

Entity Identification on the Web

Dissertation

submitted in partial fulfillment of the requirements
for the degree of

Master of Technology

by

Charu Tiwari

(Roll no. 03329029)

under the guidance of

Prof. Sunita Sarawagi



Kanwal Rekhi School of Information Technology

Indian Institute of Technology Bombay

2005

Dedicated to my family

Dissertation Approval Sheet

This is to certify that the dissertation entitled
Entity Identification on the Web

by

Charu Tiwari

(Roll no. 03329029)

is approved for the degree of **Master of Technology**.

Prof. Sunita Sarawagi
(Supervisor)

Prof. Umesh Bellure
(Internal Examiner)

Mr. Bhavin Manek
(External Examiner)

Prof. Krithi Ramamritham
(Chairperson)

Date: _____

Place: _____

INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

CERTIFICATE OF COURSE WORK

This is to certify that **Ms. Charu Tiwari** was admitted to the candidacy of the M.Tech. Degree and has successfully completed all the courses required for the M.Tech. Programme. The details of the course work done are given below.

Sr.No.	Course No.	Course Name	Credits
Semester 1 (Jul – Nov 2003)			
1.	HSS699	Communication and Presentation Skills (P/NP)	4
2.	IT601	Mobile Computing	6
3.	IT603	Data Base Management Systems	6
4.	IT619	IT Foundation Laboratory	10
5.	IT621	Foundation course of IT - Part I	3
6.	IT623	Foundation course of IT - Part II	3
7.	IT694	Seminar	4
Semester 2 (Jan – Apr 2004)			
8.	CS604	Combinatorics	6
9.	CS628	Introduction to Asynchronous Systems	6
10.	IT606	Embedded Systems	6
11.	IT680	Systems Laboratory	6
Semester 3 (Jul – Nov 2004)			
12.	CS601	Algorithms and Complexity	6
13.	HS617	Creating and Managing Intellectual Property (Institute Elective)	6
14.	IT655	Advanced Topics in Data Mining	3
M.Tech. Project			
15.	IT696	M.Tech. Project Stage - I (Jul 2004)	18
16.	IT697	M.Tech. Project Stage - II (Jan 2005)	30
17.	IT698	M.Tech. Project Stage - III (Jul 2005)	42

I.I.T. Bombay

Dy. Registrar(Academic)

Dated:

Abstract

Searching for a particular entity is one of the most frequent activities on the Web. Many a times, there is also a need to integrate information available from various online sources. In such cases, it is essential to determine when two referents (web pages) are referring to the same physical entity. But, the presence of multiple entities with the same name makes the task difficult for a user. It is, therefore, desirable to automate this process, and allow a machine to make such a distinction, thereby saving user's valuable time.

Through this thesis, we address the mentioned problem of **Entity Identification on the Web** using statistical models from the field of machine learning. In particular, we address the problem of **Disambiguating People on the Web**. We model the problem both as a *pair-wise document classification* problem to classify each pair of documents to determine if they both refer to the same entity, as well as a *single document classification* problem to classify the individual documents as positive or negative in accordance with the user's preference. We also explore use of more recent *query expansion* techniques to better address the disambiguation problem. Our experiments show that our method is useful in unpteen batch and interactive application scenarios. This report explains the theory behind our model, the experimental setup, and the results obtained by applying our model in a few important applications.

Contents

Abstract	ix
List of figures	xv
List of tables	xvii
1 Introduction	1
1.1 Compelling Applications	2
1.1.1 Meta Search Engine	2
1.1.2 Keyword Suggestions	3
1.1.3 Background Search	3
1.1.4 Set Valued Attribute Accumulation	4
2 Re-ranking: A Supervised Application Scenario	7
2.1 Problem Definition	7
2.2 Approach	8
2.3 Feature Set	9
2.3.1 Similarity Features	9
2.3.2 Link Features	10
2.4 Experiments & Results	13
2.4.1 Experimental Setup	13
2.4.2 Evaluation Criteria	14
2.4.3 Classifier Selection	15
2.4.4 Effect of Features on Performance	16
2.4.5 Variation in Performance with increase in Training Data	19
2.4.6 Effect of using a different sized Window on Performance	19

2.5	Ranking Performance	20
2.6	Co-Reference Resolver: Our Re-ranking Demo	23
3	Clustering: An Unsupervised Application Scenario	29
3.1	Problem Definition	29
3.2	Approach	29
3.3	Cautious: A Graph Partitioning Based Algorithm	31
3.4	Experiments & Results	32
3.4.1	Evaluation Criterion	32
3.4.2	Results & Conclusions	33
4	Query Expansion	37
4.1	Needs in Entity Search	37
4.2	Related Work	37
4.3	Our Approach to Query Modifications	38
4.4	Application to Re-ranking	39
4.4.1	System Design	40
4.5	Experiments	42
4.6	Results & Conclusions	42
4.7	Capturing Correlation	47
5	Application of Entity Disambiguator in Other Scenarios	49
5.1	Set Valued Attribute Accumulation	49
5.1.1	Problem Definition	49
5.1.2	Strategy	50
5.1.3	Experiments & Preliminary Results	50
5.2	Background Search	53
5.2.1	Problem description	55
5.2.2	Strategy	55
5.2.3	Experiments & Preliminary Results	57
6	Related Work	61
6.1	Disambiguating People in Search	61
6.2	Entity-Based Cross-Document Co-referencing Using the Vector Space Model	62

6.3	Anti-Aliasing on the Web	63
6.4	Extracting Query Modifications from Nonlinear SVMs	64
6.5	Disambiguating Web Appearances of People in a Social Network	66
6.6	Interactive Deduplication using Active Learning	67
7	Conclusions & Future Work	69
	Bibliography	71
	Acknowledgments	75

List of Figures

1.1	Entity Ranker System	2
1.2	Entity Clustering System	3
1.3	Keywords Suggestion System	3
1.4	Background Search System	4
1.5	Set Valued Attribute Accumulation System	5
2.1	Disambiguation as a Classification Problem	8
2.2	Performance Vs Training data Size	19
2.3	Co-Reference Resolver Front Page	25
2.4	Co-Reference Resolver Search for Sandeep Pandey	26
2.5	Co-Reference Resolver Select Results for Sandeep Pandey	27
3.1	Disambiguation as a Clustering Problem	30
3.2	The Cautious Algorithm	32
4.1	Improved Ranker System	41
4.2	Precision at top N re-ranked results	44
5.1	Set Valued Attribute Accumulation System	51
5.2	Organization Recall for top N ranked snippets	54
5.3	Visualization	55
5.4	Background Search System	56
5.5	Ashish Gupta	58
5.6	Michael Jordan	58
5.7	Recall vs Number of fetched pages	58
6.1	Query Modifications using Non Linear SVMs.	65

List of Tables

2.1	Search terms used in the dataset	13
2.2	Number of pairs in the five folds	14
2.3	Comparison of different classifiers	15
2.4	Comparison of different classifiers (contd.)	16
2.5	Effect of various features on performance	17
2.6	Effect of various features on performance (contd.)	18
2.7	Effect of using different window size on performance	20
2.8	Search terms as mentioned in a related paper	21
2.9	New search terms	21
2.10	Fraction of $\min(K, N)$ included in first N results	22
2.11	Fraction of $\min(K, N)$ included in first N results (contd.)	23
3.1	Effect of variation in δ on performance	33
3.2	Clustering Results	34
3.3	Improvement in Performance through Clustering	34
4.1	Ashish Gupta (Junglee)	39
4.2	Tom Mitchell (CMU)	39
4.3	Top 10 words using Linear SVM	39
4.4	Oracles' details	43
4.5	Some results for <i>John Miller</i> , Roller Coaster Designer	45
4.6	Some results for <i>Ashish Gupta</i> , Northwestern University	46
4.7	Component Terms of the Scoring Function (Equation 4.2)	48
5.1	Performance of the organization-tuned <i>type</i> classifier	52
5.2	Top 5 Snippets for <i>Ashish Gupta</i>	53

5.3	Organizations augmented for entity <i>Ashish Gupta</i>	54
-----	--	----

Chapter 1

Introduction

Searching for a particular entity is one of the most frequent activities on the Web. For example, a user requiring information about some person, say *Tom Mitchell* or *Michael Jordan*, queries the Web for the same. Another frequent Web related task is the integration of information collected from various online sources to facilitate easy access of information to the users. Both these operations present the problem of ***duplicate identification/deduplication*** (i.e., determining when two objects are variants of the same object), and ***co-reference resolution*** (i.e., determining when two objects are referring to the same entity).

We target towards developing an effective method for solving the problem of co-reference resolution ¹. The problem is challenging, as the presence of namesakes makes it non-trivial for a machine to identify the context of an entity. Although much work has been done to identify duplicates from among given structured documents, such as database records, and semi-structured documents, such as citations and addresses, the disambiguation problem when applied to unstructured documents is still under-explored. The existing approaches make assumptions like the presence of external databases, which might not hold always; e.g., a list of organizations may not be available for each and every country. Further, they use methods like determining the number of people belonging to a domain, which are computationally expensive. Though use of such methods provide a sufficiently good performance, the need for a general method is indispensable.

This chapter briefly describes some of the probable application scenarios where an entity disambiguator is useful.

¹The terms “co-reference resolution” and “entity disambiguation” are used interchangeably in this thesis

1.1 Compelling Applications

Our goal is to build an *Entity Disambiguator* that can be applied to both the supervised as well as the unsupervised setups. There are umpteen situations where an entity disambiguator is useful. Some of them are briefly described as under.

1.1.1 Meta Search Engine

A user often searches for a particular entity on the Web where there are a number of entities with the same name. In a typical scenario, the user just provides the name of the entity, and the result is an interleaving of pages referring to various entities having the same name. But, this requires the user to manually read, understand, and then pick those documents that are relevant to him/her. We, therefore, aim to develop a system to automate this process.

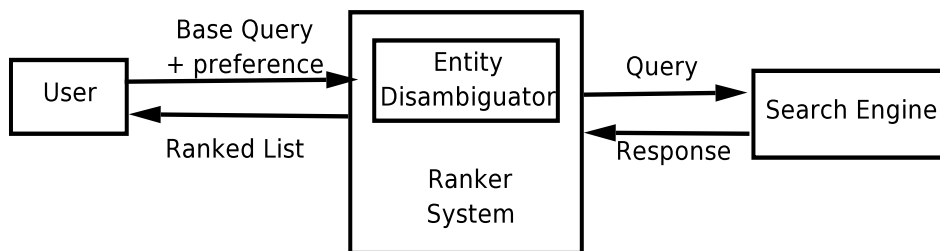


Figure 1.1: Entity Ranker System

Our aim is to provide a meta search engine like interface (Figure 1.1), whereby a user can indicate the entity of interest by selecting one of the pages from the search results, and the system then re-ranks the documents so that the relevant documents are ranked higher than the irrelevant ones. Chapter 2 describes our *re-ranking* model in detail.

A parallel line of thought is to build a clustering system (Figure 1.2) which takes as input the name of the entity the user is interested in, and then outputs a set of clusters each referring to a separate entity. This enables the user to select the cluster of interest easily. We describe an interesting modeling of the clustering problem in Chapter 3.

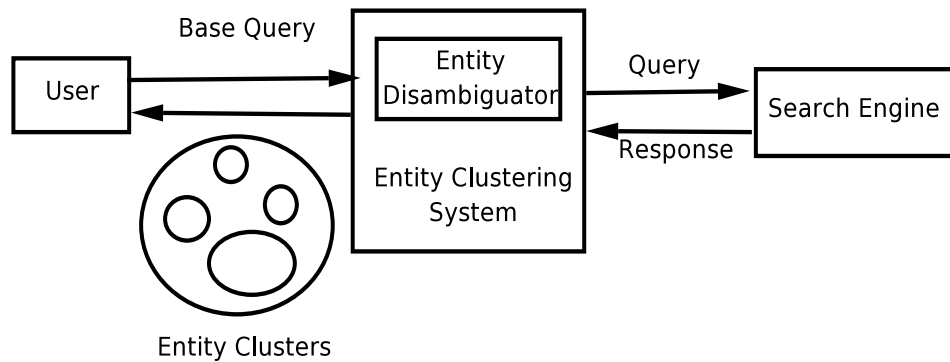


Figure 1.2: Entity Clustering System

1.1.2 Keyword Suggestions

A study of a user's behaviour on the Web indicates that a user typically starts off with a base query indicating the person name, and then modifies the query using discriminating words when the search result contains several irrelevant pages. We believe that automatic generation of important words related to the entity of interest can assist the user in forming better query expansions. Using the framework developed in Chapter 2, we developed a **Keyword Suggester** module which can provide such an assistance to the user. Figure 1.3 gives an outline of the desired system. We also explored the use of the suggested words to develop a new system, which can not only rank the relevant pages higher, but can also fetch more relevant pages. Chapter 4 gives the details of our work in this direction.

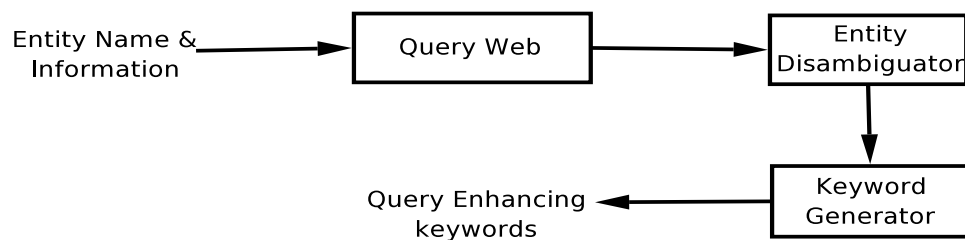


Figure 1.3: Keywords Suggestion System

1.1.3 Background Search

Consider a system that allows us to fetch as many pages about a person as possible. We call such a system as one allowing **Background Search** for a person. Such a system might serve as a platform to perform many other tasks. For example, we might require

to fetch maximum number of pages about Ashish Gupta, a venture capitalist, to extract important information about him. Search engines, like Google and Yahoo!, do not allow us to fetch more than a fixed number of pages for a query. Therefore, to fetch maximum pages referring to a person of interest, we need to pose multiple queries. Intelligent selection of the queries, together with the disambiguation of the resulting pages, then becomes very important for the mentioned coverage application. Figure 1.4 gives an outline of the desired system, and Chapter 5 provides the details of our preliminary work in this direction, where we try to make use of our keyword generation strategy to search efficiently.

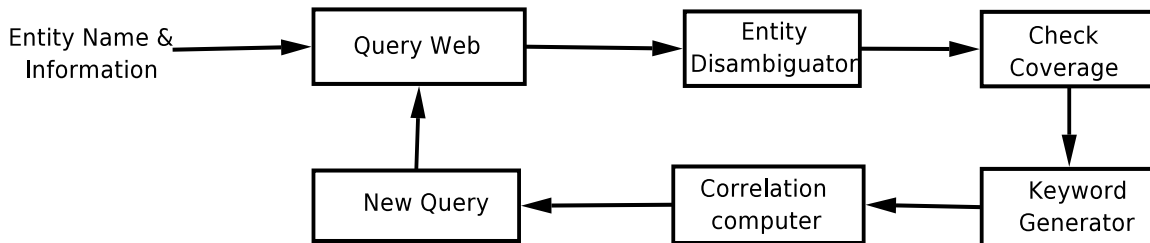


Figure 1.4: Background Search System

1.1.4 Set Valued Attribute Accumulation

Set valued attribute accumulation is one of the most interesting and motivating application of an entity disambiguator. Consider, for example, a personal information network where we have different types of information stored about people we are connected with. Now, suppose we have some information about a person in our network, and wish to automatically find all the *organizations* that the person has worked in, all the *money amounts* associated with the person, all the *co-authors* that the person has written a paper with, and the like. All these are referred to as ‘Set Valued Attributes’. Figure 1.5 gives an outline of the desired system, and Chapter 5 provides the details of our preliminary work towards developing such a system.

