COST-SENSITIVE WEB-BASED INFORMATION ACQUISITION FOR RECORD MATCHING

YEE FAN TAN

(B. Comp. (Hons.), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2011

Acknowledgements

First and foremost, I must thank my advisor, Min-Yen Kan, for all his advice, guidance, and patience in seeing me through my Ph.D. years. Without his generous and unwavering support, I would not have completed this Ph.D. thesis. He heads the Web Information Retrieval / Natural Language Processing Group (WING), and he is known among both undergraduate and graduate students as one of the most student-centric teachers. The tremendous amount of effort he put in to build relationships with his students, especially graduate students, including twice yearly WING dinners which he often pays out of his own pocket, is something I really appreciate during the years of my life as a Ph.D. candidate.

Acknowledgements also go to Dongwon Lee and Ergin Elmacioglu from The Pennsylvania State University: Dongwon Lee for suggesting collaboration opportunities as well as for providing me an annotated dataset for author name disambiguation; and Ergin Elmacioglu for being a collaborator in a few projects. I have benefited from the long-distance but fruitful discussions with them.

I would also like to thank my colleagues, both past and present, in WING as well as other members of the Computational Linguistics Laboratory. These people have provided general but insightful discussions, as well as the mutual support. Heartfelt thanks goes out to Hang Cui, Long Qiu, Hendra Setiawan, Kazunari Sugiyama, Jin Zhao, Ziheng Lin, Jesse Prabawa Gozali, Jun Ping Ng, Aobo Wang, Cong Duy Vu Hoang, Emma Thuy Dung Nguyen, Minh Thang Luong, Yee Seng Chan, Wei Lu, Shanheng Zhao, Zhi Zhong, and Daniel Dahlmeier.

Although not directly related to this thesis, I would like to thank Prof. Tat-Seng

Chua for opportunities to work on projects together with members of the Lab for Media Search (LMS). Parts of these projects had served as inspiration for my initial work in this thesis. Particular thanks go to Shi-Yong Neo and Victor Goh, who were great collaborators in these projects. These two people subsequently became founding members of KAI Square Pte. Ltd., and I am very grateful for their persistent but sincere invitations for me to join the company, for which I eventually accepted. Hellos also goes out to the following members of LMS: Ming Zhao, Mstislav Maslennikov, Huaxin Xu, Gang Wang, Yantao Zheng, Zhaoyan Ming, Renxu Sun, and Dave Kor.

Finally, my appreciation also goes out to everybody out there who have supported me in one way or another in my pursuit of a Ph.D.. These include my family members as well as my friends who are not listed above.

Portions of the work done in this thesis was partially supported by a National Research Foundation grant “Interactive Media Search” (#R-252-000-325-279).

Contents

Abstract

In many record matching problems, the input data is either ambiguous or incomplete, making the record matching task difficult. However, for some domains, evidence for record matching decisions are readily available in large quantities on the Web. These resources may be retrieved by making queries to a search engine, making the Web a valuable resource. On the other hand, Web resources are slow to acquire compared to data that is already available in the input. Also, some Web resources must be acquired before others. Hence, it is necessary to acquire Web resources selectively and judiciously, while satisfying the acquisition dependencies between these resources.

This thesis has two major goals:

1. To establish that acquisition of web based resources can benefit the task performance of record matching tasks, and

2. To propose an algorithm for selective acquisition of web based resources for record matching tasks. It should balance acquisition costs and acquisition benefits, while taking acquisition dependencies between resources into account.

This thesis has two major parts corresponding to the two goals. In the first part, I propose methods for using information from the Web for three different record matching problems, namely, author name disambiguation, linkage of short forms to long forms, and web people search. Thus, I establish that acquiring web based resources can improve record matching tasks.

In the second and larger part, I propose approaches for selective acquisition of web based resources for record matching tasks, with the aim of balancing acquisition costs

and acquisition benefits. These approaches start from the more task-specific and move towards the more general and principled. I first propose a way for adaptively combining two methods for record matching, followed by a cost-sensitive attribute value acquisition algorithm for support vector machines. This work culminates in a framework for performing cost-sensitive resource acquisition problems with hierarchical dependencies, which is the main contribution in this thesis. This graphical framework is versatile and can apply to a large variety of problems. In the context of this framework, I propose an effective resource acquisition algorithm for record matching problems, taking particular characteristics of such problems into account. Finally, I proposed two benefit functions for use in my framework, corresponding to two different evaluation measures.

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

In this thesis, I focus on using web based resources for solving record matching problems. This thesis has two major goals:

- To establish that acquisition of web based resources can benefit the task performance of record matching tasks, and

- To propose an algorithm for selective acquisition of web based resources for record matching tasks. It should balance acquisition costs and acquisition benefits, while taking acquisition dependencies between resources into account.

Among the two goals, the first goal is smaller and serves to support the premise in the second goal. The second goal is larger and is the main focus of this thesis.

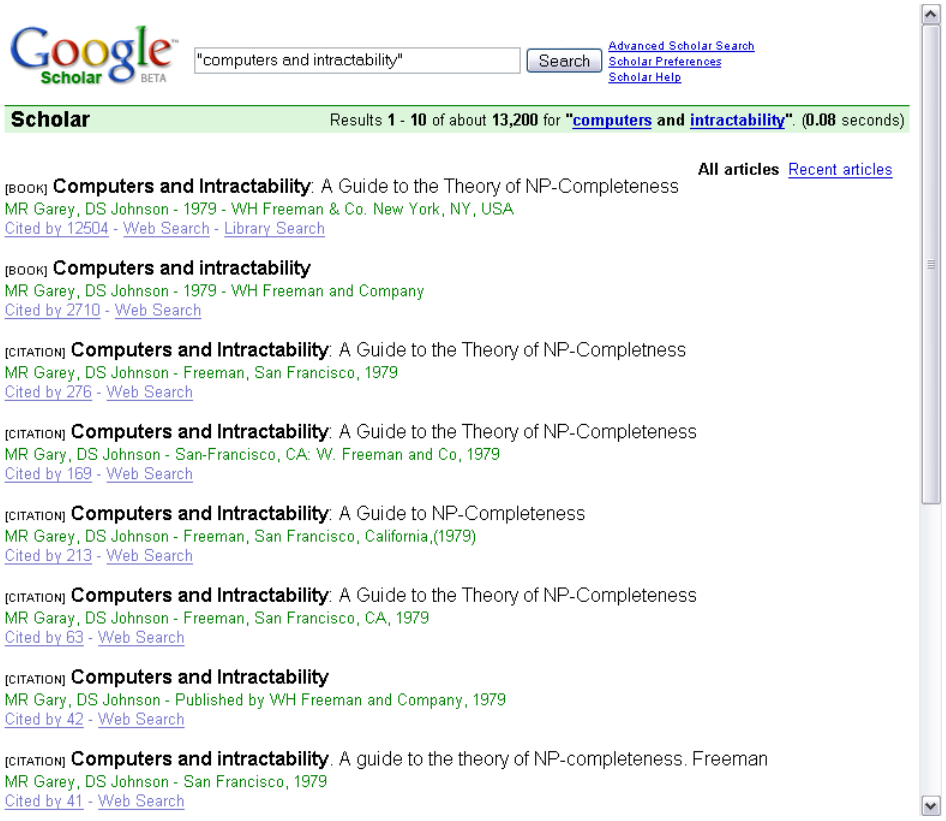
I make several contributions towards these two goals. These contributions include methods for utilizing information from the Web, as well as methods for controlling the amount of expensive web resource acquisitions required to solve record matching problems. The main contribution of this thesis is a framework for performing cost-sensitive resource acquisition problems with hierarchical dependencies, which is versatile and can apply to a large variety of problems.

Before detailing the contributions, I first provide the background for this thesis.

1.2 Background

In many domains, data can be inherently noisy. Take for instance, a large bibliographic database of scientific publication records such as ACM Portal or CiteSeer. References to a particular publication may be extracted from different sources, resulting in duplicate metadata records being stored in the database. Figure 1.1a shows duplicate metadata records returned by Google Scholar, with minor variations between these records; ideally these records should be merged and only a single result should be returned. Additionally, different authors can share the same name. This is illustrated in Figure 1.1b, where bibliographic citations belonging to two different people who bear the same name "Hui Yang" are merged into a single list, while these should be separated into two separate lists. An entity such as author or publication venue may also be represented differently, such as "M.-Y. Kan" for "Min-Yen Kan" and "JCDL" for "Joint Conference on Digital Libraries". Such problems cause difficulty in searching for relevant information, and may result in overaggregation or underaggregation of data, causing biased counts or credit misattribution. In bibliometrics of scientific articles and publication venues, these problems can lead to an inflated or deflated citation counts, leading to inaccurate values for measures such as impact factor [Garfield, 1994] or h-index [Hirsch, 2005].

Such problems are not limited to bibliographic citation records but also occur in a variety of other domains. As early as the 1940s, the matching of records and fields across large databases has been recognized as a research issue in the analysis of census data [Marshall, 1947] as well as medical records [Dunn, 1946]. The commercial and government sectors spend a large amount of time and energy to improve the integrity and quality of their expanding data records. Two examples: a high-tech equipment manufacturer saved US$6 million per year by removing redundant customer records used in customer mailings [Bell and Dravis, 2006]; and the U.K. Ministry of Defence saved over US$25 million over four years by solving key problems with their inventory and logistics [Wheatley, 2004]. Much of the record matching issues stem from reasons such as data entry errors, errors in automated extraction systems, missing or incomplete information, database schema differences, variations in names given to the same entity,



Google Scholar BETA "computers and intractability" Search Advanced Scholar Search Scholar Preferences Scholar Help

Scholar Results **1** - **10** of about **13,200** for **"computers and intractability"**. (**0.08** seconds)

All articles Recent articles

[BOOK] **Computers and Intractability**: A Guide to the Theory of NP-Completeness
MR Garey, DS Johnson - 1979 - WH Freeman & Co. New York, NY, USA
Cited by 12504 - Web Search - Library Search

[BOOK] **Computers and intractability**
MR Garey, DS Johnson - 1979 - WH Freeman and Company
Cited by 2710 - Web Search

[CITATION] **Computers and Intractability**: A Guide to the Theory of NP-Completness
MR Garey, DS Johnson - Freeman, San Francisco, 1979
Cited by 276 - Web Search

[CITATION] **Computers and Intractability**: A Guide to the Theory of NP-Completeness
MR Gary, DS Johnson - San-Francisco, CA: W. Freeman and Co, 1979
Cited by 169 - Web Search

[CITATION] **Computers and Intractability**: A Guide to NP-Completeness
MR Garey, DS Johnson - Freeman, San Francisco, California,(1979)
Cited by 213 - Web Search

[CITATION] **Computers and Intractability**: A Guide to the Theory of NP-Completeness
MR Garay, DS Johnson - Freeman, San Francisco, CA, 1979
Cited by 63 - Web Search

[CITATION] **Computers and Intractability**
MR Gary, DS Johnson - Published by WH Freeman and Company, 1979
Cited by 42 - Web Search

[CITATION] **Computers and intractability**. A guide to the theory of NP-completeness. Freeman
MR Garey, DS Johnson - San Francisco, 1979
Cited by 41 - Web Search

(a) Searching for "computers and intractability" on Google Scholar.

		2005
26	EE	Hui Yang, Jamie Callan: Near-duplicate detection for eRulemaking. DG.O 2005: 78-86
25	EE	Chonghui Song, Shucheng Yang, Hui Yang, Huaguang Zhang, Tianyou Chai: A Dual-Mode Fuzzy Model Predictive Control Scheme for Unknown Continuous Nonlinear System. FSKD (1) 2005: 876-881
24	EE	Hui Yang, Chonghui Song, Chunyan Yang, Tianyou Chai: Component Content Soft-Sensor Based on Adaptive Fuzzy System in Rare-Earth Countercurrent Extraction Process. FSKD (2) 2005: 891-896
23	EE	Hui Yang, Minjie Zhang: Ontology-Based Resource Descriptions for Distributed Information Sources. ICITA (1) 2005: 143-148
22	EE	Hui Yang, Chunyan Yang, Chonghui Song, Tianyou Chai: Intelligent Optimal Control in Rare-Earth Countercurrent Extraction Process *via* Soft-Sensor. ICNC (2) 2005: 214-223
21	EE	Hui Yang, Srinivasan Parthasarathy, Sameep Mehta: Mining Spatial Object Associations for Scientific Data. IJCAI 2005: 902-907
20	EE	Hui Yang, Srinivasan Parthasarathy, Sameep Mehta: A generalized framework for mining spatio-temporal patterns in scientific data. KDD 2005: 716-721
19		Sameep Mehta, Steve Barr, Tat-Sang Choy, Hui Yang, Srinivasan Parthasarathy, Raghu Machiraju, John Wilkins: Dynamic Classification of Defect Structures in Molecular Dynamics Simulation Data. SDM 2005
18	EE	Sameep Mehta, Srinivasan Parthasarathy, Hui Yang: Toward Unsupervised Correlation Preserving Discretization. IEEE Trans. Knowl. Data Eng. 17(9): 1174-1185 (2005)
		2004
17		Keith Marsolo, Hui Yang, Srinivasan Parthasarathy, Sameep Mehta: Discovering Spatial Relationships Between Approximately Equivalent Patterns in Contact Maps. BIOKDD 2004: 62-71
16	EE	Sameep Mehta, Srinivasan Parthasarathy, Hui Yang: Correlation Preserving Discretization. ICDM 2004: 479-482
15	EE	Hui Yang, Minjie Zhang, Zhongzhi Shi: Association-Rule Based Information Source Selection. PRICAI 2004: 563-574
14	EE	Hui Yang, Tat-Seng Chua: Effectiveness of web page classification on finding list answers. SIGIR 2004: 522-523
13	EE	Hui Yang, Tat-Seng Chua: FADA: find all distinct answers. WWW (Alternate Track Papers & Posters) 2004: 304-305
12		Hui Yang, Minjie Zhang: Hierarchical Classification for Multiple, Distributed Web Databases. I. J. Comput. Appl. 11(2): 119-131 (2004)

(b) Publications of two different Hui Yangs mixed together in a single list on DBLP.

