RESEARCHER EXPERTISE SEARCH, HOMEPAGE FINDING AND METADATA ANNOTATION

A Thesis in
Computer Science and Engineering
by
Sujatha Das Gollapalli

© 2013 Sujatha Das Gollapalli

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2013
The thesis of Sujatha Das Gollapalli was reviewed and approved* by the following:

Prasenjit Mitra  
Associate Professor, College of Information Sciences and Technology  
Dissertation Co-Advisor, Co-Chair of Committee

C. Lee Giles  
David Reese Professor, College of Information Sciences and Technology  
Dissertation Co-Advisor, Co-chair of Committee

Daniel Kifer  
Assistant Professor, Department of Computer Science and Engineering

Jesse Barlow  
Professor, Department of Computer Science and Engineering

David J. Miller  
Professor, Department of Electrical Engineering

Yanjun Qi  
Research Scientist, NEC Laboratories America, Inc.  
Special Member

Lee Coraor  
Associate Professor, Department of Computer Science and Engineering  
Graduate Program Chair

*Signatures are on file in the Graduate School.
Expert Search, the problem of retrieving people with expertise on a queried topic, has important applications [1]. For instance, conference organizers can use expert search capability while forming a panel for reviewing papers. Similarly, recruiters can use expert search to track potential employees for their companies. Although significant progress has been made on this problem, existing models for expert search are not tailored for academic domains. In academic domains, expert search and similar expert finding involve ranking researchers in response to topic and name queries based on academic documents. Academic or research documents are different from webpages in terms of their type (e.g., homepages, publications, grant proposals), structure (e.g., abstract, sections), associated metadata (e.g., venue, authors) and connections (e.g., citations).

Enabling expert search in an open-access digital library such as CiteSeer [2] is challenging since academic documents are not directly available for estimating expertise. Instead, CiteSeer acquires freely-available publications and other relevant academic documents by crawling the Web. Previous studies indicate that researchers list their publication information online using their homepages since this substantially increases the impact of their work [3]. It becomes imperative, therefore, to periodically track researcher homepage URLs in CiteSeer for obtaining up-to-date collections of academic documents. In addition to their use as a resource for academic documents, professional homepages of researchers also typically include descriptions of research interests and other metadata that is crucial in tasks such as author disambiguation and profile extraction [4].

Despite several studies on homepage finding in the context of the general web, academic homepage finding is not fully addressed in existing research. The first question we address in this dissertation is: how can we acquire an accurate homepage collection? We study this question in two settings. First, we study academic homepage finding on the Web. Given the results of web search for a researcher name query, our goal is to identify the correct homepage in the set of pages retrieved from the Web. We design features based on insights from content analysis of known academic homepages to learn a ranking function for academic homepage retrieval. In the second setting, we address homepage finding on university department websites where academic homepages need to be discriminated from other kinds of academic webpages. Despite training the classifier on “outdated” webpage instances, we show that unlabeled data and multiple views of webpages can be used to adapt our classifier to current-day academic websites.

In the second part of this dissertation, we address expert search for academic domains. We
study ranking models for researchers in response to topic and name queries. We use the content of research documents and the structural connections among documents to build query-dependent graphs for scoring researchers. We propose a simple extension to PageRank for combining evidence from multiple types of documents. This model scores researchers based on the structural connections among the documents and the importance of each document-type. In addition, we propose Author-Document-Topic graphs for scoring researchers based on the topical content of documents generated by them. Our models handle name and topic queries uniformly and show state-of-the-art retrieval performance on expert finding tasks.
Table of Contents

List of Figures viii
List of Tables x
Acknowledgments xii

Chapter 1 Introduction 1
1.1 Contributions and organization . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

Chapter 2 Abbreviations, measures and software 4
2.1 Evaluation measures . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
2.2 Software . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

Chapter 3 Academic homepage retrieval 7
3.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
3.2 Related work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
3.3 Content analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
  3.3.1 Homepages as topic mixtures . . . . . . . . . . . . . . . . . . . . . . . . . . 9
3.4 Features for ranking homepages . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
  3.4.1 Structural properties of homepages . . . . . . . . . . . . . . . . . . . . . . . 11
  3.4.2 Researcher homepage retrieval . . . . . . . . . . . . . . . . . . . . . . . . . . 12
3.5 Experiments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
  3.5.1 Results and observations . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
3.6 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17

Chapter 4 Identifying homepages on academic websites 18
4.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
4.2 Related work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
4.3 Features for homepage classification . . . . . . . . . . . . . . . . . . . . . . . . . 21
  4.3.1 URL strings as additional evidence . . . . . . . . . . . . . . . . . . . . . . . 21
4.4 Homepage classification using unlabeled data . . . . . . . . . . . . . . . . . . . . 23
  4.4.1 Imbalance in class distributions of instances . . . . . . . . . . . . . . . . . . 24
  4.4.2 Our co-training algorithm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
  4.4.3 Strategies for adding unlabeled instances for co-training . . . . . . . . . . . 25
List of Figures

3.1 Mean reciprocal rank for different methods: bl: baseline, prf: pseudo-relevance feedback, qexp: query expansion, am: ArnetMiner features, tprop: topic proportions, LDA-words: words from LDA, LDA-words+struct: words from LDA and structural features ................................................................. 15
3.2 Success rate for different methods: bl: baseline, prf: pseudo-relevance feedback, qexp: query expansion, am: ArnetMiner features, tprop: topic proportions, LDA-words: words from LDA, LDA-words+struct: words from LDA and structural features ................................................................. 16
4.1 Homepage classification performance using content features ....................................................... 30
4.2 Homepage classification performance using URL features ......................................................... 30
4.3 Homepage classification performance using content+URL features ............................................ 30
4.4 Self-training: classification performance with content features .................................................. 30
4.5 Co-training: classification performance using content features ................................................. 31
4.6 Co-training: classification performance using URL features ...................................................... 31
4.7 Performance of different co-training schemes (AddCross, AddCrossRC, AddBoth) with Naive Bayes Multinomial classifier ................................................................. 32
4.8 Performance of our gradient descent (GD) algorithm using content-based features and maximum entropy classifier ................................................................. 32
4.9 Performance of our gradient descent (GD) algorithm using URL features and maximum entropy classifier ................................................................. 34
4.10 Performance of our gradient descent (GD) algorithm using URL features and support vector machines ................................................................. 34
4.11 Performance of our gradient descent (GD) algorithm using URL features and maximum entropy classifier ................................................................. 34
4.12 Variation of the objective function value (ObjVal) and F1 as the gradient descent algorithm progresses ................................................................. 34
4.13 Reducing value of the squared-error loss as co-training progresses, MaxEnt: maximum entropy classifiers, SVM: support vector machines ................................................................. 36
4.14 Performance of on the test datasets, Org: performance with the base classifier, CT: co-training, GD: gradient descent, GET: generalized expectation ................................................................. 36
5.1 F1 variation across folds for affiliation, S1: stage 1 tagging, S2: stage 2 tagging, PR: posterior regularization ................................................................. 51
5.2 F1 variation across folds for university, S1: stage 1 tagging, S2: stage 2 tagging, PR: posterior regularization ................................................................. 51
5.3 Performance on the affiliation field with different numbers of initial training examples .......................... 52
5.4 Aggregate performance with different numbers of initial training examples ................................. 52
5.5 Performance with varying number of features, PR: posterior regularization, IG: information gain ......................................................... 53
5.6 F1 variation across folds for position, S1: stage 1 tagging, S2: stage 2 tagging, PR: posterior regularization ........................................... 53

6.1 An example author-document graph to illustrate the PageRank-based model ............... 63
6.2 An example Author-Document-Topic (ADT) graph ................................................................. 64
6.3 Comparison of baselines for name-based search on ArnetMiner dataset ................. 69
6.4 Comparison of baselines for name-based search on the UvT collection ......................... 69
6.5 ADT methods for topic-based search on the UvT collection ............................................ 70
6.6 ADT methods for name-based search on the UvT collection ............................................ 70
6.7 Recall@K for topic-based search on the ArnetMiner dataset ......................................... 71
6.8 Recall@K for name-based search on the ArnetMiner dataset ......................................... 71
6.9 ADT performance variation with number of topics on the ArnetMiner dataset .......... 71
6.10 ADT performance variation with number of topics on the UvT collection ............... 71
6.11 ADT methods for topic-based search on the ArnetMiner dataset .................................... 72
6.12 ADT methods for name-based search on the ArnetMiner dataset .................................... 72
6.13 Recall@K for topic-based search on the UvT collection ............................................ 72
6.14 Recall@K for name-based search on the UvT collection ............................................ 72
## List of Tables

3.1 Top words from topics related to homepages ........................................... 10  
3.2 Top words from topics related to subject areas ........................................ 11  
3.3 Structural features used in homepage ranking .......................................... 12  
3.4 Mean reciprocal rank for different methods: bl: baseline, prf: pseudo-relevance feedback, qexp: query expansion, am: ArnetMiner features, tprop: topic proportions, LDA-words: words from LDA, LDA-words+struct: words from LDA and structural features .......................................................... 16  
3.5 Success rate for different methods: bl: baseline, prf: pseudo-relevance feedback, qexp: query expansion, am: ArnetMiner features, tprop: topic proportions, LDA-words: words from LDA, LDA-words+struct: words from LDA and structural features .......................................................... 16  
4.1 Feature types and the size of feature sets used in homepage identification .... 21  
4.2 Example URLs with partial sets of extracted features (shown on the next line after each URL) ................................................................. 22  
4.3 List of the seed URLs used in creating homepage classification datasets .... 29  
4.4 Summary of the datasets used in classification experiments. The entry \( a/b \) represents \( a \) instances out of which \( b \) are labeled positive ........................................ 29  
4.5 Features ranked based on information gain on training and crawl datasets .... 31  
4.6 Features that are most affected after running Algorithm 2 on weight vectors from maximum entropy classifier ......................................................... 36  
4.7 Before/After performance on the test set with content features. Runs marked with “combined” use URL+content features, CT: co-training, ST: self-training, GD: gradient descent, GET: generalized expectation ................................. 37  
4.8 Comparison of feature selection using information gain (IG) and co-training (CT) .... 39  
5.1 Example illustrating homepage tagging ...................................................... 42  
5.2 Example demonstrating features added for stage 2 based on stage 1 predicted tags and window size=3 (pw: previous window, nw: next window, s1: stage 1) .... 45  
5.3 Feature templates for the Stage-1 tagger for metadata extraction ................. 45  
5.4 Sample cue words for different metadata fields ......................................... 45  
5.5 Expert-specified dictionary and proximity features for affiliation values. s1* refers to the predicted label obtained from the first-stage CRF .......................... 49  
5.6 Three-fold cross-validation F1 values for Stage 1 (S1) and Stage 2 (S2) models using expert-specified labeled features ......................................... 52  
5.7 Metadata extraction performance: Three-fold cross-validation F1 values are shown for different fields using the basic and the best performing sets of features ....... 53
5.8 Three-fold cross-validation F1 values for top 30 dictionary labeled features. #Expert features=22

5.9 Extraction performance using posterior regularization (PR) and dictionary constraints. The time in minutes and F1 values are shown for different number of training iterations

6.1 Summary of datasets used in expert search experiments

6.2 Each entry represents the number of authors having a given number of documents associated with them. For example, the number of authors in UvT having only one document is 384

6.3 Sample topics from the CiteSeer corpus

6.4 Sample topics from the UvT collection

6.5 Evaluation summary for name-based search

6.6 Evaluation summary for topic-based search

6.7 Top 10 recommendations made by author-document-topic (ADT) models for example topic queries

6.8 Top 10 recommendations made by author-document-topic (ADT) models for example name queries

6.9 Performance on topic-based search with the ArnetMiner dataset, PR: PageRank, PR-A: PageRank with “activated nodes” reset distribution, a: author, d: document, ad, dd, da: edge-type parameter settings

Acknowledgments

“It is not knowledge, but the act of learning, not possession but the act of getting there, which grants the greatest enjoyment” – Carl Friedrich Gauss

First among the people I want to thank are my advisors, Dr. Prasenjit Mitra and Dr. Lee Giles. In addition to giving me a chance to work with them during my PhD, Dr. Mitra and Dr. Giles have been a constant source of guidance and enthusiasm for me. I gratefully acknowledge their flexibility in allowing me to pursue directions of my interest and also for gently cautioning me when I lost focus. This thesis would not have happened without their constant advice with respect to my courses and research and financial support during the past few years.

Next, I thank the honorable members of my committee: Dr. Daniel Kifer, Dr. Jesse Barlow, Dr. David Miller and Dr. Yanjun Qi. Their constructive criticism, reviews and feedback have helped in shaping up my research into its current form. I particularly wish to thank Dr. Kifer for extensive discussions and ideas related to machine learning. These interactions have always helped me gain insights and perspective on the problems I was working on.

A special note of thanks to Dr. Cornelia Caragea for her invaluable inputs during various stages of my work. I am grateful to have collaborated with her during my PhD. The advice and feedback I received from her were instrumental in getting my research into a publishable form.

I gratefully acknowledge funding support through assistantship and internship positions on several projects. I thank my advisors, Dr. Kifer, Dr. Alan MacEachren (Geography, Penn State), Dr. Yanjun Qi (NEC Labs) and Dr. Frank Ritter (IST, Penn State) for giving me these opportunities and letting me have one less thing to worry about. I particularly thank Yanjun for allowing me leeway to work on the metadata extraction problem during my internship and providing me valuable directions on the problem.

On a personal front, I thank my parents, Prabhudas G. and Priyadarsini PMLR., and my sisters, Asma Shaik (Prabhavathy) and Gowry Das. I could not have made it thus far without their never-wavering support, prayers, love and encouragement. I cherish the acquaintance and friendship of several people at State College. In particular, I wish to thank my labmates, Dayu Yuan, Haibin Liu, Juan Pablo Fernandez, Madian Khabsa, Pradeep Teregowda, Puck Treerapatituk, Saurabh Kataria, Shibamouli Lahir, Sumit Bhatia, Suppawong Tuarob, Wenyi Huang and Zhaohui Wu. In addition to helpful technical exchanges, their company made the lab a conducive and fun place for me to work in. A special note of gratitude goes to my friends, (Amm) Sirinda Palahan, Anusha Guippaili, Ashwini Rao, Jingjing Zhang, Medha Atre, Sue Johnston and Vanathi Duraisamy. Their best wishes, prayers and listening ears helped me get through when I hit any low points.

As I summarize my research in this thesis, I reflect on the last several years I spent at Penn State. I took a bunch of great courses, attended several inspiring talks, made a lot of friends, learnt new things, traveled around and, in a nutshell, had a blast. I thank God for his grace and providence without which none of this could have happened.
Dedicated to my father, Prabhudas G., mother, Priyadarsini, and sisters, Asma and Gowry.
Introduction

Expert search capabilities that rank researchers in response to topic and name queries are desirable in digital libraries. Consider for instance, the following use-cases: The program chair of a conference is desirous of selecting a program committee for the “Information Retrieval” track. A potential list of panel members for such a committee can be obtained by posing an appropriate topic query to an expert search system. On the other hand, consider a student applying to graduate schools who is interested in working with researchers such as “Bruce Croft”. Both these tasks involve ranking of researchers based on the documents associated with them.

Digital Library portals such as CiteSeer [2] and ArnetMiner [4] provide access to research literature related to specific disciplines. These open-access systems typically employ automated techniques to identify indexable research-related documents and to extract metadata from them. The extracted metadata is in turn used to construct author-document, citation and co-authorship networks based on which various bibliographic metrics are computed. Needless to say, the accuracy of the extracted networks and computed measures depends crucially on acquisition and error-free “parsing” of documents and webpages associated with researchers.

Researcher homepages (also referred to as academic homepages or simply homepages in this dissertation) form an important resource for obtaining, updating and tracking document collections in digital libraries. Typically, academic homepages summarize research interests, and other metadata related to researchers. This information is useful as complementary evidence in tasks such as expertise search, extraction of academic networks, author profile extraction and disambiguation [5, 4]. A classification module is typically employed for identifying homepages from other types of pages on the web. We address researcher homepage finding in two scenarios:

- Given a researcher name, how can we find the correct homepage on the Web using the name as a search engine query?
- Given lists of hub websites where homepages are typically hosted, how can we acquire a high-quality collection of researcher homepages?
Metadata listed on researcher homepages can be used to disambiguate authors with the same name and to integrate across multiple sources of information [6, 4]. Information Extraction (IE) techniques are employed for tagging tokens on a homepage that correspond to metadata. We study “feature labeling”, a new form of semi-supervised learning, proposed by Druck, Mann and McCallum for the homepage tagging task [7]. Designer-specified labeled features imposed by learning frameworks such as generalized expectation [8] and posterior regularization [9] were shown to significantly reduce the need for fully-annotated instances for classification and tagging problems. To avail the benefits of these advancements, we address the questions: What types of labeled features can be designed for the homepage annotation task? Can we automate the extraction of these labeled features? Do automatically extracted labeled features perform on par with “designed-specified” labeled features on the tagging task?

Both “expert search” and “similar entity finding” are well-studied tasks at the Text REtrieval Conference (TREC) [137]. The expert search task at TREC targets the retrieval of people (experts) in an enterprise in response to topic queries using evidence from intranet pages and other documents available in an enterprise. On the other hand, “similar entity finding” or the “list completion task” (also studied in the INEX competition [150]) pertains to retrieving entities (e.g. cars, books, actors) based on a set of exemplar entities and a brief textual description. In digital libraries, both these tasks translate to ranking of researchers with research documents providing the evidence for expertise.

We argue that documents in the academic domain are different from webpages in terms of their type (e.g. homepages, publications), structure (e.g. abstract, sections), associated metadata (e.g. venue, authors) and connections (e.g. citations). Hence, models that account for these special features are likely to be more beneficial. Secondly, given that researcher ranking is essential to both tasks, scoring techniques that handle both the query types uniformly are desirable. Based on these conjectures, we address the following questions:

- How can we design models for ranking researchers using multiple sources of information available in a digital library?
- Can we design uniform scoring schemes to handle both the query types (name and topic)?

### 1.1 Contributions and organization

We summarize our contributions that address the questions raised so far:

- We present a comprehensive study on features that distinguish academic homepages from other webpages. Using Latent Dirichlet Allocation [10], we model the content of academic homepages in terms of topic mixtures and investigate what categories of information are “useful” from the perspective of a digital library (Chapter 3). Insights from these findings are used for learning ranking models for academic homepage retrieval on the Web.

- We address homepage identification in context of crawling hub websites such as university
department sites. We show that the available datasets for this task do not correspond to operating environments of the crawler resulting in an unacceptable yield of homepages. Instead of labeling new datasets to handle this problem, we show that a previously-learnt classifier can be adapted to new environments using multiple views of training instances and unlabeled data (Chapter 4).

- We frame metadata extraction from homepages as a tagging task and use the recent advancements in learning with feature labeling frameworks for solving this task (Chapter 5). We show that domain knowledge pertaining to affiliations between terms and metadata fields and layout conventions can be effectively captured via labeled features. Furthermore, models that take the labeled features into consideration during training not only provide significant improvements in the tagging performance but also reduce requirements on fully-annotated examples required to train models.

- Finally, we present structure-based scoring of query-dependent graphs for enabling expertise search in digital libraries. We show that multiple sources of evidence can be effectively harnessed by extending the PageRank equations to work with multiple transition matrices. For content-based scoring, we propose scoring schemes using Author-Document-Topic graphs. Our graph-based models provide unified techniques for handling both topic-based and name-based queries and at the same time perform on par with models specific to particular query-types (Chapter 6).

Some future directions for research based on the findings in this dissertation are summarized in Chapter 7 before concluding in Chapter 8.
For ease of reference, we describe some commonly used abbreviations, evaluation metrics and software implementations that we refer to in the rest of the thesis.

1. The Text REtrieval Conference (TREC) is an on-going series of annual workshops that evaluate various tasks in information retrieval. These workshops, started in 1992, are co-sponsored by the National Institute of Standards and Technology (NIST) and the Intelligence Advanced Research Projects Activity. Details related to the participating systems and the various tracks can be found at http://trec.nist.gov/. TREC tasks typically deal with enterprise and web related search scenarios. We contrast the differences between the scenarios covered by systems in TREC and academic search scenarios where applicable.

2. DBLP is a computer science bibliography website accessible at http://www.informatik.uni-trier.de/~ley/db/. DBLP tracks all the proceedings of important journals and conferences in Computer Science and listed more than 2.1 million articles on computer science in November 2012. We obtain researcher homepage information that is used in the creation of our datasets from DBLP.

3. CiteSeer/CiteSeerx is an open-source digital library portal for scientific and academic papers, in the fields of computer and information science. CiteSeer is widely considered the first search engine for academic paper search and a predecessor to Google Scholar and Microsoft Academic Search. CiteSeer is hosted at Penn State and is accessible at http://citeseerx.ist.psu.edu/index. While the problems addressed in this thesis are in the context of CiteSeer, and some of our experiments were performed using the document collections and author information from CiteSeer, we believe that our findings are

---

1http://scholar.google.com/
2http://academic.research.microsoft.com/
more generally applicable to other comparable systems [2].

4. ArnetMiner is a system for searching academic social networks in Computer Science. It has been in operation for more than three years with most of its algorithmic published in peer-reviewed conferences [4, 11]. The ArnetMiner system provides datasets for evaluating metadata extraction and expertise search for researchers in Computer Science. We used these datasets in our experiments. The ArnetMiner system is accessible at http://arnetminer.org/.

2.1 Evaluation measures

The performance of classification and tagging is typically evaluated using metrics such as precision, recall and F1 on a test dataset [12]. For binary classification with class labels, 0 and 1, let $f_{ij}$ represent the number of test instances, with correct label, $i$ and predicted label, $j$ where $i, j \in \{0, 1\}$. Then for class $i$,

$$
\text{precision} = \frac{f_{ii}}{f_{ii} + f_{ij}}, \quad \text{recall} = \frac{f_{ii}}{f_{ii} + f_{ji}}, \quad \text{F1} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}.
$$

For multi-class labeling and tagging, when computing the values for a given class $i$, $j$ represents all other classes except $i$. Traditionally, the metrics are computed over all the classes in question and weighted or unweighted averages are used for obtaining a single measure.

While evaluating retrieval models, where ranked lists of (search) results are involved, metrics that account for the position at which correct answers appear are additionally used. Let $R_q$ represent the set of relevant results for a given test query, $q$. If $S$ represents the set of results retrieved by the system for $q$, the recall and precision for $q$ are computed using:

$$
\text{Recall} = \frac{|S \cap R_q|}{|R_q|}, \quad \text{Precision} = \frac{|S \cap R_q|}{|S|}.
$$

Average precision (AvgP) refers to the average precision with $S$ after each relevant document is retrieved whereas MAP (mean average precision) aggregates the average precision value over all the queries ($Q$) to provide a single measure for precision. The mean reciprocal rank, in contrast, values the rank at which the first correct answer is found for every query in $Q$. These measures are computed using the following formulae:

$$
\text{MAP} = \frac{\sum_{q=1}^{Q} \text{AvgP}(q)}{|Q|}, \quad \text{MRR} = \frac{1}{|Q|} \sum_{q=1}^{Q} \frac{1}{\text{rank}(q)}.
$$

The term rank($q$) refers to the position of the first relevant answer in the ranked list of results found for a query, $q$. Queries for which no relevant experts are found are considered to contribute 0 to the formula for computing MRR. When only the top-$k$ predictions in $S$ are considered for computing these measures, we append @$k$ while referring to them (e.g. Recall@10). More details
on these evaluation measures can be found in standard IR textbooks (for example, [13]).

2.2 Software

- Depending on the requirement, we use either SVM$^\text{light}$ [147] or LibSVM [146] for learning classifiers based on Support Vector Machines [14]. Both these implementations are written in C, fast, and commonly used for text classification problems. However, some features are not common to both. For instance, ranking models are implemented in SVM$^\text{light}$ via RankSVM whereas LibSVM provides the implementation for one-class classification.

- Weka [148], a java-based package provides the implementation for several machine learning algorithms. We used the classification algorithms and feature selection techniques implemented in Weka for our experiments.