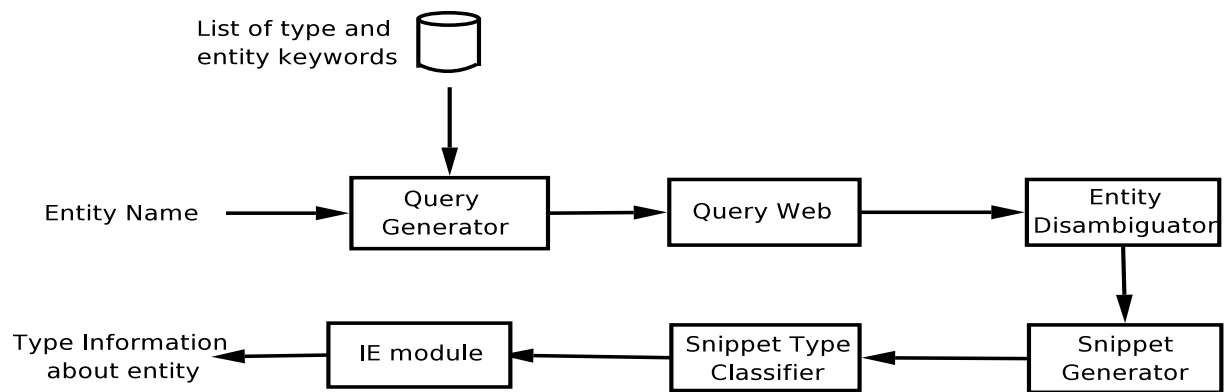


Figure 1.5: Set Valued Attribute Accumulation System

Chapter 2

Re-ranking: A Supervised Application Scenario

This chapter describes our main framework for disambiguating entities and its application to the preferential ranking of URLs in a search engine's response set.

2.1 Problem Definition

A user often searches for a particular entity on the Web where there are a number of entities with the same name. In order to narrow down the number of results, the user has to supply additional keywords related to the particular entity he/she is looking for. As we know that the addition of improper, or too many words, may degrade the quality of the result, we desire to perform entity identification without much user intervention. Understanding that many additional words may leave useful results, a user looking for a particular entity would just query on the name of the entity. But, this requires the user to manually read, understand, and then pick those documents that are relevant to him/her. E.g., a user queries a search engine for information on *Tom Mitchell*. The search engine provides a list of documents, not all referring to the *Tom Mitchell* the user is looking for. This is because more than one person can have the same name. Some result entries might refer to the *Tom Mitchell* who is a professor, others might relate to some musician, while still others might refer to an entirely different person. Now, a user searching for information on *Tom Mitchell* who is a professor, will have to read the resultant documents, and identify all those pages that are of relevance to him/her. It would be much convenient if this process can be automated.

2.2 Approach

Figure 2.1 shows the different steps of the entity classification process. As shown in the figure, the input to the system is the entity name and the user’s preference of the entity. A user indicates his/her preference by selecting a web page from the search engine’s response to the *base query*¹. The system then takes the selected page, referred to as the *base page*, as the one referring to the entity of interest, and adapts the disambiguation process to suit the user’s preference.

We model the problem as a *pair-wise binary classification* problem. We basically require a 0/1 classifier that will classify pairs of web pages as either co-referent (label 1) or non co-referent (label 0). This involves identification of features whose values when fed to a classifier would allow it to disambiguate the entities. As each web page is read, it is first cleaned to get a well formed HTML document, and then converted into a SpecialDocument object. This SpecialDocument stores and allows access to various important parts of the web page namely the URLs, the text within a window around the search term, and the like. The training data consists of a set of pages obtained by querying a Search Engine on a set of person names. For every person name, there will be pages referring to a number of different physical entities. At the time of training, each page in the training data is considered a potential base page. All possible pairs of the pages – obtained by querying on a single name – are, hence, formed, pair features extracted, and fed to a classifier for training. The classifier then learns a model based on the added features. At the time of testing, the learnt model is used to predict the label of the test pairs.

Before providing the details of the experiments and the evaluation criterion used for the same, the next section presents the identified feature set in detail.

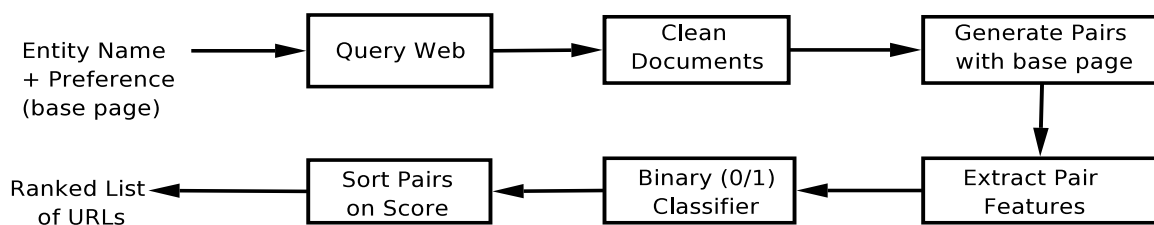


Figure 2.1: Disambiguation as a Classification Problem

¹The terms “entity name” and “base query” are used interchangeably throughout this thesis

2.3 Feature Set

The following are the main categories into which all identified features are classified.

2.3.1 Similarity Features

These features are based on the token similarity between different parts of the document pair under consideration. TFIDF and Jaccard are the main similarity feature classes of which a number of similarity features have been derived. These features are commonly referred to as the vector based similarity features.

- **Cross Text Similarity Features:** The `TFIDFCrossTextFeature` measures the TFIDF similarity of the entire text of the two documents belonging to the pair under consideration. The value of this feature is the 0-1 normalised TFIDF score. The idea behind using this feature is that – highly similar documents are more likely to refer to the same entity. The `JaccardCrossTextFeature` measures the corresponding Jaccard similarity.
- **Text Title Similarity Features:** The `TFIDFTextTitleFeature` measures the TFIDF similarity of the title of one document with the entire text of the other, and vice-versa. The value of the feature is the average of the two normalised TFIDF scores thus obtained. This feature is based on the observation that the titles of the web pages are often good indicators of the content of the page and thus may help in the process of disambiguation. For example, if the title of a page reads “Sandeep Pandey wins the Magsaysay Award”, it is likely that the other document, if referring to the same “Sandeep Pandey”, will contain the word “Magsaysay” somewhere in its text. However, titles like “Amit’s Homepage” can be quite misleading. The `JaccardTextTitleFeature` measures the corresponding Jaccard similarity.
- **Cross Title Similarity Features:** The `TFIDFCrossTitleFeature` measures the TFIDF similarity of the titles of the documents belonging to the pair under consideration. The value of the feature is the normalised TFIDF score. This feature is again based on the observation that the titles, if similar, are good indicators of the pages referring to the same entity in most of the cases. But, once again, titles like “News Report on ...” can be misleading. The `JaccardCrossTitleFeature` measures the corresponding Jaccard similarity.

- **Near Entity Features:** The `TFIDFNearEntityFeature` measures the TFIDF similarity between the snippets collected from a window around the search term, from the document text. The `JaccardNearEntityFeature` measures the corresponding Jaccard similarity. This feature is based on the intuition that important information usually appears in the visual vicinity of the entity name. But, since the web pages are highly unstructured, the text in the visual vicinity may not be in the textual vicinity. We, therefore, need a way to identify relevant blocks of information given a web page. Deng Cai et al., [CYWM04] [DCM03], propose a vision based page segmentation algorithm called **VIPS**, which segments a web page into blocks using the layout information. We believe that identification of relevant blocks by utilizing visual layout, and generation of block dependent features, should enhance the performance of our system. The disadvantage, however, is that the system becomes slow. We, therefore, look forward to explore proper utilization of such features in future.

2.3.2 Link Features

A number of link features utilizing the information rich hyperlinks have been added to this implementation. Some of them test the presence of certain type of links directly; others are based on domain name match of the links, while still others utilize the information in the anchor text. This section describes each of them in detail.

- **Domain Match Feature:** This feature tests if the two web pages in the current pair share the same domain name. The intuition behind this is that if two web pages are from the same domain then they have a high probability of being co-referent. For example, two web pages with URLs “`www.it.iitb.ac.in/~sunita`” and “`www.it.iitb.ac.in/library/sunita`” have the same domain name, and so are likely to be referring to the same entity – *Sunita*.
- **Sub-domain Match Feature:** This feature tests if the two web pages in the current pair belong to a common rare upper domain. The intuition behind this feature is that an organization commonly has a number of departments, which share a common upper level domain name. For example, two web pages with URLs “`www.it.iitb.ac.in/~sunita`” and “`www.iitb.ac.in/research.html`” share the

same domain – “iitb.ac.in”. The SubDomainMatchFeature attempts to identify such a match. A list of top level domains (such as edu, com, gov, ac, in, ...) has been identified and if the domain of the pages under consideration shows a match beyond these identified domains, then this feature is fired.

- **Cross Link Feature:** This feature tests if the two web pages in the current pair link to each other. This is one of the strongest link based similarity features.
- **Cross Link Domain Match Feature:** This feature tests if each of the two web pages in the current pair link to a page that comes from the same domain as the other page in the pair. This feature, though implemented, is not included in the reported results.
- **Cross Link Sub-Domain Match Feature:** This feature tests if each of the two web pages in the current pair link to a page that comes from the same rare top level domain as the other page in the pair. This differs from the Cross Link Domain Match Feature in that it checks for only a rare top level domain match rather than a complete domain name match. For example, consider two web pages with URLs “www.cs.cmu.edu/~wcohen/publications.html” and “www.it.iitb.ac.in/~sunita/publications.html”. Now, if the first page points to the page with the URL “www.iitb.ac.in/faculty/sunita”, and the second one points to the page with the URL “www.cs.cmu.edu/~wcohen”, then the Cross Link Domain Match Feature will not fire, while the Cross Link Sub-Domain Match Feature will. This feature, though implemented, is not included in the reported results.
- **Single Link Feature:** This feature tests if at least one of the two web pages in the current pair links to the other. The intuition behind this feature is as follows. Web pages commonly have a hierarchy with various pages in the hierarchy linking to each other through hyperlinks. The single link feature checks if the two web pages under consideration belong to some two neighbouring levels of a hierarchy. For example, if one of the web pages in a pair points to the web page with the URL “www.it.iitb.ac.in/~sunita/publications.html”, which is also the URL of the second web page in the pair, then this feature will be fired for this pair of web pages.
- **Single Link Domain Match Feature:** This feature tests if at least one of the two web pages in the current pair links to a page that comes from the same domain as the other page in the pair. If one of the web pages in a pair points to the URL

“www.it.iitb.ac.in/~sunita/publications.html”, while the other web page in the pair has the URL “www.it.iitb.ac.in/~sunita/index.html”, then this feature will be fired for this pair of web pages. This feature, though implemented, is not included in the reported results.

- **Single Link Sub-Domain Match Feature:** This feature is similar to the Single Link Domain Match Feature, with the difference that it tests for a match till a rare top level domain, rather than check for the entire domain name match. This feature, though implemented, is not included in the reported results.
- **Forward Link Features:** A number of features based on the commonality of the forward links between the web pages have been defined. Since two web pages might be quite different and still point to a common third page, the features in this category are designed to make use of important forward links.
 - a) **Forward Link on Entity Name Match Feature:** This feature tests if each of the two web pages in the pair under consideration have at least one common link having the search term name in its anchor text in both the pages.
 - b) **Forward Link on Entity Name Domain Match Feature:** This feature tests if one of the two web pages under consideration have at least one link with the search term in its anchor text, that has a matching domain with some link having the search term in its anchor text in the other web page in the pair under consideration.
 - c) **Forward Link on Entity Name Single Match Feature:** This feature tests if one of the two web pages under consideration have at least one link with the search term in its anchor text, with the same link being present in the other page in the pair under consideration.
 - d) **Forward Link Containing Entity Name Match Feature:** This feature tests if the two web pages under consideration point to at least one common web page whose URL contains the entity name in it. For example, if the entity being searched for is *Sunita* and two web pages have “www.it.iitb.ac.in/~sunita” as a common link, then this feature will be fired for this pair of web pages.
 - e) **Forward Link Containing Entity Name Domain Match Feature:** This feature is similar to the above mentioned features except that it tests for a match of the URL till the entity name rather than considering the complete

link match. For example, if one web-page contains “`www.it.iitb.ac.in/~sunita/teaching.html`” and other contains “`www.it.iitb.ac.in/~sunita/publications.html`” then this feature will be fired, because the part of the URLs till the entity name *Sunita* in both the URLs is the same. This will also be applicable in cases where the two web pages contain URLs of the type “`www.it.iitb.ac.in/faculty/sunita/papers/hmmtut.pdf`” and “`www.it.iitb.ac.in/faculty/sunita/papers/crf.pdf`”.

2.4 Experiments & Results

We performed a number of experiments to develop our disambiguation system. This section describes our experiments, together with the setup for the same, in detail.

2.4.1 Experimental Setup

The training as well as the test data was drawn from a large pool of labeled web pages which forms the dataset used in a closely related paper ([GG04]). The dataset consists of web pages obtained by performing searches on 12 different search terms (all related to people) listed in Table 2.1. The dataset, however, is only a subset of all the result pages. Each search term has about 4-5 different entities associated with it, with each entity having an average of 10-15 co-referent pages. This dataset is augmented by results from a few more search terms which are also listed towards the end in Table 2.1. However, these added entities do not have many documents associated with them. These are some of the challenging examples added to the dataset to obtain a robust classification method.

Charles Smith	Patricia Williams	Lisa Brown
Mary Brown	Barbara Johnson	Michael Jordan
John Taylor	John Thompson	Susan Davis
Michael Jones	John Miller	James Harris
Amit Gupta	Sunita Sarawagi	Rahul Sharma
Subramanyam	Mukesh Sharma	

Table 2.1: Search terms used in the dataset

2.4.2 Evaluation Criteria

In this section, we describe our evaluation criteria for the two different settings: classification and ranking.

2.4.2.1 Classification Accuracy

We measure the performance of various classification models in terms of **F1** which is the harmonic mean of precision and recall, given by Equation 2.1. The **precision** of a model is the ratio of the number of true positives to the total identified positives, whereas **recall** is the ratio of the number of true positives to the actual number of positives.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (2.1)$$

To evaluate various classification models, we perform a five fold cross validation on the dataset mentioned in Section 2.4.1, with each fold consisting of an average of 2500 pairs. All the pages belonging to a search term are included in a single fold, constraining us to create folds of different sizes. The actual number of pairs in each fold are given in Table 2.2. We compare the predicted labels against the actual labels for all the test pairs, and compute the precision, recall, and $F1$ values for each fold. To obtain a single value to denote a model’s performance, we compute Macro and Micro $F1$ from the individual $F1$ values for the five folds. The **Micro F1** is the weighted sum of the individual $F1$ values for the five folds, weighted according to the number of test pairs. The objective here to identify the features and the classifier which will give maximum value for Micro $F1$. For **Macro F1**, each fold is treated equally, regardless of the number of pairs in it, with the Macro $F1$ computed as the simple average of the individual $F1$ values.

<i>Fold No.</i>	<i>Number of Pairs</i>
1	1780
2	2863
3	5006
4	2765
5	1692
Total	14106

Table 2.2: Number of pairs in the five folds

2.4.2.2 Ranking Accuracy

To measure the performance of a ranking strategy, we used the evaluation criterion mentioned in the paper [GG04]. Each page from the test set is selected as the base page, and its pair with each of the remaining pages for the same search term formed. These pairs are then classified using our pair-wise entity disambiguator to compute the similarity scores. For our experiments, the score is taken to be the probability of label 1 assigned by the disambiguator. A pair score can now be considered as the score for the page, other than the base page, in the pair. The pages are then sorted in descending order of their scores. The number of pages co-referent with the base page, appearing in top N pages, divided by $\min(N, K)$ where K is the number of pages in the corpus that refer to same entity as the base page, gives the performance metric for our computations.