Figure 1.1: Record matching problems.

and different entities sharing the same name.

Owing to simultaneous recognition of such matching problems across different disciplines, such problems have been given a large number of names. These include *record linkage*, *duplicate record detection*, *name disambiguation*, *data cleaning*, *data cleansing*, *identity uncertainty*, *citation matching*, *merge-purge*, *reference reconciliation*, *entity resolution*, *object matching*, *approximate text join* and *authority control*. In all these problems, depending on the application, the notion of "matching" may apply to either whole records or only to particular fields such as person names. In this thesis, I shall refer to both of these problems simply as *record matching*. They come in two main flavours:

- **Record linkage.** The input is two list of records, A and B. The aim is to determine, for each pair of records $(a, b) \in A \times B$, whether records a and b are a match.

- **Clustering.** The input is a list of records, L. The aim is to determine, for each pair of records $(a, b) \in L \times L$, whether records a and b are a match.

As we can see, a large number of other problems can be cast into record matching problems. This thesis examines two problems in particular:

- **Linkage of short forms to long forms.** Given a list of short forms (*e.g.*, WWW) and a list of long forms (*e.g.*, World Wide Web), which short forms correspond to which long forms? This can be seen as a record linkage problem.

- **Author name disambiguation.** Given an ambiguous author name and a list of bibliographic citations containing the ambiguous name, which citations refer to the same author? This is illustrated in Figure 1.1b and can be seen as a clustering problem.

However, for a number of datasets, record matching can be difficult because the dataset itself lacks the required context, giving insufficient information to perform the matching [Lee et al., 2004; Mani and Sundaram, 2007]. As an example, from the information in Figure 1.1b alone, it may be easy to tell that records #13 and #14 refer

to the same Hui Yang, but it is anything but obvious that record #26 also refer to the same person. This is because this Hui Yang obtained her Masters degree at the National University of Singapore while working on question answering and retrieval, and subsequently became a Ph.D. student at Carnegie Mellon University and changed her topic to near duplicate detection. Unfortunately, this fact is never reflected in the input data, unless it is supplied to the record matching algorithm as a piece of external information.

In many problem settings, it is common to use external resources as part of the solution. External resources are auxiliary information that is not part of the input data. These include ontologies from which relationships between objects can be extracted, and corpora from which statistical information can be obtained. Often, external resources often contain knowledge that is not found in the input data, or knowledge that may be difficult to extract from the input data. In natural language processing, the famous WordNet lexical resource [Fellbaum, 1998] is employed in a variety of tasks such as word similarity (*e.g.*, [Pedersen et al., 2004]) and word sense disambiguation (*e.g.*, [Li et al., 1995]). In performing record linkage of two lists of records, auxiliary information that are external to the input lists can be used to aid the linkage process (*e.g.*, [Winkler, 2006]).

In this thesis, I focus on acquiring information from a particular kind of external resource: the Web.

1.2.1 Web Resources for Record Matching and the Acquisition Bottleneck

In these few decades, the Web has grown explosively. People and organizations have been putting up a lot of information on all kinds of topics on the Web. Therefore, different parts of the web have become different knowledge resources, serving a very large variety of information needs. To aid users to find information on the Web, search engines were invented. Search engines such as Google, Yahoo!, and Bing crawl all over the Web and index the web pages they found. Hence, a search engine has become an essential gateway to Web information, and a Web search effectively means querying a

search engine.

The enormous size of the Web results in huge indices for search engines. In 2005, Gulli and Signorini have estimated that the number of publicly indexable web pages is more than 11.5 billion [Gulli and Signorini, 2005]. In 2006, de Kunder estimated the number of distinct web pages indexed by the Google, Yahoo!, Bing, and Ask.com search engines to be at least 14 billion [de Kunder, 2006]. As of 2010 April 7, de Kunder's WorldWideWebSize.com website gave the number to be at least 20.51 billion. Given the size of the Web, not only search results make useful information resources, the hit counts from search queries also turns out to be extremely useful for compiling statistics that approximates trends and other human usage of terms. In other words, the Web, either accessed through a search engine or otherwise, can be seen as a very large corpus or data repository waiting to be exploited.

Using the Web as an external information resource is not a new idea, and has been employed by researchers in various fields. Question answering systems have used Web search engines to perform subtasks such as query expansion (*e.g.*, [Sun et al., 2006]), or mine targeted subsets of the Web such as Wikipedia to find answers to questions (*e.g.*, [Buscaldi and Rosso, 2006]). Linguistic studies sometimes treat the Web as a huge text corpus, using search engines to query and retrieve documents, because researchers may find standard corpora such as the British National Corpus inadequate for their needs (*e.g.*, [Fazly et al., 2005]). There are even proposals for building large sized non-English offline corpora through a search engine such as Google (*e.g.*, [Sharoff, 2006]). These works have demonstrated that acquiring additional information through a search engine resulted in increased task performance effectiveness.

To illustrate the application of web resources for record matching problems, consider again the case of disambiguating publication records of Hui Yang as shown in Figure 1.1b. Despite her change of affiliation, resulting in a change in her collaborators and research topics, all her past and present publications are listed in her publication web page hosted at Carnegie Mellon University. Her publication page indicates that she authored all of publications #13, #14, and #26 in the DBLP records. This example

illustrates that information that are external to the input lists can help the linkage process, in this case, the external information came from web resources. Similar to Hui Yang, many other individual researchers or research organizations have put up their publication lists on the Web as well. Therefore, a possible solution for disambiguating the publication records of an ambiguous author name might be to search for the publication web pages of the various individuals having ambiguous names, and then match the publication titles of the bibliographic records against those in the web pages.

There are many ways to obtain features for record matching from the Web. One can call a search engine and examine its returned snippets, or to crawl and download web pages. However, obtaining search engine results and downloading web page are time consuming processes. Usually the web pages provide more comprehensive features, but downloading them take a lot more time relative to merely querying the search engine. Therefore, a solution that downloads a large number of web pages may give very good performance but is highly impractical. Suppose we are matching two named entities. What kinds of web resources do we acquire? If we query a search engine for one named entity and obtain its results, is it sufficient for the matching task, or do we need to acquire more information? If so, do we download the web pages at the URLs of these results, or do we query the search engine for the other named entity? Furthermore, certain search engines perform rate limiting and restrict the number of queries one may make daily. For example, Yahoo! Search has a daily quota of $5{,}000$ queries. However, for record matching problems, even for two small lists of 100 items each can generate $10{,}000$ pairwise queries, which requires two days. While web resources are useful for record matching tasks, it also poses an acquisition bottleneck. As such, it is necessary to acquire web resources in a selective manner.

A way to limit the cost of resource acquisitions is by *blocking*, which filters out obvious mismatched record pairs before performing matching on the remainder (*e.g.*, [Winkler, 2006]). Successful applications of blocking achieve a reasonable task performance with a greatly reduced number of pairwise matchings [Goiser and Christen, 2006], which can reduce resource acquisitions significantly. However, blocking techniques are

often ad-hoc and domain-specific, and can be difficult to define properly; poor blocking decisions can degrade performance [Goiser and Christen, 2006].

In order for record matching solutions involving web based resources to be scale up to large inputs, selective acquisition of resources must be performed. In other words, the benefits of resource acquisitions must be balanced against the costs of acquiring them. Therefore, in this thesis, in addition to demonstrating the usefulness of web based resources, I also examine the problem of acquiring only a subset of resources for achieving a balance between costs and benefits.

When we consider cost-sensitive selective acquisition of web based resources, we need to take into account the acquisition dependencies between different resources. For example, it is possible to query a search engine, and then download the web pages at the URLs given in the search engine results. Obviously, if we consider the search engine results as a resource and the corresponding web pages as another resource, then the former must be acquired before the latter can be acquired. Such dependencies can be captured in a *resource dependency graph* which I describe in Chapter 6.

1.3 Contributions

In this thesis, I make two main contributions, corresponding to the two goals in this thesis.

- **Using web resources for record matching.** I demonstrate that acquiring web based resources can improve record matching tasks. This is done by means of concrete examples on three different problems, namely, author name disambiguation, linkage of short forms to long forms, and web people search. For author name disambiguation, I show that a new measure of my contribution, inverse host frequency, when used alone is effective in solving the mixed citation problem, and it can be combined together with a coauthor linkage method to achieve an even better disambiguation performance. For linkage of short forms to long forms, I evaluate a number of methods and show that a count-based method is the

most effective for this linkage problem. For web people search, I present a brief summary of our PSNUS system that participated in the Web People Search Task in SemEval 2007.

- **A framework for hierarchical cost-sensitive web resource acquisition.** For the main contribution of this thesis, I present a general framework for performing cost-sensitive resource acquisition problems with hierarchical dependences. This framework takes in as input a resource dependency graph, an acquisition cost function (for determining the cost of making particular resource acquisitions), and a benefit function (for determining the benefit value of making particular resource acquisitions), and selects resources to acquire to maximize the benefit of acquisitions while minimizing the acquisition costs. This framework abstracts away from any specific problem instance, and can be applied to many problems involving selective resource acquisitions. For record matching problems, I note that their resource dependency graphs have unique characteristics which poses particular challenges for constructing an algorithm for making resource acquisitions. In this regard, I propose a resource acquisition algorithm that overcomes these challenges in record matching problems. To complete a resource acquisition algorithm for record matching problems, I need to have a benefit function. While an acquisition cost function can be easily obtained or engineered, coming up with a benefit function can be a challenge. Here, I propose two benefit functions, for two different evaluation measures for the test instances: total misclassification cost, and F_1 measure.

In addition, there are other contributions in this thesis.

- **A framework for adaptively combining two methods for record matching.** I present an adaptive combination framework for combining two methods, such that the combined method has the better aspect of each method. I apply my adaptive combination framework to the problem of linking short forms to long forms, combining the count-based method with a query probing method to re-

duce the number of search engine queries required while maintaining linkage performance.

- **Cost-sensitive attribute value acquisition for support vector machines.** In this work, I consider supervised classification of test instances which contain missing attribute values which may individually be acquired at some cost, where the classifier is the support vector machine. Therefore, the aim is to achieve balance between acquisition costs and misclassification costs. One important feature of my proposed acquisition algorithm is that it can be applied to any kernel. Another feature is that it can compute the expected classification certainty and misclassification cost of a test instance, before and after acquiring an arbitrarily given subset of missing attribute values. The latter key feature enables me to construct benefit functions for the hierarchical resource acquisition framework, by breaking down the problem of finding the benefit of acquiring vertices into the related problem of finding the benefit of acquiring missing attribute values in classification problems.

Portions of the work done in this thesis appeared in the following publications: [Tan et al., 2006], [Elmacioglu et al., 2007b], [Kan and Tan, 2008], [Tan et al., 2008], and [Tan and Kan, 2010]. The work done in some of these papers have been expanded in this thesis.

1.4 Organization

Figure 1.2 shows an overview roadmap that gives the organization of the work done in this thesis. The bold items together with the dashed arrows indicates the progression of my work in selective resource acquisition, as I develop algorithms or methods from the more specific to the more general, cumulating in the hierarchical cost-sensitive resource acquisition framework for record matching problems (shaded), which is the main contribution of this thesis. Each solid arrow from one part to another part indicate that elements of the former part has been applied in the latter part.

Broadly speaking, this thesis can be divided into two main parts. The first and

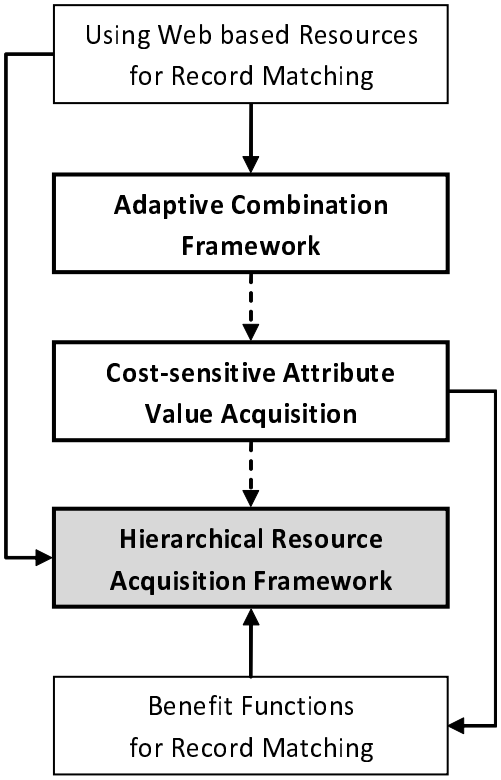


Figure 1.2: Overview roadmap of this thesis.

smaller part consists of Chapter 3, which fulfils the first goal of establishing that acquisition of web based resources can benefit the task performance of record matching tasks. The second and larger part consists of Chapters 4, 5, 6, and 7, which fulfils the second and main goal of proposing algorithms for selective acquisition of web based resources for record matching tasks.

In more detail, the remainder of this thesis is organized as follows.

In Chapter 2, I give an overview of the related work in record matching. I divide the related work into those not using web based resources and those using web based resources, where the latter group is more pertinent to my work in this thesis.

In Chapter 3, I describe my contributions towards using web based resources for record matching applications in three problems: author name disambiguation, linkage of short forms to long forms, and web people search.

In Chapter 4, I present my framework for adaptively combining two methods for record matching, and I apply my framework to the problem of linking short forms to long forms.

In Chapter 5, I present a cost-sensitive attribute value acquisition algorithm for support vector machines, driven by the expected decrease in misclassification cost for acquiring attribute values.