- MALLET provides the implementation for maximum entropy classifiers and conditional random fields. Mallet is a Java-based machine learning toolkit provided by Andrew McCallum’s group at UMass [15]. Mallet also includes several text classification tools including topic models and semi-supervised learning methods for both classification and tagging.

- We use the search engine Indri [149] to index document collections. Indri was released as part of the Lemur project; a cooperative effort between the University of Massachusetts and Carnegie Mellon University. Indri uses language modeling techniques and inference networks for ranking documents in response to various free-form and structured queries. Indri also provides facilities for query expansion and was shown to perform competitively on various tasks at TREC [137].
Chapter 3

Academic homepage retrieval

3.1 Introduction

Navigational queries where the search intent is to find specific websites corresponding to entities such as persons, organizations and products constitute a large proportion of search engine traffic [16, 17]. Since homepages typically comprise the target of navigational queries, homepage finding was extensively addressed as part of TREC.

Ranking people’s homepages in response to name queries poses significant challenges. Several people with the same name can have homepages requiring extra information for disambiguation. For example, when we specify the query “Michael Jordan”, the name of a famous Computer Science researcher, on Google (search performed on February 1, 2011), his homepage appeared at the 15th position in the search results. In this search experiment, there were several pages related to the basketball player in the remaining results.

Several webpages, apart from a homepage, can be associated with a particular person. These webpages may be found with a name query on the Web. For most researchers in Computer Science, it is common to see pages associated with the researcher from websites such as Wikipedia, DBLP, LinkedIn, and Amazon ranked higher than the academic homepages in web search results using name queries. In most cases, this ranking can be significantly improved by specifying additional information. For instance, the previous “Michael Jordan” query can be augmented by keywords such as “Berkeley” or “Computer Science” to improve the rank at which the corresponding homepage appears in the search results. However, this query-specific augmentation requires additional information that is not always available a priori.

We show that when only name information is available, using content-based similarity between the name and the webpages for ranking webpages performs poorly for academic homepage retrieval. Our goal is to study ranking techniques such that given a researcher name query, academic homepages are ranked higher (appear in earlier positions) than other kinds of webpages. We posit that an accurate academic homepage retrieval engine requires addressing two questions:
What constitutes an academic homepage? Can we extract specific properties of homepages that can be used to design a ranking function for homepage retrieval on the Web?

Contributions: We study the problem of ranking academic homepages on the Web in response to researcher name queries. First, we extract several content-based and HTML structure-based features from academic homepages that can be used to discriminate them from other webpages. The content-based features are extracted using Latent Dirichlet Allocation [10, 18, 19] on a corpus of known academic homepages from DBLP [142] whereas previously studied features in webpage genre identification [20] are extended for deriving HTML structure and URL-based features common to homepages.

We express the homepage retrieval problem using the RankSVM formulation [21]. The proposed content and structural features are used in this formulation to learn a ranking function for identifying homepages. Our ranking function obtains significant improvements with respect to success rate and the mean reciprocal rank measures, and beats both the retrieval method using language modeling and inference networks and a second method that uses query expansion.

Section 3.2 briefly describes previous works closely related to our contributions. In Section 3.3 we describe academic homepages in more detail and provide motivation for analyzing them using topic modeling tools. Section 3.4 provides an overview of the features we used in training our ranking functions and the baselines whereas Section 3.5 provides details on our experimental setup, results and observations. Finally, we conclude this chapter with directions on future work in Section 3.6.

3.2 Related work

The homepage finding task at TREC though not specifically focused on people’s homepages served as a platform for addressing homepage finding in a systematic manner and provided a dataset for evaluation. Several researchers employed machine learning techniques for addressing homepage finding using query-independent features such as URL-type (com, edu etc.) and PageRank along with features based on (query, content) and (query, anchor-text) similarity [22, 17, 23].

Academic homepage identification was addressed by Tang, et al. for mining researcher profiles and integrating with the publication data in ArnetMiner [4]. Tang, et al. address homepage finding as a binary classification task and use URL-based features on the pages retrieved using Google with researcher name queries. Japanese word features (from a manually-assembled dictionary) on the page and the surrounding pages that link to it were used by Wang, et al. for collecting homepages from university websites in Japan [24]. Macdonald, et al. address homepage finding with the goal of building candidate profiles that are used to rank experts [25]. They identify candidate homepages using the anchor-text based features, since the anchor text pointing to a homepage usually contains the candidate’s name.

Incorporation of relevance judgements or user clicks while ranking webpages is a common objective for “learning to rank” methods [26]. Several algorithms for learning to rank were
proposed in the last decade including RankSVM [21], RankNet [27] and PermuRank [28]. Each of these algorithms targets different aspects of the learning to rank problem. For instance, RankNet considers the differences in number of results associated with each query in the training data, an issue that can potentially affect the learning process if not addressed. An essential component to all these methods is the design of features based on the task at hand.

We study the learning to rank problem for a homepage retrieval engine. The input queries are assumed to be “researcher name” queries and we focus on designing features that can be potentially used with any of the learning to rank algorithms. We use content features obtained using Latent Dirichlet Allocation [10] and adapt HTML and URL features previously used in webpage genre identification for our problem [29, 20].

3.3 Content analysis

Academic homepages present a person’s academic standing and current activity. Academic homepages are a valuable resource for digital library portals such as CiteSeerX and ArnetMiner that provide access to collections of research literature and associated authors. Backend tasks of such portals include crawling for new literature, metadata extraction from research documents and document classification.

Academic homepages of authors serve as a resource that can help in several digital library tasks. For instance, a crawler can potentially obtain information related to an author’s publications from her homepage. Similarly, metadata available from homepages such as e-mail information or affiliation can help in disambiguating author names [6]. We use topic modeling tools to empirically prove that academic homepages indeed provide the content of interest to digital libraries. In the later sections, we use insights from these experiments to study homepage retrieval.

3.3.1 Homepages as topic mixtures

Latent topic mixture models posit that a document can be viewed as a mixture of a small number of latent topics and that each ‘observed’ word in the document can be attributed to one of these topics [30, 18, 19]. The process of homepage creation by its author can be visualized in a fashion similar to that of the document generation process in topic models such as Latent Dirichlet Allocation or LDA [10] where topics pertain to categories of information such as “research activity”, “contact information”, “teaching activity”, etc. Depending on the person’s preference, a given homepage may contain more information related to one category (topic) than the others. Moreover, the layout or position of each category of information may vary based on personal preferences. However, given that all these webpages pertain to “academic homepages”, the choice of categories, words and layout can be expected to come from apriori corpus-level distributions. Based on this conjecture, the homepage generation process can be explained using the following steps:
1. Sample a mixture proportion of topics. Each topic corresponds to a specific category of information such as contact information and publications.

2. For each of the $N$ terms in the homepage
   
   (a) Sample a topic for that position
   
   (b) Sample a word conditioned on the chosen topic

The details of LDA including the plate notation, sampling equations and the estimation process can be found in the references [10, 18, 19]. Previous research using LDA has shown its effectiveness as an unsupervised tool for analyzing text corpora. We now describe some quantities that LDA estimates from a collection of documents since they are used in the later chapters.

Based on co-occurrence counts in the corpus, LDA estimates a topic-term association matrix, $\phi$. The entries in this matrix corresponds to predictive distributions of words given topics, that is, $\phi_{w,i}$ is the probability of a word $w$ given the topic $i$. After an LDA run, every term in the document is randomly assigned a topic based on these probabilities. These assignments are used to express the document as a mixture of topics. $\theta_d$ refers to the topic distribution vector of length $K$ for document $d$, where $K$ is the number of topics (a parameter while running LDA). The component $\theta_{d,i}$ is the smoothed proportion of times topic $i$ was assigned to the terms in $d$.

From an analysis standpoint, the top words for a given topic (words with high probability values for that topic from $\phi$) are useful in understanding the underlying theme captured by that topic.

<table>
<thead>
<tr>
<th>talk</th>
<th>page</th>
<th>students</th>
<th>member</th>
</tr>
</thead>
<tbody>
<tr>
<td>slides</td>
<td>home</td>
<td>graduate</td>
<td>program</td>
</tr>
<tr>
<td>invited</td>
<td>publications</td>
<td>faculty</td>
<td>committee</td>
</tr>
<tr>
<td>part</td>
<td>links</td>
<td>research</td>
<td>chair</td>
</tr>
<tr>
<td>talks</td>
<td>contact</td>
<td>cse</td>
<td>teaching</td>
</tr>
<tr>
<td>tutorial</td>
<td>personal</td>
<td>student</td>
<td>board</td>
</tr>
<tr>
<td>seminar</td>
<td>list</td>
<td>undergraduate</td>
<td>editor</td>
</tr>
<tr>
<td>summer</td>
<td>updated</td>
<td>college</td>
<td>courses</td>
</tr>
<tr>
<td>book</td>
<td>fax</td>
<td>current</td>
<td>state</td>
</tr>
<tr>
<td>introduction</td>
<td>email</td>
<td>ph</td>
<td>activities</td>
</tr>
<tr>
<td>chapter</td>
<td>department</td>
<td>school</td>
<td>technical</td>
</tr>
<tr>
<td>group</td>
<td>interests</td>
<td>program</td>
<td>associate</td>
</tr>
<tr>
<td>workshop</td>
<td>phone</td>
<td>university</td>
<td>special</td>
</tr>
<tr>
<td>lectures</td>
<td>info</td>
<td>grant</td>
<td>education</td>
</tr>
<tr>
<td>presentation</td>
<td>homepage</td>
<td>news</td>
<td>present</td>
</tr>
</tbody>
</table>

**Table 3.1.** Top words from topics related to homepages

Table 3.1 shows the top words of topics indicative of homepages obtained by running LDA on known homepages from the DBLP dataset (described in Section 3.5). The top words shown in Table 3.1 are indicative of themes such as “contact information” (second column) and “professional activity” (last column), validating our speculation about categories of information on a homepage. Another group of topics identified with LDA on DBLP pertains to subject areas. Some of these topics are shown in Table 3.2. These topics are evidence that information related
to “research interests” or “area of work” commonly appear on homepages. However, the exact nature of these topics is more an artifact of the dataset. Since the DBLP dataset is composed of homepages of researchers in Computer Science, the words in this table show topics corresponding to databases, multimedia, distributed systems and circuit design, the sub-fields in Computer Science.

### 3.4 Features for ranking homepages

We use the insights obtained from Section 3.3 to design design features for our ranking function. Recall from our content analysis that certain categories of information are likely to appear more often than not on a researcher homepage. For instance, name and affiliation information, educational background and current research activities are commonly seen on a researcher homepage.

We obtain the top words of the topics pertaining to homepages to form our feature dictionary of terms (Table 3.1). We avoid the topic-term clusters that reflect subject areas (Table 3.2) to avoid features that discipline-specific. This was done by manual examination of the LDA output. Note that, isolating discipline-independent features in other feature selection methods such as mutual information is not easy due to the lack of terms groups as in output from LDA. Automatic identification of discipline-specific topics is desirable and a direction for future study.

### 3.4.1 Structural properties of homepages

Our next set of observations pertain to the HTML structure and URLs of homepages. Though it is not always the case, it is common for homepages to be hosted on the institution domains of researchers. In addition, homepage URLs that belong to a certain university sometimes follow naming conventions. For example, a string containing a tilde followed by author’s lastname after the department URL. URL conventions are less consistent across institutions as opposed to the HTML structure-based features that we describe next.

A common convention observed in academic homepages is that authors tend to put their names, sometimes coupled with the word ‘homepage’ in the title tag of the HTML. Similarly, it is
fairly uncommon for academic homepages to contain several tables or images embedded in them. Table 3.3 captures the list of URL, HTML and query dependent features we used for training our ranking function. The first three features in this table were used previously in ArnetMiner for obtaining the homepages of researchers from Google search results [4]. We show later that these features are indeed rather effective for homepage retrieval but large improvements are still possible with our full list of content-based and structural features. Some of the features in Table 3.3 are homepage-specific variations of features previously used in webpage genre identification [29, 20].

<table>
<thead>
<tr>
<th>Feature Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 name match in HTML title</td>
</tr>
<tr>
<td>2 name match in HTML content</td>
</tr>
<tr>
<td>3 name match in url</td>
</tr>
<tr>
<td>4 depth of url</td>
</tr>
<tr>
<td>5 tilde in url</td>
</tr>
<tr>
<td>6 domain of the url (com, edu, ac or other)</td>
</tr>
<tr>
<td>7 term home in HTML title</td>
</tr>
<tr>
<td>8 number of links</td>
</tr>
<tr>
<td>9 number of links in the same domain</td>
</tr>
<tr>
<td>10 ratio of 8 and 9</td>
</tr>
<tr>
<td>11 number of images</td>
</tr>
<tr>
<td>12 number of tables</td>
</tr>
<tr>
<td>13 number of links to pdf files</td>
</tr>
<tr>
<td>14 links with anchor text containing research/publications</td>
</tr>
</tbody>
</table>

Table 3.3. Structural features used in homepage ranking

3.4.2 Researcher homepage retrieval

Our goal is to provide a re-ranking module on top of the results of a standard search engine (such as Google), that ranks academic homepages of researchers better than other webpages. Thus, as opposed to the generic search scenario, the target-type of our search (viz. researcher homepage) is known in advance. We explore the use of features summarized in the previous section for learning a ranking function. Although we focus on using page-specific features, features external to the webpage such as PageRank or anchor text content can be folded into the ranking module when they are available. Content-based ranking methods typically use features based on match with the query for scoring documents. We show that using these features alone does not result in effective homepage ranking.

The Indri search engine (Section 2.2) uses a combination of inference networks and language modeling for content-based retrieval. The retrieval function in Indri has been shown to be effective in a wide range of retrieval tasks at TREC. In addition to being a competitive baseline that scores documents based on query-document match, Indri provides the pseudo-relevance feedback feature as well as the provision to specify expanded queries [31, 32].
For learning the ranking function, we use the RankSVM model proposed by Joachims [21]. During training, the RankSVM learns a retrieval function that maximizes the Kendall’s \( \tau \) measure based on the partial order information present in training examples. Kendall’s \( \tau \) is related to the Average Precision measure commonly used in ranked retrieval. It is intuitive to view a homepage, non-homepage pair as a preference pair and minimize the number of inversions in the training set while learning the ranking function. In our experiments, we compared the following sets of features for learning a function to rank homepages:

1. **Topic Proportions of Documents (tprop)** LDA embeds each document in the corpus as a vector in the topic space. We use the proportions of topics that pertain to homepages as features.

2. **Word Features (LDA-words)** After identifying the topics from LDA that relate to homepages, we use the top 20 words from these topics to form a word dictionary. Each document can be expressed as a vector in this word space using their term frequencies normalized by the document length. These values are used as features for training the ranking function.

3. **Query-dependent Features (am)** This set comprises the features 1-3 and 7 from Table 3.3. These features were previously used by Tang, et al. for training a binary classifier that identifies academic homepages in ArnetMiner [4]. We include this set for comparison with our proposed features.

4. **All Features (LDA-words+struct)** The structural features summarized in Table 3.3 are used together with the word features in this set.

We illustrate the performance benefits of learning a ranking function for homepage retrieval by comparing with the following methods:

1. **Baseline (bl)** The ranking function in Indri using language modeling and inference networks is used as the baseline with researcher names as queries.

2. **Pseudo-Relevance Feedback (prf)** We use the pseudo-relevance feedback mechanism of Indri to rank pages in response to researcher-name queries. This is a two-step process which involves, using the top words from documents retrieved with the original query to perform query expansion. This form of expansion was found to effective in certain settings where the query is precise [33, 13].

3. **Query Expansion (qexp)** During web search, when the correct result is not retrieved using the “researcher name” as query, an expanded query containing other discriminative words can potentially lead to better retrieval. For instance, for the “Michael Jordan” example, mentioned in the introduction, one could try adding “machine learning” or “publications” in the query to filter out the results related to the basketball player. In this baseline, we use the top words from homepage-related topics (Table 3.1) to form the expanded query and use the expansion mechanism provided in Indri for retrieval.
3.5 Experiments

We use the DBLP and WebKB datasets for evaluating our features.

1. **DBLP** dataset: DBLP lists the names of computer scientists whose articles are indexed in their database. At the time of our dataset creation in 2010, DBLP listed about 769,785 author names out of which for 13,290 authors, the homepage url information was also available. From the author-provided homepages on DBLP, we were able to retrieve from the specified URLs, homepages for about 6,000 authors.

2. **WebKB**: The WebKB dataset was previously used by several researchers for studying webpage and text classification [34, 35, 36, 37]. The WebKB collection contains academic webpages from Computer Science departments of four universities: Cornell, Washington, Texas and Wisconsin, categorized into 7 categories (student, faculty, staff, department, course, project, and other). The “other” class comprises pages that cannot be fit into the remaining six classes; for instance, a publications page that links to a page belonging to a faculty page. This collection was created in 1997. For the WebKB dataset, since name information is not available, we test the effectiveness of our features in a classification setting to measure their discriminating capability.

In the DBLP dataset, since negative instances (or non-homepages) are not available and we want to duplicate the web search scenario, we adopt the following process to collect a corpus for retrieval: Each author name for which the homepage URL was specified was used as a query string to search the web with Yahoo’s BOSS API [152]. The first 20 hits from this search were scanned for the homepage URL listed in DBLP. If the homepage was found in the top 20 hits, this was marked as a positive example and the remaining 19 hits comprise the negative examples. This corpus is indexed using Indri whose native ranking function forms the baseline retrieval method.

The LDA estimation process requires setting the number of topics ($K$) value and other parameters for priors of hyperparameters. We used the default settings along with the hyperparameter optimization option available in Mallet (Chapter 2.2). For number of topics, we experimented with settings 10-200 in steps of 10 and evaluated the training data likelihood. We found the likelihood value with $K = 70$ to be among the better ones. Manually examining the top words for each topic also indicated that for $K > 70$, several words are repeated under different topics indicating that multiple topics might be covering the same theme. Therefore $K$ was set to 70 in all LDA related experiments.

We obtain the top 20 terms corresponding to the homepage-related topics to form the word dictionary with 96 terms (Section 3.3). Using a larger number of terms per topic did not improve retrieval performance. We use the normalized term counts of the terms in this dictionary for generating word features. Other representations such as boolean, raw term counts and TFIDF performed worse than the normalized term count representation. For topic proportion vector, we directly use the mixture proportions output by LDA as features. These are real values between
0 and 1 indicating the proportion of a topic in a given document. Details on term feature representations can be found in Manning, et al [13].

For the ranking experiments using DBLP we provide the results averaged across five random splits of the data. Since the names of the researchers whose homepage is present in this dataset are available, for the baseline methods, we use Indri to construct an index over the collected corpus and use the underlying search engine to check if the correct page is retrieved when the name is used as a query. For training the ranking function, for each example in the training folds, the correct homepage along with a randomly chosen example from the other pages obtained during search with this name query, forms an ordered pair that is used to create a training instance for RankSVM. The ranking function learnt on the training folds is used to re-rank the top 100 results retrieved with Indri for queries in the test fold.

The success rate and mean reciprocal rank measures are used to evaluate performance on the test folds. For an example, if the correct homepage is found among the top \( k \) results, this contributes a value of 1 to the aggregate score for “success rate \( \hat{0} @k \)” and a value of \( \frac{1}{k} \) for “mean reciprocal rank \( \hat{1} @k \)”. The aggregate score is normalized by the number of test queries. For the classification setting, we use the F1 measure to evaluate performance. More information regarding the use of these measures in ranked retrieval and classification can be found in Manning, et al [13].

### 3.5.1 Results and observations

The performance for various settings is shown in the Figures 3.1 and 3.2 and Tables 3.4 and 3.5. Since user studies indicate that people rarely browse beyond the second page of search results [38], we use Success Rate (SR) and Mean Reciprocal Rank (MRR) at \( k = 1, 3, 5, 10, 20 \) for evaluating the performance of our ranking function. We also present the F1 measure for the WebKB dataset.

As shown by the numbers in these tables, compared to the baseline scoring techniques used typically by a custom search application, large improvements can be obtained by using ranking

![Figure 3.1. Mean reciprocal rank for different methods: bl: baseline, prf: pseudo-relevance feedback, qexp: query expansion, am: ArnetMiner features, tprop: topic proportions, LDA-words: words from LDA, LDA-words+struct: words from LDA and structural features](image)
functions trained with the knowledge of the expected target page. Pseudo-relevance feedback was not found to be useful but query expansion using the LDA word features is useful. Pseudo-relevance feedback choses query expansion terms based on term frequencies in the top retrieved documents irrespective of their type (homepage versus non-homepage). On the other hand, LDA terms were chosen from topics indicative of homepages. Specifically training a ranking function

### Table 3.4. Mean reciprocal rank for different methods: bl: baseline, prf: pseudo-relevance feedback, qexp: query expansion, am: ArnetMiner features, tprop: topic proportions, LDA-words: words from LDA, LDA-words+struct: words from LDA and structural features

<table>
<thead>
<tr>
<th>Method</th>
<th>MRR@1</th>
<th>MRR@3</th>
<th>MRR@5</th>
<th>MRR@10</th>
<th>MRR@20</th>
</tr>
</thead>
<tbody>
<tr>
<td>bl</td>
<td>0.0316</td>
<td>0.0588</td>
<td>0.0725</td>
<td>0.0902</td>
<td>0.1046</td>
</tr>
<tr>
<td>prf</td>
<td>0.0344</td>
<td>0.0604</td>
<td>0.0727</td>
<td>0.0884</td>
<td>0.1021</td>
</tr>
<tr>
<td>qexp</td>
<td>0.0449</td>
<td>0.0786</td>
<td>0.0974</td>
<td>0.1214</td>
<td>0.1360</td>
</tr>
<tr>
<td>am</td>
<td>0.1540</td>
<td>0.1937</td>
<td>0.2075</td>
<td>0.2207</td>
<td>0.2263</td>
</tr>
<tr>
<td>tprop</td>
<td>0.1377</td>
<td>0.2042</td>
<td>0.2235</td>
<td>0.2377</td>
<td>0.2441</td>
</tr>
<tr>
<td>LDA-words</td>
<td>0.1591</td>
<td>0.2285</td>
<td>0.2505</td>
<td>0.2664</td>
<td>0.2736</td>
</tr>
<tr>
<td>LDA-words+struct</td>
<td><strong>0.2128</strong></td>
<td><strong>0.3106</strong></td>
<td><strong>0.3364</strong></td>
<td><strong>0.3516</strong></td>
<td><strong>0.3559</strong></td>
</tr>
</tbody>
</table>

### Table 3.5. Success rate for different methods: bl: baseline, prf: pseudo-relevance feedback, qexp: query expansion, am: ArnetMiner features, tprop: topic proportions, LDA-words: words from LDA, LDA-words+struct: words from LDA and structural features

<table>
<thead>
<tr>
<th>Method</th>
<th>SR@1</th>
<th>SR@3</th>
<th>SR@5</th>
<th>SR@10</th>
<th>SR@20</th>
</tr>
</thead>
<tbody>
<tr>
<td>bl</td>
<td>0.0316</td>
<td>0.0965</td>
<td>0.1573</td>
<td>0.2955</td>
<td>0.5033</td>
</tr>
<tr>
<td>prf</td>
<td>0.0344</td>
<td>0.0970</td>
<td>0.1517</td>
<td>0.2745</td>
<td>0.4740</td>
</tr>
<tr>
<td>qexp</td>
<td>0.0448</td>
<td>0.1262</td>
<td>0.2094</td>
<td>0.3946</td>
<td>0.6023</td>
</tr>
<tr>
<td>am</td>
<td>0.1540</td>
<td>0.2456</td>
<td>0.3066</td>
<td>0.4058</td>
<td>0.4839</td>
</tr>
<tr>
<td>tprop</td>
<td>0.1377</td>
<td>0.2925</td>
<td>0.3777</td>
<td>0.4840</td>
<td>0.5757</td>
</tr>
<tr>
<td>LDA-words</td>
<td>0.1591</td>
<td>0.3203</td>
<td>0.4175</td>
<td>0.5356</td>
<td>0.6377</td>
</tr>
<tr>
<td>LDA-words+struct</td>
<td><strong>0.2128</strong></td>
<td><strong>0.4388</strong></td>
<td><strong>0.5512</strong></td>
<td><strong>0.6631</strong></td>
<td><strong>0.7225</strong></td>
</tr>
</tbody>
</table>

Figure 3.2. Success rate for different methods: bl: baseline, prf: pseudo-relevance feedback, qexp: query expansion, am: ArnetMiner features, tprop: topic proportions, LDA-words: words from LDA, LDA-words+struct: words from LDA and structural features
using RankSVMs with these features is more effective than the straightforward query expansion.