2.4.3 Classifier Selection

This section presents the details of the experiments performed to select the best classifier for our problem. A number of classifiers such as Decision Trees, Support Vector Machines (SVMs), and Logistic Regression were tested. These experiments were performed using the tool “WEKA” [WEK]. Table 2.3 gives a comparative study of different classifiers, along with the details of the five folds used in the experiment. The feature set used for these experiments included all the link features listed in Section 2.3.2, and the TFIDF and Jaccard similarity features applied to the entire text of the document without removing the search term from the text.

<i>Fold No.</i>	<i>Fold Size (in number of pairs)</i>	<i>Decision Trees</i>	<i>Linear SVMs</i>	<i>Logistic Regression</i>
1	1780	0.852	0.882	0.867
2	2863	0.872	0.854	0.877
3	5320	0.796	0.781	0.811
4	3045	0.638	0.624	0.638
5	1726	0.836	0.832	0.868
<i>Micro F1 (by Pairs)</i>		0.790	0.781	0.818

Table 2.3: Comparison of different classifiers

We also experimented with other kernels for SVMs. The quadratic SVMs did not converge for a long time with the settings of Figure 2.3, so a three fold cross validation was done to compare their performance with Logistic Regression. Table 2.4 gives the details of the experiments; $C = 100000$ was used for quadratic SVMs and $\gamma = 0.01$ was used for RBF kernels. The other settings for C and γ either resulted in reduced performance or required the model to take an unacceptably long time during training.

<i>Fold No.</i>	<i>Fold Size (in number of pairs)</i>	<i>Quadratic SVMs</i>	<i>RBF Kernel</i>	<i>Logistic Regression</i>
1	4581	0.802	0.811	0.874
2	5305	0.589	0.607	0.811
3	4731	0.632	0.656	0.724
<i>Micro F1 (by pairs)</i>		0.670	0.687	0.802

Table 2.4: Comparison of different classifiers (contd.)

The results indicate that, for our problem, Logistic Regression performs slightly better than other classifiers such as Decision Trees and Linear SVMs. So, our implementation of the co-reference resolver makes use of the Logistic Regression model for classification.

2.4.4 Effect of Features on Performance

This section gives an analysis of the variation in performance with variation in the feature set. Table 2.5 shows the results obtained by a five fold cross validation on the dataset.

As is evident from the results, the similarity features are not enough to obtain a reasonable level of performance. Though they might perform better when the entities do not have much in common, but will fail badly in case the entities share a lot. We need a method which is robust enough to handle most of the cases. Also, observe that the removal of the search term from the text before taking the similarity shows an improvement in performance. This is because some pages contain very little information about the entities, but as the search was performed on the search term, most of the result documents contain the search term – sometimes even more than once. If the document pair similarity is computed without the removal of the search term, we might get a high similarity score for a pair of documents actually referring to different entities. We conclude that removal of the search term is important before computing any type of similarity.

<i>Features</i>	<i>Performance</i>	
	<i>Micro F1</i>	<i>Macro F1</i>
Only TFIDF	0.86209	0.86793
TFIDF+Jaccard (Without search term removal)	0.86813	0.87314
TFIDF+Jaccard (After search term removal)	0.89225	0.90102
TFIDF+Jaccard +Links (After search term removal)	0.89578	0.90440
Only Links	0.18572	0.16834
All (including title related)	0.89511	0.90564

Table 2.5: Effect of various features on performance

Addition of the link features improves the performance slightly, but link features alone are not sufficient because these features are not present in most of the documents on the Web. Such features are mostly useful in cases the search term results in an entity who maintains his/her personal site, because it is expected that the mention of such an entity would have a link to the entity’s homepage. But, since the entities in our dataset are authors, election candidates, musicians, and sports persons, the search results mostly contain pages reporting some news related to these entities. The results are also indicative of the fact that the link features are not sufficient on their own, and that the major part of performance comes from various similarity features. However, the identified link features always result in high precision indicating that their presence is a very good indicator of a pair being co-referent. Table 2.6 gives the five fold Micro Precision, Micro Recall, and Micro *F1* for the link features in detail. The results show that the Sub Domain and Domain features are most important link features, and contribute a lot to the link feature performance. The other features perform poorly because they are rarely found on the web pages, but their presence is a very good indicator of the pair being co-referent.

The results in Table 2.5 also show that the inclusion of title related features results in a slight decrease in performance. Though it does improve the performance in three of the five folds, but fails in the remaining two. The reason for the failure is that – one of these two folds contain result documents for the search term *Patricia Williams* and three out

<i>Features</i>	<i>Micro Precision</i>	<i>Micro Recall</i>	<i>Micro F1</i>
Domain and Sub Domain	1.00000	0.07378	0.13625
Single and Cross link	1.00000	0.00110	0.00220
FLContainingEntityName	1.00000	0.04158	0.07903
FLOnEntityName	1.00000	0.03330	0.06433
All Link Features	1.00000	0.10378	0.18572

Table 2.6: Effect of various features on performance (contd.)

of the five entities with this name are professors. The titles of the pages mostly contain the term ‘Professor’, resulting in high similarity between such documents even when they are non co-referent. Also, the documents corresponding to one of the entities for this search term contain description of other professors, resulting in high similarity with those referring to some other entity who is a professor. Since, this fold is the biggest of all, the drop in performance is justified. The Macro $F1$, reported in Table 2.5, highlights the importance of the title related features.

We observe that of the entire text of a document, the *nouns* and the *adjectives* are the main contributors to the process of entity identification. Note, however, that POS tagging is a time consuming process, and hence, might be unacceptable in an online application environment. Similarly, techniques like phrase recognition might also prove beneficial for the disambiguation task in a batch environment.

We also observe that some of the result pages do not contain the entire search term but contain only parts of it. Continuing with the example of *Patricia Williams*, a few pages do not belong to any *Patricia Williams*, but were included in the results because they contain terms like ‘Patricia’ and ‘Williams’. Identification of such pages will help reduce irrelevant pages.

We believe that we need some way to establish the context of an entity to improve performance. In other words, we need some way to identify portions of documents which actually belong to the entity of interest. We can then rank these portions according to their relevance to the entity being referred. One can also hunt for more information – by crawling the Web graph by one level from the given pages – to improve performance, but we believe that the primary goal should be to establish the context.

2.4.5 Variation in Performance with increase in Training Data

Figure 2.2 shows how the performance increases with an increase in training data size. In the figure, size refers to the number of pairs in the data set. The results were obtained by taking a repository of size 14106 from the data set mentioned in Section 2.4.1, and averaging the results for four test data sets of sizes 4404, 6169, 3479, and 4643, drawn from the same repository but with different seeds, and using the remaining data for training. Figure 2.2 shows Micro $F1$ for the above mentioned test datasets, being plotted against the training data size (on a log scale), where Micro $F1$ is the weighted average of the individual test performances. As the graph shows that after training data size reaches 1000, the performance remains more or less the same, without much variation. Also, the graph takes a different slope after the training data size reaches nearly 100. So, we can say that we can obtain a reasonable performance with just 100 training data points, and a sufficiently high performance with nearly 1000 training data points.

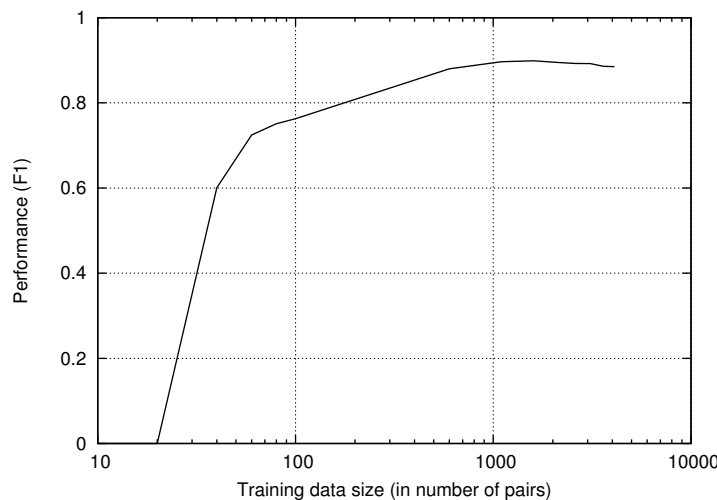


Figure 2.2: Performance Vs Training data Size

2.4.6 Effect of using a different sized Window on Performance

In this section, we show the effect of using a different sized window around the search term. In this experiment, we used all the link features mentioned in Section 2.3.2, and the TFIDF and Jaccard similarity features, but excluded the title related features. The results obtained are shown in the Table 2.7. In each test – except the one using the entire text of the document – text from a window of size W is gathered around the search term from wherever the search term appears in a page. Note that the search term here refers to any

part of the search term including both the first name as well as the last name. We remove the search term from the gathered text and measure the TFIDF and Jaccard similarity between them. The results in Table 2.7 are obtained using a five fold cross validation with the fold sizes as mentioned in Table 2.2. The results reveal a higher performance for larger window sizes, with the best performance for the case using the entire text of the document. This is because, many a times, important information appears far away from the search term that a smaller window around the search term is not able to capture that information. Using the entire text of the web page might create problems if the page does not belong to the searched entity in its entirety. But, overall, the results indicate using a larger window size.

<i>Window Size (W)</i> <i>(in number of words)</i>	<i>Performance</i> <i>(Micro F1)</i>
25	0.848832
50	0.866165
75	0.877727
100	0.877727
125	0.885442
150	0.885948
All Text	0.895789

Table 2.7: Effect of using different window size on performance

2.5 Ranking Performance

We performed a number of experiments to measure the performance of the identified features as per the evaluation criterion of Section 2.4.2.2; this section provides the details of these experiments. We used two fixed test datasets which are described as under.

1. One of the test dataset was drawn from the repository mentioned in Section 2.4.1, and consisted of all the pages belonging to the search terms mentioned in Table 2.8.
2. The second test dataset consisted of pages belonging to the search terms mentioned in Table 2.9. This test dataset was used to measure the performance of the model in case of challenging cases.

The entire data from the repository mentioned in Section 2.4.1, excluding the part included in the test dataset, was used for training. We trained three disambiguator models – all using the same training data set, but different feature sets. The feature sets used for the three models are as under.

1. **Case 1:** The first model makes use of all the features mentioned in Section 2.3 – namely all the link features, and similarity features applied to the text and title of the web pages. The search term was removed from the text and the title of the web pages before taking the similarity.
2. **Case 2:** The second model makes use of only one feature, i.e., TFIDF document text similarity feature. The search term was retained in the document text while measuring the similarity.
3. **Case 3:** The third model is the same as the second one except that the search term was removed from the text before taking the similarity.

Michael Jordan	Lisa Brown	John Taylor
Barbara Johnson	John Thompson	

Table 2.8: Search terms as mentioned in a related paper

Abhijeet Dasgupta	Sandeep Pandey	Pradeep Singh
Paul Cohen	Sudarshan	

Table 2.9: New search terms

The performance results of the three models for the two test datasets are shown in Tables 2.10 and 2.11 respectively. The results, shown in Table 2.10, for the first test data set indicate that there is not much difference in the performance for the three cases. The following points, however, are noteworthy.

- The scores assigned by the models of case 2 and 3 were such that there was not much difference in the scores of the co-referent as well as the non co-referent pairs (all scores above 0.5 in each case), with co-referent pairs getting slightly higher scores than the non co-referent ones. On the other hand, the model of case 1 was clearly able to separate out the co-referents (scores mostly in the range 1-0.75) from the non co-referents (scores as low as 0.005). The presence of the added features

N	<i>Case 1</i> (All Features)	<i>Case 2</i> (TFIDF without search term removal)	<i>Case 3</i> (TFIDF after search term removal)
5	0.961	0.956	0.951
10	0.955	0.943	0.946
20	0.957	0.957	0.956
30	0.965	0.969	0.965
40	0.960	0.972	0.963

Table 2.10: Fraction of $\min(K, N)$ included in first N results

in case 1, which makes it a better classifier, results in the TFIDF feature getting a higher weight compared to the case 2 and 3 models. The case 1 model learns a positive weight for label-0 and a negative weight for label-1 for the Jaccard features. Though, helping in some cases, this might hurt as well. For example, it might happen that a pair, having a high TFIDF than the other, might get a lower overall score because it also has a high Jaccard score. This problem arises mainly when, except the similarity features, all other features are absent.

- The type of scores assigned by the models as mentioned in the previous point dictates that the case 2 model does not prove to be a good classifier as it does not have a clear separation between the positive and the negative classes. Due to this, if the user just wants those pages that are co-referent to the selected page, instead of a sorted list of pages, or wants to eliminate those pages that are co-referent to the selected one, the case 1 model is expected to perform much better compared to the case 2 model. Similar arguments apply to the model of case 3, which though better in score assignment than the case 2 model, still has a poor separation between the co-referent and the non co-referent classes.
- The first test data set lacked challenging cases. The results on the second test data set supports this argument. The results for the top 5 and 10 pages are shown in Table 2.11. The results show that the models of case 1 and 3 outperform the case 2 model. The main reason here being the presence of the search term in the text of the web pages in case 2. This can also be inferred from the results because the model of case 3 performs nearly as well as the model of case 1. So, what was helping

case 2 in the earlier case, now causes it to perform relatively bad. The model of case 2, therefore, is not a robust one. Intuitively, we would want the search term to be removed from the web pages before taking the similarity. The reason being the presence of the search term in almost every page returned by the search engine.

N	<i>Case 1</i> (All Features)	<i>Case 2</i> (TFIDF without search term removal)	<i>Case 3</i> (TFIDF after search term removal)
5	0.987	0.941	0.986
10	0.986	0.968	0.982

Table 2.11: Fraction of $\min(K, N)$ included in first N results (contd.)

2.6 Co-Reference Resolver: Our Re-ranking Demo

We have made a demo which serves as a meta search engine for people search; Figure 2.3 shows the front page of our demo – which we call **Co-Reference Resolver**.

The demo requires a user to enter the name of the person to be searched. After the user has entered the name of the required person, say *Sandeep Pandey*, the demo contacts the Google search engine to fetch the search results, which are then displayed to the user. This set of results is exactly the same as returned by querying Google directly for *Sandeep Pandey*, except that it restricts the results to top K . Figure 2.4 shows the first page of the Resolver’s search results for the query *Sandeep Pandey* with $K=20$. A select link is displayed beside each search result. This allows the user to indicate his/her preference by clicking on the appropriate select link. The indicated page is taken as the base page – the page containing information about the entity of interest. The Resolver then classifies the search results using the pair-wise disambiguation method discussed in Section 2.2. The pages are then re-ranked in the decreasing order of their scores, and presented to the user.

Figure 2.5 shows the select results for the search of Figure 2.4, when the user has selected the page with the URL – <https://www.cs.cmu.edu/%7Espandey> – to indicate the preference. Also shown are the classifier’s scores for the pages. The indicated base page refers to *Sandeep Pandey*, a graduate student at Carnegie Mellon University. Figure 2.5 shows that the relevant pages have come at the top after re-ranking. Manual inspection

of the search pages indicated that there were in all 2 different *Sandeep Pandey*s in the top 20 search results: one is a student at CMU, and the other a social worker. We found that the social worker *Sandeep Pandey* has a large Web presence, and that, out of the 20 pages, just 3 pages referred to the *Sandeep Pandey* of CMU. These pages were overshadowed by the pages referring to the social worker *Sandeep Pandey* in the original search results, making it difficult for a user to browse the results. Observe, in Figure 2.5, that our disambiguator could get all the three pages of interest to the top. Also, observe that the scores assigned to the relevant pages are much higher compared to the scores of the irrelevant ones. Such a high separation of scores between the relevant and the irrelevant pages allows a user to focus on the required results. In figure 2.5, a green line marks the separation between the relevant pages and the irrelevant ones. Note, however, that the quality of the results depend highly on the selected base page, and on the Web presence of the desired entity. The better the information content of the selected page, and the Web presence of the desired entity, the better is the performance of our system.