In Chapter 6, I present my framework for hierarchical cost-sensitive web resource acquisition, applicable for many kinds of problems involving selective resource acquisitions. Within this framework, I propose an acquisition algorithm for record matching problems that overcomes the unique challenges posed by their resource dependency graphs.

In Chapter 7, I construct two benefit functions for record matching problems, for two different evaluation measures, thereby giving a complete description of a resource acquisition algorithm for such problems.

In Chapter 8, I conclude my thesis. I highlight the contributions in this thesis, and outline some possible directions for future work.

Chapter 2

Related Work

2.1 Introduction

Record matching is a widely studied problem and is recognized as a research issue as early as the 1940s, due to the analysis of census data [Marshall, 1947] and medical records [Dunn, 1946]. The field of record linkage is often said to be first modernized by Newcombe *et al.* [Newcombe et al., 1959]. Later, Fellegi and Sunter [Fellegi and Sunter, 1969] proposed a mathematical model consisting of the decision regions of non-match, possible match, and definite match, with the possible match region further investigated by clerical review. This model is still a basis for much of the record linkage work today.

There are probably thousands of publications dealing with record matching problems, and several papers have surveyed approaches to record matching and its variant problems (*e.g.*, [Gu et al., 2003], [Winkler, 2006], [Elmagarmid et al., 2007], [Kan and Tan, 2008], and [Smalheiser and Torvik, 2009]). In this chapter, I briefly review the related work in record matching by surveying a representative sample of these works.

For the purpose of my thesis, I have divided the related work into those that acquire information from the Web and those that do not. In this chapter, I first survey non Web-based algorithms in Section 2.2, followed Web-based algorithms in Section 2.3. While the non Web-based work serve as a good overview to record matching algorithms, the Web-based work are much more pertinent to my thesis.

Here, methods that use external resources such as ontologies and databases are con-

sidered to be non Web-based if they reside on the same machine where computations are taking place, while those that access resources that are located remotely are considered Web-based. In particular, hidden web databases [Kautz et al., 1997] residing on remote machines are considered Web-based. This distinction is because there is a cost for acquiring remote resources, while resources that are available locally are already acquired (or have their acquisition cost already paid up).

2.2 Non Web-based Record Matching Algorithms

I first survey non Web-based record matching algorithms. Non Web-based algorithms have been studied for much longer than Web-based algorithms, and can be described at multiple levels. On one level, we can simply consider how similar two strings are. At another level, we can consider the different types of record fields and how to combine the various field similarity metrics into a single unified metric between two records. At the highest level, we can consider the matching of a whole set of records, taking into consideration the interactions between record fields across different records.

2.2.1 Uninformed String Matching

In its most basic form, record matching can be simplified as string matching, which decides whether a pair of observed strings refer to the same underlying item. In such cases, we use pairwise similarity between the strings to calculate whether they are coreferential. String similarity measures can be classified as either sequence- or set-based, depending on whether ordering information is used or not.

Sequence-based similarity can be generally cast as edit distances [Levenshtein, 1966] and their generalizations, such as normalized edit distance [Marzal and Vidal, 1993], Smith-Waterman distance [Smith and Waterman, 1981], and Needleman-Wunsch distance [Needleman and Wunsch, 1970]. Typically, the transformation cost is measured by summing the cost of simple incremental operations such as insertion, deletion and substitution.

Set-based similarity considers the two strings as independent sets (or multisets) of tokens S and T, using a bag-of-words model. There are a number of similarity measures that make use of the intersections and unions of S and T. These include matching coefficient ($|S \cap T|$), Jaccard coefficient ($\frac{|S \cap T|}{|S \cup T|}$), Dice coefficient ($\frac{2|S \cap T|}{|S|+|T|}$), and overlap coefficient ($\frac{|S \cap T|}{\min(|S|,|T|)}$) [Manning et al., 2008]. Alternatively, asymmetric measures such as *degree of similarity* ($\frac{|S \cap T|}{|S|}$) [Low et al., 2001] may be more appropriate when one string is more important to match than the other. Finally, one can borrow from information retrieval and construct TF or TF-IDF vectors out of S and T, and then compute the cosine of the angle between these vectors [Manning et al., 2008].

Hybrids of both set- and sequence-based measures are often used. For example, when the string is a series of words, a sequence-based measure may employed for individual tokens, but the string as a whole may be modeled as a set of tokens [Low et al., 2001; Cohen et al., 2003].

2.2.2 Informed Similarity and Record Matching

Database records themselves contain a wide variety of data. For example, bibliographic metadata records contain personal names, URLs, controlled subject headers, publication names, and years. Each of these fields may have their own notions for what is considered acceptable variation ("Liz" = "Elizabeth"; "Comm. of the ACM" = "CACM"; 1996 $\neq$ 1997). Knowing what type of data exists in a field can inform us of what constitutes similarity and duplication. As such, string similarity measures are usually weighted differently per field.

Certain data types have been studied in depth. In fact, the need to consolidate records of names and addresses pioneered research to find reliable rules and weights for record matching. In set-based similarity, tokens may be weighted with respect to their (log) frequency, as is done in information retrieval models. In sequence-based edit operations, a spectrum of weighting schemes have been used to capture regularities in the data, basically by varying the edit cost based on the position and input. For example, in genomic data, sequences often match even when a whole substring is inserted

or deleted; the same is true when matching abbreviations to their full forms. In census data, person names are typically short and their initial letters rarely incorrect, making the Jaro [Jaro, 1989] and Jaro-Winkler [Winkler and Thibaudeau, 1991] measures suitable for matching person names. Person name matching is a widely studied topic (see *e.g.*, [Cohen et al., 2003], [Snae, 2007]).

Such models need to set parameters such as the cost for each type of edit operation in a principled way. Fortunately, data-driven methods have emerged to learn optimal weights from training data (*e.g.*, [Winkler and Thibaudeau, 1991], [Bilenko and Mooney, 2003]).

2.2.3 Iterative and Graphical Formalisms for Record Matching

Record matching can be performed iteratively. For example, the Fellegi-Sunter model provides a possible match region. We can selectively choose some of the record pairs in this region for clerical review and obtain their true match/mismatch classifications. These classifications can then be used to rebuild the record matching model, and the process repeated in an active learning fashion until convergence [Larsen and Rubin, 2001]. Alternatively, by making use of relationships between record fields, the act of consolidating the data after an iteration can cascade and provide evidence for matching on other fields in later iterations. This incremental approach can resolve duplicates when true matching records do not exceed a global similarity threshold before individual fields in the records are merged [Bhattacharya and Getoor, 2004].

In recent years, graphical formalisms are becoming popular for record matching. Typically, fields or whole records are viewed as nodes in a graph with edges connecting similar nodes, with similarity values assigned to edges, allowing global information to be incorporated in the disambiguation process. Graphical formalisms often lead to the latter type of iterative record matching. For example, dependency graphs [Dong et al., 2005] or conditional random fields [Wellner et al., 2004] nicely model incremental record matching, enabling the propagation of contextual similarity.

A common manifestation of graphical formalisms in disambiguation tasks is in the

form of social networks, such as collaboration networks. Social network analysis methods such as centrality and betweeness can be applied. For example, one may identify cliques of high weights or quasi-cliques as matching nodes [Feitelson, 2004; Pei et al., 2005]. In author disambiguation, we may be able to attribute two papers to the same "Hui Yang" when the co-author lists do not have common names, but share names with a third paper – the two nodes are connected by a path through a third node [Reuther, 2006]. Another work uses network cuts and random walks in the collaboration network of actors to disambiguate names in the Internet Movie Database [Malin et al., 2005]. A similar idea of using connection strengths between nodes, but exploited in a different way, is explored in [Kalashnikov and Mehrotra, 2006]. In the ArnetMiner system [Tang et al., 2008], which aims to extract and mine academic social networks, various modes of connections such as coauthors, publication venues, and citations were used to disambiguate names in publication metadata extracted from different sources.

Graphical formalisms in the guise of generative probabilistic models have also been suggested. In the author disambiguation problem, we can view authors are members of collaborative groups. This model first picks out collaborative groups and then assigns authors within these groups to generate references. We can then run this model in the opposite direction to infer which collaborative group (thus which disambiguated author) is responsible for a particular work [Bhattacharya and Getoor, 2006]. Such graphical models have outperformed methods using pairwise comparisons in accuracy but have yet to demonstrate efficiency on large datasets.

2.2.4 Reducing Complexity by Blocking

Record matching tasks are often performed using pairwise comparisons, which can be an computationally expensive task. However, when the number of records, n, is large, having to perform $O(n^2)$ pairwise comparisons simply takes too much time, For example, as of 2005, the number of independent articles and monographs in computer science research alone is estimated to exceeded 2.2 million [Petricek et al., 2005]. If each pairwise comparison takes 0.001 seconds, performing all $\binom{n}{2}$ pairwise comparisons will

require a few decades to complete. Therefore, the number of computationally expensive pairwise comparisons must be cut down. Observations show that the ratio of true record matches to non-matches is very low. Also, many pairs of records are obviously non-matches, *e.g.*, two person names "J. Brown" and "D. Lee" are very unlikely to refer to the same individual. Therefore, if we create a block for each encountered token in the input names, and insert each name into the blocks corresponding to its tokens. Since "J. Brown" and "D. Lee" share no common tokens, no blocks will contain both names. Such a blocking algorithm is computationally cheap, and the more computationally expensive similarity measures can then be confined to run only for records within each block (see e.g., [Winkler, 2006]). Successful applications of blocking can reduce the number of required matchings required by a few orders of magnitude while filtering out only a very small proportion of the true matches [Goiser and Christen, 2006].

More generally, blocking is performed by first defining a blocking key [Gu et al., 2003] for each record, which is typically a string. This key can be the value of a field as it is, or the values of multiple fields (or even the entire record) concatenated together. The value can also be preprocessed, such as by applying the Soundex transformation to names, or selecting the first two letters of a postcode. Where the type of the values of known, the transformation can involve rules such as normalizing person names to first character of initial name and the full last name [On et al., 2005]. Then, a variety of blocking techniques can be applied. The traditional blocking technique simply places records with identical blocking key values into the same block [Fellegi and Sunter, 1969]. However, matching records with slight variations in blocking key values will be placed into different blocks and be missed in the subsequent comparison. Thus, alternatives for overcoming this limitation has been proposed. One simple alternative places records whose blocking key values contain a common token or n-gram into a block [On et al., 2005]. More sophisticated techniques employing the common n-gram idea uses suffix arrays [Aizawa and Oyama, 2005; de Vries et al., 2009]. Blocking key values can also be sorted, and any two sorted values falling within a window of size w be placed into the same block [Hernández, 1996]. Blocking can also be performed by ap-

plying a computationally cheap similarity metric such as TF-IDF cosine similarity and then applying a threshold. A more elaborate scheme using this idea is seen in canopy clustering [McCallum et al., 2000]. Finally, blocking key values can be mapped into (high-dimensional) Euclidean spaces. The dimensions becomes blocks, or distances between data points can be used for blocking [Jin et al., 2003; Li et al., 2006].

Almost all blocking algorithms have parameters, and badly set parameters can prevent a significant portion of true matches from being compared by the more expensive but accurate similarity metric [Winkler, 2005; Gu and Baxter, 2004; Goiser and Christen, 2006]. However, for the same blocking algorithm, the optimal parameters for different datasets can be quite different. Therefore, researchers have proposed methods to make blocking algorithms adaptive or learnable [Gu and Baxter, 2004; Bilenko et al., 2006; Michelson and Knoblock, 2006; Yan et al., 2007]. There has been proposals for making blocking algorithms parameter-free, but research in these are still pretty much in progress [Goiser and Christen, 2006; Christen, 2007].

Blocking algorithms can also be done in multiple passes, each using a different blocking key, thereby creating a hierarchical system of blocks that starts from the cheapest metric and ends at one expensive metric (usually string comparators) [Broadbent and Iwig, 1999; Winkler, 2005]. Unfortunately, the selection of the blocking criteria and the order of its application has been an art rather than a science, largely guided by intuition based on past experience as well as the knowledge of the characteristics of the data itself.

2.2.5 Adaptive Methods

As we have seen, record matching algorithms often come with parameters that needs to be tuned. The word *adaptive* has been applied to a wide range of algorithms that automatically fit itself to its environmental conditions, typically by automated tuning of (possibly internal) parameters based on its input data. This technique has been largely adopted in data integration research to improve query processing in recent years (*e.g.*, [Ng et al., 1999; Zhu and Wu, 2004]). Adaptive methods have also been proposed for

blocking (*e.g.*, [Gu and Baxter, 2004; Bilenko et al., 2006; Yan et al., 2007]) as well as record matching itself (*e.g.*, [Bilenko and Mooney, 2003]). Often the aim of the adaptive methods is to improve performance or to reduce running time, or a trade-off between both. Adaptive methods can also be iterative in nature, since decisions made in one iteration can be used to adaptively influence the decisions made in the next iteration (*e.g.*, [Bhattacharya and Getoor, 2004; Dong et al., 2005]).

Many of the proposed adaptive methods can be seen as adaptively combining two or more methods or data sources such that the better aspect of each is achieved. For example, blocking may be seen as a fast method to filter out obvious mismatches, so that the more expensive linkage algorithm needs to perform comparisons on a significantly reduced portion of the data. Another example is to adaptively decide when to search and when to crawl in text-centric tasks involving a hidden web database so that execution time is minimized for a required recall level [Ipeirotis et al., 2006].

