approach, the topic distribution of each (source or target) document is studied in the context of a document pair. This approach allows flexible topic assignment if the same source document is paired up with different target documents, and vice versa. That is, the same source document can have different topic assignments in different contexts.

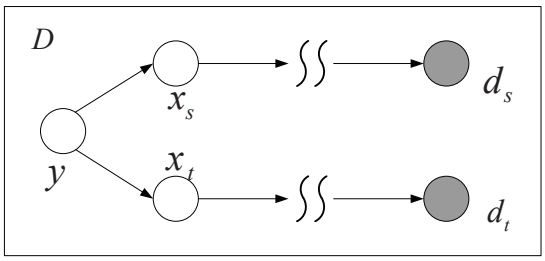


Figure 7.2: Basic structure of the LAA framework.

The Latent Association Analysis (LAA) framework described in this chapter takes the document-level correlation approach. As shown in Figure 7.2, LAA consists of two components, the correlation factor y between two latent variables x_s and x_t, and the document-generation processes for d_s and d_t. We can instantiate LAA with different correlation models and topic models. The models of generating source and target docu-

ments can even be different. Once LAA is learned based on training document pairs, it can be directly applied to solve our ranking problem. For a given query d_s, we can rank pairs (d_s, d_t) based on not only the topics of d_s and d_t, but also the correlation factor between them.

7.5 Modeling Document Pairs

In this section, we introduce an instantiation of the LAA framework with Canonical Correlation Analysis (CCA) [10] and the Correlated Topic Model (CTM) [18], and derive a variational method [17] to estimate the parameters for the model.

7.5.1 Canonical Correlation Analysis

Canonical Correlation Analysis (CCA) [88] works on two sets of random variables and their covariance matrix. Two linear transformations are found for the two sets of random variables such that the two sets of projected variables have maximum correlation with each other. Bach *et al.* [10] gave a probabilistic interpretation of CCA and considered CCA as a model-based method that could be integrated with other probabilistic methods.

In CCA, the observed random variables $x_1 \in \mathbb{R}^{m_1}$ and $x_2 \in \mathbb{R}^{m_2}$ depend on a latent correlation factor $y \in \mathbb{R}^d$. The generative process can be described as follows.

- For a pair of variables, draw the correlation factor $y \sim \mathcal{N}(\mathbf{0}, \mathbf{I_d})$ where $min\{m_1, m_2\} \geq d \geq 1$.

- For each set of random variables, draw

  $x_1|y \sim \mathcal{N}(T_1 y + \mu_1, \Psi_1)$, $T_1 \in \mathbb{R}^{m_1 \times d}$, $\Psi_1 \succeq 0$

  $x_2|y \sim \mathcal{N}(T_2 y + \mu_2, \Psi_2)$, $T_2 \in \mathbb{R}^{m_2 \times d}$, $\Psi_2 \succeq 0$.

In LAA, we can use CCA to capture the semantic association between the source document and the target document. The two random variables x_s and x_t are lower-dimensional representations of the source and target documents, respectively. The correlation factor y represents why these two documents are associated on a semantic level.

7.5.2 Latent Association Analysis

Whereas CCA can capture the semantic association in a document pair, many existing topic models can capture the topics of the two documents. Choices include CTM [18], LDA [19], PLSI [62], *etc*. If PLSI is used, the random variables x_s and x_t are the topic proportions of documents d_s and d_t. If LDA is used, the random variables x_s and x_t are the Dirichlet priors of the topic proportions in d_s and d_t. If CTM is used, the topic proportion of a document is modeled as a Gaussian variable, which naturally fits in with x_s or x_t in CCA. In this chapter, we choose CTM to instantiate LAA.

The instantiated LAA model is depicted in Figure 7.3. The LAA model comprises the model parameters in the set $M = \{\Psi_s, T_s, \mu_s, \Psi_t, T_t, \mu_t, \beta_s, \beta_t\}$. The words in the source and target documents, $w_{s,1:l_s}$ and $w_{t,1:l_t}$, where l_s and l_t are the document length of d_s and d_t, are the observable variables. The latent variables (*i.e.*, variables that are neither directly observable nor explicitly specified in the learned model) form the parameter set $V_l = \{y, x_s, x_t, z_{s,1:l_s}, z_{t,1:l_t}\}$.

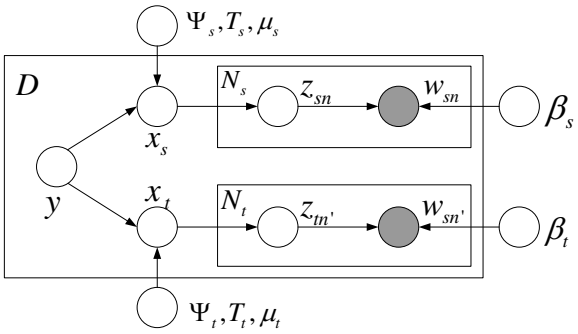


Figure 7.3: Graphical representation of the LAA model.

The generative process can be described as follows:

1. For each edge in the bipartite graph $\mathcal{G}$ (*i.e.*, a document pair), draw an L-dimensional Gaussian correlation factor: $y \in \mathcal{N}(0, I_L)$. The dimension L $<$ min $\{K_s, K_t\}$, where K_s is the number of topics in the source document set $\mathcal{D}_s$ and K_t is the number of topics in the target document set $\mathcal{D}_t$.

2. For each document pair connected by an edge, draw topic proportions as follows:

   For the source document, draw

$x_s|y \sim \mathcal{N}(T_s y + \mu_s, \Psi_s)$; $T_s \in \mathbb{R}^{K_s \times L}$, $\Psi_s \succeq 0$.

For the target document, draw

$x_t|y \sim \mathcal{N}(T_t y + \mu_t, \Psi_t)$; $T_t \in \mathbb{R}^{K_t \times L}$, $\Psi_t \succeq 0$.

3. For each word in the source document, choose:

   (a) a topic $z_{sn}|x_s \sim Mult(\theta_s)$, where

   $\theta_{si} = exp(x_{si})/\sum_j exp(x_{sj})$ for $i \in \{1, 2, ...K_s\}$.

   (b) a word $w_{sn}|z_{sn}, \beta_s \sim Mult(\beta_{sz_{sn}})$.

   The topics and words in the target document are chosen in a similar manner.

Although the topic modeling portion of LAA stems from the idea of CTM, LAA is more complicated than the existing topic models. It is built on a set of document pairs, instead of a single document set as in existing topic models. As a result, the latent topic structures in the source document set and the target document set, as well as their correlation, need to be analyzed simultaneously. LAA considers each edge in the bipartite graph as a correlation factor that initiates the connection between two documents. The generation process of the topic proportions depends on the correlation factor, which means that LAA first decides what makes the connection between the source documents and the target documents at the document level. LAA models the pair consisting of the source document and the target document as a co-occurrence in-

terpreted by the correlation factor, instead of assuming a causality relationship between the two documents, which is difficult to validate.

It is worth noting that the topic proportion of a document is context-dependent. The same piece of text, in the eyes of interpreters with different emphases, can belong to different topics. In LAA, each source or target document is put in the context of a pair, allowing the topic proportion of each document to be mutually enhanced and to be context-dependent. Doing so provides the flexibility of not deciding the topic of a document until we have learned what is emphasized in the other document paired with it.

7.5.3 Variational Inference and Parameter Estimation

Given the LAA model described above, we need to solve the following two problems: (1) Model fitting: Given a set of document pairs, how do we find model parameters that best fit the data? (2) Inference: For a new document pair, how do we decide the correlation factor y and the topic proportions x_s, x_t and the topic assignment z for each word? Because the best-fit model parameters are computationally intractable, similar to CTM, our LAA model employs a variational method to solve these two problems.