Figure 2.3: Co-Reference Resolver Front Page



Co-Reference Resolver

sandeep pandey

Search Results (Results 1 - 10 of 20.)

Homepage of Sandeep Pandey - CMU

Sandeep Pandey. Office address: **Sandeep Pandey** 8122, Wean Hall 5000 Forbes Avenue Pittsburgh, PA 15213-3891 Phone: (412)-268.3070 Email: ...

<http://www.cs.cmu.edu/~spandey/>

Fellowship - Sandeep Pandey (UTTAR PRADESH)

Fellowship - **Sandeep Pandey** Reoti, Ballia District UTTAR PRADESH, Funds Disbursed. 2001, Silicon Valley, \$800.00. 2002, Silicon Valley, \$800.00. ...

<http://www.ashanet.org/projects/project-view.php?p=285>

Asha for Education(TM): Sandeep Pandey wins 2002 Ramon Magsaysay ...

... **Sandeep Pandey**: Profile of an Activist. What made **Sandeep Pandey** give it all up? An engineering graduate of the prestigious University ...

<http://www.ashanet.org/pandey/profile-sandeep.html>

rediff.com: The Rediff Interview/Magsaysay Award Winner Sandeep ...

The Rediff Interview/**Sandeep Pandey**. A decade ago **Sandeep Pandey** quit his job at the Indian Institute of Technology-Kanpur, and plunged ...

<http://www.rediff.com/news/2002/jul/31inter1.htm>

rediff.com: Activist Sandeep Pandey wins Magsaysay award

Activist **Sandeep Pandey** wins Magsaysay award. **Sandeep Pandey** has bagged this year's Ramon Magsaysay award in the 'Emergent Leadership ...

<http://www.rediff.com/news/2002/jul/29mag.htm>

Sandeep Pandey - Asha founder - Magsaysay award winner

Sandeep Pandey wins Magsaysay award Recognition marks years of work in education and peace July - August 2002 - The Ramon Magsaysay Awards Foundation has ...

<http://www.indiatogether.org/people/pandey.htm>

DBLP: Sandeep Pandey

Figure 2.4: Co-Reference Resolver Search for Sandeep Pandey



Co-Reference Resolver

SELECT Results(Results 1 - 10 of 20.)

Homepage of Sandeep Pandey - CMU

Sandeep Pandey. Office address: **Sandeep Pandey** 8122, Wean Hall 5000 Forbes Avenue Pittsburgh, PA 15213-3891 Phone: (412)-268.3070 Email: ...

<http://www.cs.cmu.edu/%7Espandey/> 1.0

Monitoring the dynamic web to respond to continuous queries

... Authors, **Sandeep Pandey**, Indian Institute of Technology, Powai, Mumbai, India: Krithi Ramamritham, Indian Institute of Technology, Powai, Mumbai, India. ...

<http://portal.acm.org/citation.cfm?id=775245> 0.999999996770579

DBLP: Sandeep Pandey

Sandeep Pandey. ... 2004. 3, EE, **Sandeep Pandey**, Kedar Dhamdhere, Christopher Olston: WIC: A General-Purpose Algorithm for Monitoring Web Information Sources. ...

<http://www.informatik.uni-trier.de/%7Eley/db/indices/a-tree/p/Pandey:Sandeep.html> 0.999999995755023

Sandeep Pandey's critique of the education system.

More than schooling A critique of the modern education system, by **Sandeep Pandey**. Mail this ... this direction. **Sandeep Pandey** June 2001. ...

<http://www.indiatogether.org/opinions/pandey.htm> 0.098036670489609

Sandeep Pandey : Indias Professional Seccessionist

Sandeep Pandey. Professional Seccessionist. It is to solve Kashmir political ... of "its all India's fault" crowd ... INTERVIEW: **Sandeep Pandey**.. ...

<http://www.geocities.com/enemiesofbharat/sandeepandey.html> 0.07493564517751344

2002 Ramon Magsaysay Awardee for Emergent Leadership - Sandeep ...

The 2002 Ramon Magsaysay Award for Emergent Leadership CITATION for **Sandeep Pandey** Ramon Magsaysay Award Presentation Ceremonies 31 August 2002, Manila ...

<http://www.rmaf.org.ph/Awardees/Citation/CitationPandeySan.htm> 0.06212213081768955

The Importance of Disarmament Initiatives by Dr. Sandeep Pandey on ...

Figure 2.5: Co-Reference Resolver Select Results for Sandeep Pandey

Chapter 3

Clustering: An Unsupervised Application Scenario

In this chapter, we will look at the application of the Entity Disambiguator in an unsupervised setup. We describe a graph partitioning based clustering algorithm, and see how it performs when applied to the problem of entity disambiguation.

3.1 Problem Definition

In the clustering framework, we attempt to develop a system which can perform clustering of the input documents, such that all co-referring documents are in the same cluster, and the non co-referring documents are in different clusters. For example, a user queries a search engine for information on *Tom Mitchell*. Without requiring the user to intervene, our goal is to return a set of clusters, one of which entirely consists of only those documents that refer to the *Tom Mitchell* who is a professor, another one might refer entirely to the *Tom Mitchell* who is a musician, and so on. Now, it becomes extremely easy for a user to select the cluster of his/her interest, by browsing through just a few URLs from each of the clusters. Stated formally, given a search term, and a number of Web documents obtained as a result of a search query performed on the search term, the problem is to cluster together those documents that refer to the same physical entity.

3.2 Approach

Figure 3.1 shows the different steps of the entity clustering process. As shown in the figure, the system takes as input just the name of the entity the user is looking for, and

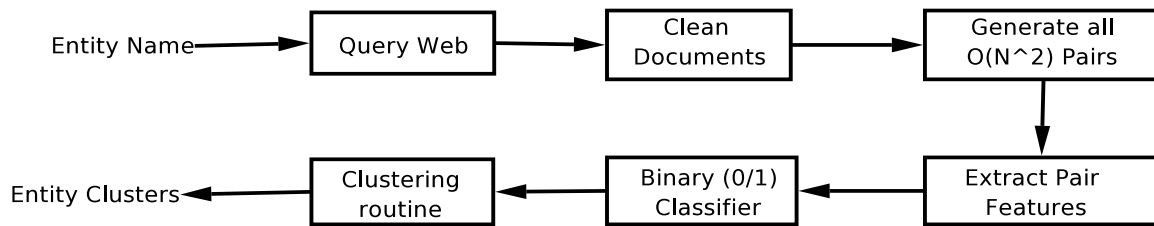


Figure 3.1: Disambiguation as a Clustering Problem

outputs a number of clusters each referring to a separate physical entity. We model the entity clustering problem as a two stage process.

1. Classification of pairs of web pages as being co-referent or not.
2. Clustering of web pages based on the classification result.

After accepting the name of the entity as input, the system queries the Web for the base query to fetch a set of pages. As before, each fetched page is cleaned to form a well formed HTML document, and then converted to a `SpecialDocument` object. This `SpecialDocument` stores and allows access to various important parts of the web page namely the URLs, the text within a window around the search term, and the like. All possible pairs of these `SpecialDocuments` are then formed, pair features extracted, and fed to a pair-wise binary classifier trained using the method discussed in Chapter 2. The prediction of the classifier is then fed to a clustering routine to cluster the documents together. The main idea behind using a clustering algorithm is to remove inconsistencies from the classifier output – inconsistencies of the type where two out of the three pairs corresponding to some three documents were given a label of 1, whereas the third one was given a label of 0.

We do not see direct application of any general clustering algorithm to our model. This is because our features are based on pairs of web pages rather than on each single web page individually. Our implementation uses a graph partitioning based algorithm known as **Cautious** [BBC02]. Cautious is an approximate algorithm to minimize disagreements (the number of 0-labeled edges inside clusters and the number of 1-labeled edges across clusters). The main advantage of using this algorithm, as opposed to other clustering algorithms such as K -means, is that it does not require the number of clusters to be specified in advance. And since, on the Web, we do not know the number of people

having the same name as the search term, a clustering algorithm that does not require the number of clusters to be specified is preferable.

3.3 Cautious: A Graph Partitioning Based Algorithm

Cautious is a correlation clustering algorithm introduced in [BBC02]. Figure 3.2 provides an overview of the algorithm.

The output of the binary classifier can be represented as a complete graph with the documents as vertices and the edges (representing pairs) being given a label of 0 or 1. The algorithm uses a parameter, δ , which specifies the amount of looseness allowed in the clusters: the larger the δ , the more is the looseness. A vertex v is called **δ -good** w.r.t a cluster C , where $C \subseteq V$, if it satisfies the following:

- $|N^+(v) \cap C| \geq (1 - \delta)|C|$
- $|N^+(v) \cap (V \setminus C)| \leq \delta|C|$

where, V is the set of vertices, and $N^+(v)$ is the set of vertices containing v and all those neighbours of v which have a 1-labeled edge with it. In words, a vertex v is said to be **δ -good** w.r.t a cluster C , if the number of positive neighbours (which includes v as well) inside the cluster is at least $(1 - \delta)$ times the cluster size and the number of positive neighbours outside the cluster is at most δ times the cluster size. If a vertex is not δ -good with respect to C , then it is called **δ -bad** w.r.t. C .

As described in Figure 3.2, the algorithm proceeds iteratively in two phases. To begin with, we take an arbitrary vertex, and build a temporary cluster $A(v)$ consisting of the chosen vertex and all those neighbours which have a 1-labeled edge with it. Then in the next phase – which is an iterative phase and is known as the **vertex removal phase** – we test each vertex from the set V against this temporary cluster to see if it is 3δ clean w.r.t. $A(v)$. Vertices failing to satisfy the condition are removed from $A(v)$; the cluster $A(v)$ gets modified in the process. We continue testing until there are vertices in $A(v)$ which fail to satisfy the said condition. This marks the end of the first phase of the algorithm. After the first phase, we have a tightly bound cluster. Then in the next phase, known as the **vertex addition phase**, we add those vertices which might not have a 1-labeled edge with the vertex chosen in the first step, but should actually belong to the same cluster as the members of $A(v)$. In the vertex addition phase, we test each vertex from the set

of vertices V , to check if it is 7δ clean w.r.t. the static cluster, $A(v)$, left at the end of the vertex removal phase. Those vertices which qualify are added to $A(v)$, and are also removed from the set of vertices V , so that they are not considered again. The algorithm then iterates over the mentioned phases starting with a new vertex from the set V , and terminates when either V becomes empty, or all the produced sets $A(v)$ are empty.

- 1: Pick an arbitrary vertex v and do the following:
 - Let $A(v) = N^+(v)$
 - **(Vertex Removal Step):** While $\exists x \in A(v)$ such that x is 3δ -bad w.r.t. $A(v)$, $A(v) = A(v) \setminus \{x\}$.
 - **(Vertex Addition Step):** Let $Y = \{y | y \in V, y \text{ is } 7\delta\text{-good w.r.t. } A(v)\}$. Let $A(v) = A(v) \cup Y$.
- 2: Delete $A(v)$ from the set of vertices and repeat until no vertices are left or until all the produced sets $A(v)$ are empty. In the latter case, output the remaining vertices as singleton nodes.

Figure 3.2: The Cautious Algorithm

3.4 Experiments & Results

We used the dataset mentioned in Section 2.4.1 to measure the performance of the clustering algorithm applied to the entity disambiguation problem.

3.4.1 Evaluation Criterion

As before, we use a five fold cross validation technique on the training dataset to measure the performance of our clustering system. For each of the five folds, we measure the goodness of the clusters obtained as follows. Regardless of the classifier prediction, we treat two documents belonging to two different clusters as predicted non co-referents, while all those in the same cluster as predicted co-referents. We then compute the precision, recall, and $F1$ values, described in Section 2.4.2.1, for the clustering results. As before, we report the Micro $F1$ for our experiments. The results are, of course, dependent on the results of the classification model.

3.4.2 Results & Conclusions

We first observed the effect of δ on clustering performance. For this experiment, we fixed our feature set to contain all the features mentioned in Section 2.3. The similarity features were applied to the text and the title of web pages after removing the search term from them. Table 3.1 gives the effect of varying the δ value on the goodness of the clustering obtained. The best $F1$ was obtained with $\delta = 0.12$, which is still smaller than that obtained without clustering. We noticed that our best performance was obtained at a δ value which was greater than that suggested in the paper [BBC02]. The paper mentions the use of a $\delta < 1/9$, which resulted in much lower performance.

δ	<i>Micro F1</i>	δ	<i>Micro F1</i>	δ	<i>Micro F1</i>
0.022	0.06545	0.116	0.87028	0.125	0.87031
0.1	0.68752	0.118	0.87028	0.14	0.82786
0.111	0.86197	0.12	0.87981	0.17	0.36286

Table 3.1: Effect of variation in δ on performance

Another important observation was that a value of 7δ in the vertex addition phase, was too loose, and was causing non co-referents to easily enter the cluster. We also observed that the vertex removal phase was much strict compared to the vertex addition phase. This indicated that the constants mentioned in the paper were not working for our case. We, therefore, modified the algorithm to use different constants in the vertex removal phase and the vertex addition phase. By modifying the 7δ condition of the vertex addition phase to 3δ – same as the one used in the vertex removal phase – we found that settings with $\delta = 0.17$ gave the best performance. We next compared the performance of the best settings for *Modified Cautious* (i.e., $\delta = 0.17$) against the best settings for *Cautious* (i.e., $\delta = 0.12$); Table 3.2 gives the details for the five folds of the cross-validation. The following conclusions can be drawn from the table.

1. *Modified Cautious* performed consistently better than the direct classifier output shown under the heading – *Before Clustering*. Error improvement was more than 66% on some folds with $F1$ varying from 0.955 before clustering to 0.985 using *Modified Cautious*.
2. *Cautious* was not consistent in performance; on some folds it gave lower $F1$ than

<i>Fold No.</i>	<i>Fold Size (in number of pairs)</i>	<i>Performance (F1)</i>		
		<i>Before Clustering</i>	<i>Cautious ($\delta = 0.12$)</i>	<i>Modified Cautious ($\delta = 0.17$)</i>
1	1780	0.93197	0.90929	0.97572
2	2863	0.93564	0.94389	0.94389
3	5006	0.87233	0.86180	0.87844
4	2995	0.77389	0.79452	0.79278
5	1692	0.95471	0.94459	0.98548
<i>Micro F1</i>		0.88154	0.87981	0.89833

Table 3.2: Clustering Results

<i>Features</i>	<i>Performance (Micro F1)</i>	
	<i>Before Clustering</i>	<i>Using Modified Cautious</i>
Only TFIDF	0.86209	0.88920
TFIDF+Jaccard (Without search term removal)	0.86813	0.90923
TFIDF+Jaccard (After search term removal)	0.89225	0.92637
Only Links	0.18572	0.19486
TFIDF+Jaccard +Links (After search term removal)	0.89578	0.92662
All (including title related)	0.89511	0.92466

Table 3.3: Improvement in Performance through Clustering

the direct classifier output.

3. *Modified Cautious* was better than *Cautious* on four out of the five folds, and was comparable on the remaining fold.

The results suggest that *Modified Cautious* boosts the performance of the classifier output by utilizing the transitive nature of the co-reference similarity.