2.3 Web-based Record Matching Algorithms

There are many real-life problems in which the given information is either insufficient or incomplete. Therefore, an increasing number of solutions elect to acquire additional information from external resources, such as by querying the web through a search engine, to achieve a better solution quality. Utilizing Web information through a search engine effectively treating the Web as a *hidden web* database [Kautz et al., 1997], in the sense that the only way to retrieve documents from the database is through a query interface. Examples of record matching work that utilizes a search engine include the creation of an ontology where entities are matched to concepts [Cimiano et al., 2005], measuring semantic similarity between words [Bollegala et al., 2007], social network extraction (finding relationships between people) [Matsuo et al., 2006], word sense disambiguation (matching words to senses) [Mihalcea and Moldovan, 1999], machine transliteration (matching terms to candidate transliterations) [Oh and Isahara, 2008], record linkage [Elmacioglu et al., 2007a], disambiguation of person names [Bollegala

et al., 2006b; Bollegala et al., 2006a], and web people search (differentiating people of the same name in search engine results) [Kalashnikov et al., 2008]. All of these works demonstrated that acquiring additional information through a search engine resulted in increased matching effectiveness.

In this section, I first discuss the form of search engines, then discuss in detail how search engine results are utilized for record matching tasks. Then, I explain the acquisition bottleneck caused by the potentially large number of queries one can make when the input contains a large number of records.

2.3.1 Form of Search Engine Queries

I first study the form of search engine queries made by Web-based algorithms by discussing a few representative works in more detail. These works deal with different problems and scenarios in fields such as information retrieval and natural language processing. In [Cimiano et al., 2005], an ontology building system is created to match the entities (*e.g.*, people, organizations, and locations) extracted from an input document D to a list of concepts. For an entity $e \in D$ and a concept c, the system submits queries formed by e concatenated with patterns associated with c to a search engine. From the retrieved snippets, if the similarity of a snippet and D exceeds a threshold then concept c gets a vote. Thus, e will be associated with the concept with the highest number of votes. In [Bollegala et al., 2007], the semantic similarity between two words a and b is measured by first obtaining hit counts from the three queries a, b, and $a \wedge b$. Using these hit counts, web versions of Jaccard, overlap, Dice, and pointwise mutual information metrics can be computed. The values of these metrics then become attribute values in a test instance which is classified using a support vector machine classifier. [Oh and Isahara, 2008] described a machine transliteration system in which the transliteration candidates of a word a are generated to be $b_1, \ldots, b_n$. For each candidate b_i, the queries a, b_i, and $a \wedge b_i$ are made, and then the system counts the frequency of occurrence of b_i as well as a and b_i in the resultant web pages for each type of query. Normalized frequencies then become attribute values in a test instance, and a trained classifier is used

to select the best transliteration candidate b_i. In [Elmacioglu et al., 2007a], a framework for finding matching entities in an input list is given. The authors first find a representative token t_c from the input data, and then query the search engine with the query $e \wedge t_c$ for each entity in the list.

Through these works, a clear pattern can be seen. To perform record matching for input lists, queries of the form a, b, and $a \wedge b$ are issued to the search engine for some or all record pairs a and b in the input. Optionally, these queries can be further augmented with additional terms or tokens t, thus we may query with $a \wedge t$ instead of query with just a. Then similarity metrics or other information such as frequency counts may be extracted from the search engine results, which may be used standalone or combined to form test instances for a classifier such as a support vector machine.

2.3.2 Using Web Information for Record Matching

Next, I go into more detail how web information obtained from search engine results can be used for record matching tasks. Recall that for a search engine query, the search engine returns the total number of results matching the query (also known as hit count), and for each result, the title of the web page and a keyword-in-context short snippet, as well as its URL.

Let us denote the hit count of a query q by $hitcount(q)$. If the search engine returns reliable hit counts, then we can view $hitcount(q) \approx |D(q)|$, where $D(q)$ is the set of Web documents indexed by the search engine containing the query string q. Also, we have $hitcount(a \wedge b) = |D(a) \cap D(b)|$ and $hitcount(a \vee b) = |D(a) \cup D(b)|$, but as the latter can also be computed using $hitcount(a \vee b) = hitcount(a) + hitcount(b) - hitcount(a \wedge b)$, therefore we focus on conjunctive queries and ignore disjunctive queries. Given the relationship between $hitcount(q)$ and the set $D(q)$, we can use hit counts to compute set-based similarity measures, such as Dice and Jaccard similarity. Using hit counts to compute pairwise similarity measures is seen in a number of works (*e.g.*, [Bollegala et al., 2007], [Elmacioglu et al., 2007a]).

Next, we consider the keyword-in-context snippets. The web page title is often

treated as part of the snippet, but it can also be treated separately. A simple way is to consider all the tokens in the snippet as a multiset and compute set-based similarity between two such multisets [Elmacioglu et al., 2007a]. In this case, TF-IDF cosine similarity is popular. Alternatively, for the snippets of a, one can count the number of snippets that b occurs in, either anywhere in the snippet or occuring within a window containing a [Oh and Isahara, 2008]. One can also apply more advanced information extraction techniques on the returned snippets. For example, [Cimiano et al., 2005] used Hearst patterns [Hearst, 1992] such as "New York and other cities" and "hotels such as the Ritz" to extract the possible concepts for an entity, such as New York is a city and the Ritz is a hotel. Instead of the snippets, similar processing can also be done on web pages at the URLs of the search engine results. However, downloading the web pages incur additional costs. While web pages contain the complete material, snippets only contain a small amount of text, typically partial sentences with possibly the search phrase crossing sentence boundaries. This characteristic of snippets poses both challenges and benefits. A major challenge of using snippets is that the very limited textual information limits the usefulness of counts such as text frequencies, turning TF-IDF essentially into IDF. However, a major benefit of the snippets is that it already performs the necessary extraction of the context. This is unlike web pages, which may contain much other content that is irrelevant to the query.

Finally, we discuss how the URLs returned by a search engine can be utilized. As the URLs or their hostnames in a search engine result can again be seen as a set, we can compute similarity measures such as overlap coefficient or Jaccard coefficient between the URLs or their hostnames of two queries [Elmacioglu et al., 2007a; Aumüller, 2009]. For solving the problem of affiliation string matching, Aumüller [Aumüller and Rahm, 2009; Aumüller, 2009] proposed a URL overlap similarity measure. For two queries a and b, retrieve their top-k search engine results, denoted by $search(a, k)$ and $search(b, k)$ respectively. Then the URL overlap similarity measure is defined as:

$$simURLdist(a, b, k) = \frac{\alpha\frac{\omega}{k} + \beta\frac{1}{1+\delta}}{\alpha + \beta}$$

where ω is the number of overlapping URLs between $search(a, k)$ and $search(b, k)$, α and β are weighting factors (*e.g.*, 2 and 1 respectively), and δ is the difference between the rank of the first URL in $search(a, k)$ contained in $search(b, k)$ and the rank of the first URL in $search(b, k)$ contained in $search(a, k)$. The components of a URL can also be exploited. For example, the URL http://wing.comp.nus.edu.sg/~tanyeefa/downloads/searchenginewrapper/ has the hostname wing.comp.nus.edu.sg, plus other components ~tanyeefa, downloads, and searchenginewrapper, which encode a path from the root directory / to the leaf directory /~tanyeefa/downloads/searchenginewrapper/. Nasraoui and Krishnapuram [Nasraoui and Krishnapuram, 2002] proposed that the overlap between two such paths can be used to compute a similarity measure between URLs. More advanced URL processing is also possible. For example, Kan and Nguyen [Kan and Nguyen Thi, 2005] demonstrated that web page classification can be performed by using the URL alone. They extract the following features from a URL: components of the URL, orthographic features, sequential n-grams, and precedence bigrams. These features were then used in a maximum entropy classifier for web page classification.

Sometimes, the search engine results are not directly used for record matching. Instead, it is used to build a corpus of documents for subsequent data mining. For example, for disambiguating author names in bibliographic citations, Pereira *et al.* [Pereira et al., 2009] queried a search engines using the titles of the citations to build a corpus of documents. Then, they try to identify single-author documents from the corpus, and use them to cluster the input citations.

2.3.3 The Acquisition Bottleneck

All of these works demonstrated that acquiring additional information through a search engine resulted in increased matching effectiveness. However, acquiring such web information is time consuming due to slow web accesses and rate limiting by search engines, and can entail other access costs. Given various sources of information, such as search engine results and Web page downloads, how do we best utilize them? Be-

cause Web information can take a relatively long time to retrieve, we naturally desire an effective solution that executes in a reasonable amount of time. In this thesis, I propose a cost-sensitive framework for selecting which pieces of Web information to retrieve.

A way to limit the cost of resource acquisitions is by blocking, as mentioned earlier, which filters out obvious mismatched record pairs before performing matching on the remainder. However, blocking techniques are often ad-hoc and domain-specific, and can be difficult to define properly. Poor blocking decisions can degrade performance significantly by either filtering out too little or too much [Goiser and Christen, 2006]. Also, if we apply multi-pass blocking, it is often not obvious how the passes should be ordered. However, if we consider each blocking algorithm as producing a value, then we can treat these values as information resources that can be acquired. In this way, I can propose a more principled approach for performing value acquisitions.

On the other hand, there is also a large pool of work that formulates such resource acquisition problems into selective and cost-sensitive acquisition of missing attribute values in a classification model (*e.g.*, [Ling et al., 2006], [Saar-Tsechansky et al., 2009]). While such works provide more principled acquisition algorithms, they generally assume that each attribute value in each instance is an independent resource. However, this is not true in the Web resource context, *e.g.*, the hit count (number of web pages matching a query) for a query a can be a common attribute value for all pairwise instances that compare a with another item. Also, such works ignore possible hierarchical dependencies between resource acquisitions, *e.g.*, acquiring the hit count of a requires first acquiring the search engine results of a, and these search engine results can also be used to generate other attribute values such as similarity metrics between a and some other query.

Chapter 3

Using Web-based Resources for Record Matching

3.1 Introduction

Record matching algorithms can be improved by acquiring web-based resources, typically through a search engine. This is particularly so when the existing contents in the input records contain incomplete or insufficient information. In Section 2.3, I have presented related work that uses web-based resources. I continue with this theme in this chapter, and present my own contributions to three applications of record matching problems by utilizing web-based resources in my solutions.

The first goal of this thesis is to show that acquisition of web-based resources can benefit record matching tasks. This chapter fulfills this goal by means of concrete examples. Section 3.2 describes my contributions towards the problem of author name disambiguation, Section 3.3 describes my contributions towards the problem of linkage of short forms to long forms, and Section 3.4 summarizes my contributions towards the problem of web people search.

3.2 Search Engine Driven Author Disambiguation

3.2.1 Introduction

Bibliographic digital libraries such as DBLP [Ley, 2002] and CiteSeer [Giles et al., 1998] contain a large number of publication metadata records and make these records searchable for academics. A common use of such repositories is to assess the impact of individual researchers on the community. A problem occurs when different individuals share the same name. This leads to mismatch problems in which citations to different authors may be mixed together in a single list (*e.g.*, W. Wang). Such problems can hinder scientific data gathering, information retrieval and even credit attribution [Han et al., 2005].

Previous works have focused on using knowledge encoded in the citation records to form appropriate author clusters. For example, Lee *et al.* [Lee et al., 2005] considered similarity between citations and authors as well as performed blocking on coauthor information, and Han *et al.* [Han et al., 2005] used spectral clustering on various fields of the citation record. However, the information contained in citation records are sometimes ambiguous. For example, two citations on the same topic may use disjoint keywords in their titles.

A key differentiating factor in my work is that I leverage resources external to the citation data to resolve this problem. In particular, I leverage the collective information on the web to do disambiguation, by employing a web search engine. Specifically, I attribute a citation to a particular author, based on the pages returned by a search engine in response to web queries. Just as importantly, my search engine based author disambiguation algorithm is complementary to other author disambiguation algorithms that utilize other kinds of information. In particular, I show that it is possible to combine my search engine based algorithm with another algorithm that uses coauthor information to produce a hybrid algorithm that gives even better disambiguation performance.

3.2.2 Using Inverse Host Frequency for Author Disambiguation

Algorithm

I deal with a restricted version of the author disambiguation problem. Given an author string name X (representing a known k unique individuals) and a list of citations C containing the name X, my system identifies which citations are attributed to which of the k authors. This problem formulation is identical to the *mixed citation problem* in [Lee et al., 2005], and can be seen as a standard k-way classification problem.

When lay people are faced with the author disambiguation task and are given unfamiliar publications, they may query a search engine with the publication titles and use the results to help them distinguish between the different authors. My method tries to approximate this process.

My algorithm is as follows: For each citation $c \in C$, I query a search engine using the title of c as a phrase search to obtain a set of relevant URLs. Each citation $c \in C$ is then represented by a feature vector, whose features are the relevant URLs and weighted by their IHFs (explained below). Next, I compute the pairwise similarity of two citations $c_1, c_2 \in C$ using cosine similarity. Finally, I perform hierarchical agglomerative clustering (HAC) on C using the similarity values to derive k clusters. The final clusters represent the k individual authors.

IHF Weighting. URLs returned by a search engine are not equally useful, as some may belong to aggregator services, such as DBLP and CiteSeer. To overcome this problem, I desire a weighting scheme that weighs aggregator web sites with low values and personal and group publication web pages with high values. Akin to the Inverse Document Frequency measure used in information retrieval, I formulate an Inverse Host Frequency (IHF) to gauge the relative rarity of an Internet host among a suitable corpus of web documents. Hosts that correspond to aggregator services will have a high frequency among searches on publication titles, thus a low IHF.

I first form a corpus to establish IHF values. I obtain the citations belonging to the top 100 author strings (by number of citations) in DBLP, and then query a search engine with each citation's title. The returned URLs are truncated to its hostname. If a

hostname h has frequency $f(h)$, then its *inverse host frequency* is computed as:

$$\text{IHF}(h) = \log_2 \frac{\max_h f(h) + 1}{f(h) + 1} + 1$$

In my implementation I notice that using the hostname alone may have problems, especially when the host has multiple names or is represented by an IP address (*e.g.*, www.informatik.uni-trier.de, ftp.informatik.uni-trier.de and 136.199.54.185, all the same host) with dissimilar distributions. To correct for these anomalies, I also tried using the domain instead (*e.g.*, uni-trier.de) and resolving all hostnames to IP addresses before processing.