Variational Inference

Consider a pair (d_s, d_t) of documents, represented as sets of words $\{w_{sn}\}$ and $\{w_{tn'}\}$, where w_{sn} is the nth word in d_s and $w_{tn'}$ is the n'th word in d_t, Equation (7.1)

evaluates the probability that the document pair arises from an LAA model represented by parameter set M.

$$P(d_s, d_t|M) = \int_y \int_{x_s} \int_{x_t} P(y)P(x_s|y, M)P(x_t|y, M) \times \prod_{k'=1}^{K_t} \prod_{n'=1}^{l_t} (P(z_{tn'} = k'|x_t)P(w_{tn'}|z_{tn'}, \beta_t))d(x_t) \times \prod_{k=1}^{K_s} \prod_{n=1}^{l_s} (P(z_{sn} = k|x_s)P(w_{sn}|z_{sn}, \beta_s))d(x_s)d(y) \quad (7.1)$$

Ideally, the latent variables in the set V_l should be chosen to maximize the probability $P(d_s, d_t|M)$ to best fit the pair of documents. Unfortunately, it is computationally intractable to determine the true posterior distribution over V_l, because the latent variables are coupled together. Thus, we introduce a variational distribution $Q(V_l)$, in which the latent variables are independent of each other, to approximate the true posterior distribution $P(V_l|d_s, d_t)$. The graphical representation of Q is shown in Figure 7.4. According to the variational distribution, $Q(y) \sim \mathcal{N}(\bar{y}, \Sigma)$, $Q(x_{si}) \sim \mathcal{N}(\bar{x}_{si}, \sigma^2_{si})$, $Q(x_{ti}) \sim \mathcal{N}(\bar{x}_{ti}, \sigma^2_{ti})$, $Q(z_{sn}) \sim Multi(\phi_{sn})$ and $Q(z_{tn}) \sim Multi(\phi_{tn})$. Note that each component in the topic proportions x_s and x_t are drawn independently. The variational parameters introduced in the variational distribution are fit such that the KL-divergence between $Q(V_l)$ and $P(V_l|d_s, d_t)$ is minimized.

Using the variational distribution and Jensen's inequality, we take the logarithm of the probability in Equation (7.1) and rewrite the objective function in Equation (7.2). Instead of maximizing the log likelihood directly, which is intractable, we maximize

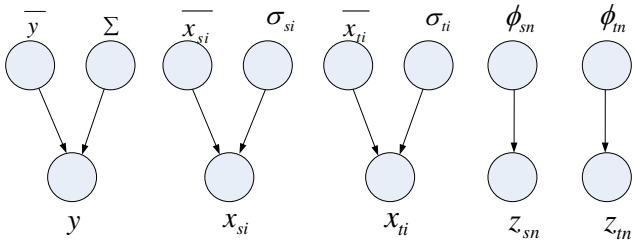


Figure 7.4: Variational distribution.

the lower bound of the log likelihood to obtain an approximation of the optimal value of the latent variables.

$$\log(P(d_s, d_t|M)) \geq E_Q \log(P(d_s, d_t|M)) + H(Q) = \lfloor \mathcal{L} \rfloor \tag{7.2}$$

The above maximization problem is a convex optimization problem and, thus, the optimal values of the variational parameters occur when the derivatives are zero. According to the decomposition of the marginal probability in Equation (7.1), we expand the lower bound of the log likelihood as follows:

$$\begin{aligned}
\lfloor \mathcal{L} \rfloor = & \sum_n E_Q \log P(w_{sn}|z_{tn}, \beta_s) + \sum_{n'} E_Q \log P(w_{tn'}|z_{tn'}, \beta_t) \\
& + \sum_n E_Q \log P(z_{sn}|x_s) + \sum_{n'} E_Q \log P(z_{tn'}|x_t) \\
& + E_Q \log P(x_s|y, \Psi_s, T_s, \mu_s) + E_Q \log P(x_t|y, \Psi_t, T_t, \mu_t) \\
& + E_Q \log P(y) + H(Q(V_l))
\end{aligned} \tag{7.3}$$

where each term on the right-hand side is a function over the variational parameters as shown in Equation (7.4) - (7.8):

$$\sum_n E_Q \log(P(w_{an}|z_{an}, \beta_a)) = \sum_{n=1}^{l_a} \sum_{k=1}^{K_a} \phi_{ank} \log(\beta_{ank}) \tag{7.4}$$

Here, a represents the source document s or the target document t in a pair. Because a document pair is symmetric, we use the same set of equations with different subscripts.

According to LAA, the topic assignment z is drawn based on the Gaussian prior x, $P(z_n = k|x) = \frac{exp(x_k)}{\sum_j exp(x_j)}$. Let $\iota = \sum_j exp(x_j)$. If we take the first-order Taylor expansion with respect to ι at point ζ to approximate $\log P(z_n = k|x)$, we have $\log P(z_n = k|x) = x_k - \log(\zeta) - \frac{1}{\zeta}(\sum_j exp(x_j) - \zeta) + \mathcal{O}((\iota - \zeta)^2)$. Thus,

$$\begin{aligned} \sum_n E_Q \log(P(z_{an}|x_a)) \geq & \sum_{n=1}^{l_a} \sum_{k=1}^{K_a} \phi_{ank} \bar{x}_{ak} \\ & - l_a \log(\zeta_a) - \frac{l_a}{\zeta_a} \sum_{k=1}^{K_a} \exp(\bar{x}_{ak} + \frac{\sigma_{ak}^2}{2}) + l_a \end{aligned} \tag{7.5}$$

where ζ is an additional variational parameter.

$$\begin{aligned} E_Q \log(P(x_a)) = & \frac{1}{2} \log(|\Psi_a^{-1}|) - \frac{1}{2} tr(\text{diag}(\sigma_a^2)\Psi_a^{-1}) \\ & - \frac{1}{2} tr((T_a \bar{y} + \mu_a - \bar{x}_a)(\bar{y}^T T_a^T + \mu_a^T - \bar{x}_a^T)\Psi_a^{-1}) \\ & - \frac{1}{2} tr(T_a \Sigma T_a^T \Psi_a^{-1}) + \text{const} \end{aligned} \tag{7.6}$$

$$E_Q \log(P(y)) = -\frac{1}{2} \log(2\pi) - \frac{1}{2} tr(\Sigma) - \frac{1}{2} \bar{y}^T \bar{y} \tag{7.7}$$

$$H(Q) = -\sum_{a=s,t}\sum_{n=1}^{l_a}\sum_{k=1}^{K_a}\phi_{ank}\log(\phi_{ank}) + \frac{1}{2}\log(det(\Sigma)) + \sum_{a=s,t}\sum_{k=1}^{K_a}\log(\sigma_{ak}) + \text{const} \tag{7.8}$$

We substitute Equation (7.4)-(7.8) into Equation (7.3), and then maximize the lower bound of the log likelihood by taking the partial derivatives with respect to each of the variational parameters and setting them to zero.

For the variational parameters ζ, ϕ, Σ and y, the optimal values that maximize the objective function are achieved by:

$$\zeta_a = \sum_k exp(\bar{x}_{ak} + \frac{\sigma_{ak}^2}{2}) \tag{7.9}$$

$$\phi_{ank} \propto \beta_{akv} exp(\bar{x}_{ak}), \text{ s.t. } w_{an}^v = 1. \tag{7.10}$$

$$\Sigma = \sum_{a=s,t} T_a^T \Psi_a^{-1} T_a + I_L \tag{7.11}$$

$$\bar{y} = \Sigma \sum_{a=s,t} T_a^T \Psi_a^{-1}(\bar{x}_a - \mu_a) \tag{7.12}$$

For the variational parameters $\bar{x}$ and σ, there are no analytical solutions. The optimal values of these variables are the solutions to Equation (7.13) and (7.14), which are solved iteratively using Newton's method.

$$\sum_n \phi_{an} - \frac{l_a}{\zeta_a} \exp(\bar{x}_a + \frac{\sigma_a^2}{2}) - \Psi_a^{-1}(T_a\bar{y} + \mu_a - \bar{x}_a) = 0 \tag{7.13}$$

$$\frac{l_a}{\zeta_a} \exp(\bar{x}_a + \frac{\sigma_a^2}{2}) + \text{diag}(\Psi_a^{-1}) - \frac{1}{\sigma_a^2} = 0 \tag{7.14}$$

For each edge in the bipartite graph, we calculate the variational parameters using Equation (7.9) - (7.14) iteratively until the log likelihood lower bound in Equation (7.3) no longer increases. The resulting variational parameter values are an approximation of the optimal values of the latent variables. Specifically, $y^* = \bar{y}$, $x^*_{ak} = \bar{x}_{ak}$, $z^*_{an} = \arg_k \max(\phi_{ank})$, where $a \in \{s, t\}$, $k \in \{1, 2, ..., K_a\}$, $n \in \{1, 2, ...l_a\}$.