The query-dependent URL features used for classifying webpages in ArnetMiner are indeed good indicators of homepages. However, both the topic proportion and word features individually yield larger improvements than using the ArnetMiner features alone. Word features are better than the topic proportion features but combining both these sets does not improve results (not shown). This is not surprising since the topic proportion vector can be viewed as an embedding of the webpage from the higher-dimensional word-space into a lower-dimensional topic space. Word features that are fine-grained were found to be more beneficial than aggregated topic proportion values for our ranking problem. The best performance was obtained when the structural features are used together with the word features. Indeed, compared to the Indri baseline, using a ranking function trained with the LDA-words+struct features gives a significant improvement in the success rate from 3.2% to 21.28% at rank position 1 and from 29.55% to 66.31% at rank position 10.

Other combinations such as using expanded queries with structural features did not beat the performance with the LDA-words+struct features. The margin trade-off parameter ‘C’ [14] in RankSVM was set to 0.01 in the experiments. We experimented with other values of C between 0.005 to 0.025 but found no considerable difference in the performance.

For the WebKB dataset, we evaluate classification performance using a classifier trained on the DBLP dataset with word and structural features. Only query-independent features were used since queries (researcher names) are not available for this set. The objective of this experiment is to evaluate the robustness of our features in identifying homepages in the absence of names. This set up is common while building a focused crawler for accumulating a homepage collection in absence of names. We obtained an F1 score of 0.5738 on the WebKB dataset. We found decision trees to be the better performing among logistic regression, two-class and one-class SVMs we tried for this task. In total, our set of features is small (about 125, LDA words and features in Table 3.3). Previously, using thousands of unigram and bigram features, average accuracies between 70 – 90% were reported for WebKB [39].

3.6 Summary

We studied homepage retrieval for researcher entities on the Web. To address this problem we use various content-based and structural aspects of researcher homepages based on topic analysis and learn a ranking function via RankSVM. The effectiveness of our features was experimentally illustrated on two datasets. From the perspective of homepage finding on the Web using researcher name queries, we point out that in contrast to humans looking for researcher homepages where it might be sufficient to have the homepage among the top 10 results, acquisition of a collection of homepages for use in a digital library requires the choice of small number (one or two) of webpages for every researcher name. Since the performance of other tasks such as metadata extraction depends crucially on using the correctly mapped homepage for a given researcher, high accuracy is desirable.
Chapter 4

Identifying homepages on academic websites

4.1 Introduction

Digital library systems such as CiteSeer [139], ArnetMiner [140], and Google Scholar [138] are primarily interested in obtaining and tracking researchers’ homepages in order to retrieve appropriate scientific research publications and to allow users to browse information related to an author. We study homepage acquisition on academic websites with the help of a classification module for identifying researcher homepages while crawling academic websites.

Given the infeasibility of collecting the entire content of the Web, a focused crawler aims to minimize the use of network bandwidth and hardware by selectively crawling only pages relevant to a (specified) set of topics [40]. A key component for such a crawler is a classification module that identifies whether a webpage being accessed during the crawl process is potentially useful to the collection. For digital libraries, the “yield” of such crawlers highly depends on the accuracy of researcher homepage classification.

Supervised methods for learning homepage classifiers rely on the availability of large amounts of labeled data. A widely used dataset for webpage classification is the WebKB dataset [143]. Due to recent changes in the information content on academic websites, this dataset collected in 1997 is becoming outdated. For example, there are now pages on academic websites that are related to various activities such as invited talks, news, events that do not occur in the WebKB dataset. We refer to university, department and research center websites as “academic websites” in this dissertation. Compared to few decades back, it is easier now to find faculty information, links to their homepages, information on research groups, course related notes and documents, and research papers from academic websites. Similarly, job postings, seminar announcements and notices are also being uploaded onto departmental websites in recent times [41].

How can a homepage classifier keep up in the face of rapidly changing types of pages on
the Web? Specifically, given a classifier that identifies homepages with reasonable accuracy (as measured on the training datasets), how does it perform in the potentially different deployment environment? Semi-supervised methods that can exploit large amounts of unlabeled data together with limited amounts of labeled data for learning accurate classifiers have received significant attention in recent research in machine learning due to the fact that labeling examples for any supervised learning problem requires intensive human labor [37].

Against this background, one question that can be raised is: Can we design techniques to effectively adjust the previously-trained classifier to the changed content on the Web, while minimizing the human effort required for labeling new data, and under what conditions, can such an adjustment be possible?

Contributions and Organization. We present two approaches to researcher homepage classification using unlabeled data readily available from academic websites. More precisely, we first adapt the well-known co-training approach [34] to reflect the change in the data distribution over time. Second, we design an iterative algorithm based on the Mini-batch Gradient Descent technique for learning a conforming pair of predictors using two different views of the data. We restrict ourselves to homepages of researchers in Computer Science we only have training datasets for this discipline. To the best of our knowledge, the problem of researcher homepage classification using unlabeled data available during focused crawling was not addressed in previous research.

The contributions in this chapter are as follows:

- We show that with the classifiers trained on existing datasets for researcher homepage classification we incorrectly identify pages of types not seen in the training datasets as homepages. Consequently, a focused crawler mis-classifies fetched webpages and results in an unacceptable yield in crawls.

- We design novel features based on URL surface patterns and terms to complement term and HTML features extracted from the content of homepages and show that these two sets of features can be treated as two “views” for a researcher homepage instance.

- We show that the URL and content-based views can be used successfully in a co-training setup to adapt classifiers to the changing academic environments using unlabeled data. This finding enables us to accurately crawl the new academic website content without having to label a new dataset for re-training the classifier.

- Inspired by the success of co-training on this problem, we investigate loss functions that capture the disparity between the classifiers’ predictions on the two views afforded by co-training. We design an iterative algorithm based on the Mini-batch Gradient Descent technique for minimizing this loss and learning a conforming pair of predictors with the two views.

- Finally, we show that minimizing our proposed loss function on unlabeled data closely corresponds to the effect demonstrated by co-training techniques. We posit that this loss
can, therefore, be used as a measure, for tracking the progress of co-training schemes even in the absence of a validation dataset.

We focus on the design of accurate approaches for researcher homepage classification, with a future objective of integrating it into a focused crawlers in digital libraries such as CiteSeer and ArnetMiner. In these usage environments, since maintaining up-to-date collections of research literature is of primary importance, having an accurate list of homepage URLs for frequent, periodic tracking is both feasible and scalable, compared to examining the entire content at academic websites each time.

4.2 Related work

Researcher homepage classification is a well-studied webpage classification problem in context of digital libraries such as CiteSeer [2] and ArnetMiner [4]. Typically, content-based term features and HTML structure-based features are used for classifying webpages [42]. We propose the use of URL features as additional evidence for homepage identification. A smaller set (compared to ours) of URL-based features (presence of part of the name, presence of the character ‘∼’, and so on), was used in isolating homepages among the search engine results for researcher name queries by Tang, et al [4]. URL-based features are widely used in tasks related to the Web. For example, URL strings were used for extracting rules to solve the webpage de-duplication problem [43]. Shih and Karger [44] used URL features, the visual placements of links in the referring pages to a URL for improving applications such as ad-blocking and recommendation, whereas a preliminary study by Kan and Thi [45] illustrates the use of URLs in performing fast webpage classification.

The problem of collecting a high-quality researcher homepage collection was studied for Japanese websites by Wang, et al. using on-page and anchor text features [24]. Tang, et al. studied homepage acquisition from search engine results using researcher names as queries [4]. In contrast, we seek to apply focused crawling using a seed list of academic websites (where researcher homepages are typically hosted) to acquire such a collection. Focused crawling first proposed by Bra, et al. is a rich area of research on the Web [46, 47]. Chakrabarti, et al. [40] present a discussion on the main components involved in building a focused crawler. Although focused crawling is our motivating application, our contributions deal with the classifier component of the crawler and not with the crawler itself.

We show that the focused crawling scenario presents novel challenges in using a pre-trained homepage classifier in identifying relevant pages. Specifically, the classifier needs to be attuned to the changing types of pages on the Web. Co-training is proposed as a solution for addressing this challenge for homepage classification. Blum and Mitchell [34] first proposed co-training, an approach for semi-supervised learning when the number of labeled examples available for training is limited, and when training instances have two independent sets of features often referred to as “views”. For examples, in webpage classification instances can be represented using two sets of features afforded by content and anchor-text. Follow up research has provided studies related to when co-training is applicable to a problem [48, 49, 50].
Co-training is a special case of multiview learning, that involves learning with instances having multiple views. In multiview learning, models are typically learnt by maximizing “consensus” or agreement among the different views [51, 52, 53]. Most solutions to multiview learning tend to frame the problem in terms of a global optimization problem and simultaneously learn classifiers for all the underlying views. In some cases, the solutions depend on underlying classification algorithm used [54, 55]. Although our proposed algorithm based on mini-batch gradient descent seeks to maximize consensus as well, our approach is a generic technique assuming only that the underlying classifiers output initial “parameter vectors” that are altered using a simple, iterative algorithm.

4.3 Features for homepage classification

“Bag of words” approaches are frequently used in webpage or text classification [13]. Specifically, the frequently occurring and discerning terms are collected from training data to form a feature dictionary that is used to represent instances as normalized term frequency or TFIDF vectors [13]. Homepage classification was previously studied as a text classification problem using term features [56, 36]. Previous work on the same problem also used other content-based features related to the HTML structure of the page such as the number of images/tables on the page, and the terms commonly found in anchor text of homepages (Chapter 3). In this study, we extracted content-based and URL-based features from our training sets. These features and the size of feature sets are summarized in Table 4.1. The term dictionaries contain terms that occur in at least three documents (i.e., webpages) and at least five times in the training set.

In addition to term dictionaries, we hypothesize that the URL strings of homepages can provide additional evidence for identifying homepages. Hence, we design novel URL-based features based on surface patterns and presence in WordNet [154]. The URL-based features are explained in the next subsection.

<table>
<thead>
<tr>
<th>Type of features</th>
<th>#features</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Content-based</strong></td>
<td></td>
</tr>
<tr>
<td>Top unigrams</td>
<td>18674</td>
</tr>
<tr>
<td>Number of tables/links/images on the page</td>
<td>3</td>
</tr>
<tr>
<td>Unigrams from anchor text on the page</td>
<td>30</td>
</tr>
<tr>
<td><strong>URL-based</strong></td>
<td></td>
</tr>
<tr>
<td>Top unigrams and bigrams from URL strings, surface pattern and wordnet features</td>
<td>1039</td>
</tr>
</tbody>
</table>

Table 4.1. Feature types and the size of feature sets used in homepage identification.

4.3.1 URL strings as additional evidence

The idea of using URL strings in academic homepage identification comes from an error analysis of a crawl obtained with the content-based classifier. Consider some example URLs we encountered in our crawl listed in Table 4.2.
Table 4.2. Example URLs with partial sets of extracted features (shown on the next line after each URL)

<table>
<thead>
<tr>
<th></th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><a href="http://www.cs.columbia.edu/robotics/projects/visual_control/allen-realtime.html">www.cs.columbia.edu/robotics/projects/visual_control/allen-realtime.html</a></td>
</tr>
<tr>
<td>2</td>
<td><a href="http://www.cs.ucla.edu/events/events-archive/2011/limits-of-communication">www.cs.ucla.edu/events/events-archive/2011/limits-of-communication</a></td>
</tr>
<tr>
<td>4</td>
<td><a href="http://www.cs.umd.edu/~djacobs/index.html">http://www.cs.umd.edu/~djacobs/index.html</a></td>
</tr>
<tr>
<td>5</td>
<td><a href="http://www.cs.umd.edu/~djacobs/CMSC828/CMSC828.htm">www.cs.umd.edu/~djacobs/CMSC828/CMSC828.htm</a></td>
</tr>
</tbody>
</table>

With some knowledge in academic browsing, one can confidently guess that the webpages at the URLs (1), (2), and (3) are unlikely to be researcher homepages. Similarly, among the URLs (4), and (5), while the former seems to be a homepage, the latter seems to lead to a course page. The above conjectures are based on the presumption that the URL strings are not “arbitrary”, but, instead conventions are observed that are indicative of the target content at the URL. For instance, in the previous examples, words such as “projects”, “events”, alphanumeric patterns of the terms in the URL indicate that the URLs, (1), (2), (3) and (5) are most possibly not researcher homepages.

Treating “/” as delimiters, we extract features from the URL string following the domain name of a webpage. The list of all unigrams and bigrams from URL strings that occur more than thrice in the training dataset, comprise the URL-term dictionary. For terms in the URL not present in this dictionary, we look for their presence in WordNet to check if they are common words or proper nouns. WordNet is a large, lexical database of nouns, verbs, adjectives and adverbs for English, organized as a concept graph [57, 58].

In addition, we capture the surface patterns of the URLs including the presence of hyphenated or underscored words, alphanumeric patterns, long words (i.e., words having greater than 30 characters), question marks and the presence of characters such as tilde. These features are designed to filter out the URLs that commonly represent course pages, announcements, calendars and other auto-generated content. For instance, a typical homepage URL string in Computer Science departments has the name of the researcher following the ~ character after the domain name (e.g., http://people.cs.umass.edu/~mccallum/). This pattern is usually captured by our “TILDENONDICT” feature, where mccallum is a non-dictionary term. Partial sets of extracted features are shown along with the URLs listed in Table 4.2.

The above sets of features perform very well on the training datasets as shown Section 4.5. We, therefore, do not study other complicated, problem-specific feature design or feature selection. Instead our focus in this work is to study how these classifiers perform “in the wild”. We also note here that, a classifier that can make accurate predictions using URL features can be quite
beneficial from the perspective of efficiency for a focused crawler. A crawler can potentially bypass examining the content of a page if a confident decision can be made based on the URL string. However, we may not be able to always extract features from the URL strings. For instance, consider the following URLs from our crawls:

http://john.blitzer.com/
http://clgiles.ist.psu.edu/
http://ben.adida.net/

In these cases, it is not clear from the URL string that the target content refers to academic homepages. Even if complicated name-extraction based features were designed for the above cases, it is rare to find academic homepages with ‘.com’ and ‘.net’ domain suffixes. Based on the URL alone, we cannot be confident if the target content is an academic homepage or a company/personal homepage. For the second case, ‘clgiles’ could refer to a machine name. In addition to the above cases, given that feature dictionaries typically comprise features that meet a frequency requirement, we may not be able to extract features for all URLs. In our training datasets (Section 4.5), we were unable to extract URL features for about 27% of the instances. Therefore, content-based and URL features complement each other while identifying homepage instances and a focused crawler might be required to use either or both of these sets of features.

4.4 Homepage classification using unlabeled data

We show in our experiments (Section 4.5) that, although content-based features perform extremely well on the training datasets, they are not very successful on the validation and test sets that were collected from the current-day academic websites. On the other hand, URL features show good performance on both training and validation datasets. However, as pointed out in the previous section, we may not be able to extract URL features for all instances and it is, therefore, imperative to have an accurate content-based classifier as well.

We now address the questions: Can we adapt the content-based classifier to perform well in the deployment environment with the help of the URL-based classifier? Can the two classifiers “teach” each other so as to perform better in the new environment, using the co-training approach? Since the URL and content features provide evidence for classifying a webpage instance independently, intuitively, it appears possible that there are instances that the URL classifier makes mistakes on, which the content-based classifier identifies correctly and vice versa.

Blum and Mitchell proposed co-training in context of webpage classification [34]. In their datasets, webpages are representable in terms of two distinct views: using terms on webpages and terms in the anchor text of hyperlinks pointing to these pages. When few labeled examples are available for training, they showed that co-training could be used to obtain predictions on the unlabeled data to enlarge the training set. The experiments by Blum and Mitchell and subsequent experiments by Nigam and Ghani [48] showed that when a natural split of features is available, co-training that explicitly leverages this split outperforms classifiers that do not.

We study the applicability and extension of co-training for our problem. Although the es-
sentential motivation is to make use of the naturally available feature split and enable classifiers to learn from each other, we highlight the following aspects of our setup: Previous studies and benefits from co-training were illustrated on datasets where the unlabeled data is arguably from a \textit{similar distribution}. That is, the positive and negative instances in the labeled datasets are representative of those in the unlabeled data. This is in contrast to our case, where our positive class is fairly well-defined (homepages), whereas the negative class is described in terms of “not positive”. More precisely, although our training dataset has examples for the negative class, webpages encountered during the crawls can belong to types not encountered in the labeled data. We present an error analysis in Section 4.5, that illustrates the “new” types of webpages encountered in our crawl, potentially causing the pre-trained content-based classifiers to underperform during crawling.

4.4.1 Imbalance in class distributions of instances

The number of negative instances encountered during our crawls is higher in comparison with the number of positive instances. While this aspect was noticed during our experiments, a previous estimation experiment using mark-recapture methods had indicated that academic homepages comprise a small fraction of the Web \cite{59}. We can expect this imbalance to become more prominent as more examples are sampled over the co-training rounds. In the algorithm studied by Blum and Mitchell, the ratio between the number of positive and negative instances added from the unlabeled data is maintained to be the same as that in the training dataset during each iteration of co-training \cite{34}. We argue that avoiding this constraint is better in our scenario since we want the datasets to be more representative of the changing distribution.

Most classification algorithms are sensitive to the number of positive and negative instances available in the training data and are known to learn biased classifiers in case of severe imbalance \cite{12, 60}. We employ the idea of altering the mis-classification costs for the underlying classifiers during each round of co-training to handle this problem. For example, if the training dataset has 10 positive and 100 negative instances, we can set the penalty incurred on making mistakes on a negative instance to be $\frac{1}{10}$th of the penalty incurred on making mistakes on a positive instance. For most implementations of classification algorithms, the mis-classification costs can be specified as a parameter during the training process \cite{61}.

4.4.2 Our co-training algorithm

Our co-training setup is detailed in Algorithm 1. $L$ and $U$ represent the labeled and unlabeled datasets, respectively, available at each iteration. They comprise instances with both the views (content-based and URL-based feature sets). For a round of co-training, we train classifiers, $C_1$ and $C_2$, on the two available views, using misclassification costs, $\rho_1$ and $\rho_2$, respectively. Next, “$s$” number of examples are sampled without replacement into $S$ from the unlabeled data and $C_1$ and $C_2$ are used to obtain predictions for these instances. The \texttt{GetConfidentEgs} method is a generic placeholder that stands for a function that determines what instances from $S$ are
Algorithm 1 Procedure for co-training

\textbf{Input:} $L, U, s$

$L_1 \leftarrow L, L_2 \leftarrow L$

$p_1 \leftarrow \phi, p_2 \leftarrow \phi$

\textbf{while} $U \neq \phi$ \textbf{do}

- Compute $p_1$ using $\frac{|L_1^+|}{|L_1|}$, $p_2$ using $\frac{|L_2^+|}{|L_2|}$.
- Train classifier $C_1$ using $(L_1^1, p_1)$.
- Train classifier $C_2$ using $(L_2^2, p_2)$.
- $S \leftarrow \phi$
- Sample ‘s’ examples from $U$ and move them to $S$.
- $U \leftarrow U \setminus S$
- $S_1, S_2 \leftarrow \text{GetConfidentEgs}(S, C_1, C_2)$
- $L_1 \leftarrow L_1 \cup S_1, L_2 \leftarrow L_2 \cup S_2$

\textbf{end while}

\textbf{Output:} Classifiers $C_1, C_2$.

chosen for addition in subsequent rounds of co-training. We use the notation $L_1^+$ to represent the positive instances in the set $L_1$ whereas $L_1^1$ indicates that the view 1 (or feature set 1) of the examples in $L_1$ is being used.

4.4.3 Strategies for adding unlabeled instances for co-training

Based on previous studies in co-training [34, 48], we studied the following strategies for this function:

- **AddBoth**: In this scheme, we add all examples from $S$ that are labeled by $C_1$ or $C_2$ confidently to the training set for the next round. This approach is similar to self-training used in semi-supervised learning where confidently predicted unlabeled instances are added to the training set for retraining the classifier in subsequent rounds [62]. However, in contrast with self-training that uses a single view, in AddBoth, confident predictions are obtained from two sources (view 1 and 2) for addition into subsequent rounds.

- **AddCross**: In this scheme, examples from $S$, confidently labeled by $C_1$ are added to view 2 for the next round and vice versa. That is, we use the examples confidently labeled by one classifier while training the other classifier in the next round. Cross-addition also seems resilient to handling the possibility of cascaded errors over the iterations. If a classifier makes a confident but incorrect prediction, we would like to avoid feeding this example in the next round to the same classifier, a common problem in self-training [62].

- **AddCrossRC**: This scheme is similar to AddCross with the additional constraint on the number of positive and negative instances added in each round. This constraint was originally studied by Blum and Mitchell and ensures that the ratio of the number of positive and negative instances added in each round is the same as that in the initial labeled dataset [34].
The co-training algorithm is general and can be applied with any choice of classifiers on the two views. Blum and Mitchell provided theoretical analysis of co-training with probabilistic classifiers and showed that co-training works when the assumptions on sufficiency and independence are met. That is, each view should be sufficient to predict the class label, and the two views are independent given the class label. Recent studies have proposed relaxed criteria under which co-training techniques still work [63]. However, in practice, it is tricky to judge if co-training works for a problem and to verify if the assumptions are satisfied [50]. These questions are more relevant in context of recent research in obtaining two views from a single view when two views are not naturally available for applying co-training [49]. With this background, we now discuss our formulation of the effect obtained with co-training, in terms of a loss function. This formulation allows us to track whether the co-training process is beneficial for a given problem, even without the use of a validation dataset. This is useful from the perspective of the human labor involved in creating labeled examples. A subset of labeled examples may or may not be available for validating the trained models.

4.4.4 Learning conforming predictors on unlabeled data

We assume that classifiers, $C_1$ and $C_2$ trained on the two views are parameterized in terms of their weight vectors, $w_1$ and $w_2$. Most classification algorithms e.g., Support Vector Machines (SVM) and Maximum Entropy (MaxEnt), output weight vectors capturing the importance of each feature as part of the training process [12].

One can expect co-training to benefit a classification problem if one classifier (say, $C_1$) can “guide” the other ($C_2$) on examples that the latter makes mistakes on. This guidance is provided by adding examples confidently labeled by $C_1$ to the subsequent round of training $C_2$. This observation hints at the possibility of directly manipulating $C_2$, based on $C_1$’s prediction for an example that $C_2$ is not confident about. This effect can be achieved by optimizing a function that directly captures the mismatch in the predictions of the two classifiers.

Elaborating further, given that the concept classes, “positive” and “negative” are still the same on unlabeled data, if $C_1$ and $C_2$ are accurate, they would make similar predictions on the unlabeled data. This intuition is the basis for “consensus maximization” widely adopted in multiview learning, of which co-training is a special case with two views [51, 52, 53]. The mismatch in predictions by $C_1$ and $C_2$ on unlabeled data can be quantified using a loss function. The squared error loss function commonly used in machine learning captures this loss as:

$$L_U(w_1, w_2) = \frac{1}{|U|} \sum_{u \in U} (f_1(w_1, u) - f_2(w_2, u))^2$$

The above formulation captures the average squared-difference in predictions from the two views on unlabeled data. $w_1$ and $w_2$ correspond to the parameter vectors corresponding to $C_1$ and $C_2$, respectively, and $u$ refers to an example from $U$, having two views, $u_1$ and $u_2$. For a given example, $u = (u_1, u_2)$, the functions, $f_1$ and $f_2$ act on $u_1$ and $u_2$ respectively, and make the
predictions from $C_1$ and $C_2$ comparable. These functions could be generic (e.g. a function that outputs the probability that the instance is positive) or classifier-dependent (for e.g. a function that outputs scaled distances from the separating hyperplane in case of Support Vector Machines). Minimizing $L$ corresponds to adjusting the weight vectors, $w_1$ and $w_2$, so that they make similar predictions on $U$.