Evaluation

To evaluate my approach I use a manually-disambiguated dataset of computer science citations with 24 ambiguous names, as used in [Lee et al., 2005]. These names represented 2 unique authors ($k = 2$) in all but one case where it represented 3. Each name is attributed to 30 citations on average, and the proportion of the largest class ranges from 50% to 97%. I used the Google search engine through its SOAP Search API, and I attempt to retrieve 10 URLs per citation.

I measure performance using classification accuracy. Suppose the classes are labeled from 1 to N. Let a_i and p_i be the actual and predicted classes of the ith citation respectively. Let $s(x, y)$ be 1 if $x = y$, 0 otherwise. Then,

$$\text{accuracy} = \max_{\pi \in S_N} \sum_i s(\pi(p_i), a_i),$$

where S_N is the set of permutations on the classes 1 to N.

I investigated using different clustering schemes with the IHF data. I tested three HAC schemes: single link, complete link and groupwise average. The accuracies averaged over all names are summarized in Table 3.1.

From Table 3.1, I see that single link always performs as well as or better than complete link and groupwise average. One of the main reasons could be that a publication

Clustering	Hostname	Domain	IP address
Single link	0.827	0.807	**0.836**
Complete link	0.726	0.798	0.734
Group average	0.805	0.811	0.812

Table 3.1: Average accuracy over all author names.

page of an author sometimes omits some of his or her publications. For example, if a publication page contains only citations c_1 and c_2, and another contains only c_2 and c_3, then single link is best suited for merging all of c_1, c_2 and c_3 into the same cluster. As highlighted, resolving all hostnames to IP addresses gives the best accuracy. I believe that single link may perform better when authors have disparate areas of research, and are not well represented by a centroid vector.

In 2009, Pereira *et al.* [Pereira et al., 2009] compared my HAC method against another non Web-based state-of-the-art unsupervised K-way spectral clustering algorithm by Han *et al.* [Han et al., 2005], on a different author name disambiguation dataset. Pereira *et al.* reported that my algorithm achieved a pairwise F_1 measure of 0.46 ± 0.14, while the K-way spectral clustering algorithm achieved a pairwise F_1 of 0.36 ± 0.05. As both algorithms are unsupervised, this comparison showed that a simple disambiguation algorithm that utilizes a search engine can significantly outperform a state-of-the-art algorithm that does not use any Web-based resources, highlighting the usefulness of Web-based resources for the task.

Discussion and Conclusion

An investigation of per-name accuracy is shown in Figure 3.1 using the single link HAC scheme. First, I see an apparent correlation between the accuracies and the average number of URLs returned per citation, which is shown in Figure 3.2. Those author names with few URLs returned per citation tend to fare poorly, because those results will mostly be aggregator web pages and serves little for the disambiguation task. Second, I do not observe any apparent relation between the accuracies and number of citations for a given author name. I take this to mean that my algorithm will scale, even

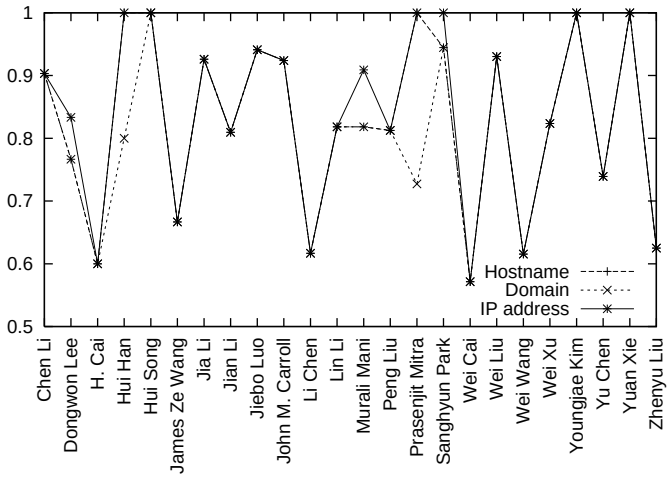


Figure 3.1: Per-name accuracies using single link.

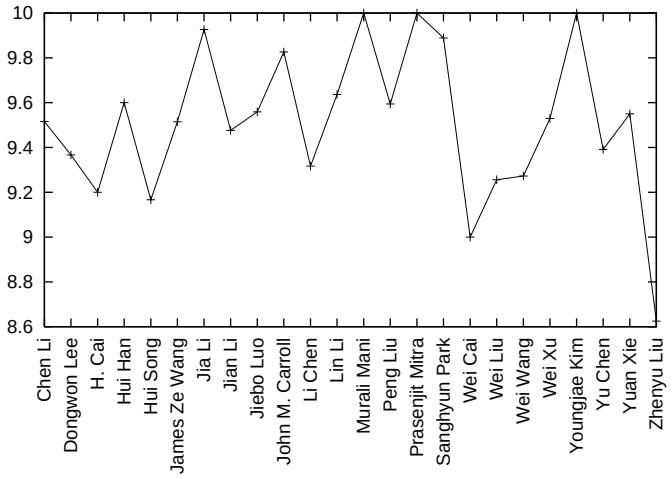


Figure 3.2: Per-name average number of URLs returned per citation.

when the number of citations is large, provided that there is enough evidence from the URLs returned. As for time efficiency, the analysis of the returned URLs is very fast, and the execution time is dominated by the search engine querying. In scholarly digital libraries, such querying may already be done during spidering, making my approach particularly time-efficient.

I have focused on using the URLs returned from searching the citation titles, and obtained a respectable average accuracy of 0.836 using IP addresses with single link HAC clustering. Next, in Section 3.2.3, I show that coauthor information is another powerful source of information for the author disambiguation task.

3.2.3 Using Coauthor Information for Author Disambiguation

I observe that researchers tend to form research groups that work on similar research areas, and people from the same group is likely to coauthor a number of papers. Therefore, coauthor information can be a strong indicator for disambiguation.

In this section, I describe how coauthor information can be used to solve the mixed citation problem. Further, I show how coauthor information can be integrated with search engine results as described in Section 3.2.2 to achieve improved results.

I first introduce a simple similarity measure that uses coauthor information. Next, I introduce the *coauthor linkage similarity*, which is more sophisticated and surpasses the simple coauthor similarity. Note that the first method uses only internal knowledge while the second uses external knowledge.

Simple coauthor similarity. Given an author name a_k and a set of citations C that contain a_k as an author. Note that each citation in C can have multiple authors. For each citation $c \in C$, we can form a binary feature vector, indexed by all the author names in C except a_k. The feature value for an author name a_i is 1 if c contains a_i as an author, 0 otherwise. In other words, the feature vector for a citation c contains the list of coauthors that has coauthored with a_k when writing the paper that is cited by c. The *simple coauthor similarity* between two citations is then the cosine similarity between their feature vectors.

Coauthor linkage similarity. A collection of external digital library records is used to build a *coauthor graph*, which is then used to determine the similarity between two authors. For each author name a_i in the external citation records, a node v_i is added to the graph. If a citation in the metadata contains two different author names a_i and a_j, indicating that a_i and a_j have coauthored a paper, then the edge (v_i, v_j) is added to the graph. Each edge in the graph is assigned unit cost.

When using the coauthor graph to disambiguate an author name a_k, I first delete v_k and its associated edges. Given two author names a_i and a_j, denote the shortest path from v_i and v_j by $D_{i,j}$. Due to the size of the coauthor graph, I do not search for shortest paths that are longer than M, and instead set $D_{i,j}$ to be M for such cases. The similarity

Clustering	Simple coauthor	Coauthor linkage
Single link	0.821	0.833
Complete link	0.808	**0.844**
Group average	0.804	0.828

Table 3.2: Average accuracy over all author names for using coauthor information.

between a_i and a_j can then be computed as:

$$sim_a(i,j) = \frac{M - D_{i,j}}{M}.$$

Suppose we have two citations C_i and C_j having the author name a_k. Let the author names in C_i and C_j, excluding a_k, be A_i and A_j respectively. The *coauthor linkage similarity* between C_1 and C_2 is defined as:

$$sim_c(i,j) = \sum_{p:a_p \in A_i, q:a_q \in A_j} sim_a(p,q).$$

In my experiments, the entire DBLP metadata, downloadable in XML form, is used to build the coauthor graph. Also, I set $M = 5$.

An evaluation is done using the same evaluation dataset and metric, as used in Section 3.2.2. The results are shown in Table 3.2. The accuracy for the simple coauthor similarity is pretty respectable, and points at the usefulness of using coauthor information for the disambiguation task. However, the coauthor linkage feature has significantly better accuracy over the simple coauthor feature. Also, the coauthor linkage feature performs best when using complete link. Due to the "small world effect", the shortest path between any two author names in the coauthor graph is expected to be between 5 and 6. Therefore, between any two citations of different individuals sharing the same target name, there will be many such paths between their respective coauthors even though these coauthors may not be very related. As such, when computing cluster-to-cluster similarity, it is best to apply complete link as it returns the similarity of the most dissimilar citations.

Method	Accuracy
IHF only (IP Address, Single Link)	0.836
Coauthor Linkage only (Complete Link)	0.844
IHF and Coauthor Linkage (Hybrid Method)	**0.850**

Table 3.3: Average accuracy over all author name after combining IHF and coauthor linkage.

3.2.4 Combining IHF with Coauthor Linkage

Recall that in Section 3.2.2, the IHF feature performs best when single link is used for clustering. However, the coauthor linkage feature performs best when complete link is used.

It is expected that combining the two features together will improve the accuracy even further. However, the difference between the intermediate cluster-to-cluster similarity measures preferred by the two features posed some difficulty when combining the two features together. During experiments, it is found that combining the two feature vectors into one does not improve the accuracy. Also, no improvement was found when the similarity between two citations is computed as a weighted average between the IHF similarity and the coauthor linkage similarity.

Therefore, I proposed a hybrid method for combining the two features, which makes use of the fact that when the HAC algorithm for coauthor linkage is run, the IHF feature prefers single link while the coauthor linkage prefers complete link. This hybrid method changes the way the intermediate cluster-to-cluster similarity is computed. For two clusters C_1 and C_2, their hybrid similarity is defined to be the average of S_1 and S_2, where S_1 is the IHF single link between C_1 and C_2, and S_2 is the coauthor linkage complete link between C_1 and C_2.

As shown in Table 3.3, we see that the two features combined using a hybrid method outperforms using only either feature in isolation. In other words, information obtained from a search engine can complement information obtained from coauthor information.

E. Agichtein, P. G. Ipeirotis, and L. Gravano. Modeling query-based access to text databases. In *WebDB*, 2003.
J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *SIGIR*, 1995.
W. W. Cohen. Learning trees and rules with set-valued features. In *AAAI*, 1996.

Figure 3.3: References with publication venues abbreviated.

3.2.5 Conclusion and Discussion

External resources is routinely used to solve research problems. I have demonstrated that two kinds of external resources, search engine results and coauthor graph built from external digital library metadata, are capable of solving the disambiguation problems in scholarly digital libraries. However, we can also explore other sources of information, such as the publication venues of the citations and utilizing the actual contents of the web pages.

3.3 Web-Based Linkage of Short to Long Forms

3.3.1 Introduction

Proper nouns, technical terms, and long words are often shortened for saving space or improving clarity, due to writing style or convenience. Figure 3.3 shows typical examples of bibliographic references, randomly selected from [Ipeirotis et al., 2006], where publication venues have been abbreviated ("WebDB", "SIGIR", and "AAAI"). Figure 3.4 shows more examples in bibliographic publication venues, stock ticker symbols, and human genome research. However, it is not obvious how short forms are generated from long forms. For example, "MOU" includes the word "of" in its abbreviation, while "MIT" does not. Also, "UbiComp" is generated phonetically rather than by selecting initial letters of words. Further, different long forms can have the same short form, such as "ACSAC". It should also be noted that a long form can have multiple valid short forms. For example, both "MIT" and "Mass. Inst. of Tech." are both short forms of the long form "Massachusetts Institute of Technology". In this work, I

DBLP Computer Science Conferences and Workshops	
ACSAC	Annual Computer Security Applications Conference
ACSAC	Asia-Pacific Computer Systems Architecture Conference
KDD	Knowledge Discovery and Data Mining
KDID	Knowledge Discovery in Inductive Databases
UbiComp	Ubiquitous Computing
WebDB	International Workshop on Web and Databases
NASDAQ Composite	
AAPL	Apple Inc.
CSCO	Cisco Systems, Inc.
DELL	Dell Inc.
INTC	Intel Corporation
MSFT	Microsoft Corporation
XRAY	DENSPLY International Inc.
Human Genome Acronym List	
MALDI	matrix-assisted laser desorption ionization
Mb	megabase
MGI	Microbial Genome Initiative
MHC	major histocompatibility complex
MIT	Massachusetts Institute of Technology
MOU	Memorandum of Understanding

Figure 3.4: Examples of abbreviations in various domains.

always use the shortest possible version of a long form as the short form, *i.e.*, "MIT" in the preceding example.

Short forms are a principal way in which variations are introduced to string representations of names, contributing to data quality issues when mining the Web. To aid knowledge discovery and uncovering implicit linkages, resolving short and long forms plays an important role in many data applications. In this section, I study the problem of linking short forms to long forms. I believe that resolving short forms to long forms is harder than the other way around, because the former requires the interpolation of missing data from abbreviations that are typically 3 to 5 letters long, while the latter only needs to discard extraneous data. My problem statement is as follows:

> Given a set of short forms SF and a set of long forms LF, for each short form in SF, find the corresponding matching long forms in LF.