Parameter Estimation

We estimate the model parameters using the variational expectation-maximization algorithm. In the E-Step, we update the variational parameters for each edge in the bipartite graph. In the M-Step, we update the model parameters, so that the sum of the log likelihood lower bound on each edge is maximized.

The process used in the M-Step is similar to that of variational inference. The goal here is to maximize the aggregated log likelihood of all the edges in the bipartite graph, rather than maximizing the log likelihood of a single edge. We sum up the lower bounds of the log likelihood in Equation (7.2) for each edge and take the partial derivative over the set M of model parameters. We then calculate the optimal values of the model parameters by setting these derivatives to zero.

$$\beta_{akv} \propto \sum_{e \in \mathcal{E}} \sum_{n} \phi_{adnk} 1(w^v_{ean} = 1) \quad \text{s.t.} \sum_v \beta_{akv} = 1. \tag{7.15}$$

$$T_a = (\sum_{e \in \mathcal{E}} (\bar{x}_{ea} \bar{y}_e^T - \mu_a \bar{y}_e^T))(\sum_{e \in \mathcal{E}} (\bar{y}_e \bar{y}_e^T + \Sigma_e))^{-1} \tag{7.16}$$

$$\mu_a = \frac{1}{|\mathcal{E}|}(\sum_{e \in \mathcal{E}} \bar{x}_{ea} - T_a \sum_{e \in \mathcal{E}} \bar{y}_e) \tag{7.17}$$

$$\Psi_a = \frac{1}{|\mathcal{E}|} \sum_{e \in \mathcal{E}} (\text{diag}(\sigma^2_{ea}) + T_a \Sigma_e T_a^T + (T_a \bar{y}_e + \mu_a - \bar{x}_{ea})(T_a \bar{y}_e + \mu_a - \bar{x}_{ea})^T) \tag{7.18}$$

The E-Step and the M-Step are performed iteratively until the model parameters converge, indicating that the model parameters are fit to the training dataset.

7.6 Ranking Document Pairs

Given an LAA model M learned from a training dataset, for a new source document d_s, we aim to rank the target documents in a test dataset according to their potential associations with the source document, In this section, we introduce three different methods to this problem. We evaluate these methods, together with the PTM method proposed by Zhang *et al.* [153], in Section 7.7.

7.6.1 Two-Step Method

First, we discuss a Two-Step method that mines the topics in the target and source document sets independently and then determines the correlation between their topic structures. This method is used as the baseline to compare with LAA.

The training process consists of two steps: (1) Find the topics in the source and target document sets, respectively, and (2) Find the correlation between the source and target topic structures. In the first step, CTM is independently applied to the two document sets $\mathcal{D}_s$ and $\mathcal{D}_t$. The topic proportion priors x_s and x_t are obtained for $\mathcal{D}_s$ and $\mathcal{D}_t$, respectively, using the variational inference method proposed in [18]. For each document pair (d_s, d_t), the corresponding topic proportion priors (x_s, x_t) form a pair. In the second step, these topic proportion priors, which follow Gaussian distributions, are fed into CCA. The CCA parameters T_1, T_2, μ_1, μ_2, Ψ_1, Ψ_2 are fit to the topic proportion pairs (x_s, x_t).

In the document retrieval task, given a new source document d_s, our goal is to evaluate the target documents in a test set. The candidates d_t are ranked based on the probability $P(d_t|d_s)$ that a target document d_t can be observed in a document pair containing the source document d_s.

We assume that the topic proportion priors x are a lower dimensional representation of the document d. Thus, $P(d_t|d_s) \propto P(x_t|x_s)$. In CCA, given x_1, the latent correlation factor y follows a normal distribution: $y|x_1 \sim \mathcal{N}(M_1^T U_{1d}^T (x_1 - \mu_1), I - M_1 M_1^T)$ [10],

whereas given y, x_2 follows a normal distribution: $x_2|y \sim \mathcal{N}(T_2 y + \mu_2, \Psi_2)$. Thus, given the topic proportion prior x_s of a source document d_s, its corresponding document d_t has a topic proportion prior x_t following the normal distribution:

$$x_t|x_s \sim \mathcal{N}(T_1 M^T (x_s - \mu_1) + \mu_2, \Psi_2 + T_1(I - MM^T)T_1^T) \tag{7.19}$$

where $M = (P_l)^{1/2}$ and P_l is the diagonal matrix of the top l canonical correlations.

Given a source document d_s and a candidate target document d_t, their topic proportion priors x_s and x_t can be inferred using CTM. Thus, the target documents can be ranked using $P(x_t|x_s)$ calculated from Equation (7.19).

7.6.2 LAA Direct Method

The LAA model derived in Section 7.5 allows us to predict, for a new source document d_s, which target document d_t is more likely to be associated with d_s. An direct way of ranking target documents is to evaluate how likely a hypothetical document pair (d_s, d_t) arises from the underlying LAA model. The lower bound of $\log(P(d_s, d_t|M))$ can be estimated by Equation (7.3) using the variational inference method discussed in Section 7.5.3. Thus, we can use function $R(d_s, d_t) = \lfloor \log(P(d_s, d_t|M) \rfloor$ to rank the target documents. Because both the source and target documents are considered as a bag of interchangeable words in the LAA model, the generation probability of a long document is smaller than the generation probability of a short document. Note that in

this prediction method, the ranking score of a document pair is inversely proportional to the document length. To avoid unfairly penalizing long documents, we normalized all of the documents to unit length.

7.6.3 LAA Latent Method

Although the LAA direct method is intuitive, it has potential drawbacks. In ranking document pairs, the most important factor should be the semantic association between the source and target documents; the exact wording of a document in expressing its semantic meanings should not be overemphasized. However, when evaluating a document pair using the probability that this document pair arises from the LAA model, the LAA direct method considers all of the words in the source and target documents as equally important. Consequently, if a target document contains rare words, it will be ranked low. The reason is that, even if the rare words in the target document might associate perfectly with the source document semantically, the probability of generating such words is still very low, which brings down the rank of the target document. Moreover, in our ranking, the popularity of the correlation factor should not matter, as long as it interprets the semantic association in a document pair. The LAA direct method cannot accommodate this feature either.

To address the aforementioned problems, we developed the LAA latent method based on the semantic association between source and target documents. In this method,

only the topic association information is used to rank the document pairs. For any given source document d_s and target document candidate d_t, first we use variational inference to calculate the most probable correlation factor $y^* = \bar{y}$, and the topic proportion $x_s^* = \bar{x}_s$ and $x_t^* = \bar{x}_t$, according to the variational distribution. Then, we evaluate how likely there exists an association between the two documents based on the topic proportion, and use the following ranking function to rank the target documents.

$$R(d_s, d_t) = P(x_s^*, x_t^*|y^*) = P(x_s^*|y^*)P(x_t^*|y^*) \tag{7.20}$$

In Equation (20), $P(x_s|y^*) \sim \mathcal{N}(T_s y^* + \mu_s, \Psi_s)$, and $P(x_t|y^*) \sim \mathcal{N}(T_t y^* + \mu_t, \Psi_t)$.

7.7 Experiments

We trained the LAA model based on real-world datasets and evaluated its performance on the document retrieval task. Two IT service datasets collected from IBM, *IT-Change* and *IT-Solution*, are used to evaluate the effectiveness of the LAA model.

7.7.1 Datasets

The *IT-Change* dataset was obtained in the context of IT change management. In IT-change management, when a change to the current IT environment is requested, the service provider needs to identify the possible problems caused by this change and, hence, assess its impact and cost. In this dataset, each document pair consists

of a change document, which describes the planned change, and a problem document, which describes the problem resulting from this change. Both the change and problem documents are in text, and the associations between them are currently established by human experts. Given a historical change-problem dataset, we built an LAA model and used it to retrieve the potential problem documents (from a set of problems reported) resulting from a new change request. This dataset contained 24,317 pairs of documents. We randomly sampled 20,000 document pairs for training and used the rest to evaluate the performance of our ranking method.