Manual inspection of the documents showed that the fourth fold, in Table 3.2, contained some documents which were labeled incorrectly. For example, 2-3 pages referred to more than one entity with the same name, but were labeled as referring to only one of them. We, therefore, removed these pages from our dataset, reducing the fold size 2765. Using the correctly labeled dataset, we performed another set of experiments to compare the performance of *Modified Cautious* against the direct classifier output using different feature sets; Table 3.3 presents the results in detail. Once again, the results confirm that clustering, in general, helps boost performance.

Chapter 4

Query Expansion

In this chapter, we explore use of Query Expansion techniques to re-ranking of the search results to suit user’s preference (as discussed in Chapter 2). We describe our approach to query modification, and see how it helps in re-ranking.

4.1 Needs in Entity Search

Our initial work on re-ranking (refer Chapter 2) focused on re-ordering the list of URLs returned by the search engine so that the interesting URLs – those referring to the entity of interest – are ranked higher than the uninteresting ones. We observed that, in many cases, the overall precision of the set was very low; the reason being the base query being too general/broad. A study of a user’s behaviour on the Web indicates that a user typically starts off with a base query denoting the person name, and then modifies this query using discriminating words using his/her discretion when the search result contains several irrelevant pages. We noticed that our re-ranking system will be of better use if it can assist the user with “proper” words to modify the query, along with the basic re-ranking of the documents. This chapter gives the details of our work towards developing a *Dynamic Keyword Suggester* module, and its use in re-ranking to provide a better ranked list to the user – the one in which the interesting pages are not only at the top, but are also large in number.

4.2 Related Work

In the field of preference mining, the techniques of automatic query expansion have been used to generate query modifications in varied scenarios; techniques like query modifica-

tions using query logs are well known. A few other techniques [RKJZ04] involve mining the anchor text to generate the query refinements. All these techniques work well where the training data is available in fair amount to generate useful suggestions. This does not fit our model as we cannot predict a priori the entity a user will be looking for. We, therefore, need a way to generate query modifications using a limited amount of training data – in the most likely case, one positive and one negative document.

4.3 Our Approach to Query Modifications

We observe that distinguishing keywords is what we need to generate effective query modifications. Such words are those which occur in abundance in relevant set of pages, and scarcely appear in the irrelevant set, if at all. Gary Flake et al., [FGLG02], demonstrate the usefulness of nonlinear Support Vector Machines (SVMs) to generate query modifications for accumulation of homepages and conference pages. Theirs is the first attempt to utilize the distinguishing and generalization capabilities of SVMs to generate effective query modifications. Their method, however, cannot be applied directly to our set up mainly because of the lack of training data. We need the capability to generate modifications using very few positive and negative documents – in the extreme case, a single positive and a single negative document. We use their idea of using SVMs, which we modify and build upon later to capture the correlation of words, to generate query modifications in our model. We proceed using the following steps to test the applicability of the method in our setup.

- **Preprocessing the Document:** We preprocess each HTML document to extract the text, and represent it by its TFIDF (IDFs learnt on a static data set as mentioned in 2.4.1) weight vector. During this step, we remove the tokens which form part of the base query from the text.
- **Learning Linear SVM:** Using the preprocessed labeled data, we train a linear SVM. We extended libsvm [lib] to generate the weight vector out of the support vectors.
- **Word Selection:** We take the largest T positive magnitude components of the weight vector as the selected words.

We experimented on 8-10 different entities, using a hand-labeled training data consisting of about 10-15 positives documents, and an equal number of negative documents, in each case. Tables 4.1 and 4.2 list the top 10 words, as given by the SVM, for the two base queries: *Ashish Gupta* and *Tom Mitchell*. The words listed in Table 4.1 are for Ashish Gupta, a venture capitalist, and those listed in Table 4.2 are for Tom Mitchell, a CMU Professor. From the domain knowledge, we know that the words like `junglee`, `stanford`, and `ibm` actually relate closely to the required *Ashish Gupta*. Similarly, words like `machine`, `carnegie`, and `computer` aptly describe the desired *Tom Mitchell*. We next explore use of this technique in re-ranking.

junglee	mumick
kanpur	inderpal
venture	vldb
stanford	daksh
ibm	materialized

Table 4.1: Ashish Gupta (Junglee)

learning	machine
cmu	mellon
carnegie	computer
ist	artificial
mccallum	science

Table 4.2: Tom Mitchell (CMU)

Table 4.3: Top 10 words using Linear SVM

4.4 Application to Re-ranking

Our experiments (refer Section 4.3) confirmed that our method of generating keywords work if the positive and the negative set contains at least 10-15 documents. We next explored the application of our method in the most likely case where the user points to a single positive and a single negative document ¹. We found that the text from a single positive and a single negative page does not provide the necessary boost to the important keywords in some cases. Also, in an online setting, expecting a user to label more than one positive and one negative document, is too strong an assumption. We combined our pairwise disambiguation strategy (refer Section 2.2), with the keyword generation strategy, to generate useful suggestions, even in the restrictive case of one labeled positive and one labeled negative document. Such a utility provides a **Dynamic Keyword Suggester**

¹Now on, these two sets will be referred to as the “base positive” and the “base negative” sets respectively

module to the user – a suggester which adapts to the user’s preference.

We also developed an **Improved Ranker** system which uses the Dynamic Keyword Suggester as a guide to fetch pages intelligently, and hence improve the precision of the fetched set. We next describe our overall Improved Ranker system.

4.4.1 System Design

Figure 4.1 describes the high level working of the Improved Ranker system.

Suppose the user is looking for some entity with the name w_b . The user indicates his/her preference to the system by submitting a set of positive (BP) and negative (BN) documents on the base query w_b . The system’s job is then to finally return a ranked list of URLs, such that those referring to the entity of interest are ranked higher than the others, while also ensuring a high precision of the fetched set.

To augment the base sets (BP and BN), the system initially fetches top K_1 documents by querying the search engine for w_b . After forming the pairs of each fetched document with each of the base page, the system uses a pair-wise entity disambiguator (refer Chapter 2) to classify the pairs. The disambiguator assigns each pair a score representing the probability of the two documents in the pair being co-referent. We used a logistic regression based pair-wise entity disambiguator, built using all the features implemented in the disambiguator mentioned in Chapter 2, for our experiments. The system then aggregates the scores for each document using Equation 4.1.

$$AggregatedScore_k = \frac{\sum_{i=1}^{|BP|} Pr(y = 1|pair(d_k, d_i)) + \sum_{j=1}^{|BN|} (1 - Pr(y = 1|pair(d_k, d_j)))}{|BP| + |BN|} \quad (4.1)$$

These scores are used to augment BP with documents having a score above a threshold T_1 , and BN with documents having a score less than a threshold T_2 . For our experiments, we kept $T_1 = 0.9$, and $T_2 = 0.1$. The augmented base sets serve as input to the linear SVM to generate the query modifications. From each of the top D suggested modifications, the system fetches new top K_2 documents, and uses an entity disambiguator to classify them. We used the SVM, learnt using the augmented base sets, to generate the final score for each document as the distance from the hyperplane. Finally, the system returns a ranked list of URLs consisting of all the documents it has fetched so far.

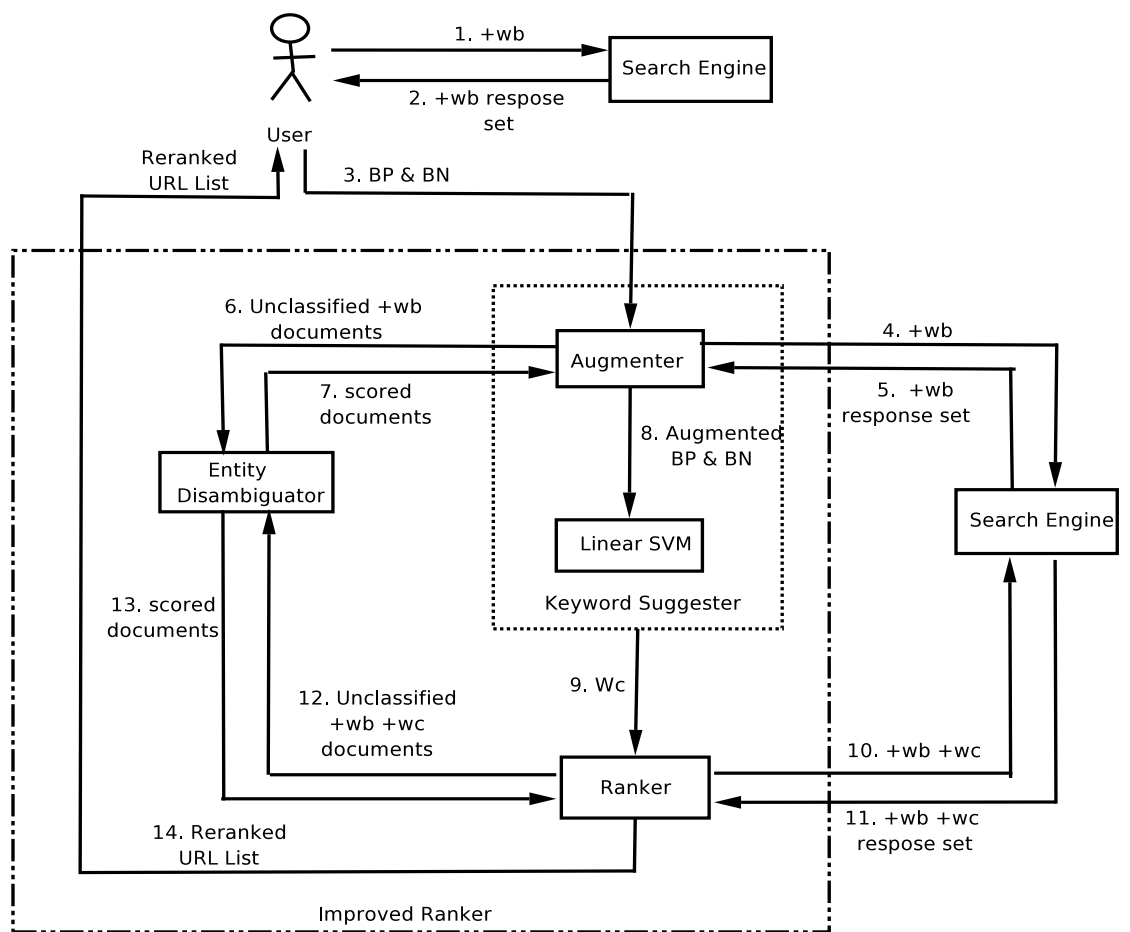


Figure 4.1: Improved Ranker System

4.5 Experiments

We compared the performance of the Improved Ranker against our earlier method (refer Chapter 2), which we call the Baseline Ranker, by fetching an equal number of pages in both the cases. For the Baseline case, we fetch top 150 documents on the base query, and classify them according to our pair-wise scoring method. We also aggregate the document scores using Equation 4.1, similar to the case of the Improved Ranker. The Baseline then finally re-ranks the documents according to these aggregated scores. By fixing the number, K_1 , of documents to fetch on the base query, as 50, the number, D , of keyword suggestions, as 10, and the number, K_2 , of documents to fetch on each of the suggested queries, as 10, we fetch the same number of pages in both the Baseline as well as the Improved case. We measured the performance on 12 different entities taken from the augmented dataset that we had used in Chapter 2. For each base query w_b , we queried Google to take the first good positive and negative document in its response list as our base positive and base negative respectively.

Since the manual labeling of the documents is a time consuming process, we trained an Oracle for each case independently, using the training data, and used the labels assigned by it as the true labels of the fetched documents. In each of the 12 cases, we trained a linear SVM to act as an Oracle. Table 4.4 gives our confidence in the oracle for each of the 12 cases. Table 4.4 reports values obtained using 3 fold cross validation on the training data, and averaged over 5 runs.

4.6 Results & Conclusions

Figure 4.2 shows the precision at different values of top N documents in the final ranked list for the three cases: the Improved Ranker, the Baseline Ranker, and the unaltered Google returned search list. Note that the Improved Ranker fetches documents from different query streams, and subsequently merges the results intelligently, while both the Baseline Ranker and the Google graphs are for the documents fetched using the base query. The following conclusions are noteworthy.

1. The graph shows that, given the same number of documents to fetch, the Improved Ranker is able to fetch more positives (if not equal) than can be obtained by fetching

<i>Entity Name (w_b)</i>	<i>Description</i>	<i>Precision</i>	<i>Recall</i>	<i>F1</i>
Ashish Gupta	Fashion Designer	1.000	0.78	0.874
Ashish Gupta	Junglee Corporation	1.000	0.825	0.904
Ashish Gupta	Northwestern University	1.000	0.729	0.839
Barbara Johnson	Author	1.000	0.946	0.971
Barbara Johnson	Governor	1.000	0.743	0.845
John Miller	Roller Coaster Designer	1.000	0.933	0.966
Michael Jordan	Football Player	1.000	0.975	0.987
Michael Jones	Piano Player	1.000	0.708	0.827
Michael Jones	Sculptor	1.000	0.927	0.959
Patricia Williams	Law Professor	1.000	0.778	0.873
Patricia Williams	Marketing Professor	1.000	0.958	0.978
John Miller	Computation Economics Professor	1.000	0.787	0.88

Table 4.4: Oracles' details

the pages on the base query alone. This indicates that the words picked by the Dynamic Keyword Suggester using the augmentation strategy are effective for query modifications even in the case of limited training data.

2. The method of fetching pages from several streams, and then re-ranking them, proves to be effective in obtaining a high precision at different values of top N . The graph shows that the precision of the Improved Ranker was always better than the Baseline Ranker.

Table 4.5 shows the base URLs, the URLs picked by the system to augment the base sets, the keywords generated by the keyword suggester, and the top 10 URLs of the Baseline as well as the Improved Ranker for the base query – *John Miller*. Manual inspection of the top 10 shown URLs indicates that the Improved Ranker could get 9 positives in the top 10 URLs, whereas the Baseline Ranker has a precision of 0.3.

3. We, however, note that the set of pages fetched by the Improved Ranker can be biased towards some particular aspect of the entity. There is not much we can do here as the user's preference is itself biased. Although, our augmentation strategy

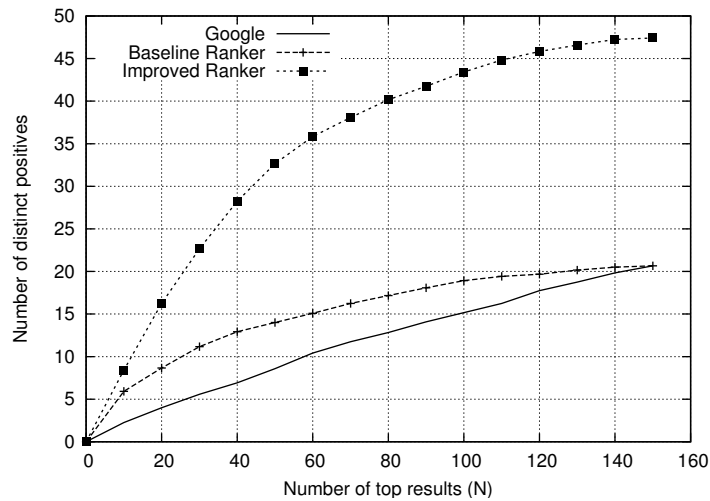


Figure 4.2: Precision at top N re-ranked results

improves the coverage of the set – by utilizing various features – to some extent, it fails when there is no strong positive or negative document in the initial set of top K_1 pages. Our framework, however, allows the user to specify as many base pages as the user thinks are necessary to ensure a good recall.