In contrast with multiview learning methods, where learning the classifiers is folded into a global objective function in sophisticated ways [51, 52, 53], we adopt a simpler approach that works off the initial parameter vectors and iteratively modifies them in a “co-training like” manner. Note that this initialization plays a crucial role in avoiding trivial solutions (such as $w_1 = w_2 = 0$) that are potentially possible since the loss is optimized only on unlabeled instances. Our proposed technique for obtaining the “pair of conforming classifiers” is described in Algorithm 2.

Algorithm 2 Learning a pair of conforming classifiers

Input: $w_1, w_2, U, 's', #oIteRs, #iIteRs, \alpha$

\[o = 0\]

while $o \leq #oIteRs$ do

\[i = 0\]

while $i \leq #iIteRs$ do

\[S = \varnothing. \text{Sample } 's' \text{ examples from } U \text{ into } U_t\]

for $u \in U_t$ do

  if $f_1(u_1, w_1)$ is confident then
    Add $u$ to $S$
  end if

end for

\[w_2 \leftarrow w_2 - \alpha \frac{\partial L_S}{\partial w_2}\]

\[i \leftarrow i + 1\]

end while

%Perform Mini-batch Gradient Descent to obtain a new $w_1$

\[i = 0\]

while $i \leq #iIteRs$ do

\[S = \varnothing. \text{Sample } 's' \text{ examples from } U \text{ into } U_t.\]

for $u \in U_t$ do

  if $f_2(u_2, w_2)$ is confident then
    Add $u$ to $S$
  end if

end for

\[w_1 \leftarrow w_1 - \alpha \frac{\partial L_S}{\partial w_1}\]

\[i \leftarrow i + 1\]

end while

\[o \leftarrow o + 1\]

end while

Output: $w_1, w_2$

In Algorithm 2, we start with the original parameter vectors $w_1$ and $w_2$ from classifiers $C_1$ and $C_2$, respectively, and iteratively adjust these vectors so that the values of $f_1(w_1, u_1)$ and
$f_2(w_2, u_2)$ look similar for all $u \in U$. The input parameter, $\#oIte$, refers to the number of times the inner loop comprising of the two gradient descent steps is executed, whereas, the $\#iIte$, and $\alpha$ are parameters for the gradient descent algorithm. Overall, the values of $\#oIte$, $\#iIte$, and $\alpha$ control the rate of convergence of the algorithm and can be set experimentally. These parameters can be set based on the base classifiers used, noting when the decrease in the objective function value is below a threshold. Adaptive tuning of these parameters by tracking the change in the value of the objective function in every iteration is a subject for future study [64].

In each iteration, we employ mini-batch gradient descent to minimize the loss function, once with respect to $w_1$ and next with respect to $w_2$. The mini-batch gradient descent algorithm is a hybrid approach often used for large-scale machine learning problems. This approach combines the best of stochastic (on-line) gradient descent and batch gradient descent to obtain fast convergence during optimization by running gradient descent on small batches of randomly selected examples [65].

In our algorithm, in each iteration, a small batch of instances are randomly sampled from the unlabeled data, $U$ and the loss function calculated, using instances for which $w_1$ makes confident predictions from this sampled set. This loss is minimized using gradient descent to adjust $w_2$. A similar process is then applied for adjusting $w_1$ using confident predictions from $w_2$. In effect, as the algorithm proceeds, we are adjusting the parameters of each classifier so that it makes predictions that are aligned with those of the other classifier’s confident predictions. Upon convergence, both $w_1$ and $w_2$ are adjusted so that they make conforming predictions on the unlabeled data.

In our experiments, we used the differentiable, logistic sigmoid function for $f_1$ and $f_2$. Typically, classifiers use the parameter vector, $w$, for computing decision values for each instance. That is, given an instance $x$, the dot product value, $\langle w, x \rangle$, is used for determining the label assignment for the instance. This value can be ‘squashed’ to a number between 0 and 1 indicating that the probability that instance has a particular label with the logistic function [12]:

$$P(t) = \frac{1}{1 + e^{-t}} \quad \text{with} \quad \frac{dP(t)}{dt} = P(t) \cdot (1 - P(t))$$

Given, the simple form for the derivative, we can directly use the values of $f_1$ and $f_2$ (that we compute anyway), for computing the gradients in Algorithm 2. Although the effect obtained by Algorithm 2 is similar to that of co-training, the conformity loss directly measures the effect of co-training as it is being applied. In contrast, Algorithm 1 is typically terminated either when no more examples are available or by tracking the performance on a validation dataset.

We provide a preliminary, experimental demonstration of the connection between co-training and our proposed algorithm in Section 4.5. A more detailed analysis, study of other choices for the loss function $\mathcal{L}$ and the functions, $f_1$ and $f_2$, are a subject of future work. Nevertheless, quantifying the discrepancy in predictions from the two views and an algorithm to directly address this aspect is an exciting step in understanding when co-training works. We show in Section 4.5 that our method can be used in lieu of a validation dataset for tracking the performance of
co-training.

4.5 Experiments

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://cs.illinois.edu">http://cs.illinois.edu</a></td>
<td><a href="http://www.cc.gatech.edu">http://www.cc.gatech.edu</a></td>
<td><a href="http://www.cse.ucsd.edu">http://www.cse.ucsd.edu</a></td>
</tr>
<tr>
<td><a href="http://www.cs.princeton.edu">http://www.cs.princeton.edu</a></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.3. List of the seed URLs used in creating homepage classification datasets

We discuss 3 types of experiments: First, we study the performance of content-based and URL-based features on the training and validation datasets. Second, we show that co-training can successfully address the problem of mismatch in the training and deployment environments for homepage classification. Finally, we show that our proposed algorithm (Algorithm 2), achieves the same effect as co-training.

4.5.1 Datasets

For studying homepage classification, the WebKB dataset is available in the public domain. However, since this dataset is slightly old and is not suitable for some of our problem setups, we also use the homepages specified for authors in DBLP.

1. **DBLP** dataset: DBLP lists the names of computer scientists whose articles are indexed in their database. At the time of our dataset creation in 2010, DBLP listed about 769785 author names out of which for 13290 authors, the homepage url information was also available. From the author-provided homepages on DBLP, we were able to retrieve from the specified URLs homepages for about 6000 authors.

2. **WebKB**: The WebKB dataset was previously used by several researchers for studying webpage and text classification [34, 35, 36, 37]. The WebKB collection contains academic webpages from Computer Science departments of four universities: Cornell, Washington, Texas and Wisconsin, categorized into 7 categories (student, faculty, staff, department, course, project, and other). The “other” class comprises pages that cannot be fit into the remaining six classes; for instance, a publications page that links to a page belonging to a faculty page. This collection was created in 1997.

<table>
<thead>
<tr>
<th>Training(WebKB+DBLP)</th>
<th>Unlabeled(Crawl)</th>
<th>Test(Crawl)</th>
<th>Validation(Crawl)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9263/4719</td>
<td>143145</td>
<td>1600/89</td>
<td>500/42</td>
</tr>
</tbody>
</table>

Table 4.4. Summary of the datasets used in classification experiments. The entry a/b represents a instances out of which b are labeled positive
We use the positive instances in the DBLP dataset and the WebKB dataset described in Chapter 1 for studying academic homepage classification. For the deployment scenario, we crawled the university websites listed in Table 4.3. These websites were selected arbitrarily from the list of top US graduate schools in Computer Science (obtained from rankings in US News [156]). We seeded our crawl with these URLs and used the open-source crawling software, Heritrix [155] (version 1.14.3), for obtaining all web pages of content-type ‘text/html’, within a depth of 5 starting at the parent URL. In total, we were able to obtain 162,369 webpages using this process. This crawl was performed in April, 2012 and hence, represents a relatively recent snapshot of content at these URLs. Note that, this setup was used for the purpose of experiments. Our final goal is to embed accurate classifiers into the crawler, so as to only obtain homepages.

To validate the performance of our classifiers, we randomly selected sets of 100 webpages from each of the 16 universities listed in Table 4.3 and manually labeled them. From the remaining pages, another set of 500 pages were randomly chosen for validating or tuning the methods proposed in Section 4.4. For our experiments, we only consider instances for which both the views are available, that is, pages from which we are able to extract both URL and content features. A summary of the datasets just described is shown in Table 4.4.

### 4.5.2 Classification experiments

![Figure 4.1](image1.png) **Figure 4.1.** Homepage classification performance using content features

![Figure 4.2](image2.png) **Figure 4.2.** Homepage classification performance using URL features

![Figure 4.3](image3.png) **Figure 4.3.** Homepage classification performance using content+URL features

![Figure 4.4](image4.png) **Figure 4.4.** Self-training: classification performance with content features
We study the performance of our content-based and URL features using the classification algorithms: Naive Bayes (NB), Naive Bayes Multinomial (NBM), Random Forests (RF), Support Vector Machines using a linear kernel (SVM) and Maximum Entropy (MaxEnt). Naive Bayes and Naive Bayes Multinomial are generative models whereas Random Forests is an ensemble method using Decision Trees. Discriminative algorithms such as Support Vector Machines [66, 67] and Maximum Entropy classifiers [36] are being used extensively for text classification problems in the recent times. These methods output parameter vectors as part of the training process that can be manipulated directly in our Algorithm 2. Further details on different classification algorithms and their parameters can be found in a standard machine learning textbook (e.g. [12]). The Naive Bayes algorithm is well-studied in context of co-training and its theoretical analysis, although co-training is a classifier-independent technique. For text classification, the Naive Bayes Multinomial classifier is different from Naive Bayes in terms of its modeling of term counts with multinomials [35].

<table>
<thead>
<tr>
<th>URL</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>training crawl</td>
<td>training crawl</td>
</tr>
<tr>
<td>TILDENODICT</td>
<td>ALPHANUMBER</td>
</tr>
<tr>
<td>TILDENODICT_SEQEND</td>
<td>TILDENODICT</td>
</tr>
<tr>
<td>ALPHANUMBER</td>
<td>ALPHANUMBER_ALPHANUMBER</td>
</tr>
<tr>
<td>NONDICTWORD</td>
<td>HYPHENATEDWORD</td>
</tr>
<tr>
<td>courses</td>
<td>ALPHANUMBER_SEQEND</td>
</tr>
<tr>
<td>ALPHANUMBER_SEQEND</td>
<td>TILDENODICT_SEQEND</td>
</tr>
<tr>
<td>users_NONDICTWORD</td>
<td>QMARK</td>
</tr>
<tr>
<td>users</td>
<td>NUMBER</td>
</tr>
<tr>
<td>NONDICTWORD_SEQEND</td>
<td>courses</td>
</tr>
<tr>
<td>homes</td>
<td>NUMBER_SEQEND</td>
</tr>
</tbody>
</table>

Table 4.5. Features ranked based on information gain on training and crawl datasets

![Figure 4.5](image1.png) ![Figure 4.6](image2.png)

**Figure 4.5.** Co-training: classification performance using content features  **Figure 4.6.** Co-training: classification performance using URL features

Figure 4.1 and 4.2 show the weighted F1 measure on our training (five-fold cross-validation) and validation datasets using the various classification algorithms with content-based and URL features respectively. We used classifier implementations provided by Weka [61], libSVM [68] and Mallet [15]. Where applicable, we tuned the parameters on the training datasets for the
best performance (e.g., the \( C \) parameter for SVM, the number of trees in RF). Similarly, to handle potential imbalance in instances belonging to different classes during co-training, we use appropriate misclassification costs (e.g., using the CostMatrix option in Weka and the “w” setting in libSVM). Our goal is to illustrate with these experiments the difference in training and validation scenarios. Consequently, validation dataset is not used for fine-tuning parameters in these experiments.

As the figures illustrate, discriminative algorithms such as SVM and MaxEnt generally outperform the generative models such as NB and NBM on the homepage identification task using both content (Figure 4.1) and URL (Figure 4.2) features. The performance of URL-based classifiers (Figure 4.2) is typically higher compared to that of content-based classifiers (Figure 4.1), especially on the validation set that was collected from the crawled data. We performed an error analysis on the validation set and noticed that the content-based classifiers suffer from a high false-positive rate, and incorrectly labeled negative instances as positive. The WebKB dataset includes negative instances coming from types such as course, department, and project-related pages. However, about 212 out of the 500 validation instances could not be categorized into any of the 7 types present in WebKB. Instead, we can capture these pages under the following new types:

1. Webpages related to colloquium, seminars, lectures, publications, papers, talks, slides.
2. Webpages that describe code, widgets, scripts, datasets.
3. Webpages related to department activities such as picnics, pages with embedded photos, and personal pages.
4. Webpages pertaining to information on news, events, highlights, faq, forms.
5. Webpages pertaining to alumni-related information, job and contest calls.

![Figure 4.7](image1.png)  
**Figure 4.7.** Performance of different co-training schemes (AddCross, AddCrossRC, AddBoth) with Naive Bayes Multinomial classifier

![Figure 4.8](image2.png)  
**Figure 4.8.** Performance of our gradient descent (GD) algorithm using content-based features and maximum entropy classifier

Given that our validation set only comprises 500 instances, it is reasonable to suspect that other types of webpages exist in our crawled collection. We used Information Gain [12] on the training
and crawl-based datasets to understand the feature-class correlation between the two datasets. The top 10 features ranked by this measure, shown in Table 4.5, also point to the difference between the two environments. However, our aim is not to model new types of webpages, but rather, we wish to learn a discriminator that isolates academic homepages from non-homepages. In our experiments, we noticed that webpages belonging to the new types 1 and 2 above were often misclassified as academic homepages. However, surface patterns and cue words such as “seminars” in the URLs are effective for classifying these instances correctly.

Further, we studied the following approaches for improving performance of content-based classifiers for the new environment: First, we learned classifiers on the combined set of features for each instance (content+URL). The performance of these classifiers is shown in Figure 4.3. As the plots indicate, using the combined set of features on the validation set is better than using content features alone (Figure 4.1), but still worse than using only URL features alone (Figure 4.2).

Second, we trained a one-class SVM (linear kernel) for identifying homepages. One-class classifiers learn discriminators for a class by using just the positive instances without explicitly modeling the other classes [69, 70]. One-class methods are typically used for detecting outliers and novelty detection. However, one-class SVMs do not work as well as binary classifiers for our problem on the training datasets. Their performance on the validation sets is better, but not comparable to the best performing classifiers on the validation datasets (see Figures 4.1, 4.2, and 4.3, 1C-SVM).

Finally, we employed self-training [71] to train the content-based classifiers using the unlabeled data. Self-training is an iterative approach commonly used in semi-supervised learning. Unlabeled examples, predicted confidently by the classifier are added back to enlarge the training dataset used in retraining the classifier in the subsequent iteration. The intuition behind self-training has been compared to pseudo-relevance feedback employed in information retrieval [48, 13]. Although, this approach was successfully applied to some problems such as word sense disambiguation and parsing [71], self-training is known to suffer when classifiers in the initial iterations add incorrectly-labeled examples to subsequent iterations. This results in error cascading with the final classifier performing worse than the initial one [62]. Our content-based classifiers do not show any improvements with self-training and, depending on the classification algorithm, even show decreased accuracy on the validation set. This behavior is illustrated in Figure 4.4.

We chose the better performing of all classification algorithms: NBM, SVMs and MaxEnt as base classifiers for the remaining experiments.

### 4.5.3 Co-training experiments

We studied co-training (Algorithm 1) using the different schemes, AddBoth, AddCross, and AddCrossRC for selecting unlabeled instances for the next round. We use the same type of algorithm for training classifiers on both views (e.g., SVM for URL features as well as for content-based features). We sampled 5000 instances from the unlabeled dataset in each iteration
and consider an instance for addition only if a prediction was made for this instance with a confidence probability $\geq 0.9$. Note that the labeled training set is WebKB+DBLP, whereas the unlabeled training set is the data from our crawl. Figure 4.5 shows the performance of content-based classifiers (within co-training) on the validation set. Figure 4.6 shows similar results using URL-based classifiers. As shown in the figures, co-training successfully manages to pull up the performance of the content-based classifiers using the unlabeled data. The URL classifiers being more stable add relevant instances to the labeled datasets over successive rounds, enabling the content-based classifier to retrain itself over the iterations and learn to discriminate better among the current-day pages on the web. Although the initial rounds do not result in improvements in the performance for the URL classifier, once the content-based classifiers are up in accuracy, they are able to provide useful examples to the URL classifiers, in turn, resulting in improvements even for the URL classifier (Figure 4.6).

Figure 4.9. Performance of our gradient descent (GD) algorithm using URL features and maximum entropy classifier

Figure 4.10. Performance of our gradient descent (GD) algorithm using URL features and support vector machines

Figure 4.11. Performance of our gradient descent (GD) algorithm using URL features and maximum entropy classifier

Figure 4.12. Variation of the objective function value (ObjVal) and F1 as the gradient descent algorithm progresses

Figure 4.7 shows the effect of different schemes for instance selection during co-training iterations for the NBM algorithm. Difference in performance on the validation dataset was noticed for the co-training schemes with NBM as base classifiers. For SVM and MaxEnt pairs, there were no observable difference in the performance attained by the different schemes, AddBoth and AddCross. The AddCross scheme does either better or on par with the other schemes.
In addition, since we use cross addition of instances in Algorithm 2, we chose AddCross for comparisons with the Gradient Descent based approach in the next section.

In our experiments, not accounting for class imbalance results in mistakes on the positive instances due to the large number of negative instances in the training data and the performance on this class reduces drastically over the co-training iterations for both the AddBoth and AddCross schemes. The performance on AddCrossRC is not affected due to the ratio constraint being maintained over the iterations since we started with an almost balanced training dataset (see Table 4.4). However, the performance with the AddCrossRC is not as good as the AddCross scheme with class imbalance accounted for.

4.5.4 Gradient descent experiments

Figures 4.8 and 4.9 show a run of Algorithm 2 with initial weight vectors obtained with MaxEnt classifiers. These initial vectors are obtained by running the MaxEnt trainer over the labeled (training) dataset. The figures show the classification performance on the validation set after the termination of each iteration (from 1 to 20) of Algorithm 2, which minimizes the loss function in each round. Similar plots with the initial weight vectors obtained from SVMs are shown in Figures 4.10 and 4.11.

The figures also show the comparison of the proposed gradient descent algorithm with co-training. We plot the maximum $F_1$ score that was obtained with co-training experiments (previous sub-section) alongside the curves in Figures 4.8-4.11, to illustrate that our proposed algorithm in effect attains similar performance improvements that are possible with co-training. Although these plots show the $F_1$ variation over the validation set, note that, the validation set is not used for tracking the optimization process. Instead, the algorithm can terminate either after a pre-set number of iterations or by explicitly tracking the objective value for convergence. Each iteration of the algorithm involves running mini-batch gradient descent twice, once for each classifier.

We observed the objective values to be converging in about 20 iterations when the algorithm is initialized with the weight vectors from MaxEnt whereas it takes about 50 iterations with those from SVM. These values correspond to the #oIters in Algorithm 2. The #iIters and $\alpha$ values for Mini-batch Gradient Descent were set to 50 and 0.1, respectively, in all experiments. About 1% of unlabeled data was randomly sampled in each round and examples that were predicted with 90% confidence were used for computing the loss function that is minimized with Gradient Descent. The #iIters and $\alpha$ values affect the rate of convergence for Gradient Descent. We chose these values based on experimentation, instead of adaptively using techniques such as line search [64]. Experimenting with these parameters is left for future work. With the settings just described, the run times for convergence were similar to that of co-training. In general, depending on the classification algorithms used, the co-training experiments took times ranging between 5min-4hrs on a 16-core, 800MHz, 32GB RAM, AMD Opteron, Linux server.

We plot the $F_1$ on the validation dataset against the computed objective function value in Figure 4.12. The plot depicts the close correspondence between reducing the discrepancy between
predictions based on the two views and the improving performance on the validation dataset. We also illustrate the connection between the effect of co-training and our proposed loss function by plotting the value of our loss function on unlabeled data available in that iteration as co-training progresses in Figure 4.13. This plot highlights the fact that when co-training works, it seems to be due to the reducing discrepancy between the predictions from the two views used in co-training. The list of top 10 features that undergo the most change in the MaxEnt weight vectors after Algorithm 2 converges are shown in Figure 4.6.

![Figure 4.13](image1.png)

**Figure 4.13.** Reducing value of the squared-error loss as co-training progresses, MaxEnt: maximum entropy classifiers, SVM: support vector machines

![Figure 4.14](image2.png)

**Figure 4.14.** Performance of on the test datasets, Org: performance with the base classifier, CT: co-training, GD: gradient descent, GET: generalized expectation

<table>
<thead>
<tr>
<th>URL</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONDICTWORD</td>
<td>numLinks</td>
</tr>
<tr>
<td>TILDENODICT</td>
<td>numImages</td>
</tr>
<tr>
<td>NONDICTWORD_SEQEND</td>
<td>research</td>
</tr>
<tr>
<td>LONGWORD</td>
<td>here</td>
</tr>
<tr>
<td>LONGWORD_SEQEND</td>
<td>courses</td>
</tr>
<tr>
<td>QMARK</td>
<td>slide</td>
</tr>
<tr>
<td>TILDENODICT_SEQEND</td>
<td>academics</td>
</tr>
<tr>
<td>HYPHENATEDWORD</td>
<td>education</td>
</tr>
<tr>
<td>grads</td>
<td>publications</td>
</tr>
<tr>
<td>ALPHANUMBER</td>
<td>document</td>
</tr>
</tbody>
</table>

**Table 4.6.** Features that are most affected after running Algorithm 2 on weight vectors from maximum entropy classifier

### 4.5.5 Evaluation on the test set

So far, we have been showing performance results using the validation set (Table 4.4) for the sake of illustration. We summarize our evaluation with our final classifiers on the test dataset in Figure 4.14 and Table 4.7. The table shows the weighted precision, recall and F1 measures [13] whereas the figure illustrates the improvements in F1 via bar charts. We compare the original content-based classifiers trained on the training datasets (the ‘Before’ entries) with our proposed methods. The ‘After-CT’ entries use the classifiers obtained after co-training was employed
<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>NBM</td>
<td>0.8910</td>
<td>0.4700</td>
<td>0.5890</td>
</tr>
<tr>
<td>NBM-After-CT</td>
<td>0.9350</td>
<td>0.7980</td>
<td>0.8470</td>
</tr>
<tr>
<td>SVM</td>
<td>0.9413</td>
<td>0.4947</td>
<td>0.6048</td>
</tr>
<tr>
<td>SVM-After-CT</td>
<td>0.9147</td>
<td>0.8167</td>
<td>0.8559</td>
</tr>
<tr>
<td>SVM-After-GD</td>
<td>0.9175</td>
<td>0.8826</td>
<td>0.8977</td>
</tr>
<tr>
<td>MaxEnt</td>
<td>0.8806</td>
<td>0.4167</td>
<td>0.5380</td>
</tr>
<tr>
<td>MaxEnt-After-CT</td>
<td>0.9234</td>
<td>0.9295</td>
<td>0.9262</td>
</tr>
<tr>
<td>MaxEnt-After-GD</td>
<td>0.9272</td>
<td>0.9401</td>
<td>0.9336</td>
</tr>
<tr>
<td>GET-1</td>
<td>0.8668</td>
<td>0.5598</td>
<td>0.6719</td>
</tr>
<tr>
<td>GET-2</td>
<td>0.8761</td>
<td>0.5614</td>
<td>0.6724</td>
</tr>
<tr>
<td>MaxEnt-Combined</td>
<td>0.9469</td>
<td>0.7219</td>
<td>0.7938</td>
</tr>
<tr>
<td>MaxEnt-Combined-After-ST</td>
<td>0.9418</td>
<td>0.7098</td>
<td>0.7849</td>
</tr>
<tr>
<td>GET-1</td>
<td>0.9273</td>
<td>0.4765</td>
<td>0.5895</td>
</tr>
<tr>
<td>GET-2</td>
<td>0.9365</td>
<td>0.7591</td>
<td>0.8201</td>
</tr>
</tbody>
</table>

Table 4.7. Before/After performance on the test set with content features. Runs marked with “combined” use URL+content features, CT: co-training, ST: self-training, GD: gradient descent, GET: generalized expectation.

with unlabeled data and terminated after convergence was attained on the validation datasets (typically in about 20 iterations). The ‘After-GD’ entries use the weight vectors obtained after running Algorithm 2 starting with the original weight vectors.