The *IT-Solution* dataset was obtained in the context of IT problem management. In IT-problem management, each solved problem needs to be documented with a solution. In practice, it is extremely challenging for a service agent to identify the correct solution for a new problem, from a solution repository that contains a large number of solution documents accumulated in the past. In this dataset, each document pair consists of a problem document and its corresponding solution document identified by human expert. LAA is used to predict possible solutions for new problems. This dataset contains 19,696 pairs of documents. We randomly selected 15,000 document pairs for training and the rest for testing.

7.7.2 Accuracy Analysis

We compare our two LAA-based methods, *i.e.*, LAA Direct (LAA-D) and LAA Latent (LAA-L), against the Two-Step method and the PTM method developed by Zhang *et al.* [153], in terms of their accuracy in retrieving target documents for a given source document. For a source document, PTM predicts a word distribution of its potential target document and compares it with the word distributions of the candidate documents. The word distribution in PTM has two components: one from the model, and the other from the similarity between the source and target documents. In LAA, we do not assume any overlap between the vocabularies of the source and target documents, which provides a key advantage over PTM. For comparison purposes, we use only the model component in PTM,

$$P_{PTM}(w_t|d_s) = \sum_{i=1}^{K_s} P(w_t|\theta_i)P(\theta_i|d_s) \tag{7.21}$$

and adopt the KL-divergence distance [75] to evaluate candidate target documents, as proposed in [153].

From each of the two datasets, we randomly select a batch of 100 document pairs, with only one-to-one mappings between the source and target documents for evaluation. Given a source document randomly selected within these 100 document pairs, we then rank the 100 target documents based on the four different methods. We use the average rank of the correct target document (the one actually paired with the selected source

document) to measure the performance. This process is repeated for five batches (*i.e.*, 500 queries in total) for both datasets.

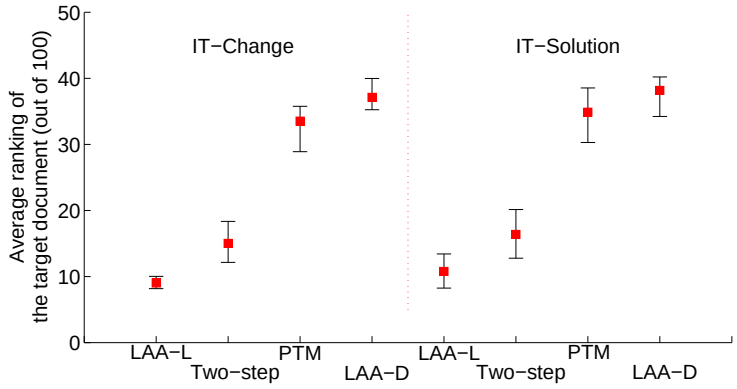


Figure 7.5: Comparison of retrieval accuracy of four methods on two datasets.

We set the number of topics to be 20 for both the source document set and the target document set to train the three models. In the Two-Step method and the LAA method, we set the dimension of the correlation factor to be 10. Figure 7.5 compares the performance of these four methods on the two datasets. The y axis shows the average rank of the correct target document out of the 100 target document candidates. Each bar in the figure shows the performance range of one method over the five batches of test cases. The average over the five batches is marked in red on each bar. For both datasets, LAA-L significantly outperforms all other methods, and the Two-Step method performs the closest to LAA-L. The key difference between the LAA-L method and the Two-Step method is that the topic structures of the source and target documents in the Two-Step method are learned independently without considering the correlations between them.

As a result, the performance of the Two-Step method is not as good as that of LAA-L. On the other hand, LAA-D suffers from the problems discussed in Section 7.6.2 and does not perform well in our document retrieval task. Due to the noisy nature of word-level correlation (as highlighted in Section 7.4), the PTM method does not show good performance either. We also experimented a modified version of PTM that compares the topic distributions, rather than the word distributions, between the source and target documents. The performance of this modified method is similar to that of the Two-Step method, but significantly worse than LAA-L.

7.7.3 Robustness Analysis

In this section, we address the robustness of the LAA-L method in capturing the semantic associations in document pairs, We trained the model with different numbers of topics and compared the results of the document retrieval task in an experimental setting similar to that in Section 7.7.2.

Figure 7.6 shows the experimental results on the *IT-Change* dataset. We do not show the results on the *IT-Solution* dataset, but our observations are quite similar. We chose the same number of topics for the source document set and the target document set, and the dimension of the correlation factor $L = \frac{1}{2}K_s = \frac{1}{2}K_t$. With different numbers of topics, the performance of the LAA-L method remains stable, and is consistently better than that of the other methods.

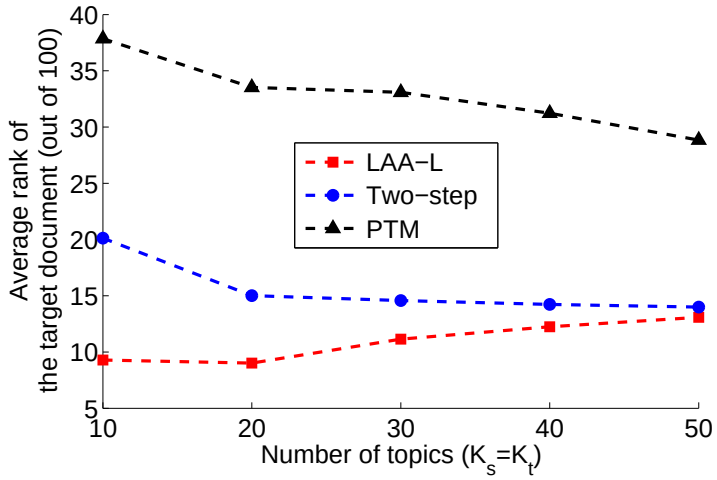


Figure 7.6: Performance comparison with different numbers of topics.

7.7.4 A Case Study

The LAA framework assumes a correlation factor. The topic portion priors of a pair of documents are drawn centered around a point in their corresponding topic simplex. Because each point in the topic simplex implies a mixture of the topics and each topic is represented by a probability distribution of words, the point in the topic simplex can also be mapped to a distribution over words. We now give examples of correlation factors and the corresponding top-ranked words in the source documents and the target documents. In these examples, the dimension of the correlation factor was set to 10. The numbers of topics in both the source and target document sets were set to 20. Note that the topic numbers in the source and target documents do not have to be the same.

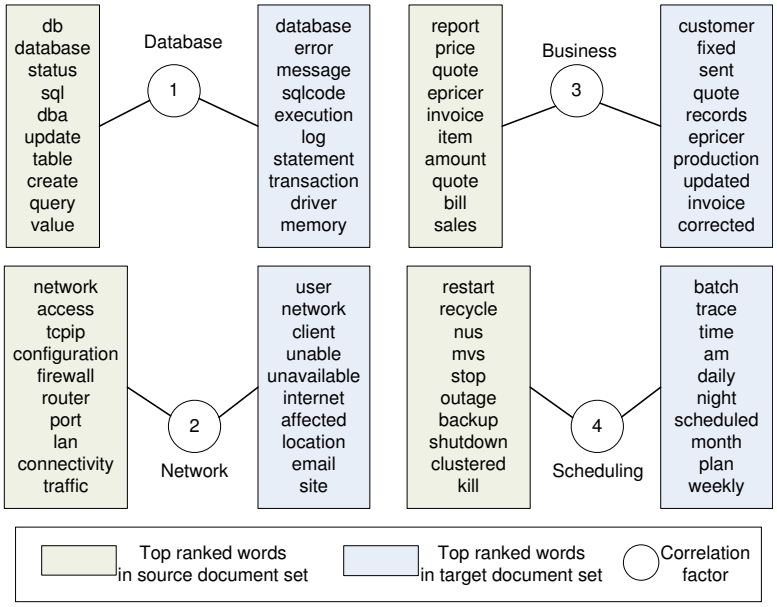


Figure 7.7: Sample top ranked words linked to the same correlation factor.

As shown in Figure 7.7, the LAA model successfully captures the semantic-level connections between the source documents and the target documents. Cases 1 and 2 were extracted from the *IT-Change* dataset, whereas Cases 3 and 4 were extracted from the *IT-Solution* dataset. For Cases 1 through 4, the top-ranked words indicate the correlations between source and target documents are around Database, Network, Business and Scheduling, respectively[1].