In this section, I assume that no contextual information is available when linking short forms to long forms. For example, in Figure 3.3, we see the short forms of publication venues ("WebDB", "SIGIR", and "AAAI") but not their corresponding long

forms. Hence, algorithms that detect short forms and link them with their corresponding long forms in full-text documents (*e.g.*, [Chang et al., 2002], [Schwartz and Hearst, 2003], [Ao and Takagi, 2005], [Okazaki and Ananiadou, 2006], [Torii et al., 2006]) are inappropriate for my problem setting as they depend on the missing contextual information. Also, while a number of short to long form lists (*e.g.*, acronym lists) are freely available, they are usually incomplete and quickly outdated, as new short forms are continually created. To remedy these problems, I explore the use of the Web as an additional knowledge source. In particular, I propose to use a *search engine* such as Google, Yahoo!, or Bing to obtain additional information that facilitates linking decisions. My major contributions are as follows:

- I propose to exploit the external knowledge from the Web to obtain the required contextual information that is typically missing from such data. While similar work has employed the Web for other tasks, my work is unique, in being the first to specifically tackle the problem of short-to-long form matching and in unifying related threads of research on this theme. I view the task as two related facets: a) query composition, and b) search engine evidence analysis. In particular, I propose a count-based method that is effective for linking short forms to long forms.

- I compare my proposed method with other types of search engine evidence on three datasets of different domains. The results show that my claims consistently hold for all the three datasets.

I first describe related work in the general linkage area in Section 3.3.2. Next, I describe a framework that unifies various approaches that use a search engine to perform linkage in Section 3.3.3. I then propose a count-based method in Section 3.3.4 for linking short forms to long forms. I show its effectiveness compared to other linkage methods in Section 3.3.5, before concluding with a discussion in Section 3.3.6.

3.3.2 Related Work

Most closely related to my work, [Jain et al., 2007] described an approach that combined three kinds of data available in a search engine backend to extract short and long form pairs. In contrast, my approach does not assume access to search engine internals, and deals directly with constructing long-short form pairs using only the front end querying interface.

More generally, the linkage problem has been widely studied and is known by various names. A comprehensive survey is beyond the scope of this thesis; introductions can be found in [Elmagarmid et al., 2007] and [Winkler, 2006]. A number of linkage works focus on finding suitable string similarity metrics (*e.g.*, [Bilenko et al., 2003; Cohen et al., 2003; Snae, 2007]). However, most linkage works assume that information can be extracted from multiple record fields, whereas my problem has only one "record" field readily available. Among all record linkage works, web-based approaches are most relevant (*e.g.*, [Cimiano et al., 2004; Cimiano et al., 2005; Oh and Isahara, 2008]). Some works used conjunctive keyword queries – querying for $sf \wedge lf$ to see whether a short form sf and a long form lf are linked. As this results in quadratic time complexity, non-conjunctive keyword approaches have also been developed. These only query from one side – querying for sf to see whether the results have any evidence for lf. While efficient, this can lead to problems in accuracy and coverage. One key contribution of my work is to further improve upon this by introducing bi-directional non-conjunctive querying, resulting in higher accuracy while retaining linear complexity.

Recent work have also dealt with the finer details of using the Web for linkage evidence. [Sahami and Heilman, 2006] and [Kwok et al., 2007] looked at effective query expansion by using the Web. [Matsuo et al., 2006] extended this by identifying characteristic terms that differentiate namesakes. [Elmacioglu et al., 2007a] suggested that if two strings refer to the same entity, then each string will frequently co-occur with some common information piece on the Web.

Algorithm 3.1 Overall algorithm.

```
1: for each sf ∈ SF do
2:    for each lf ∈ LF do
3:       obtain information for sf and lf using search engine
4:       compute score_sf(lf) using obtained information
5:    rank the long forms in LF according to score_sf(lf)
```

3.3.3 Linking Short to Long Forms

In this work, I use the following notation. Uppercase SF and LF denote the sets of short forms and long forms in the dataset, respectively. Lowercase sf and lf denote specific instances from SF and LF, respectively. $score_{sf}(lf)$ denotes a scoring function that ranks the long forms in LF that potentially match a particular short form sf. The function $SearchEngineTop(q, k)$ retrieves the top-k results from a search engine for the query q.

My intuition is that if a short form and a long form indeed refer to the same real-world entity, people would use them interchangeably on the Web. To link a set of short forms SF to a set of long forms LF, I consult a search engine for external linkage evidence, as illustrated in Algorithm 3.1. For a short form sf and a long form lf, I can derive information from a search engine, and use it to compute a *scoring function* $score_{sf}(lf)$ to rank the long forms in LF for a given short form sf. How to do this efficiently – in terms of query bandwidth – is the central focus of my work. Next, I investigate open details, such as the form of the queries and the computation of the scoring function.

Designing the Search Queries. As discussed, to gather linkage evidence, one can issue conjunctive keyword queries, *i.e.*, $q = sf \wedge lf$, or issue non-conjunctive keyword queries, *i.e.*, $q = sf$ or $q = lf$, or both. If $|SF| = m$ and $|LF| = n$, then conjunctive keyword queries require $O(mn)$ queries, which is of quadratic complexity and infeasible given long lists. On the other hand, non-conjunctive keyword queries only require $O(m)$ or $O(n)$ queries, or $O(m + n)$ queries, which is still feasible. Hence, I only consider non-conjunctive keyword queries in this work. Note that we can add domain knowledge to the query to further filter results. For example, for DBLP publi-

Human Genome Project - Wikipedia, ... "Genomes: 15 Years Later A Perspective by ...
More on the sequencing of the human genome The international **Human Genome Project** (HGP) ...
More on the sequencing of the human genome ... Approximately 60% of the underlying sequence data ...

Figure 3.5: Snippets (simplified) from the query "HGP".

cation venues, we may add the keywords "workshop" and "conference" to the query to promote web pages with publication metadata in the returned results.

Two possible modifications to this general framework should be noted. First, it is possible to add domain knowledge to the query to further filter results. For example, in the case of DBLP publication venues, we may add the keywords "workshop" and "conference" to the query to attempt to promote web pages with publication metadata to the top of the returned results. Second, previously executed queries may provide evidence that can be applied in a transitive closure. For example, if lf is linked to sf and lf' is linked to sf, then we may believe that lf is also linked to lf' [Dong et al., 2005]. I do not explore these issues further, but I note that these modifications do not change the number of queries needed.

Search Engine Evidence. A search engine typically returns a page of results (usually 10), and each result contains its rank, title, Keyword-In-Context (KWIC) snippet, and URL. The total number of results is also reported. We can choose how to process these information to determine linkage evidence, and specify follow-up actions as needed, *e.g.*, download the web pages in the results. We can make repeated calls to the search engine to obtain multiple pages of results. The main focus here is on processing the search engine evidence, *i.e.*, defining the scoring function via information obtained from a search engine.

3.3.4 Count-based Linkage Methods

I propose methods for linking short forms to long forms by counting the terms in the returned results. These methods use the $SearchEngineTop(q, k)$ function, which

Algorithm 3.2 Computing $count(sf \rightarrow lf)$ by obtaining top-k search engine results for each short form.

```
1: for each sf ∈ SF do
2:     D = SearchEngineTop(sf, k)
3:     for each lf ∈ LF do
4:         count(sf → lf) = number of results in D containing lf
```

queries a search engine with q and retrieves top-k results. I define the following scoring functions:

- $count(sf \rightarrow lf)$ is the number of results (snippets or web pages) of the short form sf containing the long form lf. For a simplified illustration, suppose sf is "HGP" and lf is "Human Genome Project". We query a search engine with "HGP" and suppose we consider only the top-3 results, whose snippets are as shown in Figure 3.5. As two of these snippets contain "Human Genome Project", we have $count(sf \rightarrow lf) = 2$. Algorithm 3.2 shows this algorithm more formally.

- $count(sf \leftarrow lf)$ is the number of results of the long form lf containing the short form sf. It can be obtained by interchanging sf and lf in Algorithm 3.2.

- $count(sf \leftrightarrow lf) = count(sf \rightarrow lf) + count(sf \leftarrow lf)$ is a combination of the previous two.

While web-based and traditional record linkage techniques have been applied to other tasks, to the best of my knowledge, no study has yet to examine the efficacy of these techniques on the task of short form to long form matching. To my knowledge, my less computationally expensive $O(m + n)$ $count(sf \leftrightarrow lf)$ scoring function is a new contribution that may assist in other web-based linkage tasks.

3.3.5 Evaluation

Comparison with Other Types of Evidence

I will evaluate my count-based methods against three other methods adapted to solve the same problem.

Schwartz and Hearst. This algorithm [Schwartz and Hearst, 2003] is a state-of-the-art abbreviation extraction algorithm that expects full text input and does not use the Web. It scans full-text articles for fragments of the form "lf (sf)", and tries to extract the long form lf with the best alignment with the short form sf in a greedy manner. I apply this algorithm by creating a file containing all combinations of short forms and long forms in the dataset, with lines of the form "lf (sf)". This extraction algorithm returns candidate matching pairs without scoring their quality, but instead indicates which part of the long form matches (*e.g.*, the "Key Cryptography" part of "Public Key Cryptography"). Therefore, I define the scoring function to be the fraction of characters in the long form that was matched. For the above example, this fraction is $\frac{15}{21} = 0.714$.

Inverse host frequency. This web-based method uses URL information from search engine results. For a query q, I form a feature vector v_q of hostnames (or domain names) from the URLs of its top-k search engine results, weighted by its *inverse host frequency* (IHF) (see Section 3.2), *i.e.*, $\text{IHF}(h) = \log_2 \frac{\max_h freq(h)+1}{freq(h)+1} + 1$. Here, $freq(h)$ is the number of short and long form queries whose top-k results contain the hostname h. For a short form sf and a long form lf, the cosine similarity between v_{sf} and v_{lf} is the scoring function. I experimented with $k \in \{10, 20\}$.

Sahami and Heilman. This is a web-based information retrieval method [Sahami and Heilman, 2006] that I reimplemented. For each query q, I download the web pages at the URLs of its top-k results, $w_{q,1}, \ldots, w_{q,k}$. For each web page $w_{q,i}$, I compute its tf-idf vector $v_{q,i}$. Following Sahami and Heilman, I truncate $v_{q,i}$ to include only the 50 tokens with the highest tf-idf weights, and then normalize it. Next I compute $v_q = \frac{1}{k} \sum_{i=1}^{k} v_{q,i}$, and when normalized, it is the query expansion vector of q. For a short form sf and a long form lf, the cosine similarity between v_{sf} and v_{lf} is the scoring function. Also, Sahami and Heilman suggested that snippets can be used instead of web pages.

Dataset	Short forms	Long forms	Matching pairs
DBLP	906	920	926
Description: DBLP conference and workshop titles Source: http://www.informatik.uni-trier.de/~ley/db/conf/indexa.html Query: "$\langle title \rangle$" conference OR conferences OR workshop OR workshops			
NASDAQ	3084	3061	3084
Description: NASDAQ Composite stock symbols Source: http://www.nasdaq.com/asp/index_component.asp?symbol=IXIC Query: "$\langle title \rangle$" nasdaq			
GENOMES	307	307	307
Description: Human genome acronym list Source: http://www.ornl.gov/sci/techresources/Human_Genome/acronym.shtml Query: "$\langle title \rangle$" genome OR genomes			

Table 3.4: Evaluation datasets.

Datasets and Search Engine

To validate my methods, I examine real-world problems of matching short and long forms. In all experiments, I used the Google search engine via its SOAP Search API. I used three datasets from different domains for my evaluation: DBLP, NASDAQ and GENOMES which have very different characteristics. Each dataset contains the SF and LF sets, as well as the solution set S of matching short and long form pairs. I applied all the methods using SF and LF as input, and used S as the gold standard. Table 3.4 summarizes these datasets, and lists the form of queries used. The additional keywords in the queries were selected based on the domain of the dataset, with all web-based methods using the same queries. Examples from each dataset are shown in Figure 3.4.

The DBLP dataset consists of computer science conferences and workshops in the DBLP digital library. This dataset is generally clean, because DBLP was manually constructed and consistently uses full words in more than 98% of its long forms. The retrieved web pages tend to be conference and workshop web sites, and publication lists on academic homepages and research groups. The web pages sometimes contain typos and spelling mistakes. The NASDAQ dataset consists of stock symbols in the NASDAQ Composite index. This large list itself is also fairly clean, but there are stock

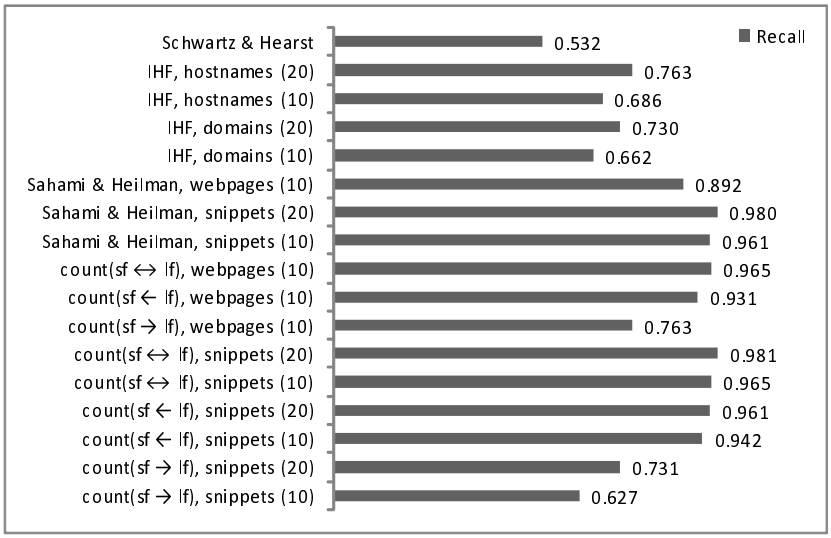
symbols that have no resemblance to the company name at all, such as "XRAY" for "DENSPLY International Inc.". The retrieved web pages come from a large variety of domains and mainly consist of financial news and stock information. Some of these web pages appear to be automatically generated from databases, hence these can be fairly clean. The GENOMES dataset consists of abbreviations that are commonly used in the human genomes domain. It is the smallest and noisiest among the three datasets. Some long forms have abbreviated words like "Univ." and "Intl.", but their usage is inconsistent. Worse, this list includes academic institutions, academic conferences, government organizations and biological terms. Hence, the retrieved web pages have all kinds of sources and information.