We also compared MaxEnt with “expectation regularization”, a technique proposed by Mann, et al [72]. In this semi-supervised learning method, additional regularization terms are added to the objective function while learning classifiers of the exponential family. These additional terms encourage the predictions on unlabeled data to meet certain expectations that capture label priors or feature-correlations. The expectations are expressed as soft constraints while learning the classifiers [7]. For example, for our problem, we could specify constraints, which indicate that the feature ‘homepage’ in the HTML title strongly suggests that the class-label is positive.

We obtained the code for automatically generating constraints from the labeled data using information gain and an implementation of the expectation regularization framework with MaxEnt classifier (GETrainer) from the authors. Using the parameter settings suggested by the authors, the performance of the GETrainer on the test dataset is listed in the “GET-1” and “GET-2” rows in Table 4.14. Labeled data can be used to extract constraints but class labels are not required for learning the GETrainer. The “GET-1” row refers to the performance using unlabeled data from the crawl environment while training, whereas the “GET-2” row shows the performance when the labeled dataset was used during training. Based on the performance of the GETrainer on the validation dataset with different number of constraints, we chose the number of constraints to be 8000 for the “GET-1” setting and 6000 for “GET-2”.

Figure 4.14 and Table 4.7 indicate that the “expectation regularization” technique is successful in capturing feature constraints and improves the performance of our content-based classifier. We also evaluated the GETrainer using feature vectors containing both content-based and URL

---

1 We thank G. S. Mann, G. Druck and A. McCallum for sharing their implementation with us.
features. For the combined set of features, the GE method achieves an improved performance although it is not very high compared to that obtained with the original classifier (‘Before’ entry). However, larger benefits are obtained by harnessing the split of features in terms of the two views.

4.5.6 Comparison of co-training and feature selection

Our experiments in Section 4.5.3, validate the findings by Nigam, et al. on the applicability of co-training [48]. Nigam, et al. show that in classification problems where instances have multiple views, models that explicitly harness this split tend to perform better than models that do not. However, for several classification problems, multiple views may be unavailable or unknown [49]. In these cases, feature selection and boosting are commonly evaluated for their potential benefit with respect to classification [12].

Feature selection (FS) is a popular technique employed in machine learning problems [12]. When data contains irrelevant or redundant features, FS methods that reduce the variable space using various measures such as information gain, class correlation and mutual information can yield benefits not only with respect to classification accuracy but also by reducing the model training times and complexity. In addition, most relevant features with respect to a particular domain are often used to “interpret” the results of a trained model.

AdaBoost (short for Adaptive Boosting), is a “meta algorithm” proposed by Freund and Schapire for use in conjunction with other classifiers [73, 74]. AdaBoost is an iterative algorithm, where weights are assigned to instances indicating their importance. In each iteration, a classifier is trained on weighted instances followed by prediction. Instances for which the classifier makes incorrect predictions are assigned higher weights for use in the next round. The intuition behind this re-assignment of weights is to enable the learning algorithm to focus on the “hard” cases in the next round. Freund and Schapire theoretically showed that when the classifiers perform slightly better than random (“weak” classifiers), the AdaBoost algorithm will improve the final classification performance.

We perform experiments to test if feature selection and boosting benefit homepage classification and compare them against co-training. Table 4.8 provides a summary of the performance on the test dataset using feature selection and co-training with the Naive Bayes Multinomial classifier. We evaluated the performance of different feature selection methods implemented in Weka [148] and different feature set sizes and chose the settings that give the maximum performance on the validation dataset. Selecting the top 30 and top 1200 features based on information gain (IG) were found to be the best performing settings for content-only and the combined classifiers respectively.

As can be seen in Table 4.8, although feature selection and AdaBoost both provide improvements, the improvements are not as high as those obtained with co-training. However, as indicated in the final two rows of Table 4.8, combining co-training with feature selection can yield even higher improvements. A thorough comparison of feature selection, boosting and co-training on other classification datasets (with two views) is a subject for future study.
4.6 Summary and future work

We studied the problem of adapting a classifier trained on a labeled dataset of webpages to a related environment containing newer types of webpages in the context of focused crawling for researcher homepages. We showed that co-training with unlabeled data can be used effectively to improve the classification performance in the deployment (crawling) scenario. Although our evaluation is specifically for homepage classification, we conjecture that our findings hold for other scenarios where a mismatch in training and test environments occurs. For instance, this is likely to be true in most focused crawling situations given the changing rate of content on the Web. Intuitively, we can expect our techniques to work well when the following criterion is met: a view $v_1$ must be able to predict at least one unlabeled example confidently that view $v_2$ cannot and vice versa. When this condition is satisfied, the views can “help each other” over the iterations.

We also proposed a novel formulation of co-training in terms of a loss function. This loss can be directly minimized via a mini-batch gradient descent algorithm. Our results indicated that even without a validation set, one can track the effect of the co-training process via our loss function. Some interesting directions for future study are:

1. For learning a “conforming pairs of classifiers”, other optimization algorithms, loss functions and function choices for comparing classifier predictions need to be studied. Further analysis is required with respect to convergence and parameter sensitivity during optimization. In addition, other optimization algorithms such as l-BFGS [64] can also be explored for optimizing the objective function.

2. How does our gradient-descent algorithm compare against other approaches to multiview learning (for example, multiview Laplacian SVMs [75])? In this chapter, we only provided comparisons for single-view and two views and co-training compared to self-training and feature labeling. Some initial results with feature selection were also included. On average, which approach or combination works the best from the perspective of the classification performance on multiview datasets?

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>NBM</td>
<td>0.932</td>
<td>0.689</td>
<td>0.770</td>
</tr>
<tr>
<td>NBM+AdaBoost</td>
<td>0.938</td>
<td>0.736</td>
<td>0.804</td>
</tr>
<tr>
<td>NBM+IG</td>
<td>0.932</td>
<td>0.734</td>
<td>0.802</td>
</tr>
<tr>
<td>NBM+IG+AdaBoost</td>
<td>0.931</td>
<td>0.745</td>
<td>0.810</td>
</tr>
<tr>
<td>Content-only with NBM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pre-CT</td>
<td>0.891</td>
<td>0.470</td>
<td>0.589</td>
</tr>
<tr>
<td>Post-CT</td>
<td>0.934</td>
<td>0.803</td>
<td>0.850</td>
</tr>
<tr>
<td>Pre-CT with IG</td>
<td>0.902</td>
<td>0.728</td>
<td>0.796</td>
</tr>
<tr>
<td>Post-CT with IG</td>
<td>0.899</td>
<td>0.847</td>
<td>0.870</td>
</tr>
</tbody>
</table>

Table 4.8. Comparison of feature selection using information gain (IG) and co-training (CT)
3. The objective function currently is only defined on the unlabeled instances and we do not track the performance on the original training instances. In the crawling scenario, our goal was to adjust to the more recent crawl environment due to which the performance on the older dataset (such as WebKB) was not a concern. However, it is possible to conceive of situations (such as domain adaptation [76]) where high accuracies are required on both the “train” and “test” environments. For these cases, objective functions that include both labeled and unlabeled instances need to be designed.

4. An interesting angle in the co-training algorithm is the potentially changing “labels” of the unlabeled instances added during iterations. In the current setup, once a highly confident prediction is made for an unlabeled instance in an iteration, a label is “assigned” to the instance and is assumed to be “fixed” for subsequent iterations. However, does the prediction for this instance, remain consistent and confident over the co-training iteration rounds? To the best of our knowledge, no work provides a detailed analysis of these aspects of the co-training algorithm.

5. Another future direction is the possibility to optimize an objective function using both labeled and unlabeled instances in a single step instead of the current two-step approach in which optimization is followed by a co-training like addition of unlabeled instances in multiple rounds.

6. Finally, from the perspective of focused crawling, our motivating scenario, a study is required on the benefits of folding in our proposed URL and content-based classifiers into the crawl process both in terms of yield and efficiency. Using the URL features, can we design a rule-based filters for discarding the unlikely URLs? How does this compare with a simple rule-based system using “expert knowledge” (For example, only follow links with “people” or “tilde” in the URLs)?
Chapter 5

Metadata extraction from academic homepages

5.1 Introduction

Academic homepages summarize research and academic interests of researchers and contain other metadata that can be useful in tasks such as expertise search, academic network extraction and name disambiguation [5, 4]. In this chapter, we address the task of metadata extraction from homepages. That is, given a researcher homepage, our goal is to identify values for a number of pre-defined metadata fields: employment position, university and department affiliations and contact information such as email, phone and fax.

The homepage metadata extraction problem can be converted to a sequence labeling (also known as tagging or annotation) problem in a straightforward manner: Given the stream of tokens corresponding to the content on a homepage, assign to each token a tag/label from the set: \{ AFFL, EMAIL, FAX, PHN, POS, UNIV, O \} where these labels correspond to “affiliation”, “email id”, “fax number”, “phone number”, “employment position”, “university” and “other” fields respectively. An example is illustrated in Table 5.1. Previous research on this task showed that tagging or sequence labeling approaches that capture dependencies among tags out perform classification-style approaches [4, 77]. This is not surprising given that researchers tend to observe certain conventions while laying out their metadata on their homepages. For instance, it is common to find phone and fax information close together on a researcher homepage. Similarly, it is common to find employment position information followed by the affiliation information on a homepage (e.g., “professor” in the “Computer Science department” at “Stanford”).

We highlight some challenges in tagging metadata fields on homepages compared to the more general NLP task of tagging fields in general English text [78].

1. Presence of cue words does not always indicate the metadata of the homepage’s subject researcher. Although ‘student’ and ‘professor’ are commonly seen values for the position
field, in Table 5.1, only ‘student’ needs to be annotated with the ‘POS’ label since this value corresponds to the “owner” of the homepage.

2. A related challenge pertains to similar words occurring with multiple labels where certain labels are more common than the others. For instance, in the above snippet, the first ‘State’ corresponds to a university field whereas the second ‘state’ refers to a research problem the student is working on (to be marked as “other”). Since webpages give rise to lengthy sequences of tokens and most of them are “other”, we found that terms such as “research, department, student” that are indicative of potential metadata fields tend to occur more often with the “other” tag. Learning algorithms that depend on co-occurrence counts may not be able to model such parameters accurately.

3. Values for certain metadata fields exhibit diverse patterns (e.g., affiliation values Department of Computer Science, EECS Department, and Computer Science and Electrical Engineering Department). In addition, patterns with certain cue words may occur rarely in a dataset. For instance, in our dataset, we found that affiliation values containing the term “department” occur about 30% of the time whereas values that contain the term “centre” only occur 1.3% of the time.

How can we account for imbalance in token-label pairs and rare patterns without having to label more examples? For instance, given that we know that the term, “centre” corresponds to affiliation, can we use this information to guide the training process? More generally, can we extract and incorporate domain-specific hints while training homepage annotation models? We address these questions with feature labeling, a recent advancement in semi-supervised learning [79].

Contributions: We study the use of “labeled features” for annotating metadata on researcher homepages. We capture term and layout hints associated with metadata fields via dictionary and proximity labeled features respectively. These hints enable us to train annotation models with fewer training instances. Our contributions are summarized below:

1. First, we propose and evaluate a set of basic features for annotating homepages. In contrast with previous works that use rule-based patterns, noun phrases and visual information our set of features is minimalistic with domain information separated to dictionary features alone. We posit that this approach is likely to be more robust across datasets unlike previously proposed intricate features [11, 77].

2. To the best of our knowledge, annotation of researcher homepages using semi-supervised models was not studied before. We adopt the recently proposed feature labeling approach
where supervision is provided using (feature, label) distributions. These distributions are incorporated into the training process via posterior regularization. Our experiments demonstrate the effectiveness of this approach when the number of annotated instances are limited.

3. Finally, we study strategies to extract labeled features when training instances are available. In absence of a large number of such instances, we show that automatic methods may not be capable of extracting labeled features whose value is comparable to that of expert-specified labeled features.

In the next section, we summarize the work closely related to our contributions. In Sections 5.3 and 5.4, we describe our methods. Section 5.5 covers our experimental setup, results and observations while Section 5.6 concludes this chapter.

5.2 Related work

Information extraction problems are of great interest in the web and natural language processing communities. Some widely-studied information extraction problems include identifying named-entities from news articles, street names and cities from address strings, header information from research papers, tables extraction from pdf documents, and product information from the web [80, 81]. Linear chain Conditional Random Fields (CRFs) that address information extraction as sequence tagging problems are shown to be widely successful on various IE tasks [78]. CRFs can be trained discriminatively using arbitrary features unlike generative models such as HMMs where folding in arbitrary features typically requires independence assumptions on the features for computational reasons.

In particular, metadata extraction from academic homepages was studied for the ArnetMiner project [140, 11] and for CiteSeer [139, 77]. Tang, et al. designed several sets of noun-phrase, dictionary, pattern and term features for identifying the metadata fields. Zheng, et al. instead classify the HTML DOM nodes that correspond to metadata fields using visual features such as font-style, and position of the block in the page. A second stage inter-field probability model is used for the final extraction. Based on comparisons and observations from these works, we chose CRFs for studying the homepage annotation task. Our focus is on using simpler features and semi-supervised learning for this task in lieu of the intricate features proposed by previous works.

A recent advance in semi-supervised learning pertains to the use “labeled features” for training models [82, 83, 79]. Mann, et al. and Druck, et al. proposed the Generalized Expectation (GE) criterion for using labeled features within discriminative classifiers and taggers [8, 7]. Ganchev, et al. proposed “Posterior Regularization” (PR), a more general framework for incorporating “side information” into models for structured prediction by imposing linear constraints on posterior expectations [9]. We use the CRF trainer, labeled features and the PR framework implemented in Mallet, the information extraction package provided by UMass [15].
Our focus is on capturing homepage-specific aspects as labeled features for use within the PR framework. Feature extraction for semi-supervised models was previously studied for classification [7, 84] and tagging [83, 8]. However, these works focus on term-based features that frequently correspond to labels. In addition to term features, we design “proximity” features that capture the layout of metadata fields on a homepage.

Our proximity features are similar in spirit to self-labeled features previously studied for tagging problems [85]. Qi, et al. proposed an iterative scheme, where feature vectors in each iteration are augmented with the predicted word-level class label distributions from the previous iteration in a semi-supervised manner. Similarly, we use predictions from a first-stage CRF for use as “labeled features” in a second CRF, effectively combining the two ideas.

5.3 Method

5.3.1 Motivation for a two-stage process

It is reasonable to assume that researchers do not arrange their metadata on their professional homepages arbitrarily. For example, it is unlikely that the phone contact information appears at the top of the page while the fax information appears towards the end. Similarly, it is common to find employment information of a researcher closely listed with the affiliation information (e.g. “I am an assistant professor in the Computer Science department at Stanford”). Indeed, researchers follow certain conventions in placing their metadata and this aspect was captured partially via visual dependencies [77] and transition features [4] in previous research. However, the proposed visual layout features are very intricate while the transition features are limited to a single step in linear CRFs [78].

Given that the metadata fields on an academic homepage tend to occur in close proximity with each other, can the knowledge of certain fields aid in the identification of the other fields? We seek to answer this question via a two-step approach as follows:

1. Use the basic set of features (Table 5.3) to train a tagger for the first stage.
2. Next, use predicted tags from the first stage tagger as additional features to train a second-stage tagger.

We posit that this two-stage process is better in modeling next labels in addition to previous labels as well as label information within a window rather than just the previous step label dependencies (as in the case of linear-chain CRFs). More precisely, in the second stage, for every token position, we add the closest tag within a window of positions with respect to the current token position. An example is shown in Table 5.2.

5.3.2 Stage 1: Training the first CRF tagger with basic features

We train a homepage tagger using features corresponding to simple surface patterns, terms and dictionaries. In contrast with previous work that used pattern-based and visual features that are
Table 5.2. Example demonstrating features added for stage 2 based on stage 1 predicted tags and window size=3 (pw: previous window, nw: next window, s1: stage 1)

 unlikely to be robust across datasets, we chose simple unigram, bigram features based on terms, surface patterns and dictionaries available for this task. We use the following features:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unigram features</td>
<td>$F_i$, $i = {-2, \ldots, 2}$</td>
</tr>
<tr>
<td>Bigram features</td>
<td>$F_{i-1} F_0$ and $F_0 F_i$</td>
</tr>
<tr>
<td>Skip features</td>
<td>$F_{i-1} F_i$</td>
</tr>
<tr>
<td>Conjunction features</td>
<td>$F_{i-1} G_0$ and $F_0 G_i$</td>
</tr>
</tbody>
</table>

Table 5.3. Feature templates for the Stage-1 tagger for metadata extraction

1. **Canonical term features**: These features refer to basic terms corresponding to the textual content on a homepage. We use whitespace tokenization and convert all tokens to lowercase after removing punctuation.

2. **Dictionary features**: We use boolean features corresponding to the presence in field-specific dictionaries. These dictionaries were obtained from previous work related to ArnetMiner where homepage annotation was studied using CRFs and SVMs [4]. These dictionaries comprise a total of 147 cue words often seen with metadata fields. For example, values for the phone field usually appear as numeric strings following the cue words, ‘phone’ or ‘ph’ (sample words in Table 5.4).

| AFFL: | center, centre, college, department, dept, dipartimento, laboratory |
| UNIV: | universiteit, universitat, university, univ |
| PHN:  | cell, ext, extn, homophone, mobile, numbers, ph, phonefax, phone |
| FAX:  | ext, extn, facsimile, fax, faxno, faxnumber, telefax, tel/fax |
| EMAIL:| contact, email, firstname, lastname, gmail, mail, mailbox, mailto |
| pre POS: | administrative, affiliate, assistant, associate, asst, co, chief, deputy, scholar, scientist, |

Table 5.4. Sample cue words for different metadata fields

3. **Surface-form features**: Surface patterns provide valuable hints in annotation tasks. For instance, phone numbers are typically numeric fields and values for affiliation fields are often
capitalized. We use boolean features indicating if the token matches one of the surface patterns: singleLetter, allletters-capitalized, is-a-capitalized-word, all-digits-in-word, and word-has-digits.

4. **Name-based features**: To capture the empirical observation that most metadata fields on a homepage appear in close proximity with the researcher name, we indicate the presence of a researcher name within a neighborhood of five lines within the line containing the token via this feature.

5. **Sentence delimiter features**: We add sentence boundary features indicating whether the token starts, is inside or ends a sentence. These features are designed to capture the observation that labels do not often extend across line boundaries.

Given a string of terms corresponding to the content of a homepage, let $F, G$ represent feature-types described above. We use subscripts to denote the feature corresponding to a particular position in the text. The feature templates used for training the initial classifier are listed in Table 5.3. We refer to a CRF model trained on these features as “Basic CRF” or “Stage 1 or S1” in Section 5.5.

### 5.3.3 Stage 2: Training the second CRF tagger with predicted labels from stage 1

In our initial experiments with basic features, we found that the tagger was able to accurately identify fields corresponding to phone, fax and e-mail but was not very accurate on fields such as affiliation, position and university (Section 5.5.3). The second-stage tagger is designed to avail the “most likely correct” fields identified in stage 1 to zero in on other fields in the next round. We obtain the set of predicted tags from the Stage 1 CRF to form the second-set of features for each token position on a homepage.

Table 5.2 shows an example for generating additional features using predicted tags for a window of size 3. In this table, “pw” and “nw” indicate respectively, the previous and next positions within the specified window size, where the stage 1 tagger labeled a metadata field. For example, “pws1UNIV” indicates that within the previous 3 positions, the stage 1 tagger marked a token with the “UNIV” tag.

The set of features for stage 2 are generated using the conjunction feature templates by combining the predicted tags from stage 1 with term, dictionary and surface pattern features. For example, the features added in stage 2 for the token “Penn” in the example from Table 5.2 are: pennₙ₁UNIV, capitalizedₙ₁UNIV and nodictₙ₁UNIV.
5.4 Feature labeling to improve tagging

5.4.1 Motivation

We further study techniques to improve the tagging performance on the affiliation, position, and university fields. An error analysis of results obtained using the taggers from the previous section indicated two reasons for low numbers on these fields:

1. Cue words that are indicative of metadata fields occur with the “other” tag more often than with the specific field potentially making it hard for the model to estimate weights for the associated parameters accurately. For instance, the term, “student” occurs with the “POS” tag only 23% of the time in our dataset, whereas the term “university” occurs with the “UNIV” tag about 36% of the time. The remaining percent corresponds to their occurrence with the “O” tag.

2. The affiliation, position and university fields exhibit various patterns in their values with a severe imbalance in the training examples for each pattern. For example, there are about 300 labeled instances having affiliation field values with the term “department”, whereas only 14 labeled instances have the term “centre” in their affiliation values.

Supervised learning algorithms require a number of labeled instances to estimate accurately the weights corresponding to feature parameters. For taggers, edge transition parameters need to be additionally estimated [12]. Compared to classification tasks where labels are assigned at an instance level (e.g., for a document or an image), labeling examples for sequence tagging where each token position in the sequence needs to be marked up involves considerably more human effort. However, recent research indicated that while labeling instances is hard, labeling features is considerably easier and faster for an expert in the domain [84].

Given the advancement in semi-supervised learning with labeled features, we ask the question: What kind of hints can we provide via labeled features to the learning algorithms to enable better tagging of researcher metadata on a homepage?

5.4.2 Background

Labeled features were studied by Mann, Druck and McCallum for both classification and tagging problems [8, 7]. They proposed adding “supervision” to learning algorithms by providing (feature, label) affinities rather than fully-annotated instances. Consider the example in Table 5.1. Even without annotating the entire snippet, from domain knowledge, one can expect the correct label for the token “student” to be “POS”, “most” of the time. This hint can be imposed as a soft preference or a constraint by specifying the (feature, label) distribution. For example, the labeled feature “student POS:0.8, O:0.2”, indicates a preference for marking the token “student” with the label “POS” 80% of the time. Generalized Expectation (GE) and Posterior Regularization (PR) are two frameworks studied previously for imposing such preferences in discriminative models [79, 9]. We choose the PR framework since it handles more general constraints and was
found to be better performing in our experiments. In the next subsection, we briefly describe posterior regularization with labeled features for completeness.

5.4.2.1 Posterior regularization

Using the notation from Sutton, et al [78], let the pair \((x, y)\) represent an instance for sequence labeling where \(x\) corresponds to the token sequence and \(y\) represents the label sequence for \(x\). The regularized, conditional log likelihood function optimized in CRFs is given by:

\[
l(\theta) = \sum_{i=1}^{N} \log p(y^{(i)}|x^{(i)}) - \frac{1}{2} \sum_{k=1}^{K} \frac{\theta_k^2}{\sigma_k^2}
\]  

(5.1)

In the above equation, Euclidean norm is used for regularizing the parameter vector, \(\theta\) with the regularization factor given by \(\frac{1}{2\sigma^2}\). Using the Markov assumption, the feature expectations in CRF can be written as \(\sum_y p(y|x)F_k(x, y) = \sum_x \sum_{y_{t-1}} p(y_t, y_{t-1}|x)f_k(y_t, y_{t-1}, x, t)\). The distributions specified via labeled features are converted into expectation constraints in the PR framework as follows: Let \(\phi(x_v, y)\) represent the value of the feature expectation estimated by the model when \(x = v\). If \(b\) represents the target expectation, the constraints corresponding to \(x\) can be written as

\[
Q_{x_v} = \{q_{x_v}(y) : \mathbf{E}_q[\phi(x_v, y)] \leq b\}
\]

Let \(X, Y\) represent the training instances and \(Q\), the desired distribution space representing all constraints, the objective for PR captures the KL-divergence between \(Q\) and the model posteriors to be minimized as:

\[
J_Q(\theta) = l(\theta) - \text{KL}(Q||p_\theta(Y|X))
\]  

(5.2)

The posterior expectations are specified as linear constraints during the parameter estimation process in the PR framework. More details on PR, optimization issues and other forms of the objective function are described in [9].

5.4.3 Labeled features for homepages

We study two types of labeled features:

1. **Dictionary features** capture terms that commonly occur with certain fields. For example, (student, position), and (dept, affiliation).

2. **Proximity features** capture the layout conventions observed by researchers on their homepages. For example, using the notation described in Section 5.3.1, (department, pws1POS, affiliation) indicates a preference for affiliation if the current token is department and few tokens previous to this token have values corresponding to position.