[1]The notations to these correlation factors were added by the authors.

7.8 Summary

This chapter presented a topic modeling approach that analyzes the topic structures of two document sets linked by a bipartite graph. The Latent Association Analysis (LAA) method draws the topic proportion priors of a pair of documents based on a latent correlation factor. Unlike other topic models, the goal of LAA is not only to provide a semantic-level explanation of the topics contained in document pairs, but also to retrieve the associated target document, when a new source document is given. Based on LAA, we introduced a document-level ranking method that can help retrieve target documents associated with a source document. Experiments on real datasets confirm the effectiveness of our method in extracting semantic concepts of associated document pairs, and substantiates that LAA outperforms state-of-the-art algorithms for ranking document pairs.

Chapter 8

Conclusions and Future Work

This Ph.D. Dissertation addresses large-scale unstructured or semi-structured data on the Web and in social networks and contributes toward semantic understanding of the data with emphasis on parallel and distributed computing, data extraction and integration, information flow analysis, and topic modeling. In this chapter, we summarize the specific contributions and propose possible future work.

8.1 Parallel Spectral Clustering Algorithm

This Ph.D. Dissertation presented a parallel approach for spectral graph analysis, including spectral clustering and co-clustering. The scalability of spectral methods has been increased in both computation time and memory use by using multiple computers

in a distributed system. This approach makes it possible to analyze Web-scale data using spectral methods. Experiments show that our parallel spectral clustering algorithm performs accurately on artificial datasets and real text data. We also applied our parallel spectral clustering algorithm to a large Orkut dataset to demonstrate its scalability.

In future work, we plan to reduce the inter-computer communication cost to improve scalability further. We also plan to investigate incremental methods for community mining and discovery to achieve even greater performance speed up.

8.2 Information Extraction and Integration

To extract information from heterogeneous sources, this Ph.D. Dissertation presented a novel approach to data record extraction from Web pages. The method first detects the visually repeating patterns on a Web page and then extracts the data records. The novel idea of visual signal is introduced to simplify the Web page representation as a set of binary vectors instead of the traditional DOM tree. A data record list corresponds to a set of visual signals that appear regularly on the Web page. The normalized cut spectral clustering algorithm is employed to find the visual signal clusters. For each visual signal cluster, data record extraction and nested structure detection are conducted to extract both atomic-level and nested-level data records.

Experimental results on flat data record lists are compared with state-of-the-art algorithms. Our novel visual signal algorithm shows significantly higher accuracy than existing algorithms. For data record lists with a nested structure, we collected Web pages from the domains of business, education, and government. Our extraction algorithm demonstrates high accuracy for both atomic-level and nested-level data records. The execution time of the algorithm is linear in the document length for practical datasets.

Our algorithm depends only on the Web page structure without examining the Web page content, which makes it a domain-independent approach. The algorithm is suitable for handling Web-scale data because it is completely automatic and does not need any information other than the Web page.

In the future, we plan to extend this work to support data attribute alignment. Each data record typically contains multiple data attributes. Unfortunately, there is no one-to-one mapping from the HTML code structure to the data record structure. Identification of the data attributes offers the potential of better use of data on the Web.

The work presented here extracts data records from single Web pages. However, the Web is composed of billions of Web pages each with their own data records. Future work will include integration of heterogeneous data records across different Web pages.

In this Ph.D. Dissertation, we also described algorithms for partially recovering the semantics of tables on the Web that aims to integrate the hundreds of millions data tables on the Web. We explored an intriguing interplay between structured and un-

structured data on the Web, where we used text on the Web to recover the semantics of structured data on the Web. Because the breadth of the Web matches the breadth of structured data on the Web, we are able to recover the semantics effectively. In addition, we provided a detailed analysis of when our techniques will not work and how these limitations can be addressed.

In future research, we will investigate better techniques for information extraction to recover a larger fraction of binary relationships and techniques for recovering numerical relationships (*e.g.*, population, GDP, *etc.*). The other major direction of future research is increasing our table corpus by extracting tables from lists [50], structured Web sites, and PDF files.

8.3 Modeling Information Flow in Collaborative Networks

To address information flow in collaborative networks, this Ph.D. Dissertation presented generative models that characterize ticket routing in a network of expert groups, using both ticket content and routing sequences. These models capture the capability of expert groups either in resolving the tickets or in transferring the tickets along a path to a resolver. The Resolution Model considers only ticket resolvers and builds a resolution profile for each expert group. The Transfer Model considers ticket routing sequences

and establishes a locally optimized profile for each edge that represents possible ticket transfers between two groups. The Optimized Network Model (ONM) considers the end-to-end ticket routing sequence and provides a globally optimized solution in the collaborative network. For ONM, we present a numerical method to approximate the optimal solution which, in general, is difficult to compute.

Our generative models can be used to make routing predictions for a new ticket and minimize the number of transfer steps before it reaches a resolver. For the generative models, we presented three routing algorithms to predict the next expert group to which to route a ticket, given its content and routing history. Experimental results show that the proposed algorithms can achieve better performance than existing ticket resolution methods.

8.4 Collaborative Network Routing Efficiency Analysis

This Ph.D. Dissertation examined a special type of social network – collaborative networks. Detailed observations of three real-world collaborative networks were presented along with the static network topology and dynamic information routing for each network. Collaborative networks exhibit not only the truncated power-law node degree distribution but also organizational constraints. Information routing in collaborative networks is different from routing in conventional complex networks, such as computer

networks and airline networks, because of random factors in human decision making. The routing steps also follow a truncated power-law distribution, which implies that a considerable number of tasks travel along long sequences of steps before they are completed. Our results and observations for several different kinds of collaborative networks are consistent with each other, and can be generalized to other real-world collaborative networks. They help in understanding the complicated behavior exhibited in human collaboration.

Based on real-world data, we developed a graph model to generate networks similar to real collaborative networks, and a stochastic routing algorithm to simulate the human dynamics of collaboration. The models are independently validated using real-world data. We demonstrated that the two models can be used to answer real-world questions, such as: *"How can one design a collaborative network to achieve higher efficiency?"* To the best of our knowledge, our work is the first attempt to understand and quantify the complex human dynamics exhibited in collaborative networks and to estimate analytically the efficiency of real collaborative networks.

8.5 Latent Association Analysis

To analyze the semantic association between multiple document sets, *e.g.*, problems and solutions, symptoms and treatments, *etc.*, this Ph.D. Dissertation tackled the prob-

lem of analyzing the topic structures of two document sets linked by a bipartite graph. The Latent Association Analysis (LAA) model draws the topic proportion priors of a pair of documents based on a latent correlation factor. Unlike other topic models, the goal of LAA is not only to provide a semantic-level explanation of the topics contained in document pairs, but also to retrieve the associated target document, when a new source document is given. Based on LAA, we introduced a document-level ranking method that can help retrieve target documents associated with a source document. Experiments on real datasets confirm the effectiveness of our model in extracting semantic concepts of associated document pairs, and substantiates that LAA outperforms the state-of-the-art algorithms in ranking document pairs.

In future work, we plan to extend the LAA model to more complex association structures over multiple document sets. The symmetric structure of the source and target documents can be replaced by an asymmetric structure, when it is appropriate to do so for other applications.

Bibliography

[1] Bugzilla: http://www.bugzilla.org/.

[2] Cobra: Java HTML renderer and parser, http://lobobrowser.org/cobra.jsp.

[3] DBLP: http://www.informatik.uni-trier.de/∼ley/db/.

[4] Eclipse: http://www.eclipse.org/.

[5] Mozilla: http://www.mozilla.org/.

[6] Orkut: http://www.orkut.com/home.aspx.

[7] J. Anvik, L. Hiew, and G. C. Murphy. Who should fix this bug? In *Proceedings of the 28th International Conference on Software Engineering*, pages 361–370, Shanghai, China, 2006.

[8] A. Arasu and H. Garcia-Molina. Extracting structured data from Web pages. In *Proceedings of the 2003 ACM International Conference on the Management of Data*, pages 337–348, San Diego, CA, 2003.