For the count-based methods, as well as Sahami and Heilman method, I evaluated on both the snippets and the downloaded web pages. For all methods except Schwartz and Hearst, I evaluated with $k \in \{10, 20\}$, *i.e.*, using top-10 or top-20 search engine results, except for web pages where I only show results for $k = 10$.

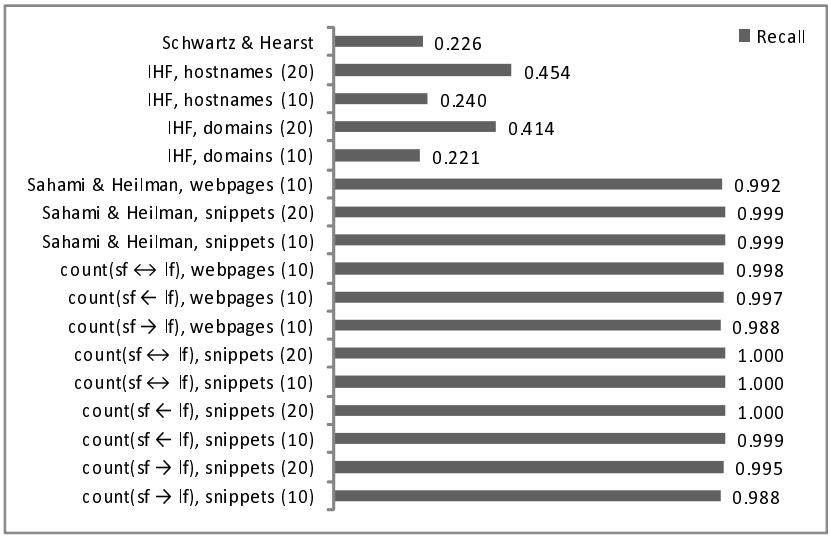
Experimental Results

To evaluate the various methods, I use average recall and average ranked precision as my evaluation metrics. Suppose a short form sf corresponds to R long forms, and the top-10 candidate long forms in the ranked list for sf contains r correct long forms. Then, the recall is $\frac{r}{R}$. To account for the quality of rankings, I use *ranked precision* [Hull, 1993] instead of traditional precision. Let P_i be the fraction of correct long forms within the top-i candidates. Let C be the set of positions of the correct long forms within the top-10 candidates. For example, if the 1st, 2nd, 4th and 8th long form candidates are correct, then $C = \{1, 2, 4, 8\}$. Thus, the ranked precision is $\frac{\sum_{i \in C} P_i}{r}$. The *average recall* and *average ranked precision* are the averages of recall and ranked precision over all short forms in my dataset.

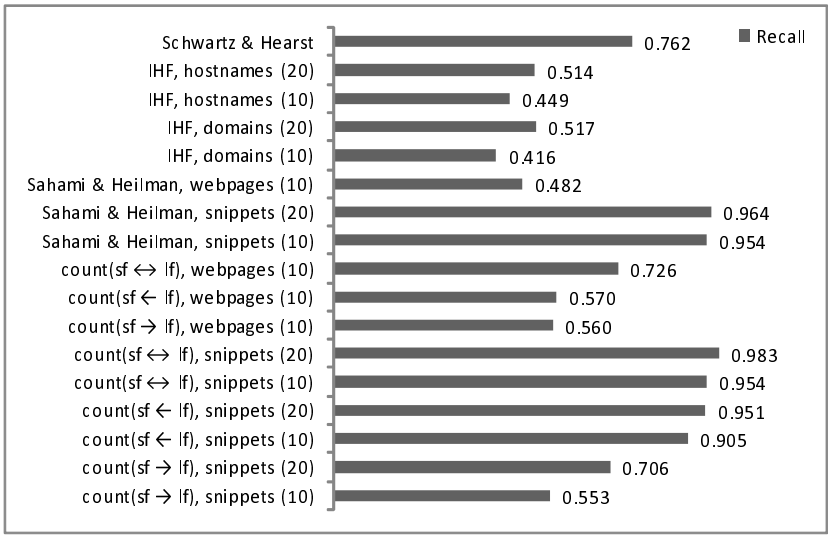
Figure 3.6 and Figure 3.7 shows the results. Overall, I found that the count-based methods with snippets tend to produce the best performance. In particular, I found that $count(sf \leftrightarrow lf)$ consistently produces the best results. The $count(sf \leftarrow lf)$ and



(a) DBLP

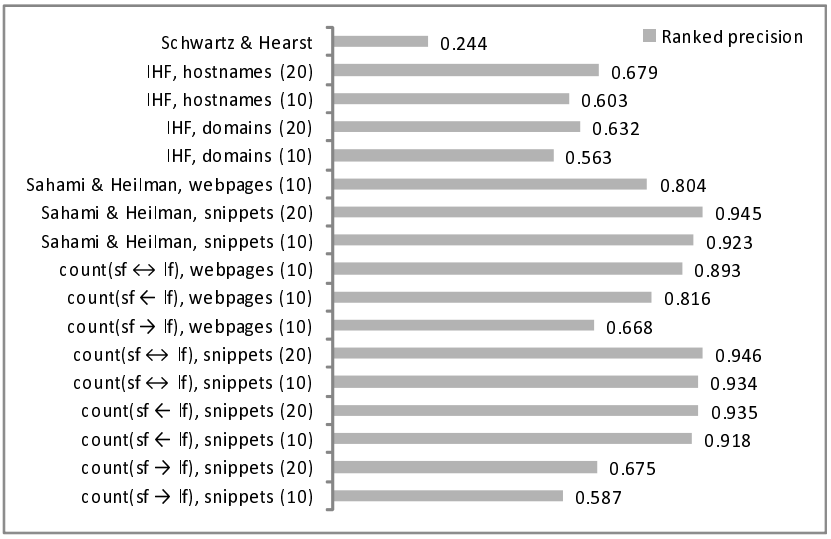


(b) NASDAQ

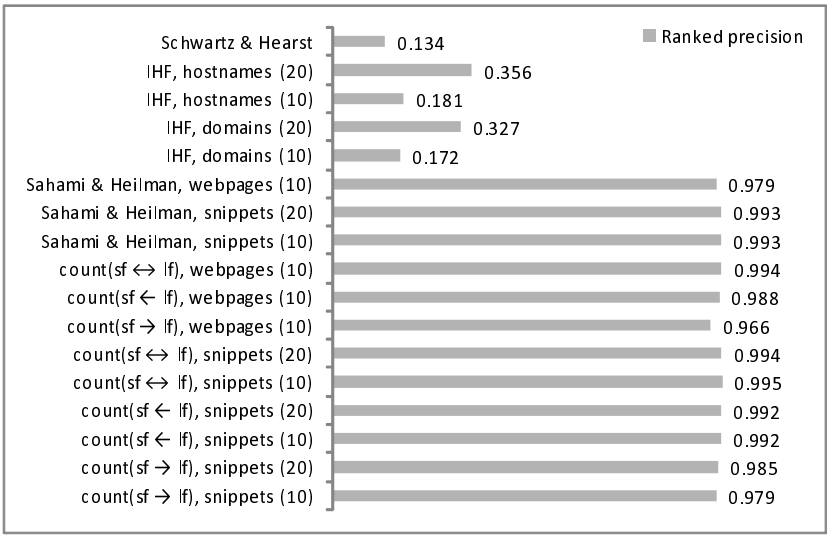


(c) GENOMES

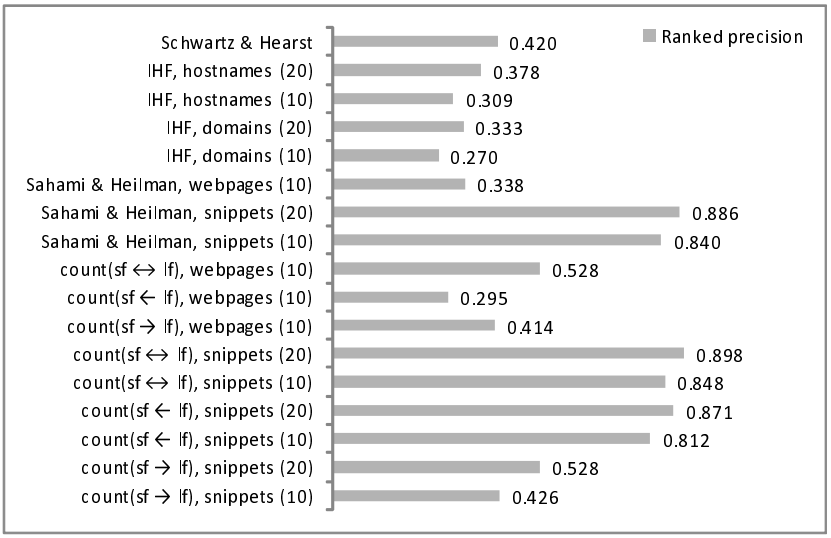
Figure 3.6: Average recall for the various kinds of evidence for the three datasets. Numbers in parentheses represent number of results retrieved per query.



(a) DBLP



(b) NASDAQ



(c) GENOMES

Figure 3.7: Average ranked precision for the various kinds of evidence for the three datasets. Numbers in parentheses represent number of results retrieved per query.

$count(sf \leftrightarrow lf)$ methods usually achieved average recall and average ranked precision higher than 0.9. Microanalysis revealed that current writing styles tend to associate the short and long forms together somewhere in the text. The count-based methods exploit this phenomenon to the benefit of the linkage task.

As noted, the count-based methods with snippets tend to produce the best performance, particularly $count(sf \leftrightarrow lf)$. Interestingly, $count(sf \leftarrow lf)$ is significantly better than $count(sf \rightarrow lf)$, though they differ only in direction. For $count(sf \rightarrow lf)$, many short form queries have no snippets containing the corresponding long forms, and some short forms are common words, leading to irrelevant results, affecting both precision and recall. Examples include "STEP" for "Symposium on Operating Systems Principles" and "ZUM" ("to the" in German) for "Z Users Conference". On the other hand, long forms make more informative multi-word queries, and often appear where short forms are defined. Finally, $count(sf \leftrightarrow lf)$ combines the best of the two unidirectional methods, and gives the best performance at the expense of doubling the number of queries. This is because $count(sf \leftarrow lf)$ is often able to obtain relevant information when $count(sf \rightarrow lf)$ misses out.

The Schwartz and Hearst algorithm generally did not perform very well, and is one of the worst performing algorithms for the DBLP and NASDAQ datasets. This is not surprising because many long form candidates share many common letters as the short form. Therefore, Schwartz and Hearst is unable to distinguish between them. On the other hand, this algorithm performs better on the GENOMES dataset because this dataset is much smaller and hence fewer candidates to choose from. However, its recall suffers because it is unable to match when the short form contains a letter that is not found in the long form. Also, the greedy algorithm sometimes gives only a partial match to the correct long form (*e.g.*, for "AAAI", only the last four words of "American Association for Artificial Intelligence" is matched). Such a weakness suggests that the given data itself lacks the required context to solve the problem, hence we need to obtain Web information to perform the matching.

The Sahami and Heilman algorithm is a strong contender for the top spot, but in all

three datasets, $count(sf \leftrightarrow lf)$ always wins by at least a small margin, for both snippets as well as web pages. While both algorithms are effective in linking short forms to long forms, the processing required to compute $count(sf \leftrightarrow lf)$ is both simpler and faster. The main difference between their performance comes from those cases where there are very few search engine results containing both the short form and its correct corresponding long form: $count(sf \leftrightarrow lf)$ tends to get it right while Sahami and Heilman finds all web pages are almost equally dissimilar.

IHF works well when the web pages of a matching short form and long form come from common sources, *i.e.*, share common domains or hostnames. This is somewhat true for the DBLP dataset, therefore using only the URLs alone gives a fairly competitive algorithm. However, for the NASDAQ dataset, some web sites aggregate information for almost all of the stock symbols, therefore the common domains or hostnames have little discriminating power and make IHF ineffective. In all cases, using hostnames is slightly better than using domain names.

In practically all cases, obtaining 20 results gives better performance than obtaining only 10 results, which is not surprising. Also, the performance of the different methods are comparable and show similar trends. What is more interesting, is that using web pages is not as good as using snippets for most of the methods, particularly for the GENOMES dataset. However, the performance are also comparable and show similar trends. This is likely because web pages contain a lot of noisy information, *e.g.*, many web pages for the query "Advances in Data Base Theory" contain "AAAI" in navigation bars and external links. Hence, snippets might be better as passage retrieval is already done. This means that the extra step of downloading the actual web pages, which takes up additional time and bandwidth, is not very useful as utilized in my current scheme.

In summary, my $count(sf \leftrightarrow lf)$ method on snippets outperforms the other methods in terms of accuracy.

3.3.6 Conclusion and Discussion

The presence of both short forms and long forms in record fields poses unique challenges to various searching and record linkage tasks. Approximate string matching does not tend to work well due to drastic differences in the application scenario. My work formalizes the use of search engines as the two facets of query formulation and search engine evidence analysis. In formulation, I proposed the use of inverted and bidirectional non-conjunctive queries. In evidence analysis, I surveyed techniques using gathered URLs and snippets, as well as downloaded web pages. My proposed count-based methods, particularly $count(sf \leftrightarrow lf)$, is found to be the most effective in linking short forms to long forms. Also, I found that using snippets outperforms using entire web pages, mainly because passage retrieval is already done to produce snippets so that noisy information is usually already excluded from the snippets. However, it might be possible to achieve even better results by applying more advanced information extraction techniques on either the snippets or the actual web pages, although this might mean increased processing time.