In the description above, we specified the majority labels for the features. More generally, labeled features refer to label-probability distributions. For example, “(student position:0.9, other:0.1)”
indicates a distribution on labels, position and other. Mann and McCallum showed that given limited annotation time, models that were trained using expert-specified labeled features outperform other semi-supervised approaches that use fully-labeled instances in their experiments [7]. Sample expert-designed features pertaining to affiliation are shown in Table 5.5. In this table, the feature “daffl_pws1POS” reads as the token is found in the affiliation dictionary and has a predicted tag, POS from stage-1 within the previous window.

<table>
<thead>
<tr>
<th>Dictionary features</th>
<th>Proximity features</th>
</tr>
</thead>
<tbody>
<tr>
<td>laboratory</td>
<td>nws1UNIV</td>
</tr>
<tr>
<td>centre</td>
<td>capitalized_pws1POS</td>
</tr>
<tr>
<td>college</td>
<td>capitalized_nws1UNIV</td>
</tr>
<tr>
<td>department</td>
<td>dafl_pws1POS</td>
</tr>
<tr>
<td>dipartimento</td>
<td>capitalized_pws1UNIV</td>
</tr>
<tr>
<td>institute</td>
<td>science_nws1UNIV</td>
</tr>
</tbody>
</table>

Table 5.5. Expert-specified dictionary and proximity features for affiliation values. s1+ refers to the predicted label obtained from the first-stage CRF

5.4.4 Extracting labeled features automatically

A labeled feature is a specification of a feature along with a probability distribution related to labels associated with it. Is it possible to extract labeled features automatically given labeled data? This question was briefly addressed in context of classification by using mutual information between the features and labels and Latent Dirichlet Allocation [7]. In context of tagging, frequently occurring features with a given label “that do not also occur frequently with other labels” were used to extract features automatically [83, 8].

To obtain label-probability distributions automatically, a few options were studied in the same works: (1) Majority distribution (MAJ) where the majority label associated with a feature gets the majority of probability mass whereas the remaining mass is distributed uniformly among the remaining labels; (2) Schapire distribution (SCH) where the majority of the mass is uniformly distributed among all labels associated with a feature while the remaining mass is uniformly distributed among the non-associated labels [82]; and (3) Feature-voted (VOTE) distributions where co-occurrence counts of (feature, label) pairs in the training data are normalized to obtain probability distributions. We also propose and evaluate two variant schemes: (4) (MAJ*) is a variant of MAJ where only associated labels in the training data are taken into account. That is, among the associated labels, the winning label gets the majority of the probability mass, whereas the remaining mass is distributed uniformly among the remaining associated labels, and (5) (MAJ**) is similar to MAJ* except that the winning label gets the majority of the mass only if it is a clear winner (has co-occurrence with the feature more than or equal to twice that of other associated labels).

As an example, let \((A, 10), (B, 6), (C, 0) (D, 0)\) be the co-occurrence counts in the training data for a feature for which a distribution is to be generated for labels \(A, B, C\) and \(D\). If we
choose majority probability mass value to be 0.9, the label distributions generated by the various schemes are given by:

- **MAJ**: \{A: 0.9, B: 0.033, C: 0.033, D: 0.033\}
- **SCH**: \{A: 0.45, B: 0.45, C: 0.05, D: 0.05\}
- **VOTE**: \{A: 0.625, B: 0.375\}
- **MAJ\***: \{A: 0.9, B: 0.1\}
- **MAJ\**: \{A: 0.5, B: 0.5\}

Using the (feature, label) co-occurrences in training data we study feature extraction with (1) Information Gain (IG), and (2) Frequency with a given label (Freq). We found that Mutual Information (MI) selected rare words for each label, such as university names among the top words and did not benefit from PR. This “known” issue with MI and details on IG and MI can be found in [86]. In our experiments section, we compare the features extracted using different extraction methods and probability distribution options with the expert-designed labeled features.

### 5.5 Experiments

#### 5.5.1 Datasets and settings

We summarize our experiments on the ArnetMiner profile tagging dataset. This dataset comprises of 898 annotated researcher homepages collected by Tang, et al. for studying record linkage across homepages and DBLP records [11, 4]. These homepages were annotated by seven annotators using the researcher’s profile ontology used in the ArnetMiner system [140]. To the best of our knowledge, this is the only publicly-available dataset available for our problem.

All experiments were performed on a 16-core, 800MHz, 32GB RAM, AMD Opteron, Linux server with default parameter settings in Mallet (Section 5.2). First, the stage 1 CRF is trained with the basic set of features described in Section 5.3.2. Next, predictions from stage 1 are used to generate additional features for the stage 2 CRF. Predictions on the test data are made in a transductive setting with PR. That is, the PR training phase treats the test instances as unlabeled instances and imposes the constraints described by the labeled features on them. The model obtained after this phase is used to obtain predictions for the test instances.

The dictionary labeled features can be used with PR for both the stage 1 and stage 2 CRFs whereas by design, the proximity features can be used only in the second stage. We train the CRF until convergence is obtained on the training split of the data in each experiment. For about 600 training instances, the CRF training time was 30 minutes on average for both stage 1 and stage 2 CRFs. An EM-style optimization algorithm is used in the posterior regularization framework [9]. We analyzed the performance versus running time trade-off for a few sample runs.
(Figure 5.9) and set the number of iterations for the PR part to 50 in the rest of the experiments. The window size was set to 10 for stage 2 experiments since the expert-labeled features were generated based on this size.

As is common in annotation evaluation [86], we use the F1 measure that provides a single metric capturing both precision and recall to study performance with different models.

5.5.2 Expert-labeled features

The affiliation and university field-specific dictionary terms obtained from ArnetMiner (Section 5.3.2) were directly used as dictionary labeled features. For proximity features, we manually examined the homepages in the dataset to derive a list of common layout conventions. For example, “affiliation fields are often preceded by position and followed by university information”, or “phone information is usually listed before fax information”. Next, dictionary terms and surface patterns were used to form conjunctions within a window with the label set $T = \{AFFL, POS, UNIV\}$ (e.g., capitalized pwPOS, department nwPHN). This list was manually examined to obtain a subset of features that satisfy the layout conventions. These “expert” lists of 22 dictionary and 20 proximity features along with their labeled distributions are available in the Supplementary Material (Table 5.5 contains a sample).

5.5.3 Results and observations

![Figure 5.1. F1 variation across folds for affiliation, S1: stage 1 tagging, S2: stage 2 tagging, PR: posterior regularization](image)

![Figure 5.2. F1 variation across folds for university, S1: stage 1 tagging, S2: stage 2 tagging, PR: posterior regularization](image)

Figures 5.3 and 5.4 illustrate the benefits of using expert-specified dictionary labeled features to provide “supervision” when the number of labeled homepage instances to train the initial CRF is small. In this situation, as the plots indicate, (feature, label) distributions are able to effectively boost the tagging performance. The figures show improvements in affiliation F1 from 0.1438 to 0.2088 and in aggregate F1 from 0.4222 to 0.4601 when only ten labeled examples are available. As the number of labeled examples increase, the initial CRF becomes more accurate, with a reduction in the boost with PR.

We summarize our three-fold cross-validation experiments comparing Stage 1 and Stage 2 CRF in Table 5.6. About 600 labeled examples are available in each run and the overall boost in
F1 is not as high as in the previous experiment. However, notice the field-specific improvements obtained by enforcing the expert-specified feature label constraints using PR in Figures 5.6, 5.1 and 5.2. Both dictionary and proximity features provide improvements over that obtained using CRF alone for both the stages.

<table>
<thead>
<tr>
<th>Setting</th>
<th>AFFL</th>
<th>Agg</th>
<th>Agg/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>0.5219</td>
<td>0.7937</td>
<td>0.7615</td>
</tr>
<tr>
<td>S1+PR (Dictionary)</td>
<td>0.5548</td>
<td>0.8124</td>
<td>0.7835</td>
</tr>
</tbody>
</table>

Predictions from Stage 1 (basic)

<table>
<thead>
<tr>
<th>Setting</th>
<th>AFFL</th>
<th>Agg</th>
<th>Agg/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2</td>
<td>0.5431</td>
<td>0.7974</td>
<td>0.7656</td>
</tr>
<tr>
<td>S2+PR (Proximity)</td>
<td>0.5679</td>
<td>0.8047</td>
<td>0.7743</td>
</tr>
<tr>
<td>S2+PR (All)</td>
<td>0.5527</td>
<td>0.8065</td>
<td>0.7767</td>
</tr>
</tbody>
</table>

Predictions from Stage 1 (basic + dict PR)

<table>
<thead>
<tr>
<th>Setting</th>
<th>AFFL</th>
<th>Agg</th>
<th>Agg/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2</td>
<td>0.5303</td>
<td>0.8001</td>
<td>0.7689</td>
</tr>
<tr>
<td>S2+PR (Proximity)</td>
<td>0.5621</td>
<td>0.8082</td>
<td>0.7784</td>
</tr>
<tr>
<td>S2+PR (All)</td>
<td>0.5554</td>
<td>0.8147</td>
<td>0.7862</td>
</tr>
</tbody>
</table>

Table 5.6. Three-fold cross-validation F1 values for Stage 1 (S1) and Stage 2 (S2) models using expert-specified labeled features

A comparison of the F1 values between the “S1” and “S2” rows of Table 5.6 validates our intuition regarding the use of two stages for annotating homepages (Section 5.3.1). It appears that performance-wise a two-staged approach along with proximity constraints is comparable to using a single stage process with dictionary constraints and combination schemes do not provide large enhancements.

The field-specific improvements comparing the basic stage 1 CRF and our best-performing model are shown in Table 5.7. The entries where the improvement in F1 is more than 2% are marked in bold in this table. Imposing constraints via labeled features increases recall for all the fields at the cost of a dip in precision, with the F1 improving for all the fields. The best performance was obtained by imposing all constraints in stage 2 using predictions from stage 1 and dictionary PR.

Our experiments with the feature extraction and distribution assignment schemes described in Section 5.4 are summarized in Table 5.8 for dictionary labeled features (similar trends were
<table>
<thead>
<tr>
<th>Field</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Basic</td>
<td>Best</td>
<td>Basic</td>
</tr>
<tr>
<td>AFFL</td>
<td>0.6670</td>
<td>0.4571</td>
<td>0.4302</td>
</tr>
<tr>
<td>EMAIL</td>
<td>0.9178</td>
<td>0.8889</td>
<td>0.8136</td>
</tr>
<tr>
<td>FAX</td>
<td>0.9543</td>
<td>0.9501</td>
<td>0.9295</td>
</tr>
<tr>
<td>PHN</td>
<td>0.9370</td>
<td>0.9310</td>
<td>0.8899</td>
</tr>
<tr>
<td>POS</td>
<td>0.8048</td>
<td>0.7470</td>
<td>0.5995</td>
</tr>
<tr>
<td>UNIV</td>
<td>0.7203</td>
<td>0.6596</td>
<td>0.5827</td>
</tr>
</tbody>
</table>

Table 5.7. Metadata extraction performance: Three-fold cross-validation F1 values are shown for different fields using the basic and the best performing sets of features.

<table>
<thead>
<tr>
<th>Setting</th>
<th>AFFL</th>
<th>Agg</th>
<th>Agg/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic CRF</td>
<td>0.5219</td>
<td>0.7937</td>
<td>0.7615</td>
</tr>
<tr>
<td>Posterior Regularization Expert</td>
<td>0.5548</td>
<td>0.8124</td>
<td>0.7835</td>
</tr>
<tr>
<td>IG (VOTE)</td>
<td>0.5493</td>
<td>0.8059</td>
<td>0.7757</td>
</tr>
<tr>
<td>IG (MAJ)</td>
<td>0.5224</td>
<td>0.7944</td>
<td>0.7623</td>
</tr>
<tr>
<td>IG (SCH)</td>
<td>0.5512</td>
<td>0.8062</td>
<td>0.7760</td>
</tr>
<tr>
<td>IG (MAJ*)</td>
<td>0.5230</td>
<td>0.7940</td>
<td>0.7623</td>
</tr>
<tr>
<td>IG (MAJ**)</td>
<td>0.5423</td>
<td>0.8046</td>
<td>0.7742</td>
</tr>
<tr>
<td>Freq (VOTE)</td>
<td>0.5415</td>
<td>0.8030</td>
<td>0.7723</td>
</tr>
<tr>
<td>Freq (MAJ)</td>
<td>0.5173</td>
<td>0.7903</td>
<td>0.7576</td>
</tr>
<tr>
<td>Freq (SCH)</td>
<td>0.5514</td>
<td>0.8061</td>
<td>0.7759</td>
</tr>
<tr>
<td>Freq (MAJ*)</td>
<td>0.5261</td>
<td>0.7966</td>
<td>0.7649</td>
</tr>
<tr>
<td>Freq (MAJ**)</td>
<td>0.5459</td>
<td>0.8048</td>
<td>0.7745</td>
</tr>
</tbody>
</table>

Table 5.8. Three-fold cross-validation F1 values for top 30 dictionary labeled features. #Expert features=22

Table 5.9. Extraction performance using posterior regularization (PR) and dictionary constraints. The time in minutes and F1 values are shown for different number of training iterations.

Using information gain along with feature-voted distribution seems to be the best performing scheme among all the configurations although this is lower than that obtained with expert-specified features. Increasing the number of IG features does not help as can be observed in
Figure 5.5. It is likely that IG features might get more accurate when a large amounts of labeled data is available for estimating them. However, the boost obtained using labeled features via the PR framework seems to reduce as more labeled instances become available (Figure 5.3).

5.6 Conclusions

We studied feature labeling for annotating metadata on researcher homepages. We proposed dictionary-based features to capture field-specific hints whereas proximity features capture the layout information among metadata fields. Our proximity features are different from labeled features typically studied in previous work in that they are designed to be used in a second-stage model using predictions from a first-stage model. We showed that posterior regularization can effectively impose the dictionary and proximity hints during the training process to obtain significant improvements in tagging performance when the labeled examples available are limited.
Expert Search

6.1 Introduction

Entity search and retrieval where the goal is to retrieve “objects” (such as cars, books, people) in response to user queries is an emerging research interest in the Information Retrieval community. In particular, expert finding where the goal is to rank people with expertise (experts) in response to a topic query was well-studied in the TREC [137] community. Similarly, the list completion tracks in TREC and INEX [150] competitions address similar entity finding or exemplar-based search for the general domain. In these setups, the evidence for expertise is derived based on webpages or documents on an intranet. Documents in the academic domain are different from webpages in terms of their type (e.g. homepages, publications), structure (e.g. abstract, sections), associated metadata (e.g. venue, authors) and connections (e.g. citations). In this chapter, we focus on expert (researcher/author) search in academic domains in response to both topic queries (e.g. “Information Retrieval”) and name queries (e.g. “Bruce Croft”).

In academic domains, an expert search system that allows queries based on topics and expert names has several potential applications. Consider the following use-cases: The program chair of a conference is desirous of selecting a panel of researchers for the “Information Retrieval” (IR) track. A potential list of program committee (PC) members can be obtained by firing an appropriate topic query to an expert search system. On the other hand, consider a student applying to graduate schools who is interested in working with researchers such as “Bruce Croft” who work on IR problems.

We refer to the use case, where the input query is a “topic” as expert finding or topic-based search whereas similar expert finding or name-based search is used to refer to cases when the input query is an example expert name. For the example in the previous paragraph, since “Bruce Croft” might be associated with several other expertise areas apart from IR, the results of the two queries need not be the same. Previous work for handling expert search involved designing separate models for handling the two query types. Our focus, in this chapter, is to study models...
that are **generic** in that they allow ranking of experts in response to either a topic or a name query.

**Contributions and organization:** Graph-based models are commonly used in IR problems [87]. Although nodes and edges in a graph can be associated with a type (for example, author node, webpage node, citation edge), for the purpose of computing graph-based measures such as path weights, node degree and centrality, type information is unnecessary. We study graph-based models for ranking experts in response to topic-based and name-based queries due to this property. We show experimentally that despite this “indifference” to type information, graph-based models perform on par with models that take type information into account.

Our first model is an extension of PageRank for graphs having multiple edge-types and can be used effectively for ranking experts in response to name queries as well as topic queries by constructing an appropriate query-specific graph in each case (Figure 6.1). The second set of scoring models is based on modeling the underlying corpus using a weighted, undirected, tripartite graph representing the Authors, Documents and Topics (Figure 6.2). In contrast with the PageRank-based models that score nodes depending on their structural connections with other nodes in the graph, the ADT models, are designed to capture the content-based similarity between nodes via edges to the “topic” nodes in the graph. Indeed, author nodes in ADT graphs are scored with respect to document nodes or other author nodes based on the paths connecting them.

We study the performance of our proposed ranking models and baselines on two datasets. The first dataset is based on ArnetMiner and CiteSeer and represents the retrieval environments of academic domains we seek to model whereas the second dataset, the UvT collection from Tilburg University provides a sparse retrieval environment [5]. Our experimental results indicate that despite being “generic” enough to handle both the querying scenarios, graph-based models rank experts on par with other query-specific models.

First, we describe previous work on expert finding both for name-based and topic-based queries in Section 6.2. We focus specifically on similar expert search in Section 6.3 since this problem has not been extensively studied in the academic domain previously. Details of our proposed graph-based models are covered in Sections 6.4 and 6.5. Datasets and experimental setup along with results is presented in Section 6.6. We include some preliminary results of our experiments related to the effect of edge-type weights in PageRank models (Section 6.6.4). Finally, we present conclusions and future directions on expert search in Section 6.7.

### 6.2 Related work

The expert search task was studied in various contexts such as enterprise collections (of TREC’s Enterprise track), sparse data university environments [5] and for bibliographic data and digital libraries [88, 89].

Previously, for topic-based expert finding, experts were ranked using probabilistic models [90, 5], topic models [88, 91, 92], graph-based approaches [93, 89, 94], vector-space [95] and voting
models [96]. By and large, probabilistic methods seem to be the commonly applied models because of their simplicity and run-time efficiency. Balog, et al. proposed probabilistic models for expertise profiling and expert finding in context of sparse data environments such as webpages pertaining to research institutes and universities where the documents are more structured and relatively noise-free [90, 5]. For bibliographic data, Deng, et al. showed that complicated topic models only provide slight benefits compared to probabilistic models with appropriate priors [88]. Indeed, previous studies show that intricate topic models need not always correspond to more effective retrieval [97].

In contrast, with the popular approach to expert search, viz. generative probabilistic models, Fang, et al. proposed a discriminative approach for expert finding where using training data, a known (query, expert) pair is treated as positive example whereas “irrelevant” pairs are treated as negative pairs to learn a ranking function for scoring experts in TREC corpora [98]. These techniques are complementary to our contributions in that if sufficient training data is available, we can apply the same principles to learn a scoring function or even the graph walk and scoring parameters in our model [99, 100, 101]. These aspects are discussed in Section 6.6.4 and 7.2.

Different applications employ variants of HITS [102] and PageRank [103] algorithms for scoring objects [104, 105, 106, 107]. In particular, relevance propagation models for expert search were studied in e-mail collections by Dom, et al. [93] and enterprise collections by Serdyukov, et al [94]. Zhou, et al. [89] proposed a coupled random walk model between authorship networks and citation networks for ranking authors and documents together. Liu, et al. [108] evaluate the influence of an individual author by computing AuthorRank, a query-independent measure similar to PageRank based on the co-authorship network in digital libraries. Most of these models use random walk on graphs containing only author-document links. For including links within author nodes or document nodes, a two-step model is adopted. These models are difficult to extend when more than two types of nodes are involved. We provide a simple extension to PageRank that is capable of folding in evidence of expertise from multiple sources by using multiple transition matrices. In addition, the combining factors for each matrix represents the importance of each matrix on overall scoring. Our proposed model has an additional advantage with respect to the computation of scores for author nodes. The sparsity in the underlying matrices is retained due to which the efficient algorithms proposed by Haveliwala, et al [109], can be adapted to handle our extensions.

The list-completion tasks in TREC [137] and INEX [150] address the similar-entity finding task for the general domain. The proceedings of these competitions discuss various approaches for handling this task. In contrast to our problem, the input queries in these systems, include a query topic description with examples of entities. The participating systems need to extract the relation between the example entities and the topic description and propose entities that hold a similar relation with the topic description, as part of the answer. For similar expert finding the goal is slightly simpler in that the system ranks people in response to a “name” query. Balog and Rijke studied similar expert finding on the TREC data using the relations a candidate expert has with other experts, documents and terms [110]. Hofmann, et al. considered the contextual
factors such as organizational setup and combined them with content-based retrieval scores to find similar experts within an organization [111].

We study similar expert search in academic domains. That is, our goal is to rank other researchers based on their similarity with the queried researcher (name). Although we could not find previous work on similar-entity finding in academic disciplines, previous work exists for predicting researchers to collaborate with. For instance, Chen, et al. presented CollabSeer that uses the structure of the co-author network to predict research collaborators in academic environments [112]. Xu, et al. use a two-layer network model that combines co-author network and researcher-concept network for making researcher recommendations [113]. While co-authorship provides evidence of similarity, we wish to design models that are capable of exploiting other sources of evidence, such as content similarity, citation behavior, etc. Our goal is to predict researchers with similar expertise profiles based on content they generate and not necessarily “co-authors”.

6.3 Similar expert search

We discuss in this section some baseline models for handling similar-expert ranking. In this chapter, “similarity” corresponds to similarity in terms of research interests and expertise (“profile”) and do not take other factors such as professional status (for example, student versus professor) into account.

Various choices are available for representing profiles. For example, an expert profile could be represented using term vectors based on the documents authored by the researcher, probability distribution of topics s/he worked on or structural attributes describing the researcher [88, 94, 5]. For the academic domain, it is typical to measure expertise in an area in terms of a researcher’s publications, descriptions of projects he or she has previously worked on, course contributions, citation information and the academic network involving a researcher.

Irrespective of the underlying representation, for enabling “similar researcher/expert search”, we require that there exists a function that acts on two researcher profiles and outputs a real value, \( \exists f(s_1, s_2) \rightarrow R \), that can be used to compare the closeness between two profiles. Therefore, given the set of researchers, \( E = \{e_1, e_2, \ldots e_n\} \) with profiles \( S = \{s_1, s_2, \ldots s_n\} \), an input researcher name, \( e_q \), and a parameter \( k \), a recommender system retrieves \( E_q \subseteq E \), ranks them using \( f \) and outputs the top \( k \) researchers with profiles most similar to \( s_q \).

Given the set of expertise profiles for researchers, we study the following baseline methods for computing the similarity between two researcher profiles. These methods are based on the computation of document similarity except that the “documents” pertain to author expertise profiles.

1. **Okapi BM25 (OKAPI):** A researcher profile is represented using a vector corresponding to terms in a vocabulary derived based on the content associated with the researcher. Treating one profile as the query and the second as a document, ranking functions employed in IR
can be used to obtain the similarity between them. Consider for instance, the Okapi BM25 ranking function widely used in various IR systems and across text collections. The similarity between two profiles is computed using the BM25 formula as follows:

$$\sum_{w \in s_1} IDF(w) \times \frac{tf(w, s_2) \times (k_1 + 1)}{tf(w, s_2) + k_1 \times (1 - b + b \frac{|D|}{avgdl})}$$

In the above formula, \(IDF(w)\) refers to the inverse document frequency of the word, a measure of rareness of the word computed as:

$$IDF(w) = \log \frac{N}{N(w) + 0.5}.$$  

\(N\) is the total number of profiles in the collection, \(N(w)\), the number of profiles containing \(w\) and \(tf(w, s_2)\), the number of times, the term \(w\) appears in the profile of \(e_2\). The parameter \(k_1\) is typically set to a value between \([1.2, 2]\) whereas \(b\) is typically set to 0.75 in this formula in absence of other information. Additional details on this formula and parameter settings can be found in Jones, et al. [114].