[9] W. Arnoldi. The principle of minimized iteration in the solution of matrix eigen value problems. *Quarterly of Applied Mathematics*, 9:17–29, 1951.

[10] F. R. Bach and M. I. Jordan. A probabilistic interpretation of canonical correlation analysis. Technical report, Department of Statistics, University of California, Berkeley, 2006.

[11] K. Balog, L. Azzopardi, and M. de Rijke. Formal models for expert finding in enterprise corpora. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–50, Seattle, WA, 2006.

[12] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the Web. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2670–2676, Hyderabad, India, 2007.

[13] M. Banko and O. Etzioni. The tradeoffs between open and traditional relation extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 28–36, Columbus, OH, 2008.

[14] A. L. Barabasi and R. Albert. Emergence of scyaling in random networks. *Science*, 286(5439):509–512, 1999.

[15] M. Belkin and P. Niyogi. Towards a theoretical foundation for laplacian-based manifold methods. In *Proceedings of the Conference on Learning Theory*, pages 486–500, Bertinoro, Italy, 2005.

[16] M. Belkin, P. Niyogi, V. Sindhwani, and P. Bartlett. Manifold regularization: A geometric framework for learning from examples. 2004.

[17] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1st ed. 2006., 2nd ed. 2007 edition, October 2007.

[18] D. M. Blei and J. D. Lafferty. Correlated topic models. In *Proceedings of the Neural Information Processing Systems Conference*, pages 147–154, Vancouver, British Columbia, Canada, 2006.

[19] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March 2003.

[20] M. Boguna, D. Krioukov, and K. C. Claffy. Navigability of complex networks. *Nature Physics*, 5(1):74–80, 2008.

[21] J. Boyd-Graber and D. M. Blei. Multilingual topic models for unaligned text. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 75–82, Montreal, Quebec, Canada, 2009.

[22] T. Brants. TnT — a statistical part of speech tagger. In *Proceedings of the 6th Applied Natural Language Processing Conference*, pages 224–231, Seattle, WA, 2000.

[23] D. Buttler, L. Liu, and C. Pu. A fully automated object extraction system for the World Wide Web. In *Proceedings of the 21st IEEE International Conference on Distributed Computing Systems*, pages 361–370, Washington DC, 2001.

[24] M. Cafarella, A. Halevy, and N. Khoussainova. Data integration for the relational Web. volume 2(1), pages 1090–1101, Lyon, France, 2009.

[25] M. Cafarella, A. Halevy, D. Wang, E. Wu, and Y. Zhang. WebTables: Exploring the power of tables on the Web. volume 1, pages 538–549, Auckland, New Zealand, 2008.

[26] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Uncovering the relational Web. In *Proceedings of the 11th International Workshop on the Web and Databases*, Vancouver, BC, Canada, 2008.

[27] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. WebTables: Exploring the power of tables on the Web. volume 1, pages 538–549, Seattle, WA, 2008.

[28] P. Calado, M. Cristo, E. Moura, N. Ziviani, B. Ribeiro-Neto, and M. A. Gonçalves. Combining link-based and content-based methods for Web document classification. In *Proceedings of the ACM Conference on Information and Knowledge Management*, pages 394–401, New Orleans, LA, 2003.

[29] D. Carmel, H. Roitman, and N. Zwerding. Enhancing cluster labeling using Wikipedia. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 139–146, Boston, MA, 2009.

[30] C. Chang and S. Lui. IEPAD: Information extraction based on pattern discovery. In *Proceedings of the 10th International Conference on the World Wide Web*, pages 681–688, Hong Kong, China, 2001.

[31] K. C. Chang, B. He, C. Li, M. Patel, and Z. Zhang. Structured databases on the Web: Observations and implications. *ACM SIGMOD Record*, 33(3):61–70, 2004.

[32] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. In *Proceedings of the Neural Information Processing Systems Conference*, pages 281–288, Vancouver, British Columbia, Canada, 2007.

[33] F. Chung. *Spectral graph theory*. Number 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Society, 1997.

[34] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51:661–703, 2009.

[35] V. Crescenzi, G. Mecca, and P. Merialdo. RoadRunner: Towards automatic data extraction from large Web sites. In *Proceedings of the 27th International Conference on Very Large Data Bases*, pages 109–118, San Francisco, CA, 2001.

[36] D. Cutting, D. Karger, and J. Pedersen. Constant interaction-time scatter/gather browsing of very large document collections. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 126–134, Pittsburgh, PA, 1993.

[37] H. T. Dang, D. Kelly, and J. J. Lin. Overview of the trec 2007 question answering track. In *Proceedings of the Sixteenth Text REtrieval Conference*, Gaithersburg, MD, 2007.

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

[39] H. Deng, I. King, and M. R. Lyu. Formal models for expert finding on DBLP bibliography data. In *Proceedings of the IEEE International Conference on Data Mining*, pages 163–172, Pisa, Italy, 2008.

[40] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the 7th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 269–274, 2001.

[41] I. S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: Spectral clustering and normalized cuts. In *Proceedings of the 10th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 551–556, Seattle, WA, 2004.

[42] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors: A multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1944–1957, 2007.

[43] I. S. Dhillon and D. S. Modha. A data-clustering algorithm on distributed memory multiprocessors. In *Large-Scale Parallel Data Mining*, pages 245–260, 1999.

[44] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1–2):143–175, 2001.

[45] C. H. Q. Ding and X. He. K-means clustering via principal component analysis. In *Proceedings of the 21st International Conference on Machine Learning*, pages 225–232, Banff, Alberta, Canada, 2004.

[46] C. H. Q. Ding and X. He. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the SIAM International Conference on Data Mining*, pages 606–610, Newport Beach, CA, 2005.

[47] C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings of the IEEE International Conference on Data Mining*, pages 107–114, San Jose, California, 2001.

[48] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 57–66, San Francisco, CA, 2001.

[49] D. Downey, O. Etzioni, and S. Soderland. A Probabilistic Model of Redundancy in Information Extraction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1034–1041, Edinburgh, Scotland, UK, 2005.

[50] H. Elmeleegy, J. Madhavan, and A. Halevy. Harvesting relational tables from lists on the Web. volume 2, pages 1078–1089, Lyon, France, 2009.

[51] P. Erdös and A. Rényi. On random graphs I. *Publicationes Mathematicae*, 6:290–297, 1959.

[52] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam. Open Information Extraction: The second generation. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 3–10, Barcelona, Spain, 2011.

[53] H. Fang and C. Zhai. Probabilistic models for expert finding. In *Proceedings of the 29th European Conference on Information Retrieval*, pages 418–430, Rome, Italy, 2007.

[54] P. Forner, A. Penas, E. Agirre, I. Alegria, C. Forascu, N. Moreau, P. Osenova, P. Prokopidis, P. Rocha, B. Sacaleanu, R. Sutcliffe, and E. Tjong Kim Sang. Overview of the CLEF 2008 multilingual question answering track. In *Evaluating Systems for Multilingual and Multimodal Information Access*, Lecture Notes in Computer Science 5706, pages 262–295. Springer, Berlin/Heidelberg, 2009.

[55] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225, 2004.

[56] S. Galam. Minority opinion spreading in random geometry. *The European Physical Journal B - Condensed Matter and Complex Systems*, 25(4):403–406, 2002.

[57] S. Galam. Modelling rumors: The no plane pentagon French hoax case. *Physica A: Statistical Mechanics and Its Applications*, 320:571–580, 2003.

[58] X. Guardiola, A. Diaz-Guilera, C. J. Perez, A. Arenas, and M. Llas. Modeling diffusion of innovations in a social network. *Physical Review E*, 66:026121, 2002.

[59] R. Gupta and S. Sarawagi. Answering table augmentation queries from unstructured lists on the Web. volume 2, pages 289–300, Lyon, France, 2009.

[60] L. Hagen and A. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9):1074–1085, 1992.

[61] M. Hearst. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 539–545, Nantes, France, 1992.

[62] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57, Berkeley, CA, 1999.

[63] P. Ipeirotis and A. Marian, editors. *Proceedings of the Fourth International Workshop on Ranking in Databases*, 2010.