4. If the profile of the entity of interest is too ambiguous on the web, i.e., there are too many entities sharing that profile, and the entity of interest does not have a large web presence, the performance of our method degrades. For example, suppose the user is searching for *Ashish Gupta* who is a computer science graduate from IIT Delhi, and is a post graduate student at North Western university. We see that the augmentation method fails to distinguish between the entity of interest and other entities having similar profile. This leads to the inclusion of false positives to the base set, and subsequently to the generation of noisy keywords.

Table 4.6 shows the base URLs, the URLs picked by the system to augment the base sets, the keywords generated by the keyword suggester, and the top 10 URLs of the Baseline as well as the Improved Ranker. We, however, observe that the Improved Ranker is still able to fetch more positives (3 out of top 10) compared to the Baseline (1 out of top 10).

5. We also observed that some of the suggested words were highly correlated, and that we can improve on the keyword suggestions by capturing such correlation, and also prevent unnecessary queries.

<i>Base Positive URL</i>	http://home.nyc.rr.com/johnmiller/
<i>Base Negative URL</i>	http://zia.hss.cmu.edu/econ/
<i>URLs Augmented to Positive Set</i>	http://home.nyc.rr.com/johnmiller/ https://www.wfhm.com/wfhm/md-... http://search.eb.com/coasters/i_miller.html
<i>URLs Augmented to Negative Set</i>	http://zia.hss.cmu.edu/econ/ http://www.nervana.montana.edu/people/jpmbio.html http://zia.hss.cmu.edu/miller/
<i>Generated Keywords</i>	coasters, builder, roller, coaster, fargo, rockville, mortgage, amusement, cyclone, athey
<i>Top 10 URLs of the Improved Ranker</i>	http://home.nyc.rr.com/johnmiller/ https://www.wfhm.com/wfhm/md-... http://search.eb.com/coasters/i_miller.html http://www.needs.org/needs/public/search/search_... http://web.mit.edu/invent/iow/miller.html http://home.nyc.rr.com/johnmiller/pages.html http://library.thinkquest.org/5384/Early_Roller... http://www.ultimaterollercoaster.com/.../index.shtml http://www.ultimaterollercoaster.com/.../coney7.shtml http://history.amusement-parks.com/allenmain.htm
<i>Top 10 URLs of the Baseline Ranker</i>	http://home.nyc.rr.com/johnmiller/ https://www.wfhm.com/wfhm/md-... http://search.eb.com/coasters/i_miller.html http://www.cambridge2000.com/.../html/architect... http://www.britannica.com/ebi/article... http://www.seanet.com/%7Ejimxc/Politics/ http://www.seanet.com/%7Ejimxc/Politics/June2005_2.... http://www.artifacts.net/index.php/.../exhibition... http://www.qnews.com.au/content.asp?personid=282 http://www.gutenberg.org/browse/authors/m

Table 4.5: Some results for *John Miller*, Roller Coaster Designer

<i>Base Positive URL</i>	http://www.cs.northwestern.edu/%7Eagupta/
<i>Base Negative URL</i>	http://web.mit.edu/ashishg/www/
<i>URLs Augmented to Positive Set</i>	http://www.cs.northwestern.edu/%7Eagupta/ http://www.cs.utexas.edu/users/agupta/
<i>URLs Augmented to Negative Set</i>	http://www.geocities.com/ashish_iitm/ http://web.mit.edu/ashishg/www/ http://widget.ecn.purdue.edu/%7Egupta7/main2.html http://allthatbothers.blogspot.com/2005/05/... http://ashishgupta.bravehost.com/funny01.html
<i>Generated Keywords</i>	cs, graduate, delhi, computer, austin, invaders, ee, spring, fall, programming
<i>Top 10 URLs of the Improved Ranker</i>	http://www.cs.utexas.edu/users/agupta/ http://www.cs.northwestern.edu/%7Eagupta/ http://www.cs.utexas.edu/users/agupta/Resume.html http://www.cs.utexas.edu/users/almstrum/cs373/sp05/... http://www.cse.iitk.ac.in/alumni/who.html http://www.cs.utexas.edu/%7Ealmstrum/cs373/sp04/... http://www.cs.northwestern.edu/%7Eagupta/graduate... http://www.cs.northwestern.edu/%7Efabianb/classes/... http://www.people.vanderbilt.edu/%7Eashish.gupta/ http://www.ncb.ernet.in/placements/Contents/Students...
<i>Top 10 URLs of the Baseline Ranker</i>	http://www.cs.northwestern.edu/%7Eagupta/ http://www.cs.utexas.edu/users/agupta/ http://home.iitk.ac.in/student/gashish/ http://www.tribuneindia.com/2005/20050128/delhi.htm http://www.cs.utexas.edu/users/agupta/Resume.html http://www.hindu.com/2005/06/13/stories/... http://www.public.asu.edu/%7Eagupta20/ http://grace.che.ufl.edu/Symposium/2005/... http://www.tribuneindia.com/2004/20040822/cth3.htm http://widget.ecn.purdue.edu/%7Egupta7/main1.html

Table 4.6: Some results for *Ashish Gupta*, Northwestern University

4.7 Capturing Correlation

We notice, from Tables 4.1 and 4.2, that the SVM suggested list of keywords contains words, such as `materialized` and `vldb`, which have similar co-occurrence statistics w.r.t the base query. Capturing such a correlation is beyond the capabilities of a phrase recognition system. We also note that, if we are to issue T queries, and fetch documents from these streams, we will end up getting many duplicate URLs if we do not capture the correlation and eliminate unnecessary queries. So, we next focused towards re-ordering the SVM suggested list of words by exploiting their correlation.

We tried to improve upon the basic framework of generating important keywords by capturing correlation in two different ways, which form the two main terms of our scoring function (refer Equation 4.2). The first term measures the **Pointwise Mutual Information (PMI)** [Tur01] to capture the correlation of the candidate word w.r.t the positive set, and the second term uses count queries on the web to capture the correlation between the already used words and the candidate word; Table 4.7 describes various parts of the Equation 4.2 in detail.

$$score_{w_c} = \frac{|P \cap w_c|}{|C|} \log \left(\frac{\frac{|P \cap W_c|}{|C|}}{\frac{|P|}{|C|} \times \frac{|W_c|}{|C|}} \right) - \max_i \log \left(\frac{count_{web}(+w_b + w_i + w_c)}{count_{web}(+w_b + w_c)} \right) \quad (4.2)$$

where,

w_b = Base query: Entity Name

C = Corpus: Set of all documents fetched

P = Set of positive documents in the Corpus

N = Set of negative documents in the Corpus

W_c = Set of candidate words; $w_c \in W_c$

W_i = Set of words already used for query modification in some previous iteration of the algorithm; $w_i \in W_i$

$count_{web}(x)$ = Count of results returned by the search engine for the query x .²

$score_{w_c}$ = Score for the candidate word w_c .

For a base query, w_b , we use a carefully trained oracle to classify the fetched documents as positive or negative according to their relevance to the entity of interest. We generate

²Courtesy Yahoo! and Google

$P \cap w_c$	is the subset of P containing w_c
$\frac{ P \cap w_c }{ C } \log \left(\frac{\frac{ P \cap w_c }{ C }}{\frac{ P }{ C } \times \frac{ w_c }{ C }} \right)$	represents the pointwise mutual information at $(1, 1)$. This measures correlation of w_c with positive set.
$\log \left(\frac{\text{count}_{web}(+w_b+w_i+w_c)}{\text{count}_{web}(+w_b+w_c)} \right)$	measures correlation between already issued queries w_i and the candidate words w_c

Table 4.7: Component Terms of the Scoring Function (Equation 4.2)

a set of candidate words, W_c , using an SVM trained on a smaller set of documents, and initialize the set of explored words, W_i , to be empty. Our algorithm then iteratively performs the following steps:

1. Scores each word in the set W_c using the Equation 4.2, and picks the best word w_t – the one having the maximum score.
2. Queries the web for $+w_b + w_c$, and classifies the set of pages, fetched K at a time, till we reach a state where we do not get any positive in the recently fetched set of K pages. Augments the dataset to our corpus C .
3. Removes w_c from W_c , and adds it to W_i which represents the set of explored words. Stops when W_c becomes empty, i.e., no candidate word left to be explored.

This gives an order of words which can be used to query the Web. We compare this order with the ordering obtained by taking words in the decreasing order of their SVM weights, in relation to an interesting application of an entity disambiguator which we call *Background Search*. In Chapter 5 (Section 5.2) we present an experimental evaluation of our proposed methods for handling correlation.

Chapter 5

Application of Entity Disambiguator in Other Scenarios

In the previous chapters, we discussed some of the applications of an entity disambiguator, such as *Re-ranking*, *Clustering*, and *Suggesting Query Enhancing Keywords*, which suit more to an interactive environment. In this chapter, we briefly discuss some batch application scenarios, such as those mentioned in Sections 1.1.3 and 1.1.4, where an entity disambiguator finds useful application.

5.1 Set Valued Attribute Accumulation

In this section, we discuss one of the batch application of an entity disambiguator, giving an outline of a probable strategy, and our preliminary results for the same.

5.1.1 Problem Definition

Suppose we have some information about a person, and wish to gather all the possible values for some attribute in association with that person. For example, we might desire to find all the organizations that the person has been associated with, all the co-authors that the person has written a paper with, or all the cities where the person has been to. We call such attributes as **Set Valued** attributes. The values for some of these attributes, like cities, are hard to gather, while others like organizations are defined relatively easily. Stated formally, given the name of an entity, along with some information about the entity (say a few positive and a few negative documents), and the type of the set valued attribute whose values are to be collected, the problem is to collect all the possible values for the desired attribute in association with the entity of interest.

5.1.2 Strategy

Figure 5.1 gives an outline of the desired system. As shown in the figure, we model the problem to be made up of two independent classification problems: **entity disambiguation** and **‘type’ based classification**. The first stage disambiguates the collected pages to collect those referring to the entity of interest, and the second stage helps in identifying snippets/pages that contain the desired *type* information. We assume that we would have some information – about the entity of interest – that will be used by the entity disambiguator to disambiguate the collected pages. For example, the available information can be in the form of a set of positive and/or negative documents. We also assume the availability of *type* related information. In Figure 5.1, the disambiguator is shown to be a black box, which could classify the fetched pages either using the aggregated score method as discussed in Section 4.4.1 (Equation 4.1), or can be based on other strategies.

We proceed as follows. We query the Web using a combination of *type* and *entity* related keywords extracted using the available information. We pass the fetched pages through the entity disambiguator to collect the relevant pages which we then pass through a snippet generator to generate a set of *snippets*. We hope that the required information will be available in the proximity of the entity name within the page. We, therefore, collect text within a window around the entity name, from wherever the entity name appears in the page, and generate a set of snippets. We neglect indirect references to the entity so as to stay out of NLP for the time being. We next pass the generated snippets through a second classifier trained to recognize snippets containing the desired *type*. Finally, we extract the required *type* information by passing the qualifying snippets, in a ranked order, through an information extraction module designed to extract the required information. The details of the information extraction module are left open; it can be a simple rule-based IE module which simply extracts the *type* information based on some heuristics, or it can be a complicated system involving aggregation/deduplication of the extracted type values.

5.1.3 Experiments & Preliminary Results

Suppose we are interested in finding out all organizations that Ashish Gupta, a venture capitalist, has been associated with. We used a dictionary of type keywords, such as `inc.`,

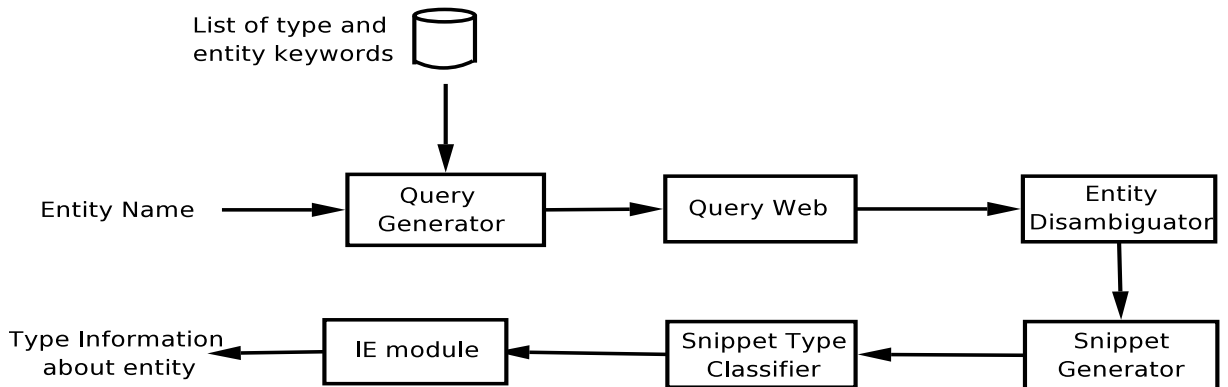


Figure 5.1: Set Valued Attribute Accumulation System

organization, startup, and institute, that are usually associated with an organization. We, however, did not use any entity related keyword – except for the name of the entity – for query generation. Using a combination of words from the *type* dictionary, and the entity name, we query the Web and fetch top 20 pages from each query, i.e., 20 pages from each of the queries +“Ashish Gupta” +industry, +“Ashish Gupta” +inc., and the like. In this section, we describe our settings for the *entity* and *type* classifiers.

5.1.3.1 Entity Disambiguator

We trained our SVM based disambiguator using about 10 – 15 positive and an equal number of negative documents. We used a single-document based entity disambiguator instead of a pair-wise disambiguator for our experiments. We, however, believe that the performance can be improved by using the pair-wise method and employing the aggregated score method discussed in Section 4.4.1 (Equation 4.1).

5.1.3.2 Type Based Snippet Classifier

We built a binary SVM classifier for classifying the generated snippets to identify those containing an organization in them. The details of the classifier are as under.

1. **Preprocessing:** We preprocess each snippet by passing it through a POS tagger; we used the Brills tagger [Bri92] for our experiments. We then group a sequence of proper noun words into a compound token.
2. **Feature Set:** We identified the following set of features.

- *Dictionary Features*: These features identify presence of words like *CEO* or *chairman* in the snippet.
 - *Distance Based Features*: These features measure the average distance of the proper noun phrases from the name of the person, of the proper nouns with more than two capital letters from the name of the person, and the like.
 - *Count Based Features*: These features count the number of proper nouns with all capital letters, number of proper nouns with more than two capital letters, and the like.
 - *Reg-ex Features*: These features test the existence of a pattern where a preposition is followed by a proper noun, like in ‘works in IBM’, and where two proper noun phrases are separated by a preposition, like in ‘University of California’.
3. **Training**: We trained a binary SVM classifier using about 50 positive and 50 negative, manually tagged snippets. All the snippets contained name of at least one person in them, but the positives ones contained an organization name as well. This is because, for this application we are sure that every snippet to be classified will contain an entity name within it. Table 5.1 gives our confidence in the classifier for snippet window sizes of 20, 100, and 200 words. The values in the Table were obtained by averaging over 5 runs of 3-fold cross validation. We found that taking a window of 100 words gives a decent performance, and therefore used this window size for our experiments.

<i>Measure</i>	<i>Window Size</i>		
	20	100	200
Accuracy	0.732	0.842	0.788
Precision	0.725	0.860	0.850
Recall	0.740	0.858	0.843
F1	0.732	0.859	0.847

Table 5.1: Performance of the organization-tuned *type* classifier

5.1.3.3 Ranking Snippets

The entity disambiguator assigns a score to each snippet, which is the distance of the snippet’s feature vector from the learnt hyperplane. This gives a measure of the classifier’s confidence in the snippet to contain an organization. We use this score to rank the snippets; Table 5.2 shows the top 5 snippets for *Ashish Gupta*. The proper nouns, as tagged by the POS tagger, are highlighted.