2. **KL divergence (KLD):** In this method, a researcher profile is represented using a probability distribution. For instance, given a set of documents associated with a researcher, the corresponding terms can be modeled using a multinomial distribution [13]. The similarity between two probability distributions is quantified using Kullback–Leibler divergence in machine learning [12]. Given \(\theta_{s_1}\) and \(\theta_{s_2}\), the multinomial probability distributions associated with the profiles of researchers, \(e_1\) and \(e_2\) respectively, KL divergence [12] between them is given by

$$KL(\theta_{s_1} || \theta_{s_2}) = \sum_{w \in s_1} p(w|\theta_{s_1}) \log \frac{p(w|\theta_{s_1})}{p(w|\theta_{s_2})}$$

3. **Probabilistic modeling (PM):** Researchers tend to work on multiple related areas and it might be more more appropriate to model their profiles as topic mixtures instead of a single multinomial distribution. Latent Dirichlet Allocation is a commonly-used topic modeling tool for unsupervised clustering of data and exploratory analysis [10]. We model the set of expertise profiles using \(T\) topics and obtain the topic distribution corresponding to each profile. The similarity between two profiles, \(s_1\) and \(s_2\) can now be measured in terms of the conditional probability of generating the profile \(s_2\), from the profile \(s_1\), \(P(s_2|s_1)\). Assuming conditional independence between \(s_1\) and \(s_2\) given a topic and a uniform distribution on topics and entities this probability can be computed as follows

$$p(s_2|s_1) \propto \sum_{t \in T} p(s_2 \cap s_1|t) * p(t) \propto$$

$$\sum_{t \in T} p(s_2|t)p(s_1|t) \propto \sum_{t \in T} p(t|s_2)p(t|s_1)$$
The above formulae only show terms that affect the relative ranking of profiles with respect to \( s_1 \).

4. **Relevance-based similarity** (REL): He, et al [115]. extended van Rijsbergen’s proposal [116] to use Gleason’s theorem in IR by modeling concepts as vector subspaces that are represented using density matrices. A density matrix is a symmetric, positive semi-definite matrix whose trace is 1. In this formulation the density matrix for a document \( d \) with concepts \( c_i, i = 1 \ldots k \) can be written as \( T_d = \frac{1}{k} \sum_{i=1}^{k} c_i c_i' \) and the probability that a concept \( c \) is relevant to a document \( d \) is computed using the formula, \( p_d(c) = Tr(c' T_d c) \) where “Tr” refers to the matrix trace operator. Let \( c_i, i = 1 \ldots k_1 \) and \( b_j, j = 1 \ldots k_2 \) refer to the concept vectors associated with two researcher profiles, \( s_1 \) and \( s_2 \). The relevance score between these profiles is computed using the formula

\[
Rel(s_1, s_2) = \frac{1}{k_1 k_2} \sum_{i=1}^{k_1} \sum_{j=1}^{k_2} c_i' b_j^2
\]

where \( c_i' b_j \) refers to the dot product value of vectors \( c_i \) and \( b_j \). In our experiments, we use unit TFIDF vectors to build the density matrices corresponding to researcher profiles.

The performance of these baselines for name-based search is presented in Section 6.6. In the next two sections, we describe our graph-based models for expert search. These models provide a uniform means of scoring author nodes in response to name and topic queries.

### 6.4 Extending PageRank for expert search

Objects in digital libraries have nodes corresponding to different objects such as authors, papers, venues and homepages. Similarly, edges in such graphs can represent authorship association, citation links, publication-venue links, etc. Let \( G = (V, E) \) represent a directed graph where \( V \) is the set of nodes and the edges in \( E \) have types assigned to them \( (t \in T) \). We use distinct transition matrices \( (P_t) \) to capture edges corresponding to each edge type, \( t \) and describe the random walk process associated with this graph: a surfer picks a node in the graph arbitrarily to initiate the walk. At every node, \( n \), the surfer chooses at random (with probability, \( w_t \)) an edge-type, \( t \), after which she proceeds to select one of outgoing edges of type, \( t \), uniformly at random. The \( w_t \) values that we refer to as mixing coefficients are indicative of the importance of each edge-type and can be set by using task-dependent domain knowledge.

To ensure that a PageRank vector exists for the underlying graph, we need to ensure that the transition matrix representing the random walk probabilities is aperiodic and irreducible [117]. Given the set of transition matrices capturing edges of a given type, if each matrix is constructed to be aperiodic and irreducible, the aggregate transition matrix obtained by a linear combination
of individual transition matrices is also irreducible and aperiodic.

\[ P = \sum_t w_t P_t \text{ where } \sum_t w_t = 1 \text{ and } \forall w_t, 0 \leq w_t \leq 1 \]

Then, per the Ergodic theorem [117], the matrix \( P \) is guaranteed to have a unique stationary (or limiting) distribution over the nodes of \( V \). This distribution is referred to as the PageRank vector in context of a graph representing webpages and hyperlinks. The PageRank vector can be obtained by computing the principal eigen vector of the transpose of \( P \) (for example, using the power method [118, 119]). The limiting distribution of the nodes or PageRank scores, in effect, capture the “importance” values of the node in the underlying graph based on its edge connections with the other nodes.

Note that the walk process we just described is similar to the random surfer model on web graphs where the steady state distribution is also referred to as the PageRank vector. In the web surfing scenario, PageRank is computed over a graph of nodes representing web-pages and edges representing hyperlinks between webpages. In contrast, the graphs we use for expert search have multiple types of nodes with multiple types of edges between nodes. In the following discussion, by “PageRank”, we in fact refer to our extension of the basic PageRank algorithm to graphs with multiple edge-types.

To ensure that a transition matrix is irreducible and aperiodic, we adopt the approach used by Haveliwala, et al [109]. That is, dangling nodes or nodes with no outgoing edges are handled by adding uniform jumps from them to all other nodes and irreducibility is ensured by adding a teleportation matrix, \( E = [1]_{n \times 1} \times v^T \). Here \( n \) is the number of nodes in \( G \) and \( v = \left[ \frac{1}{n} \right]_{n \times 1} \). \( E \) constitutes one of the transition matrices that combine to give the aggregate transition matrix. The vector \( v \), is called the teleport or personalization vector or reset distribution. Topic-sensitive and personalization variants of PageRank manipulate the entries in \( v \) to bias jumps to distributions of interest [120]. Minkov experimented with random walk models for several entity-similarity tasks and suggested setting the teleport vector based on the nodes initially activated by the input query for capturing the query-dependent aspects [121].

Due to our formulation of the aggregate matrix as a linear combination of individual transition matrices, it becomes straightforward to extend the efficient techniques proposed by Haveliwala, et al. [109] (for computing PageRank) for our case. We next describe the details of these extensions. Let \( P_t \) refer to an individual transition matrix (describing edges of the edge-type \( t \)) and let

\[ P'_t = P_t + d_t v^T \]  
\[ E = [1]_{n \times 1} \times v^T, v = \left[ \frac{1}{n} \right]_{n \times 1} \]  
\[ P'' = \sum_t w_t P'_t + (1 - \sum_t w_t) E \]  

Here \( d_t \) is an \( n \)-dimensional vector for edge-type \( t \) identifying nodes having no edges of type \( t \). That is \( d_t(i) = 1 \) if node \( i \) has no outgoing edges of type \( t \) and is 0 otherwise. The final scores
are obtained by computing the eigenvector of the transpose of the transition matrix, \( A = P^{nT} \). Starting with an initial vector \( x \), this involves repeated computation of \( Ax \) until convergence.

For a single transition matrix, Haveliwala, et al. propose an efficient algorithm for this computation that makes use of the sparsity of the underlying transition matrix and avoids having to explicitly maintain rows for dangling nodes in the transition matrix and instantiating \( E \). We extend these techniques for our model by defining \( P \) and \( t \) as follows:

\[
P = \sum_t w_t P_t
\]

\[
t = \sum_t w_t d_t + (1 - \sum_t w_t) \vec{1}
\]

Algorithm 3 enumerates the steps that need to be repeated until convergence for obtaining the scores for all nodes in the graph for a given query. In particular, nodes of the desired type (author nodes for expert search) can be relatively ranked with respect to each other based on their scores. Using a sparse matrix library, the product \( P^T x \) can be computed efficiently and the remaining computations only require operations on vectors.

**Algorithm 3** Computing \( Ax \) where \( A = P^{nT} \)

\[
\alpha = t^T x
\]

\[
y = P^T x + \alpha v
\]

\[
x = y
\]

### 6.4.1 Obtaining the query-dependent graphs for PageRank

For Expert Finding in response to topic queries, we compute PageRank on query-dependent graphs obtained using the following process: The topic query, \( q \) is used to retrieve the top matching documents in a corpus via a search engine. This set of documents, \( D_q \) is used as an initial set of nodes from which the remaining graph, \( G_q \) is built by expanding the available relations. For instance, the author nodes of the documents can be added via the written-by edges whereas other documents can be added via the cited-by edges. Unless otherwise mentioned, we use a uniform reset distribution for \( v \). We also study the benefit of using the retrieval scores of the initial set of document nodes to obtain the reset distribution. This setting was found to be beneficial in some similarity computation tasks by Minkov, et al [121].

The PageRank model can also be used to score other author nodes in response to a query-author node to enable name-based search: We construct the query-specific graph for this scenario as follows. The documents associated with the queried author, \( a \), are first retrieved to form the initial set of documents. This document set is expanded by adding other documents (via citation edges or neighbors based on content similarity). Note that this expansion step is crucial to ensure that we do not only capture “co-authors”. However, once an initial document set is obtained, the process is similar to that of the topic-based search case. For example, consider the graph in
Figure 6.1. An example author-document graph to illustrate the PageRank-based model

Figure 6.1 and let $A_3$ correspond to the query node. The documents, $D_2$ and $D_3$ are retrieved via their direct association with $A_3$ (documents related to author $A_3$) whereas $D_4$, $A_2$ and $A_4$ correspond to the nodes added via our expansion process (co-authors of documents, $D_2$ and $D_3$, document $D_4$ cited by $D_3$). The shaded nodes comprise the query-dependent graph used for computing PageRank scores.

6.5 The Author-Document-Topic graph for expert search

Most recent works in text and document analysis adopt the view that a document is a mixture of a small number of topics. Indeed, models such as Latent Dirichlet Allocation (LDA) and probabilistic latent semantic analysis (pLSA) target the extraction of abstract concepts or topics given a collection of documents [30, 10]. These models also enable the expression of a document in the corpus in terms of its topic proportion vector that corresponds to a low-dimensional representation of the document. Given a set of documents, authored by an expert, generative distribution on topics and terms can be estimated for that expert. LDA was effectively used to model scientific documents and their authors previously [18, 122].

Although modeling of authors in terms of their topic distribution is intuitive, we demonstrate experimentally that using similarity between author topical profiles did not yield good performance for similar expert finding (Section 6.6). For expert retrieval in response to topic queries, LDA-style models were shown to only yield marginal benefits over simpler probabilistic counterparts [88]. Are there alternate ways to compute the similarity between authors while retaining the topical aspect of document representation? We seek to address this question via the Author-Document-Topic (ADT) representation. We start by expressing documents in terms of their topics using content modeling tools. Next, the author-document and document-topic associations are represented via edges in a graph and paths within this graph are used to measure the similarity between any pair of nodes in the graph.
6.5.1 Constructing the Author-Document-Topic graph

Let $T$ represent the set of topics associated with a document collection, $D$. Intuitively, an expert on a topic, $t \in T$ would have authored documents related to $t$ and other closely-related topics. Similarly, if an author, $a$ has expertise on a topic $t \in T$, authors similar to $a$ could be expected to write about $t$ and topics related to $t$. The associations between documents and their authors and documents and their topics can be represented by a weighted tri-partite graph as follows: Let $G = (V, E)$ represent such a graph where the vertex set, $V = A \cup D \cup T$ is the union of author, $A$, document, $D$ and topic nodes, $T$. Edges between $A$ and $D$ reflect the authorship relation between documents and authors whereas edges between $D$ and $T$ reflect the topical association of documents.

![Figure 6.2. An example Author-Document-Topic (ADT) graph](image)

Weights assigned to the edges in ADT capture the association strength between two nodes. For instance, in our experiments, we assign a uniform weight of 1 to all edges between author and document nodes whereas edges between document and topic nodes are assigned weights in correspondence with the proportion of that topic in the document (a positive real number). Document modeling tools such PLSI and LDA can be used for estimating these topic proportions. An example ADT graph is shown in Figure 6.2.

We now describe how the ADT is constructed for a given query. Consider for example, a given document node, $d$. First, the weighted links between this document node and its associated topic nodes are added using the proportions obtained from LDA (or PLSI). Next, other document nodes related to these topics are added to the graph along with the related edges. Finally, the known author links between the document and author layers are added to obtain a document-specific graph. If a set of documents is given as input instead of a single document (for example, 1Note: In this chapter, we use the term ‘topic’ in two senses. The first is in the context of the ADT graph and refers to the topics or concepts as extracted by tools such as LDA. We also use the term in the sense of topic-queries (such as “information extraction”) in context of expert finding.)
documents retrieved in response to a topic query), we proceed, as in the case of a single document, by incrementally building the subgraph related to each document in the set to obtain the final graph. Instead, if for a name-query, an author node, $a$, is specified as input we first retrieve the documents associated with $a$ and then follow the same procedure as with that of a set of documents.

Several well-known measures exist for comparing two nodes in weighted graphs (particularly for directed graphs). For instance, if the edge weights represent distances between nodes, one can compute proximity between two nodes in terms of the shortest path between them. In the field of network analysis, researchers have studied measures for computing vertex similarity and importance scores. A recent comparison of network similarity measures for various recommendation tasks was studied by Boldi, et al [123]. In particular, vertex similarity measures in co-authorship networks were evaluated for collaborator recommendation by Chen, et al [112].

6.5.2 Scoring author nodes using paths in the ADT graph

We found that network centrality measures such as degree centrality and PageRank values of nodes in the ADT graph are not useful for expert ranking tasks. This is not surprising since in general centrality measures capture a node’s impact with respect to the overall graph structure. These measures are not very meaningful in context of the ADT graph due to its undirected nature and the heterogeneity of the underlying nodes. Moreover, our focus is on estimating a node’s similarity with reference to the query nodes rather than its influence at a macroscopic level. To capture this aspect, we propose measuring the similarity between two nodes in a graph in terms of strength of paths between the nodes. We used the phrases “association strength” and “path weights” interchangeably in this chapter. Weight of an edge corresponds to assignments made during construction of the ADT. Author-document edges get a weight of 1, whereas document-topic edges are weighed according topic proportions estimated from LDA (Section 6.5.1). Let $weight(e)$ correspond to the weight for edge $e$, and $p$ be a path between nodes $a$ and $d$ comprising of edges such that $p = e_1e_2...e_n$. Let

$$sweight(p) = \sum_i weight(e_i)$$

$$pweight(p) = \prod_i weight(e_i)$$

Let $P(a,d)$ be the set of all paths between nodes $a$ and $d$. We studied the following schemes for computing similarity between $a$ and $d$:

1. **MaxPath** The similarity between two nodes is given by the path between them having the maximum association strength. If we assign, weights to edges using a transformation function that assigns weights to edges that are inversely proportional to association strengths,
this scheme picks the path having the shortest distance between the nodes.

$$score(a, d) = \max_{p \in P(a, d)} sweight(p)$$

2. **SumPaths** Consider a document that is related to two topics. An author who is associated with both the topics should be assigned credit for both the topics. The SumPaths scoring method seeks to capture this intuition by aggregating scores of all paths between two nodes in the ADT graph.

$$score(a, d) = \sum_{p \in P(a, d)} sweight(p)$$

3. **ProductPaths** This scoring method is similar to SumPaths but we use a multiplicative scheme for aggregating edge association into a path association.

$$score(a, d) = \sum_{p \in P(a, d)} pweight(p)$$

For a running example, suppose we would like to score the node \(A_1\) with respect to \(D_1\) in Figure 6.2. Assume the topic proportions for the documents are given as:

\(D_1 = \{0.2, 0.8, 0, 0, 0\}\), \(D_2 = \{0.5, 0.5, 0, 0, 0\}\)

\(D_3 = \{0.7, 0, 0, 0, 0.3\}\), \(D_4 = \{0, 0.4, 0, 0.6\}\)

There are three paths between \(D_1\) and \(A_1\), viz.

\{(\(D_1, T_1\), (\(T_1, D_2\), (\(D_2, A_1\))\)}

\{(\(D_1, T_2\), (\(T_2, D_2\), (\(D_2, A_1\))\)}

\{(\(D_1, T_1\), (\(T_1, D_3\), (\(D_3, A_1\))\)}

With the \textit{sweight} function, these paths have weights, 1.7, 2.3 and 2.1 respectively whereas with the \textit{pweight} function, the paths are assigned scores, 0.1, 0.4 and 0.14 respectively. Therefore, the score of \(A_1\) with respect to \(D_1\) is 2.3 with MaxPath, 6.1 with SumPaths and 0.64 with ProductPaths.

Notice that our scoring schemes are indifferent to the choice of node in the sense that all nodes are treated equally and given a node of any type, other types of nodes are assigned scores purely based on the association strengths as captured by the scoring schemes described above. This aspect is a virtue of graph-based models where, query objects and the objects to be scored are both nodes and the assigned scores are simply due to some property of the underlying graph (e.g. paths, degree of nodes). There is an important difference between the ADT graph models when compared with the PageRank-based models. Due to the types of nodes and semantic meaning.
behind edges between nodes, similarity of nodes in the ADT models pertains to similarity based on the **topical profiles** whereas the PageRank scores capture importance based on **structural connections** in the graph. For example, an “influential” author with highly-cited relevant documents is likely to be scored higher in the PageRank scoring model due to multiple connections in the underlying graph.

### 6.6 Experiments and results

To the best of our knowledge, no standard datasets exist for evaluating the similar expert finding task in the academic domain. However, sets of topic queries and lists of experts (traditionally referred to as qrels in TREC) for each topic are available for evaluating topic-based expert finding from ArnetMiner and Tilburg university (the UvT collection).

1. The ArnetMiner (AM) dataset provides topic+experts lists previously used by Tang, et al. and Deng, et al. [88] for studying topic-based expert finding in academic domains. We mapped these researcher names to the author names in CiteSeer\(^X\) using exact string matching. To obtain a suitable corpus for modeling the researchers, we collected a subset of document abstracts from CiteSeer\(^X\) by matching venues of documents with the keywords from venue names listed on Wikipedia [144].

2. The UvT collection was made available via the Webwijs system developed at Tilburg University [151]. This collection contains information on UvT employees who are involved in research or teaching along with their homepages, research profiles, publications and course pages. The list of topics+experts available with this collection was used to study expert finding and profiling tasks in sparse environments by Balog, et al [5]. We only considered pages in this collection that are in English.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Corpus Size</th>
<th>Total Authors</th>
<th>Queries</th>
<th>QRel Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM</td>
<td>ArnetMiner/CiteSeer</td>
<td>103838</td>
<td>27108</td>
<td>13</td>
<td>901</td>
</tr>
<tr>
<td>UvT</td>
<td>The UvT Collection</td>
<td>19127</td>
<td>1168</td>
<td>203</td>
<td>1751</td>
</tr>
</tbody>
</table>

**Table 6.1.** Summary of datasets used in expert search experiments

Both the AM and the UvT datasets have ‘topic’ queries and manually-identified lists of authors with expertise for each topic. For evaluating similar expert finding, we need the list of experts that are similar to a given expert. We created test datasets for similar expert finding as follows: for a given topic query, from the set of experts listed with the query, we randomly choose one of the experts as the “name query”. The other experts in the set comprise the similar researchers (or the ‘gold’ list) for this query. Due to the manner of construction, our ‘gold’ lists are in fact conditioned on the topic. Despite this dependence, we think that this dataset can be used for comparing the relative performance across ranking models. We restrict ourselves to topics that have at least five experts listed with them while forming our test set. A summary description of the datasets thus obtained is given in Table 6.1.
68

Table 6.2. Each entry represents the number of authors having a given number of documents associated with them. For example, the number of authors in UvT having only one document is 384.

<table>
<thead>
<tr>
<th>NumDocs</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM/CSX</td>
<td>n/a</td>
<td>8153</td>
<td>2994</td>
<td>717</td>
<td>167</td>
</tr>
<tr>
<td>UvT</td>
<td>384</td>
<td>65</td>
<td>40</td>
<td>22</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 6.3. Sample topics from the CiteSeer corpus

<table>
<thead>
<tr>
<th>99</th>
<th>estimation statistics probability regression model statistical distribution estimators</th>
</tr>
</thead>
<tbody>
<tr>
<td>98</td>
<td>lines prior summary top half reflects implication patterns trends greater</td>
</tr>
<tr>
<td>90</td>
<td>index cluster clusters space target ranking coming multi group collected</td>
</tr>
<tr>
<td>89</td>
<td>mind important made sense relation make arguments common consists remarks</td>
</tr>
<tr>
<td>87</td>
<td>markets industrial journal firms organization competition economics collusion oligopoly</td>
</tr>
<tr>
<td>86</td>
<td>ethics law moral ethical social legal morality politics human society theory</td>
</tr>
<tr>
<td>83</td>
<td>republic europe poland czech hungary eastern state german west east central</td>
</tr>
<tr>
<td>78</td>
<td>asia regions areas india africa rural agricultural urban historical america</td>
</tr>
<tr>
<td>72</td>
<td>criminal crime law justice police european investigation court victims prosecution</td>
</tr>
<tr>
<td>71</td>
<td>face brain related expressions facial cognitive emotion affective emotional expression</td>
</tr>
</tbody>
</table>

Table 6.4. Sample topics from the UvT collection

6.6.1 Baselines

We briefly summarize some models previously proposed for topic-based expert finding and similar expert finding.

1. **Expert finding**: Balog, et al. [90, 5] use the estimates of $p(ca|q)$ where $q$ is the query and $ca$ is a candidate for ranking experts. $p(ca|q) = \frac{p(ca,q)}{p(q)}$ and $p(ca,q)$ is defined as

   $$p(ca,q) = \sum_{d \in D} p(d)p(ca,q|d) = \sum_{d \in D} p(d)p(q|d)p(ca|d,q)$$

   (6.4)

$D$ is the set of documents related to the query that a candidate $ca$ is associated with. Assuming $ca$ is conditionally independent of $q$ given a document one can write $p(ca|d,q) = p(ca|d)$ and treating $p(d)$ and $p(q)$ as uniform:

   $$p(ca|q) \propto \sum_{d \in D} p(q|d)p(ca|d)$$

$p(ca|d)$ is defined as $\sum_{c \in C} a(d,ca)$ where $C$ is the set of all candidates and $a(d,ca)$ is the
association between document \( d \) and candidate \( ca \). For example, assuming uniform association among authors, if a document has three authors, the \( p(ca|d) \) score for each author is \( \frac{1}{3} \). The \( p(q|d) \) scores for a document are estimated using language modeling [90, 13].

Deng, et al. extended the probabilistic model proposed by Balog, et al. for bibliographic data [88]. The \( p(ca|d) \) values were defined as \( \frac{1}{n_d} \) or 0 depending on whether \( ca \) is the author of \( d \) and \( n_d \) is the number of authors for \( d \). The prior probability, \( p(d) \), was defined in terms of the number of citations that the document has. For example, \( p(d) \propto \ln(e + c_d) \) where \( c_d \) is the citation number for \( d \). This probabilistic model forms a competitive baseline and complicated author-topic models were shown to obtain only marginal improvements for topic-based expert finding in bibliographic data [88].

2. Similar expert finding: Adaptations of expert finding techniques were studied for similar expert finding by Balog, et al. [110] where the authors in the collection are represented via their profiles constructed using the documents associated with them. Experts are recommended based on the profile similarity between two authors. We implemented the baseline methods proposed in Section 6.3 and chose the best performing of them for comparison with the graph-based models. For profile based similarity in our datasets, using the TFIDF vector representation along with Okapi scoring out-performed the other techniques.

![Figure 6.3. Comparison of baselines for name-based search on ArnetMiner dataset](image)

![Figure 6.4. Comparison of baselines for name-based search on the UvT collection](image)

We see from Table 6.1, that the AM and UvT datasets correspond to rather different operating environments. The number of documents and authors and documents per author is higher in the CiteSeer/ArnetMiner although known expert-qrels are less in number. Documents in CiteSeer represent research publications and have citation links between them. In contrast, the UvT collection is sparse in that the documents mostly comprises webpages (such as homepages and research descriptions). The number of documents per expert is limited and very few experts have more than ten documents associated with them (Table 6.2).