[64] Z. G. Ives, C. A. Knoblock, S. Minton, M. Jacob, P. P. Talukdar, R. Tuchinda, J. L. Ambite, M. Muslea, and C. Gazen. Interactive data integration through smart copy & paste. In *Proceedings of the 4th Biennial Conference on Innovative Data Systems Research*, Asilomar, CA, 2009.

[65] J. Jagaralamudi and H. Daumé. Extracting multilingual topics from unaligned corpora. In *Proceedings of the European Conference on Information Retrieval*, Milton Keynes, UK, 2010.

[66] A. Jamain and D. J. Hand. The naive Bayes mystery: A classification detective story. *Pattern Recognition Letters*, 26(11):1752–1760, 2005.

[67] E. R. Jessup and D. C. Sorensen. A parallel algorithm for computing the singular value decomposition of a matrix. *SIAM Journal on Matrix Analysis Applications*, 15(2):530–548, 1994.

[68] T. Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, Germany, 1998.

[69] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 137–146, Washington D.C., 2003.

[70] J. Kleinberg. Small-world phenomena and the dynamics of information. In *Proceedings of the Neural Information Processing Systems Conference*, page 2001, Vancouver, British Columbia, Canada, 2001. MIT Press.

[71] T. Konda, M. Takata, M. Iwasaki, and Y. Nakamura. A new singular value decomposition algorithm suited to parallelization and preliminary results. In *Proceedings of the 2nd IASTED International Conference on Advances in Computer Science and Technology*, pages 79–84, Anaheim, CA, 2006.

[72] Z. Kou and W. Cohen. Stacked graphical models for efficient inference in Markov random fields. In *Proceedings of the SIAM International Conference on Data Mining*, pages 533 – 538, Minneapolis, MN, 2007.

[73] N. Kushmerick, D. Weld, and R. Doorenbos. Wrapper induction for information extraction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 729–737, Kobe, Japan, 1997.

[74] A. H. F. Laender, B. A. Ribeiro-Neto, A. S. da Silva, and J. S. Teixeira. A brief survey of Web data extraction tools. *ACM SIGMOD Record*, 31(2):84–93, 2002.

[75] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th Annual ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 111–119, New Orleans, LA, 2001.

[76] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Proceedings of the Neural Information Processing Systems Conference*, pages 556–562, Denver, Colorado, 2000.

[77] R. B. Lehoucg, D. C. Sorensen, and C.Yang. *ARPACK iser's guide: solution of large scale eigenvalue problems by implicitly restarted arnoldi methods*. SIAM, 1998.

[78] K. Lerman, L. Getoor, S. Minton, and C. Knoblock. Using the structure of Web sites for automatic segmentation of tables. In *Proceedings of the 2004 ACM International Conference on Management of Data*, pages 119–130, Paris, France, 2004.

[79] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching Web tables using entities, types and relationships. In *Proceedings of the 36th International Conference on Very Large Data Bases*, pages 1338–1347, Singapore, 2010.

[80] D. Lin and X. Wu. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 1030–1038, Singapore, 2009.

[81] B. Liu. Mining data records in Web pages. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*, pages 601–606, Washington DC, 2003.

[82] B. Liu and Y. Zhai. NET: System for extracting Web data from flat and nested data records. In *Proceedings of the Conference on Web Information Systems Engineering*, pages 487–495, New York, NY, 2005.

[83] A. Lloyd and R. May. How viruses spread among computers and people. *Science*, 292(5520):1316–1317, 2001.

[84] Q. Lu and L. Getoor. Link-based text classification. In *Proceedings of the IJCAI Workshop on Text Mining and Link Analysis*, Acapulco, Mexico, 2003.

[85] F. T. Luk. A parallel method for computing the generalized singular value decomposition. *Journal of Parallel and Distributed Computing*, 2(3):250–260, 1985.

[86] D. Makowiec. Evolving network - simulation study. *The European Physical Journal B - Condensed Matter and Complex Systems*, 48:547–555, 2005.

[87] K. Malarz, Z. Szvetelszky, B. Szekf, and K. Kulakowski. Gossip in random networks. *ACTA Physica Polonica B*, 37, Nov. 2006.

[88] K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate analysis*. Academic Press, 1979.

[89] K. Maschhoff and D. Sorensen. A portable implementation of arpack for distributed memory parallel architectures. In *Proceedings of the Copper Mountain Conference on Iterative Methods*, Copper Mountain, CO.

[90] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. In *Technical Report*. http://www.cs.cmu.edu/ mccallum/bow, 1996.

[91] Q. Mei, D. Cai, D. Zhang, and C. Zhai. Topic modeling with network regularization. In *Proceedings of the 17th International Conference on the World Wide Web*, pages 101–110, Beijing, China, 2008.

[92] G. Miao, L. E. Moser, X. Yan, S. Tao, Y. Chen, and N. Anerousis. Generative models for ticket resolution in expert networks. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 733–742, Washington D.C., 2010.

[93] S. Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.

[94] D. Mimno, H. M. Wallach, J. Naradowsky, D. A. Smith, and A. Mccallum. Polylingual topic models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 880–889, Singapore, August 2009.

[95] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[96] R. M. Nallapati, A. Ahmed, E. P. Xing, and W. W. Cohen. Joint latent topic models for text and citations. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 542–550, Las Vegas, Nevada, 2008.

[97] J. Neville and D. Jensen. Iterative classification in relational data. In *Proceedings of the AAAI Workshop on Statistical Relational Learning*, pages 42–49, Austin, TX, 2000.

[98] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Proceedings of the Neural Information Processing Systems Conference*, pages 849–856, Vancouver, British Columbia, Canada, 2001.

[99] H. Ning, W. Xu, Y. Chi, Y. Gong, and T. S. Huang. Incremental spectral clustering with ppplication to monitoring of evolving blog communities. In *Proceedings of the SIAM International Conference on Data Mining*, Minneapolis, MN, 2007.

[100] M. Paşca. The Role of Queries in Ranking Labeled Instances Extracted from Text. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 955–962, Beijing, China, 2010.

[101] M. Paşca and B. Van Durme. Weakly-supervised acquisition of open-domain classes and class attributes from Web documents and query logs. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, Columbus, OH, 2008.

[102] P. Pantel and M. Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics*, pages 113–120, Sydney, Australia, 2006.

[103] M. Pennacchiotti and P. Pantel. Entity extraction via ensemble semantics. In *Proceedings of the 2009 Conference on Empirical Methods on Natural Language Processing*, pages 238–247, Singapore, 2009.

[104] H. H. Permuter, J. M. Francos, and I. Jermyn. A study of Gaussian mixture models of color and texture features for image classification and segmentation. *Pattern Recognition*, 39(4):695–706, 2006.

[105] J. Platt, N. Cristianini, and J. Shawe-taylor. Large margin DAGs for multiclass classification. In *Proceedings of the Neural Information Processing Systems Conference*, pages 547–553, Denver, CO, 2000.

[106] S. Ponzetto and R. Navigli. Large-scale taxonomy mapping for restructuring and integrating Wikipedia. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2083–2088, Singapore, 2009.

[107] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical recipes in C: The art of scientific computing*. Cambridge University Press, 2nd edition, Oct. 1992.

[108] D. P. Putthividhya, H. T. Attias, and S. Nagarajan. Independent factor topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 833–840, Montreal, Quebec, Canada, 2009.

[109] S. Ree. Power-law distributions from additive preferential redistributions. *Physical Review E*, 73:026115, February 2006.

[110] E. M. Rogers. *Diffusion of innovations*. Free Press, 4th edition, 1995.

[111] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[112] A. Sala, L. Cao, C. Wilson, R. Zablit, H. Zheng, and B. Y. Zhao. Measurement-calibrated graph models for social network experiments. In *Proceedings of the International Conference on the World Wide Web*, pages 861–870, Raleigh, NC, 2010.

[113] R. P. Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Physical Review*, 86(14):3200–3203, 2001.

[114] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3), 2008.

[115] P. Serdyukov, H. Rode, and D. Hiemstra. Modeling multi-step relevance propagation for expert finding. In *Proceedings of the ACM Conference on Information and Knowledge Management*, pages 1133–1142, Napa, CA, 2008.