10.131	DBLP : Ashish Gupta Ashish Gupta List of publications from the DBLP Bibliography Server - FAQ Ask others : ACM - CiteSeer
10.125	its PC division. . He left Oracle in 1984 to become Founder , Chairman and CEO of his own software company , Gupta Corporation which in 1993 became the first Indian-American run software company to go public in the world. . Gupta Corporation was responsible
9.499	advisor and investor in several information technology companies , including Bodha (acquired by Peregrine , Obongo (acquired by AOL , and Pointcross . . Ashish was an investor and board member at Daksh , a a leading customer services company acquired by IBM for over \$160 million. . He
8.123	articles on technical topics he has edited a book on the topic of “ materialized views ” published by MIT press. . Ashish holds a Ph.D. . in database technology from Stanford and a bachelors degree in computer science from the Indian Institute of Technology
8.002	among a Cluster of Heterogeneous Unix Workstations . . CCGRID 2001 : 668-673 33 EE Ashish Gupta: X-tegration - Some Cross-Enterprise Thoughts . . ICLP 2001 : 6 32 EE Ashish Gupta ,

Table 5.2: Top 5 Snippets for *Ashish Gupta*

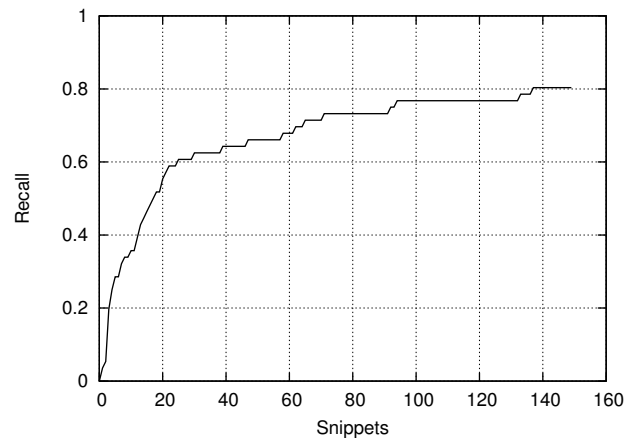
5.1.3.4 Extracting and Aggregating Organizations

We manually extracted and deduplicated the organizations from the ranked snippets. Table 5.3 gives an aggregated list of the organizations along with their aggregated scores. We divide the score of each snippet equally among all the organizations occurring in it. The score of each organization is then taken to be its accumulated score across all snippets. The system was not able to recall the organizations shown with a 0 score in the figure. Figure 5.2 shows the organization recall of the top N ranked snippets for *Ashish Gupta*. Out of the manually tagged 54 organizations related to him, the system could recall 45 – giving a recall of 0.83. The top 20 snippets alone gave a recall of 0.6.

5.2 Background Search

In this section, we will look at another interesting batch application of an entity disambiguator, namely *Background Search* for a person.

Rank	Organization	Score	Rank	Organization	Score
1	Information Systems	39.5872	28	PDIS	3.3766
2	Junglee	30.7679	29	AOL	3.3576
3	Woodside	25.5882	30	Pointcross	3.3576
4	IEEE Data Engineering	18.5530	31	JICSLP	3.0039
5	SIGMOD	17.4856	32	CIKM	2.7512
6	IBM	16.9193	33	AT&T Bell Laboratories	2.4618
7	IIT Kanpur	15.7175	34	HPDC	2.2542
8	Stanford	15.4907	35	Intell. Data Anal.	2.2519
9	Oracle	14.6795	36	Haathi	1.4167
10	International Conference Database	14.2168	37	PPCP	1.3757
11	PODS	13.8912	38	VeriWave	1.2485
12	Amazon	13.2493	39	Bodha	1.1875
13	IEEE	11.8917	40	Peregrine	1.1875
14	ACM	11.7416	41	Obongo	1.1875
15	Distributed Parallel Databases	11.3999	42	DAISD	1.1274
16	DBLP	10.6319	43	Skyblaze	0.8116
17	MIT	9.4099	44	J. Parallel Distributed Computing	0.5008
18	VLDB	8.5482	45	Internet Measurement Conference	0.1254
19	ICDE	8.0119	46	CIFE	0
20	Almaden	7.9807	47	GIGACT	0
21	EDBT	6.7950	48	WebDB	0
22	DOOD	6.7526	49	PLANX	0
23	Daksh	6.6456	50	Odesk	0
24	CCGRID	4.0013	51	TODS	0
25	ICLP	4.0013	52	Jasper	0
26	Tavant	3.7910	53	Desk	0
27	NGITS	3.3766	54	SIGART	0

Table 5.3: Organizations augmented for entity *Ashish Gupta*Figure 5.2: Organization Recall for top N ranked snippets

5.2.1 Problem description

Consider a scenario where we require to fetch as many pages referring to a person of interest as possible. We call such an application as **Background Search** for the person. Such a system might serve as a platform to perform many other tasks. If we have a background search system, which also ensures a reasonable recall, we can extract important information about a person, such as the job of the person, the organizations that the person has been associated with, the names of the news papers where the person's interviews have appeared, or any other background information. We, however, want to do this job efficiently. Search engines, like Google and Yahoo!, do not allow us to fetch more than a maximum number of pages on a query. We, therefore, need to pose multiple queries, and do so intelligently.

We see that every query results in a set of pages, and every page is composed of a set of words; Figure 5.3 gives a visualization of the same. In order to get a good coverage using a minimum number of queries, we want to move in the query plane so that not only the precision of each query is high, but also there is minimum/no overlap of the documents/words between the posed queries. Our problem, therefore, is to fetch maximum number of relevant pages, while posing minimum number of queries.

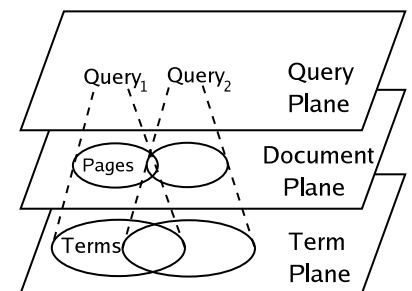


Figure 5.3: Visualization

5.2.2 Strategy

Figure 5.4 gives an outline of our system. We assume availability of some information in the form of a few positive and a few negative documents about the entity of interest. We train an entity disambiguator using the available training data, and then use it to disambiguate the fetched pages and identify the relevant ones. Once again, we treat the disambiguator as a black box, which can either use the aggregated score strategy discussed in Section 4.4.1 (Equation 4.1), or can be based on any other strategy. The available training data can also be used by the keyword generator to generate the initial set of query enhancing keywords – for example, by using the method discussed in Section 4.3.

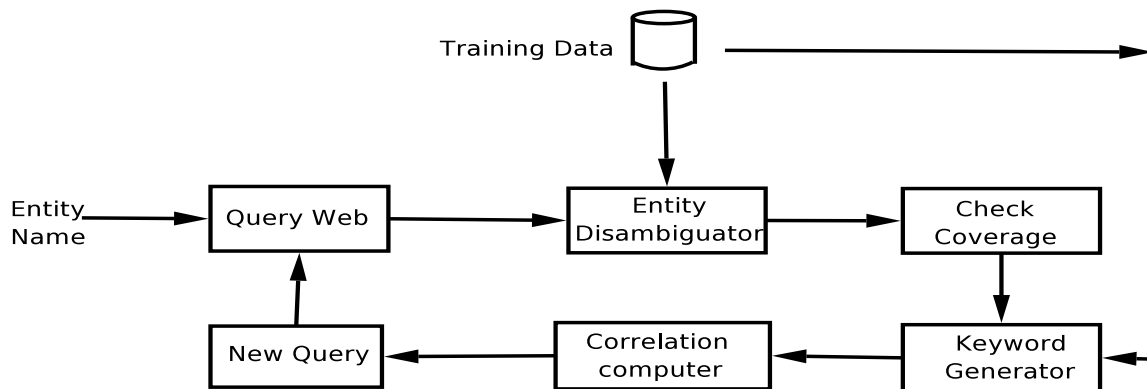


Figure 5.4: Background Search System

Initially, we query the Web for the entity name, fetch the documents in sets of N pages, and disambiguate them using the Entity Disambiguator. We continue fetching the pages till the precision for the recently fetched set of N pages drops to 0. In case limited training data is available, we can use the fetched obvious positives and negatives to augment the training data. Using the available or augmented training data set, the keyword generator generates a set of T keywords which will be used to move on the query plane. The generated keywords are then used, one by one, in some order, to query the Web, and the coverage checked after each such query. As before, for each posed query, we continue fetching the pages in sets of N , till the precision for the recently fetched set of N pages drops to 0. We stop the process when we have acquired a desired recall.

5.2.2.1 Issues

Our preliminary results (Section 5.2.3) show that the described strategy coupled with a good keyword generation method results in a good recall. The following points, however, are noteworthy.

- Our strategy makes use of query expansion using single words only. We, however, note that it might be more efficient to generate composite queries, so as to maximize precision while minimizing the number of queries posed.
- Our strategy generates keywords only once at the beginning of the process using the training data. A better strategy can consider adaptive generation of the keywords as the documents get accumulated.

We see that making decisions adaptively to move on the query plane is a complex task. We look forward to exploring strategies for the same in future.

5.2.2.2 Keyword Generation

We first convert each training document into its corresponding TFIDF weight vector. These vectors are then used to generate 10 keywords using one of the three strategies described here.

1. **Baseline Method:** The 10 keywords are taken to be the top 10 highly weighted words as per the TFIDF weighting using only the positive set of training documents. We use this method as our baseline.
2. **Naive SVM Method:** An SVM is trained using the training documents, and the top 10 keywords are taken to be the top 10 positively weighted words, using the SVM based query generation strategy discussed in Section 4.3. The words are used in the decreasing order of their SVM weights for querying.
3. **SVM (Correlation) Method:** Same as 2, but using our strategy of capturing correlation between the SVM generated keywords, discussed in Section 4.7, to change the order of their use in querying.

We next discuss the performance of the described strategies on a simulated web environment.

5.2.3 Experiments & Preliminary Results

We notice that there is no defined notion of recall on the Web because it hosts a massive amount of data. We, therefore, simulated the Web using a text search engine called *Lucene* [luc]. Since the motive behind this set of experiments is to check the performance of the proposed query strategy, we assumed availability of a human entity disambiguator. We show the results for the following two cases.

1. Ashish Gupta, a venture capitalist.
2. Michael Jordan, a professor at University of California, Berkeley.

We populated Lucene using a randomly sampled set of nearly 4000 documents. The sampling was biased to fetch more pages related to the entity of our interest. For the

first experiment, our simulated Web contained about 40 documents which referred to the desired *Ashish Gupta*, and about 110 documents which referred to his namesakes. For the second experiment, on *Michael Jordan*, the corpus contained about 52 pages referring to the desired entity. We used the populated Lucene as our Search Engine. Since our simulated web is small as compared to the WWW, we reduced the *pagesize* of the result set to 5, i.e., Lucene will return 5 results at a time. Also, we allow a maximum of 40 pages to be fetched for each query. We took the top 10 positive and the top 10 negative documents – obtained by querying Lucene for the entity name – for training.

Figure 5.7 compares the performance (recall) of the three methods of generating keywords discussed in the Section 5.2.2.2. The graph measures the number of new relevant pages fetched against the total number of fetched pages.

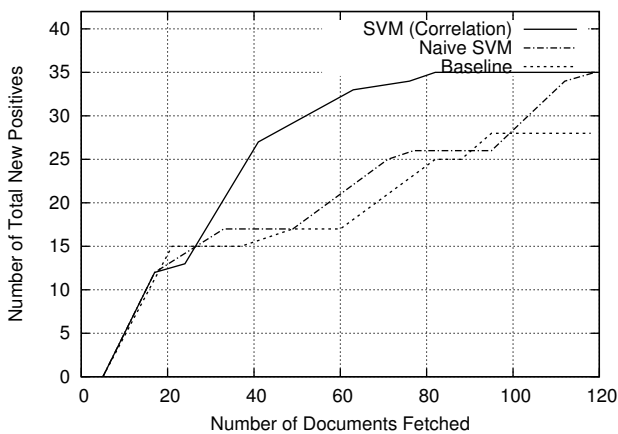


Figure 5.5: Ashish Gupta

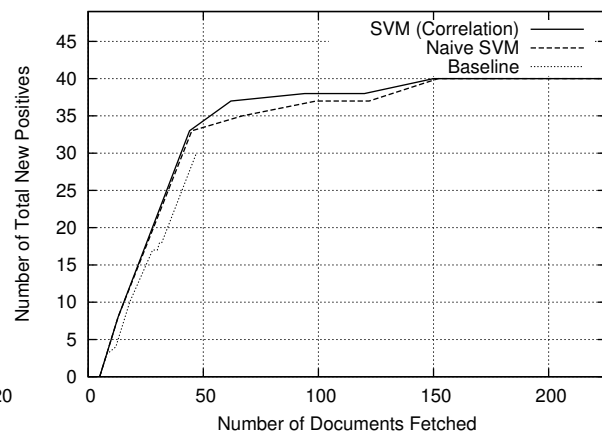


Figure 5.6: Michael Jordan

Figure 5.7: Recall vs Number of fetched pages

The following conclusions can be drawn from the graph.

- The SVM based methods could recall 35 out of 40 positive documents – a recall of 0.875 – for the first experiment, and 40 out of 52 positive documents – a recall of 0.769 – for the second one. The baseline could recall just 28 and 30 positive documents respectively for the two experiments.
- Naive SVM performs better than the baseline because of its discriminative power that boosts words which are important to distinguish the required *Ashish Gupta* from his namesakes. The baseline picks only the important words from the positive set. For example, the baseline picked the word ‘ee’ compared to the word

‘stanford’, which is more important for disambiguation, and is picked by the SVM. This causes the Naive SVM to have a higher coverage than that of the baseline.

- The coverage of the correlation capturing SVM method is the same as that of the Naive SVM, but the former method achieves it faster because of the correct ordering of the queries.
- The ranking used by Lucene is a naive one. The Search engines use a complex ranking method, e.g. Page-rank, compared to Lucene which uses a TFIDF based score to rank the documents. As we took the top fetched positives and negatives for training, we had to fetch more documents to obtain a good set of positives. On the Web, we expect our method to work even with fewer positives because good positives will be found early.

Chapter 6

Related Work

This chapter presents some of the work done in areas that relate to our problem of co-reference resolution.

6.1 Disambiguating People in Search

Guha et al., [GG04], work on the same line as ours. Their proposed method relies on the use of existing knowledge bases to help identify information relevant to the process of disambiguation.

The proposed method is based on the observation that web pages always contain some contextual information in the vicinity of an entity's reference. The idea is to represent each page by a sketch, where a sketch is a vector, the attributes of which are the values for the functions like first name, last name, profession, and location. The paper uses shallow mechanisms like lexical analysis to extract values for easily recognizable attributes like dates and emails, and relies on prior knowledge in the form of databases (like TAP and WordNet) to recognize attributes like organizations and places. A sketch can also be imagined to be a graph with all edges sharing one end point representing the entity under consideration. The other end points give the values of the attributes for the page. Since these attribute values might have been extracted using statistical methods, a probability can be associated with each edge. Weights can also be assigned to the edges depending on their importance in disambiguating people. E.g., given that two people, referred to by two different pages, work in the same company, the probability of their being referring to the same person will be more if the company is small. Hence, more weight can be assigned to the corresponding edge. In the naive case, the attributes are assumed to be independent, and all the pages are compared with the sketch of a known relevant page (or base page) by

taking the TFIDF similarity of their sketches. A second proposed method uses cardinality of the underlying domain of entities to measure the distance between two pages a and b according to Equation 6.1. The paper refers to this distance as **Inverse Network Size** or **INS**. The pages are then ranked in the decreasing order of their similarity with the base page. The result is a re-ordered list in which the relevant pages are ranked high.

$$INS_{a,b} = \prod_{i \in ai \in b}^{|S|} \sqrt{(1 - (1 - c_{ia}N_i/K))(1 - (1 - c_{ib}N_i/K))} \quad (6.1)$$

where,

INS = Inverse Network Size

c_{ix} = Uncertainty associated with page x and the dimension i ; depends on the statistical method used to extract attribute values

S = Set of dimensions

K = Cardinality of the underlying domain of entities

N_i = Number of entities having a non-zero value along the dimension i

The paper [GG04] claims a performance of over 93% in the case of people, but the assumption of existence of external databases – as assumed by the paper – is not always valid. Also, different pages use different descriptions, which only partially overlap. Textual similarity based features will fail to disambiguate pages in such a case. Ours is an attempt to utilize a combination of link and similarity features to successfully disambiguate even those pages that do not contain much information. Moreover, our features are quite general, and do not depend on any existing information base.