For ADT models the underlying document collection is represented in terms of topics. The “number of topics” comprises a tunable parameter for most document modeling tools such as pLSA or LDA. This number is typically chosen to maximize the “likelihood” or “perplexity” of a held-out set of documents in the corpus [19]. However Azzopardi et al. showed that low perplexity
representations do not necessarily result in high precision/recall for retrieval problems [97]. For this reason, we tune the “number of topics” parameter separately for each task on each dataset by using a held-out set of queries. In each dataset, 20% of the queries (randomly-selected) comprise the held-out set whereas the other 80% of the queries are used for evaluation and comparisons across models (Tables 6.9 and 6.10). The number of topics is set to that value which gave the maximum recall on the held-out queries.

We present sample topic-terms for both the CiteSeer and UvT document collections in Tables 6.3 and 6.4 respectively. These topics clearly highlight the difference in the two document collections. The CiteSeer collection corresponds to documents in Computer Science and related areas and consequently the identified topics are closely related in the area. In contrast, the document collection in UvT pertains to a broader set of topics. We contend that the retrieval performance is typically better on the UvT collection since it is easier to discriminate between expertise areas corresponding to the authors.

6.6.2 Evaluation setup

The document collections in both datasets were indexed using the search engine, Indri. As described in Chapter 2, Indri uses language modeling techniques for ranking documents in response to topic queries. We evaluate the performance of our models at different number of retrieved results \(k = 10, 20, 30, 40, 50\). In our expert finding experiments, we set the size of document set retrieved in response to the topic queries to 100 per query. The document-topic associations were obtained by running the LDA implementation provided as part of Mallet.

We measure the performance of our proposed and baseline models using the recall, mean average precision (MAP), mean reciprocal rank (MRR) measures described in Section 2.1. For expert finding, recall is an important measure since it captures the number of correct experts in the gold lists that are retrieved by a method. Therefore, we use recall as the criterion for choosing the number of topics or the appropriate baselines to compare with.
6.6.3 Results and observations

The performance of ADT models on the held-out queries for different number of topics is shown in Figures 6.9 and 6.10. As can be seen in these figures, “number of topics” for modeling the documents in the corpus forms a crucial parameter that affects the performance of the ADT models. Moreover, the best setting for number of topics is task-dependent. We found $num\text{Topics} = 400$ works well for both the tasks on the UvT dataset whereas for the AM dataset, $num\text{Topics} = 450, 500$ are the best settings for name-based and topic-based search respectively.

Our next set of experiments compares the different scoring schemes proposed for ADT models in Section 6.3 on the AM dataset for both topic-based search and name-based search (Figures 6.11 and 6.12). The SumPaths and ProductPaths schemes clearly out-perform the MaxPath scheme that assigns scores based on the best-path connecting two nodes. For our tasks, it appears important to accumulate the scores along various paths between a given set of nodes, a feature captured by both SumPaths and ProductPaths scoring schemes. This accumulation corresponds to assimilating scores from multiple documents associated with an authors unlike MaxPath where only the most relevant document is considered. Similar behavior is observed on the UvT dataset in Figures 6.5 and 6.6. For comparison with other models, we choose ProductPaths since it is slightly better than SumPaths on the held-out queries.

Next, we compare our proposed models (ADT and PR) with the state-of-the-art baselines for
these tasks. Figures 6.7 and 6.8 show the recall of the models for different number of retrieved results for the ArnetMiner dataset whereas figures 6.13 and 6.14 show similar plots for the UvT dataset. As can be seen in these figures, graph-based models (PageRank or ADT) typically outperform or perform on-par with the problem-specific baseline models in terms of recall on both the datasets. The other retrieval measures are summarized in Tables 6.5 and 6.6. For the UvT collection, simpler TFIDF-based methods seem to be better at ranking the results (as captured by MRR, for instance). We used the default setting of assigning edge-type weights \( (w_t) \) equally in our PageRank models. The citation edges were used to expand the document set in case of name-based search in the ArnetMiner dataset whereas content-similarity (top 100 neighbors) was used to expand the set for the UvT collection (Section 6.4).

Our experiments highlight results that indicate that unified models can be designed for handling both the querying scenarios in expert search without compromising on the retrieval performance. The probabilistic model was chosen as baseline for topic-based search based on its competitive performance shown in previous work [88]. For name-based search, since existing work on this problem is still preliminary, we evaluated all applicable models previously proposed for this problem [110] on our datasets and choose the best performing models as baselines (Figures 6.3 and 6.4).

Finally, for the sake of illustration, we provide anecdotal examples in Tables 6.7 and 6.8. These lists of top 10 recommendations were retrieved using the ADT model with the ProductPaths
scoring scheme. We chose popular subject areas in Computer Science as “topic queries” and known experts in these fields as “name queries”. While not all entries are perfect we found upon manual examination that most recommended authors typically publish in conferences in the related subject areas (as listed on DBLP). The entries that do not seem relevant to the specified query are highlighted in italics in these tables.

### 6.6.4 Role of edge-type weights in PageRank-based models

In all our experiments presented so far, we set the teleport vectors to uniform distribution over all nodes. Similarly, all edge types were assigned equal weights. This weight was obtained by uniformly splitting the mass corresponding to (1-the damping factor) where the damping factor was set to 0.15 according to common practice [119]. However, the edge-type weights and the

<table>
<thead>
<tr>
<th>ArnetMiner</th>
<th>UvT</th>
</tr>
</thead>
<tbody>
<tr>
<td>BL(Okapi)</td>
<td>PR</td>
</tr>
<tr>
<td>Prec@10</td>
<td>0.2699</td>
</tr>
<tr>
<td>MRR@10</td>
<td>0.5792</td>
</tr>
<tr>
<td>MAP@10</td>
<td>0.1614</td>
</tr>
<tr>
<td>Prec@50</td>
<td>0.1720</td>
</tr>
<tr>
<td>MRR@50</td>
<td>0.5908</td>
</tr>
<tr>
<td>MAP@50</td>
<td>0.0793</td>
</tr>
</tbody>
</table>

Table 6.5. Evaluation summary for name-based search

### 6.6.6 Role of edge-type weights in PageRank-based models

In all our experiments presented so far, we set the teleport vectors to uniform distribution over all nodes. Similarly, all edge types were assigned equal weights. This weight was obtained by uniformly splitting the mass corresponding to (1-the damping factor) where the damping factor was set to 0.15 according to common practice [119]. However, the edge-type weights and the

<table>
<thead>
<tr>
<th>ArnetMiner</th>
<th>UvT</th>
</tr>
</thead>
<tbody>
<tr>
<td>BL(Prob)</td>
<td>PR</td>
</tr>
<tr>
<td>Prec@10</td>
<td>0.3300</td>
</tr>
<tr>
<td>MRR@10</td>
<td>0.5009</td>
</tr>
<tr>
<td>MAP@10</td>
<td>0.1844</td>
</tr>
<tr>
<td>Prec@50</td>
<td>0.1980</td>
</tr>
<tr>
<td>MRR@50</td>
<td>0.5009</td>
</tr>
<tr>
<td>MAP@50</td>
<td>0.0987</td>
</tr>
</tbody>
</table>

Table 6.6. Evaluation summary for topic-based search

### Table 6.7. Top 10 recommendations made by author-document-topic (ADT) models for example topic queries

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Hermann Ney</td>
<td>Raymond J. Mooney</td>
<td>W. Bruce Croft</td>
<td>Ian Horrocks</td>
</tr>
<tr>
<td>Aravind K. Joshi</td>
<td>Vasant Honavar</td>
<td>Douglas W. Oard</td>
<td>Dieter Fensel</td>
</tr>
<tr>
<td>Raymond J. Mooney</td>
<td>Manuela Veloso</td>
<td>Hermann Ney</td>
<td>Enrico Motta</td>
</tr>
<tr>
<td>Bonnie J. Dorr</td>
<td>Jude Shavlik</td>
<td>Jamie Callan</td>
<td>Amit Sheth</td>
</tr>
<tr>
<td>Alex Waibel</td>
<td>David B. Leake</td>
<td>Hector Garcia-molina</td>
<td>Steffen Staab</td>
</tr>
<tr>
<td>Martha Palmer</td>
<td>Peter A. Flach</td>
<td>Justin Zobel</td>
<td>Frank Van Harmelen</td>
</tr>
<tr>
<td>Kathleen Mckeown</td>
<td>Pat Langley</td>
<td>C. Lee Giles</td>
<td>Stefan Decker</td>
</tr>
<tr>
<td>Udo Hahn</td>
<td>Yoram Singer</td>
<td>Shih-fa Chang</td>
<td>Rudi Studer</td>
</tr>
<tr>
<td>Alon Lavie</td>
<td>Ryszard S. Michalski</td>
<td>Alex Waibel</td>
<td>Wolfgang Nejdl</td>
</tr>
<tr>
<td>Bonnie Webber</td>
<td>Johannes Furnkranz</td>
<td>Jaap Kamps</td>
<td>Tin Finin</td>
</tr>
</tbody>
</table>
reset distribution are parameters that can be used to tune the performance of our PageRank model. We illustrate this point via anecdotal runs summarized in Tables 6.9 and 6.10.

<table>
<thead>
<tr>
<th>Method</th>
<th>Pr@30</th>
<th>R@30</th>
<th>MAP@30</th>
<th>MRR@30</th>
</tr>
</thead>
<tbody>
<tr>
<td>ad=0.1 da=0.4 dd=0.4</td>
<td>PR-A 0.1238 0.0188</td>
<td>0.1608 0.3776</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PR 0.1143 0.0160</td>
<td>0.1394 0.5187</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ad=0.3 da=0.3 dd=0.3</td>
<td>PR-A 0.1143 0.0169</td>
<td>0.1462 0.3004</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PR 0.1143 0.0156</td>
<td>0.1364 0.5163</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ad=0.4 da=0.4 dd=0.1</td>
<td>PR-A 0.0905 0.0136</td>
<td>0.1219 0.2595</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PR 0.1000 0.0142</td>
<td>0.1116 0.3792</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ad=0.1 da=0.2 dd=0.6</td>
<td>PR-A 0.1238 0.0189</td>
<td>0.1709 0.3805</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PR 0.1238 0.0180</td>
<td>0.1431 0.5099</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.9. Performance on topic-based search with the ArnetMiner dataset, PR: PageRank, PR-A: PageRank with “activated nodes” reset distribution, a: author, d: document, ad, dd, da: edge-type parameter settings

The edge-type weights and reset distributions can be estimated either based on training data or set using domain knowledge. For example, a domain expert might indicate that for the expert search application, author-publication links are more important than author-coursepage links in the UvT collection. Recent work has illustrated that when suitable training data is available, these parameters can be learnt automatically [124, 125, 99, 100, 101].

In Table 6.9, we set the initial set of retrieved documents from which the graph for PageRank was extracted to 100. That is, the graph is constructed by first obtaining the top 100 set of document nodes retrieved with the query and expanding it with the set of author nodes, citation edges, and author edges. ad represents the weight on edges connecting authors to documents, da the weight on edges connecting documents to authors and dd, the weight of the citation edges. The PR rows represent the PageRank runs whereas PR-A represent runs with the vector v set to the initially activated nodes whose distribution depends on their query-dependent retrieval scores.
For the UvT collection, since the different available document types can be retrieved, we use the query to obtain the top 100 homepages(h), research descriptions(r), publications(p) and course pages(c) and use the associations provided in the dataset to obtain the associated authors (a) to complete the graph on which PageRank is run. In Table 6.10, ac, ah, ap and so on refer to the edge-type weights connecting these different nodes.

As indicated in Tables 6.9 and 6.10, the reset distribution and the edge-type weights affect the performance of the PageRank models and these parameters need to be tuned for best performance. We argue that the PageRank-style models are preferable due to the extendability they afford when additional sources of expertise become available.

### 6.7 Summary

In this chapter, we presented graph-based models for enabling expert search in response to name and topic queries. We proposed the ADT models to rank experts based on content-similarity captured via document-topic edges. In contrast, our PageRank-based models rank experts using the structural connections between authors and documents and within documents. We showed via experiments that our graph-based models are capable of providing a unified framework for ranking experts in response to both name and topic queries. In addition, we showed that these models demonstrate retrieval performance on-par with problem-specific models.
A good direction to pursue is extending our expert retrieval models for panel recommendation. In contrast with topic or name-based search, recommending panels of experts imposes several problem-specific constraints. For instance, we may wish to ensure diversity in the list of experts with respect to affiliations. Similarly, personal preferences of authors with respect to their compatibility each other may need to be accounted for during panel recommendation [126].
Future directions

In this chapter, we summarize a couple of future directions for research based on the findings in this thesis. We present a brief background and outline possible solutions for two problems.

7.1 Committee selection

7.1.1 Background

Committee selection or recommendation is an extension to the Expert Search problem. Instead of retrieving experts for one (specific) topic as in expert search, the goal in committee selection is to select a group of individuals who collectively have expertise on a set of topics. While it may appear that expert search systems can be easily used to address this problem, the challenge in committee selection is that the task includes a set of constraints that the group of chosen individuals (i.e., the solution) must satisfy.

Consider as an example, selecting a committee for reviewing papers in a conference. Apart from requiring the chosen individuals have expertise on the topics pertaining to the conference, it is desirable to include restrictions such as the number of members belonging to a particular affiliation, preferences in terms of experience of the individual and ensure diversity in the committee. Similarly, while selecting a thesis committee for a graduate student, the system might be required to satisfy constraints related to university-specific requirements on the committee members. Indeed, most real-world committee selection problems involve constraints.

7.1.2 Problem details

The committee recommendation problem can be described concisely as follows: Let $E$ represent the set of all experts $\{e_1 \ldots e_N\}$ and $A$ be the set of all expertise areas such that $A = a_1 \ldots a_M$. Let $S = \{S_1 \ldots S_N\}$ represent the collection of subsets of expertise areas associated with each expert. That is, $S_i$ is the set of expertise areas for expert $e_i$ and $S_i \subseteq A$. Each individ-
ual also has attributes that are categorical values related to metadata of the individual (e.g. affiliation='psu', position='professor').

In this section, we refer to the set of expertise areas for which a committee needs to be chosen, that is, the input to the problem as a task, T. We use a multiset of expertise areas to represent T in order to capture the notion of requiring multiple individuals to be proficient in a given expertise area. We also assume that constraints are expressed using boolean functions on attributes of the individuals. For example, “at least one member should have affiliation='psu'”.

In cases where a committee needs to be chosen for a subset of expertise areas, such that $T \subseteq A$, Lappas, et al. noted that the solution corresponds to the set cover problem [127] in the absence of constraints. Their goal was to select committees that meet the expertise requirements as well as have the minimum communication cost among the members where cost is defined using the weight of edges between the member nodes in the compatibility graph [126]. Consider instead, the committee selection problem where tasks are represented as multisets and constraints specified via boolean functions. To the best of our knowledge, this setting has not been studied previously although it is a more accurate representation of committee selection in the real world.

In the following discussion, we restrict ourselves to constraints on attribute values that cannot be violated in the solution. For example, “cannot have any member who is a grad student” and “cannot have 3 members belonging to the same affiliation”.

The boolean functions that describe constraints take as input a set of attribute values and return “true” or “false”. For example, to enforce a constraint that not more than 2 people in the chosen committee must be from the same university, we could define a predicate function that takes as input the committee chosen so far, and a proposed member, check if the proposed member has the same affiliation as the two members already in the committee and appropriately return true or false. We propose a solution based on budgeted maximum coverage for the committee selection problem under the above assumptions [128].

### 7.1.3 A possible solution using set coverage

**Algorithm 4 Greedy algorithm for Committee Selection**

```
greedyCommitteeSelection(S, C, F)
G = φ
repeat
    Si ← F(S, G, C)
    if cost(G) + ci ≤ L then
        G ← G ∪ {Si}
    end if
    S ← S \ Si
until S = φ
```

In the budgeted maximum coverage problem, the collection of sets, $S = \{S_1, S_2, \ldots S_N\}$ with costs $\{c_i\}_{i=1}^N$ is defined over the domain of elements $A = a_1, a_2, \ldots a_M$ with associated weights $\{w_i\}_{i=1}^M$. The goal is to find a collection of sets $S' \subseteq S$, such that the total cost of elements in
$S'$ does not exceed a budget $L$ and the total weight of elements covered by $S'$ is maximized. Khuller, et al. proposed a greedy algorithm that is a $(1 - 1/e)$–approximation algorithm for the budgeted maximum coverage problem [128].

Let $G \subseteq S$ and $w(G)$ and $\text{cost}(G)$ represents the total weight and cost of elements covered by the sets in $G$. Further, $W_i$ is the total weight of elements covered by a set $S_i$, whereas $W'_i$, $i = 1, \ldots, m$ denotes the total weight of elements covered by the set $S_i$ but not covered by any set in $G$. The greedy heuristic picks at each step a set that maximizes the ratio $W'_i/c_i$ until the total cost does not exceed the budget, $L$, for inclusion into $G$.

Algorithm 4 is a slight modification of the greedy algorithm proposed by Khuller, et al. In the algorithm, $F$ is a black-box function that takes as input $C$, the set of predicate functions, $G$ the subsets representing the experts added into the committee so far and $S$ the set of expertise areas. The weights specify the expertise values that need to be satisfied (or maximized for task, $T$). The function $F$ returns $S_i$ the set in $S$ that has the maximum value of $W'_i/c_i$ among the sets in $S - G$ that satisfy $C$.

### 7.2 Learning edge-type weights for PageRank models

#### 7.2.1 Background

As illustrated in Section 6.6.4, the performance of the expert search task using our extended PageRank model depends on the parameter settings for the edge-type weights and the reset distribution. Learning parameters for random-surfer models given task-specific training data is a well-studied problem [124, 125, 99, 100, 101]. Given a set of training examples with desired rankings, most approaches design techniques for learning reset distributions or entries of the transition matrices by framing the mismatch in the model-predicted and “correct” distributions as an optimization problem. Due to specific requirements on the parameters (for e.g. the matrix should be a stochastic matrix), the output solutions also need to satisfy certain constraints.

Most previous approaches to learning parameters for PageRank and related algorithms do not study the optimization of the jump coefficients (or edge-type weights) and instead set these to small values between 0.1 and 0.2. However, Becchetti, et al. showed that distributions in random-surfer models are not necessarily power-distributions and only certain damping co-efficients lead to power-law distributions [129, 130]. This finding is consequential since although behavior at web-scale usually corresponds to power distributions, this need not be the case when random surfer models are used in smaller, query-specific graphs.

#### 7.2.2 A possible solution by minimizing inversions

We propose a simple algorithm for tuning our PageRank-based model (Section 6.4). Our approach used task-specific training data to learn appropriate edge-type weights. Recall that in our setting,
the aggregate transition matrix can be represented as

$$ P = \sum_t w_t P_t $$

where each $P_t$ refers to the transition matrix that captures a particular type of edges (in set $E_t$) and is irreducible and stochastic. At equilibrium, the scores for all the nodes are obtained by solving the eigen vector equation,

$$ x = (\sum_t w_t P_t') x. $$

For an individual node $x_i$, this score can be written as

$$ x_i = \sum_t w_t \sum_{(j \rightarrow i) \in E_t} P_t(i|j)x_j $$

Given training data corresponding to correct rankings, or a set of “relevant” nodes, it is easy to derive pairs, $\{(a, b)_k\}, k = 1 \ldots n$, where the relative ordering $x_a > x_b$ corresponds to correct rankings that need to be satisfied by our scoring framework. Suppose, our current parameter vector, $w$ produces scores that leads to ranking different from the correct ranking. The mismatch in the desired and the current ranking can be captured via number of inversions among the two lists. One way to quantify this mismatch based on the set of inversion pairs, $I$ is: $\sum_{(b,a) \in I} (x_b - x_a)$.

That is, we seek the weight vector $w$ that solves the following optimization problem

$$ w^* = \arg \min_w \sum_{(b,a) \in I} \sum_t w_t (\sum_{(c \rightarrow b) \in E_t} P_t(b|c)x_c - \sum_{(c \rightarrow a) \in E_t} P_t(a|c)x_c) $$

where $\sum_t w_t = 1$ and $w_t \geq 0$ for all edge-types, $t$ in the graph.

The above formulation is used to illustrate the basic procedure that can be adopted for finding $w$ given task-specific training data. In general, while formulating the loss function, one also needs to consider optimization issues such as stability and efficiency, how to compute gradients etc. In addition, while extending the loss function over multiple queries, and extracting training pairs over several query-dependent graphs, one must also consider issues such as, imbalance in training pairs per query, query-specific weights and so on. Some of these aspects were previously studied as part of learning to rank [27].
Chapter 8

Conclusions

In this dissertation, we discussed the two information retrieval tasks, **Expert Search** and **Homepage Finding**, in the context of academic domains. Academic homepages are important sources of research publications and other researcher-related metadata. This aspect makes homepages essential for enabling expert search in open-access digital libraries.

We showed that existing webpage and text classification approaches are inadequate for academic homepage classification and a deeper understanding of homepage content and changing academic environments is necessary for addressing this task. To the best of our knowledge, we are the first to systematically analyze homepage content [59], and use insights from this analysis to address homepage retrieval on the Web [131]. We are also the first to address homepage finding in the context of crawling academic websites [132].

We proposed graph-based models for ranking researchers in response to either topic and name queries. Our models though generic, obtain performance on par with models specific to each query-type (name and topic) studied in previous works. We extend PageRank for ranking authors by integrating expertise evidence across multiple sources [133]. Our second model supports the topic mixture view of research documents and scores author nodes based on association strengths of paths in the underlying author-document-topic graph [134]. To the best of our knowledge, we are the first to study similar expert search for academic domains [135] and to apply subtopic phrases for expanding general topic queries in expert search [136].

To conclude, this dissertation advances “the state of the art” by proposing models for expert search and homepage finding in the context of academic environments. Our contributions complement existing solutions for these tasks in enterprise settings and on the Web. On basis of our experimental findings, we expect our proposed models to produce favorable results in systems comparable to CiteSeer® [2].
Bibliography


URL http://dl.acm.org/citation.cfm?id=1625275.1625718


URL http://dx.doi.org/10.1023/A:1007692713085


URL http://doi.acm.org/10.1145/1008992.1009079


URL http://doi.acm.org/10.1145/988672.988699


URL http://doi.acm.org/10.1145/1099554.1099649


URL http://doi.acm.org/10.1145/1015330.1015350


Vita
Sujatha Das Gollapalli

Research interests:
Data Mining, Information Retrieval, Algorithms, Machine Learning, Recommendation.

Education

- Fall 2007 - Summer 2013: Ph.D. in Department of Computer Science and Engineering, Penn State University. Thesis Advisors: Prasenjit Mitra and C. Lee Giles
  Dissertation title: Researcher Expertise Search, Homepage Finding and Metadata Annotation
- Fall 2003 - Fall 2005: M. Tech. in Department of Computer Science and Engineering, IIT-Bombay, India. Thesis Advisor: Soumen Chakrabarti
  Thesis title: Type Prediction and Proximity Search in Question Answering

Recent work experience

- Research Intern, Inome Inc., Bellevue, Wa (Jun 2013 - present)
  Project: Record linkage in Big Data environments
  Mentor: Dr. Ang Sun
  Project: Applying multitask learning for Chinese natural language processing tasks.
  Mentor: Dr. Yanjun Qi
- Research Assistant, Intelligent Information Systems Lab (during PhD.)
  Project: Academic homepages: identification and metadata extraction
  Mentors: Dr. Prasenjit Mitra and Dr. C. Lee Giles

Representative publications

1. Sujatha Das, Yanjun Qi, Prasenjit Mitra, C. Lee Giles
   Extracting metadata from academic homepages using labeled features (in review)
2. Sujatha Das, Prasenjit Mitra, C. Lee Giles
   Ranking experts using Author-Document-Topic graphs
3. Sujatha Das, Cornelia Caragea, Prasenjit Mitra, C. Lee Giles
   Improving academic homepage classification using unlabeled data.

Familiar languages and platforms

- Languages: Java, C, C++ and Matlab
- Platforms: Linux and Windows
- Machine learning software: Mallet, Weka

Full vita and references Available on request

Contact information

- Email: sujatha[dot]das[at]gmail[dot]com
- Mobile: 814-321-8184