[116] M. Serrano, D. Krioukov, and M. Boguna. Self-similarity of complex networks and hidden metric spaces. *Physical Review*, 078701, 2008.

[117] Q. Shao, Y. Chen, S. Tao, X. Yan, and N. Anerousis. Efficient ticket routing by resolution sequence mining. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 605–613, Las Vegas, NV, 2008.

[118] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[119] J. Shlens. A tutorial on principal component analysis, December 2005.

[120] K. Simon and G. Lausen. ViPER: Augmenting automatic information extraction with visual perceptions. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 381–388, Bremen, Germany, 2005.

[121] R. Snow, D. Jurafsky, and A. Ng. Semantic Taxonomy Induction from Heterogenous Evidence. In *Proceedings of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics*, pages 801–808, Sydney, Australia, 2006.

[122] X. Song, B. L. Tseng, C.-Y. Lin, and M.-T. Sun. ExpertiseNet: Relational and evolutionary expert modeling. In *Proceedings of the 10th International Conference on User Modeling*, pages 99–108, Edinburgh, 2005.

[123] M. Steyvers, P. Smyth, M. Rosen-Zvi, and T. Griffiths. Probabilistic author-topic models for information discovery. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 306–315, Seattle, WA, 2004.

[124] D. Strang and S. A. Soule. Diffusion in organizations and social movements: From hybrid corn to poison pills. *Annual Review of Sociology*, 24(1):265–290, 1998.

[125] A. Strehl and J. Ghosh. Cluster ensembles – A knowledge reuse framework for combining multiple partitions. *Journal on Machine Learning Research*, 3:583–617, 2002.

[126] T. Strohman, W. B. Croft, and D. Jensen. Recommending citations for academic papers. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 705–706, Amsterdam, Netherlands, 2007.

[127] F. Suchanek, G. Kasneci, and G. Weikum. YAGO: A core of semantic knowledge unifying wordnet and Wikipedia. In *Proceedings of the International Conference on the World Wide Web*, pages 697–706, Banff, Alberta, Canada, 2007.

[128] P. Talukdar, J. Reisinger, M. Paşca, D. Ravichandran, R. Bhagat, and F. Pereira. Weakly-supervised acquisition of labeled class instances using graph random walks. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing*, pages 582–590, Honolulu, HI, 2008.

[129] B. Taskar, M. F. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *Proceedings of the Neural Information Processing Systems Conference*, Vancouver, BC, Canada, 2003.

[130] J. Tatemura, S. Chen, F. Liao, O. Po, K. S. Candan, and D. Agrawal. UQBE: Uncertain query by example for Web service mashup. In *Proceedings of the 2008 ACM International Conference on Management of Data*, pages 1275–1280, Vancouver, Canada, 2008.

[131] R. Taylor. *Constrained switchings in graphs*. Research report. University of Melbourne, Department of Mathematics, 1980.

[132] J. Travers and S. Milgram. An experimental study of the small world problem. *Sociometry*, 32(4):425–443, 1969.

[133] P. Treeratpituk and J. Callan. Automatically labeling hierarchical clusters. In *Proceedings of the International Conference on Digital Government Research*, pages 167–176, New York, NY, 2006.

[134] T. W. Valente. Network models of the diffusion of innovations. *Computational and Mathematical Organization Theory*, 2:163–164, 1996.

[135] D. Wang, Z. Wen, H. Tong, C. Y. Lin, C. Song, and A. L. Barabási. Information spreading in context. In *Proceedings of the International Conference on the World Wide Web*, pages 735–744, India, 2011.

[136] J. Wang and F. H. Lochovsky. Data extraction and label assignment for Web databases. In *Proceedings of the 12th International Conference on the World Wide Web*, pages 187–196, Budapest, Hungary, 2003.

[137] R. Wang and W. Cohen. Iterative set expansion of named entities using the Web. In *Proceedings of IEEE International Conference on Data Mining*, pages 1091–1096, Cancun, Mexico, 2008.

[138] R. Wang and W. Cohen. Automatic set instance extraction using the Web. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 441–449, Singapore, 2009.

[139] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.

[140] F. Wu and D. Weld. Automatically refining the Wikipedia infobox ontology. In *Proceedings of the International Conference on the World Wide Web*, pages 635–644, Beijing, China, 2008.

[141] S. Wu, J. M. Hofman, W. A. Mason, and D. J. Watts. Who says what to whom on Twitter. In *Proceedings of the International Conference on the World Wide Web*, pages 705–714, India, 2011.

[142] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory andits application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, 1993.

[143] X. Xue, J. Jeon, and W. B. Croft. Retrieval models for question and answer archives. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 475–482, Singapore, 2008.

[144] Y. Yamada, N. Craswell, T. Nakatoh, and S. Hirokawa. Testbed for information extraction from deep Web. In *Proceedings of the 13th International Conference on the World Wide Web*, pages 346–347, New York, NY, 2004.

[145] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the Annual Internation ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 42–49, Berkeley, CA, 1999.

[146] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the ACM International Conference on Machine Learning*, pages 412–420, Nashville, TN, 1997.

[147] J. Yedidia, W. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.

[148] J. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *Proceedings of the Neural Information Processing Systems Conference*, pages 689–695, Denver, CO, 2000.

[149] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *Proceedings of the Neural Information Processing Systems Conference*, pages 1601–1608, Vancouver, British Columbia, Canada, 2005.

[150] H. Zha, C. H. Q. Ding, M. Gu, X. He, and H. Simon. Spectral relaxation for k-means clustering. In *Proceedings of the Neural Information Processing Systems Conference*, pages 1057–1064, Vancouver, British Columbia, Canada, 2001.

[151] H. Zha, X. He, C. H. Q. Ding, M. Gu, and H. D. Simon. Bipartite graph partitioning and data clustering. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management*, pages 25–32, Atlanta, GA, 2001.

[152] Y. Zhai and B. Liu. Web data extraction based on partial tree alignment. In *Proceedings of the 14th International Conference on the World Wide Web*, pages 76–85, Chiba, Japan, 2005.

[153] D. Zhang, J. Sun, C. Zhai, A. Bose, and N. Anerousis. PTM: Probabilistic topic mapping model for mining parallel document collections. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1653–1656, Toronto, ON, Canada, 2010.

[154] H. Zhang. The optimality of naive Bayes. In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*, Miami Beach, FL, 2004.

[155] B. Zhao and E. P. Xing. Bitam: Bilingual topic admixture models for word alignment. In *Proceedings of the International Conference on Computational Linguistics/ACL 2006 Main Conference Poster Sessions*, pages 969–976, Sydney, Australia, July 2006.

[156] B. Zhao and E. P. Xing. HM-BiTAM: Bilingual topic exploration, word alignment, and translation. In *Proceedings of the Neural Information Processing Systems Conference*, Vancouver, BC, Canada, 2007.

[157] H. Zhao, W. Meng, Z. Wu, V. Raghavan, and C. Yu. Fully automatic wrapper generation for search engines. In *Proceedings of the 14th International Conference on the World Wide Web*, pages 66–75, Chiba, Japan, 2005.

[158] H. Zhao, W. Meng, and C. Yu. Automatic extraction of dynamic record sections from search engine result pages. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 989–1000, Seoul, Korea, 2006.

[159] H. Zhao, W. Meng, and C. Yu. Mining templates from search result records of search engines. In *Proceedings of the 13th ACM International Conference on Knowledge Discovery and Data Mining*, pages 884–893, San Jose, CA, 2007.

[160] S. Zhong and J. Ghosh. Generative model-based clustering of documents: A comparative study. *Knowledge and Information Systems*, 8:374–384, 2005.

[161] D. Zhou, J. Bian, S. Zheng, H. Zha, and C. L. Giles. Exploring social annotations for information retrieval. In *Proceedings of the 17th International Conference on the World Wide Web*, pages 715–724, Beijing, China, 2008.

[162] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. S. Olkopf. Learning with local and global consistency. In *Proceedings of the Neural Information Processing Systems Conference*, pages 321–328, Vancouver, BC, Canada, 2004.

[163] J. Zhu, Z. Nie, J. Wen, B. Zhang, and W. Ma. Simultaneous record detection and attribute labeling in Web data extraction. In *Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining*, pages 494–503, Philadelphia, PA, 2006.