6.2 Entity-Based Cross-Document Co-referencing Using the Vector Space Model

Bagga et al., [BB98], describe a way to resolve co-references across documents. Given a number of documents as input, the paper proposes a method which uses the TFIDF similarity among the summaries of the documents to result in a set of clusters/chains such that all similar documents are chained together. We describe their proposed method in brief.

First each document is submitted to a system (called the CAMP system), which identifies all the co-reference chains in the document. Each co-reference chain corresponds to an entity. The entity chain of interest is selected, and the document is submitted to a summary extractor which extracts the summary for that entity. Each such summary represents an entity. The goal is to group together summaries that belong to the same entity. For this, each summary is represented by a vector of terms. Using the TFIDF similarity measure, similarity between pairs of documents is computed. If similarity comes out to be more than a predetermined threshold value, the summaries are assumed to represent the same entity. This way the documents are chained together so that all the documents in a chain correspond to the same entity; transitive relation is assumed while chaining the documents together. The paper [BB98] also gives an evaluation method to compute the goodness of chains formed. Rather than taking into account just the count of missing links for evaluation, a more intuitive method is to compute the recall and precision values for each entity in the chain separately. The total precision/recall of each chain is then equal to a weighted sum of the precision/recall values for all the entities within the chain. The paper refers to the method as the **B-cubed method**.

The paper claims an $F1$ of 84.6% on two datasets of 197 and 216 articles from the New York Times. But the assumption of availability of similar information in all the co-referring documents, fails when applied on the Web. Two documents may have entirely different contents, but may still refer to the same entity; our link features attack this part of the problem.

6.3 Anti-Aliasing on the Web

A problem related to co-reference resolution is the identification of authors of the text given only the text and the alias using which the text was posted. Novak et al., [NRT04], give a way to demystify an author's identity in blogs and other scenarios where the author of an article is hidden. There, the goal is to cluster together the aliases that refer to the same underlying physical entity. The paper makes use of the writing styles of the authors to disambiguate their identities. The method is briefly described here.

Given a number of aliases, and the text written by them, the idea is to represent each alias by a vector of features. Though a number of features such as words, misspellings,

punctuations, and emoticons, have been identified, the authors claim that just using the text of the postings gives a reasonable performance. Therefore, each alias is represented by a vector whose dimensions corresponds to words, and the value for each dimension gives the probability of the corresponding word appearing in the text for the alias. The paper also suggests a number of similarity measures such as the TFIDF similarity and the KL similarity. The KL similarity measures the number of extra bits required to encode the text of an alias using the optimum code for another alias. The authors found that the KL similarity gives better performance. After computing the similarity for each pair of aliases, for each alias all other aliases are ranked according to their similarity values. Using this rank value measure, a clustering algorithm is used to cluster together the aliases belonging to the same author. The algorithm merges the clusters such that the cohesion is minimized. This is done to ensure that all the aliases included in one cluster are ranked high in the list for each other.

One problem with this method is that it makes use of just the frequency of words to compute the similarity between two aliases. Since people discussing on the same topic tend to use same vocabulary during the discussion, this method tend to cluster together aliases belonging to different people – just because they engage heavily in the discussion about the same topic. Our problem is different from the anti-aliasing problem because we deal with web pages, and not all web pages that refer to a particular entity are written by the entity itself.

6.4 Extracting Query Modifications from Nonlinear SVMs

Gary Flake et al, [FGLG02], demonstrate the usefulness of nonlinear Support Vector Machines (SVMs) to generate effective query modifications for accumulation of homepages and conference pages. Theirs is the first attempt to utilize the distinguishing and generalization capabilities of SVMs to generate effective query modifications. Their goal is to generate a set of query modifications that satisfy a given precision criterion while maximizing the recall. We briefly describe their method here.

Their method is a 5-step process to automatically identify effective query modifications from the given training data. The first stage is to preprocess the textual content of an

HTML document to generate features having discriminatory power. During this step, the method performs dimensionality reduction by first eliminating those features that are too rare, and subsequently by collecting words which occur frequently in only one set, and scarcely in the other, if at all. Next, a nonlinear SVM is trained to classify the documents. Since an SVM is sensitive only to the support vectors, the method performs sensitivity analysis of the learnt SVM at each positive support vector to generate a set of candidate query modifications. The candidate modifications are found by identifying the largest d components of the sensitivity vector. These candidate modifications are then exhaustively analyzed to find the best set of modifications to maximize the recall while maintaining a desired precision on the training data ($size > 250$ documents, in their experiments). Figure 6.1 gives the procedure to find effective query modifications given an SVM and a working dataset.

```
for all  $i : \lambda_i > 0 \wedge y_i = 1$  do
  Calculate sensitivity,  $df/dx|_{x=x_i}$ 
  Find the largest  $d$  magnitude components,  $c$ , of sensitivity.
  for all  $2^d - 1$  combinations of  $c$  do
    Test query modification and note statistics
    if precision rate is above desired value and recall is greater than best found so far, then save query modification.
  end for
end for
return best found query modification.
```

Figure 6.1: Query Modifications using Non Linear SVMs.

After finding the first effective query modification, the true positives are removed from the dataset. The wrongly classified positives are then used along with the entire negative set to retrain the SVM, and generate more modifications. The search process terminates when no more useful modifications can be generated. The statically generated query modifications are then applied to modify new queries of the same type. The paper reports good performance on personal homepages and conference pages.

6.5 Disambiguating Web Appearances of People in a Social Network

McCallum et al., [BM05], propose a way to simultaneously disambiguate Web appearances of a group of people. Instead of solving single person disambiguation problem, they solve the problem of disambiguating web appearance of a set of related people simultaneously. The paper proposes two unsupervised frameworks to solve the problem. One is based on the link structure of the pages, and the other on recent multi-way distributional clustering. We briefly describe their approaches here.

Given a set of N related people to be disambiguated, the method is to initially fetch top K documents on each of the N queries to the web, and then cluster the documents to get the relevant pages. The **link based approach** forms a graph of the $N \times K$ pages, where the nodes represent the web pages, and two nodes are connected by an edge iff their hyperlinks share something in common – as defined in [BM05]. The largest connected component of this graph, which consists of pages from more than one (out of the posed N) queries, is taken to be the *central cluster*, C_0 . The central cluster, together with those that are similar to it by more than some threshold, is considered to consist of relevant pages. The method is based on the idea that the relevant pages that refer to different people are likely to interconnect, while the irrelevant pages that refer to different people will probably not connect to each other.

The second method employs **Agglomerative/Conglomerative Double Clustering (A/CDC)** which does not directly compute distances between clusters. The idea is to perform simultaneous top-down clustering of the set of words, and bottom-up clustering of the set of documents, conditioning one clustering system on the other, with the aim of maximizing their mutual information. The most interconnected cluster is selected to be the relevant cluster. The paper also proposes a hybrid approach which combines the link based and A/CDC clustering to improve the performance.

The paper claims a performance of over 80% F_1 on their dataset which consist of top 100 pages on 12 different queries of related person names. Our method largely differs from their method in that rather than disambiguating a group of people, we disambiguate individual entities – which is a much more difficult problem to solve.

6.6 Interactive Deduplication using Active Learning

Another related area is the deduplication of semi-structured records, i.e., to identify records that are actually the same but appear to be different due to difference in writing styles. Sarawagi et al., [SB02], study this problem in relation to the deduplication of the bibliographic entries, and suggest an efficient method for suppressing aliases using Active Learning. The paper proposes a number of features, such as similarity measures between the bibliographic fields, to determine if a pair of bibliographic entries are duplicates. The paper also suggests some optimizations, such as evaluating simpler predicates before the expensive ones, for efficient evaluation of the predicate learnt by the classifier.

The problem is different from ours in two ways. First, there is a structure that is inherent in the citations, while web pages are hardly structured. Second, even if two web pages are very similar, they might actually belong to two entirely different entities. These differences require a different approach to be able to correctly disambiguate the entities on the web. The idea of Active Learning, however, needs to be explored for our problem to reduce the amount of labeled training data.

Chapter 7

Conclusions & Future Work

In this thesis, we have explored the problem of disambiguating entities on the Web under various application scenarios, including re-ranking of search results, extraction of words for keyword suggestions, clustering of search results, background search for a person, and accumulation of set valued attributes. We observed that the problem is inherently complex and presents interesting challenges.

A central problem is identifying when two pages refer to the same physical entity. We designed a number of features derived from the words and links in a page, and the URLs for the corresponding two pages. We observed that the identified link features are very good indicators for disambiguation, but are found scarcely on the Web. We also observed that of the entire text of a document, the *nouns* and the *adjectives* are the main contributors to the process of entity identification. But, we also realize that POS tagging is a time consuming process, and hence cannot be applied in an online environment. We believe that identification of relevant parts of web pages using visual layout will not only improve the performance of our system, but will also make it efficient by eliminating unnecessary processing of the irrelevant text.

We also developed an entity clustering system to better solve the disambiguation problem. We observed that the transitivity relationship among the web pages – exploited by the clustering system – is indeed important for the process of disambiguation.

We explored use of linear SVMs in extracting distinguishing words to aid in keyword suggestion. We modeled a combined keyword suggestion and disambiguation system to obtain better precision in a search environment. We observed that the extracted keywords are almost always relevant given enough training data. But, we also realize the limited availability of the training data in an online environment. We noticed that the generated keywords maybe correlated, and capturing such correlation could aid in many applications.

We explored the applicability of an entity disambiguation system to applications requiring a high recall. We observed that obtaining high recall is an inherently complex problem, which is exacerbated by the inability to measure it on the Web.

In future, we would like to explore strong statistical techniques to capture correlation of query enhancing keywords. Detailed experiments to explore performance of our techniques – for set valued attribute accumulation and background search – also form part of our future work. We would also like to explore performance of our strategies in cases where the base query is less general than simply the name of a person. Another interesting area that can be explored is the use of Semi-supervised and Active learning approaches to obtain improved performance in cases of limited training data.

Bibliography

- [BB98] Amit Bagga and Breck Baldwin. Entity-Based Cross-Document Coreferencing Using the Vector Space Model. In *COLING-ACL*, 1998.
- [BBC02] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation Clustering. In *FOCS '02: Proceedings of the 43rd Symposium on Foundations of Computer Science*, page 238, Washington, DC, USA, 2002. IEEE Computer Society.
- [BGMZ97] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic clustering of the web. In *Proc. Sixth Int'l. World Wide Web Conference*, pages 391–404. WWW Consortium, 1997.
- [BM05] Ron Bekkerman and Andrew McCallum. Disambiguating web appearances of people in a social network. In *WWW*, pages 463–470, 2005.
- [BMC⁺03] Mikael Bilenko, Ray Mooney, William Cohen, Pradeep Ravikumar, and Steve Fienberg. Adaptive Name-Matching in Information Integration. *IEEE Computer Society*, 2003.
- [Bri92] Eric Brill. A simple rule-based part of speech tagger. In *Proceedings of the third conference on Applied natural language processing*, pages 152–155, Morristown, NJ, USA, 1992. Association for Computational Linguistics.
- [CRF03a] William Cohen, Pradeep Ravikumar, and Steve Fienberg. A Comparison of String Metrics for Matching Names and Records. In *Proceedings of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, pages 13–18, 2003.
- [CRF03b] William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings*

- of the IJCAI-2003 Workshop on Information Integration on the Web*, pages 73–78, Acapulco, Mexico, August 2003.
- [CYWM04] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Block-based web search. In *SIGIR '04: Proceedings of the 27th annual international conference on Research and development in information retrieval*, pages 456–463. ACM Press, 2004.
- [DCM03] Ji-Rong Wen Deng Cai, Shipeng Yu and Wei Ying Ma. Vips : a vision based page segmentation algorithm. Technical report, Microsoft Research, Microsoft Corporation, Nov 1, 2003.
- [ECD⁺04] Oren Etzioni, Michael J. Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Web-scale information extraction in knowitall: (preliminary results). In *WWW*, pages 100–110, 2004.
- [FGLG02] Gary Flake, Eric Glover, Steve Lawrence, and C. Lee Giles. Extracting query modifications from nonlinear SVMs. In *International World Wide Web Conference*, Honolulu, Hawaii, May 7–11 2002.
- [GG04] R. Guha and A. Garg. Disambiguating People in Search. 2004. <http://tap.stanford.edu/PeopleSearch.pdf>.
- [Guh04] R. Guha. Object Co-identification on the Semantic Web. 2004. <http://tap.stanford.edu/CoIdent.pdf>.
- [lib] Libsvm homepage. <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>.
- [LMP01] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001.
- [luc] Lucene text search engine library. <http://jakarta.apache.org/lucene>.

- [NRT04] Jasmine Novak, Prabhakar Raghavan, and Andrew Tomkins. Anti-Aliasing on the Web. In *Proceedings International WWW Conference*, New York, USA., 2004.
- [RKJZ04] Reiner Kraft and Jason Zien. Mining anchor text for query refinement. In *Proceedings International WWW Conference*, New York, USA., May 17-22 2004.
- [SB02] Sunita Sarawagi and Anuradha Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*. ACM, 2002.
- [Sea] Search engines used.
- <http://www.google.com>
 - <http://www.yahoo.com>
- .
- [Tur01] Peter D. Turney. Mining the web for synonyms:pmi-ir versus lsa on toefl. In *Proceedings of the Twelfth European Conference on Machine Learning (ECML-2001), Freiburg, Germany*, pages 491–502., 2001.
- [Wal02] Hanna M. Wallach. Efficient Training of Conditional Random Fields. Master’s thesis, School of Cognitive Science, Division of Informatics, University of Edinburgh, 2002.
- [WEK] Weka homepage. <http://www.cs.waikato.ac.nz/~ml/weka/>.
- [XC96] Jinxi Xu and W. Bruce Croft. Query expansion using local and global document analysis. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11, 1996.
- [YHCC02] H. Yu, J. Han, and K. C-C. Pebl: Positive example-based learning for web page classification using svm. In *Proc. ACM SIGKDD Int’l Conf. Knowledge*

Discovery in Databases (KDD02), ACM Press, New York, pages 239–248., 2002.

Acknowledgments

I take this opportunity to express my sincere gratitude for **Prof. Sunita Sarawagi** for her constant support and encouragement. Her excellent guidance has been instrumental in making this project work a success.

I would like to thank **Prof. Soumen Chakrabarti**, **Mr. Jeetendra Mirchandani** and **Mr. Sandesh D. Tawari** for their valuable help.

I also express my sincere gratitude for **Mr. V. G. Vinod Vydiswaran** and **S. Anusha** for their valuable suggestions and discussions.

I would also like to thank members of the **Data Mining Research Group** at KReSIT and **Diggers** — the Special Interest Group in Data Mining, for their valuable suggestions and helpful discussions.

Last but not the least, I would also like to thank my **family** and **friends**, who have been a source of encouragement and inspiration throughout the duration of the project.

Charu Tiwari

I. I. T. Bombay

July 13th, 2005