In any web-based method, including mine, time consuming search engine queries and/or web page downloads are required. In subsequent chapters, I remedy this drawback by looking into methods that can decrease the number of queries or downloads, yet maintain linkage performance. Firstly, Chapter 4 will consider how the count-based method can be combined with a faster but weaker query probing method in an adaptive combination framework to achieve this goal for this short form to long form linkage problem. Chapter 6 then generalizes the adaptive combination framework further into a resource acquisition framework which can be applied for a very wide variety of problems.

3.4 Disambiguation of Names in Web People Search

A person name can refer to multiple different individuals. Thus, a web search for a particular person name typically returns the results describing the different individuals

all mixed up in a single list. This is the problem addressed by the Web People Search Task in SemEval 2007 [Artiles et al., 2007]. In this task, a system receives as input a person name, as well as a set of up to 100 web pages retrieved from the Yahoo! search engine using that person name as the query. The task is then to cluster the input web pages such that each cluster contains belongs to a namesake. For the SemEval task, the training data consists of 32 names from the US Census, 7 names from Wikipedia, and 10 names from the ECDL 2006 conference, and the test data consists of 10 names from Wikipedia, 10 names from the ACL 2006 conference, and 10 names from the US Census. This section gives a very brief overview of our collaboration with fellow researchers from the Pennsylvania State University, which resulted in our PSNUS system which participated in the Web People Search Task.

If we were to manually disambiguate these mentions of the person name in the given web pages, we would usually look for information such as his or her affiliations, associations, and titles. In other words, we are looking for information contained in named entities. As such, for this inaugural competition, we treated this problem as a hard clustering problem, and submitted a run from a very simple system. For each web page, we extracted named entities, and then formed a feature vector consisting of tokens from the named entities, with TF-IDF as the feature values. Our clustering algorithm is the standard single-link hierarchical agglomerative clustering, using cosine similarity as the similarity measure between feature vectors. From the training data, we empirically set the similarity cutoff to be 0.2. More details in the methods and experimention can be found in our PSNUS task description paper [Elmacioglu et al., 2007b].

Much to our surprise, our system is able to achieve purity of 0.73, inverse purity of 0.82, $F_{\alpha=0.5}$ of 0.75, and $F_{\alpha=0.5}$ of 0.78. Based on the $F_{\alpha=0.5}$ scores, our system is placed third out of the sixteen participating systems. We also observed that the top two systems, [Chen and Martin, 2007] and [Popescu and Magnini, 2007], have both also utilized information from named entities, and the top three systems have F scores that are rather tightly clustered compared to the remainder of the systems. Therefore, our conclusion is that named entities can be an important source of information for the Web

People Search Task.

The simplicity of our system allowed it to be easily reproduced, and hence it can serve as a baseline for the disambiguation task. Indeed, our system is used as a baseline in [Kalashnikov et al., 2008].

3.5 Conclusion

In many record matching problems, there is a lack of context in the input data that makes it difficult or even impossible for humans to manually perform the matching task with sufficient certainty. This argues for the need to acquire additional resources, particularly web-based resources where large amounts of information is available and searchable through search engines. Through three different problems serving as concrete examples, I have demonstrated that acquiring web-based resources can benefit record matching tasks, fulfilling the first goal of this thesis. However, web-based resources are costly to acquire in terms of running time, and hence the second and larger goal of this thesis is to propose an algorithm for selective acquisition of web based resources for record matching tasks. Having shown that the premise of the second goal is true, I turn to the selective acquisition aspect in the subsequent chapters of this thesis. The main contribution of this thesis is in Chapter 6, where I propose a framework for hierarchical cost-sensitive web resource acquisition.

Chapter 4

A Framework for Adaptively Combining Two Methods for Record Matching

4.1 Introduction

In this chapter, I present a framework for adaptively combining two methods for record matching, and apply it to the problem of linking short forms to long forms. Generally speaking, adaptive methods combine two or more base methods to obtain the better aspect of each. For example, one may combine a method with high precision with another method that is high in recall to obtain a method that has both high precision and high recall, or one may combine a method that has high accuracy with another method that executes faster to achieve a combined method that is more balanced in accuracy and running time. Adaptive methods have been adopted in various problems such as data integration (*e.g.*, [Ng et al., 1999; Zhu et al., 2004]), duplicate detection (*e.g.*, [Bilenko and Mooney, 2003]), blocking (*e.g.*, [Bilenko et al., 2006; Gu and Baxter, 2004]), and determining when to search or when to crawl for text-centric tasks. [Ipeirotis et al., 2006].

In Chapter 3, a number of web-based methods for record matching have been proposed. However, any web-based method requires slow and sometimes unpredictable network accesses. Furthermore, search engines impose both rate limiting and daily limits on the number of queries one can make; *e.g.*, as of 2009 June, Google SOAP Search API limits one to 1000 queries in any 24-hour window, while Yahoo! Search API limits

Algorithm 4.1 Adaptively combining a weaker method with a stronger method.

Input: a weaker method M_w, a stronger method M_s, and a heuristic H

for each $sf \in SF$ **do**
  for each $lf \in LF$ **do**
    compute $score_{sf}(lf)$ using M_w
  if H determines that scores by M_w gives a poor ranking of long forms **then**
    for each $lf \in LF$ **do**
      compute $score_{sf}(lf)$ using M_s
    return scores computed by M_s
  else
    return scores computed by M_w

one to 5000 queries per day. Therefore, it is highly desirable for any web-based methods to reduce the amount of web accesses needed to solve the problem. In this chapter, I continue to ground my evaluation of the methods explored in this chapter with the application of linking short forms to long forms. While my count-based method as proposed in Section 3.3 is effective in linking short forms to long forms, it may be further improved by reducing the number of search engine queries required. Therefore, in this chapter, I first propose a query probing algorithm that uses significantly fewer queries. This query probing method is highly precise, but suffers from low recall which makes the algorithm unsuitable to be used by itself. However, by using my framework to adaptively combine the more accurate but slower count-based method with the query probing method, I demonstrate a combined method that uses much fewer queries than the count-based method with no appreciable change in the linkage performance.

Previous works have proposed ensemble methods to entity resolution, such as [Zhao and Ram, 2005] and [Chen et al., 2009], in which the multiple entity resolution algorithms are combined using techniques such as stacking or boosting to improve data quality. However, unlike such works, I do not merely focus on the accuracy of the combined algorithm. Instead, my aim is to reduce the number of search engine queries and hence the running time, while maintaining the task performance.

4.2 Adaptive Combination

For the record matching application of linking short forms to long forms, we can

adaptively combine a more accurate but slower method M_s with a weaker but faster method M_w, resulting in a combined method whose accuracy is close to that of M_s and yet runs much faster than M_s. This combination uses a general adaptive combination framework as shown in Algorithm 4.1. In the combined method, M_w is first allowed to resolve each short form to the corresponding long forms, and then M_s is applied only to those short forms whose candidate long forms appear to be incorrect, according to some heuristic H. In this way, the execution time can be reduced by making fewer calls to M_s.

Such an adaptive framework is very versatile and can apply to many kinds of methods. Here, I use a query probing method as our M_w and a count-based method as our M_s. Now I describe my query probing method.

4.2.1 Query Probing

Query probing is the automatic extraction of information from a "hidden web" database [Kautz et al., 1997] by selecting suitable terms, known as *query probes*, to query [Gravano et al., 2003]. This approach has been used to obtain language models [Callan and Connell, 2001], and to estimate word frequency in different languages [Grefenstette and Nioche, 2000]. In this context, we can issue query probes to a search engine to derive approximate Web statistics, and use it to reduce the number of queries.

As an example, consider three conferences — "Joint Conference on Digital Libraries", "European Conference on Digital Libraries" and "Digital Libraries" — and their respective short forms: "JCDL", "ECDL" and "DL". Normally, we will query all three short forms to obtain $count(sf \rightarrow lf)$, query all three long forms to obtain $count(sf \leftarrow lf)$, and query all six short and long forms to obtain $count(sf \leftrightarrow lf)$. However, it can be observed that many long forms share common n-grams. Here, *n-gram* is used at the token level. Hence, a 3-gram refers to a three-token subsequence. As the conference names' long forms all contain the common 2-gram "digital libraries", I can use a single query probe "digital libraries", together with the domain-specific keywords that were used in Section 3.3, to yield results for all three conferences. In the

Algorithm 4.2 Query probing.

1: $NG =$ set of n-grams contained by the long forms in LF
2: $D = \emptyset$
3: **for each** $ng \in NG$ **do**
4: **if** number of long forms in LF containing the n-gram $ng \geq min\_freq$ **then**
5: $D = D \cup SearchEngineTop(ng, k_p)$
6: **for each** $sf \in SF$ **do**
7: **for each** $lf \in LF$ **do**
8: $count_p(sf, lf) =$ number of results in D containing both sf and lf

top-10 result snippets conveniently contain both the long forms and their corresponding short forms for all the three conferences. Thus, when compared to $count(sf \rightarrow lf)$, $count(sf \leftarrow lf)$ and $count(sf \leftrightarrow lf)$, query probing can save two, two, and five queries, respectively.

With this observation, I devised a query probing algorithm that probes the search engine with n-grams that occur in at least min_freq of the long forms. My query probing algorithm is shown in Algorithm 4.2. I use the top-k_p search engine results to obtain $count_p(sf, lf)$, the number of results that contain both sf and lf, and use it as a scoring function.

4.2.2 Adaptively Combining Query Probing with Count-based Methods

The main weakness of the query probing scoring function, $count_p(sf, lf)$, is that typically some of the short forms sf have no candidate long forms, *i.e.*, $count_p(sf, lf) = 0$ for all long forms lf, and vice versa. However, for short forms with candidate long forms, query probing is able to save a significant number of search engine queries.

Therefore, I propose adaptively combining query probing with a count-based method using the proposed framework of Algorithm 4.1. In other words, I use $count_p(sf, lf)$ as my weaker method M_w and use one of $count(sf \rightarrow lf)$, $count(sf \leftarrow lf)$, or $count(sf \leftrightarrow lf)$ as my stronger method M_s. The heuristic H I use is:

If there is no long form lf in LF with $count_p(sf, lf) > 0$, then the stronger method M_s is applied.

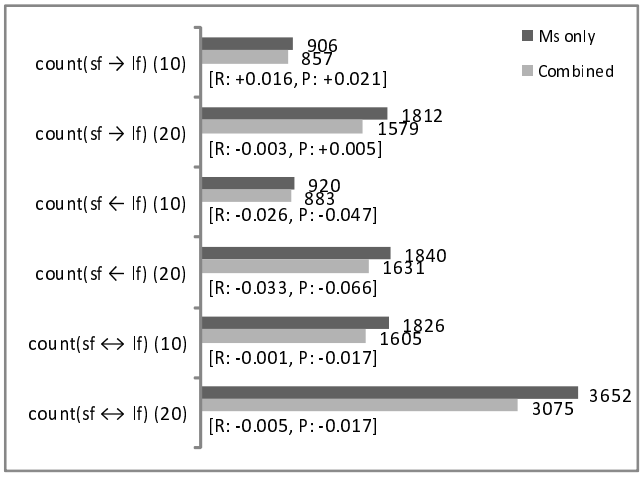
4.3 Evaluation

I compare adaptively combining M_s with query probing against using only M_s alone. For consistency with Section 3.3, I maintain my choice of Google SOAP Search API as the search engine; use DBLP, NASDAQ, and GENOMES as my three evaluation datasets; and use average recall and average ranked precision as my evaluation metrics. In addition, I also use the number of search engine calls to evaluate the effectiveness of our adaptive combination of query probing and count-based methods. As each search engine call returns 10 results, two calls are needed for 20 results. This is especially so when most search engine providers impose a daily limit on the number of calls one can issue through their APIs.

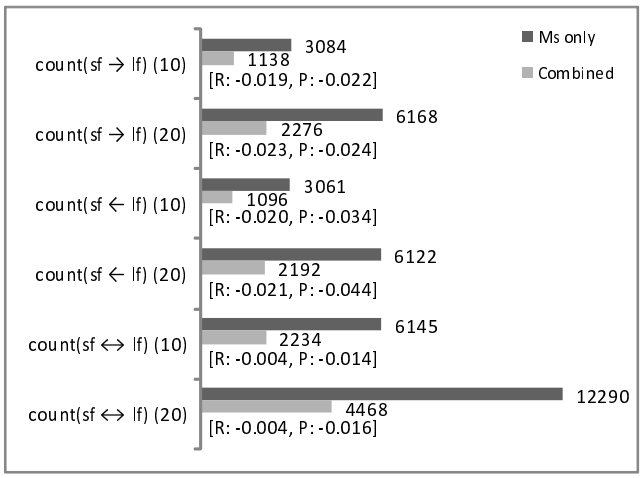
I experimented with various parameter settings, and found that usually $k_p = 10$ is sufficient to provide the largest decrease in the total number of search engine calls needed. The results for selected values of n and min_freq for each dataset are shown in Figure 4.1. Compared to the count-based methods alone, my adaptively combined methods can reduce the number of search engine calls in practically all cases, with better savings when compared against the stronger count-based method using 20 snippets. The largest dataset, NASDAQ, also gave the most significant savings of 18.0% to 31.6%. The change in average recall and average ranked precision is relatively small, with decreases of up to 0.036 and 0.066, respectively. Task performance actually improved in a few cases. Other query probing parameter settings also gave similarly insignificant changes in average recall and average ranked precision. Therefore, I conclude that combining query probing and the count-based method can reduce the number of search engine calls significantly while maintaining task performance, making the combined method applicable towards large record linkage problems.

4.4 Discussion

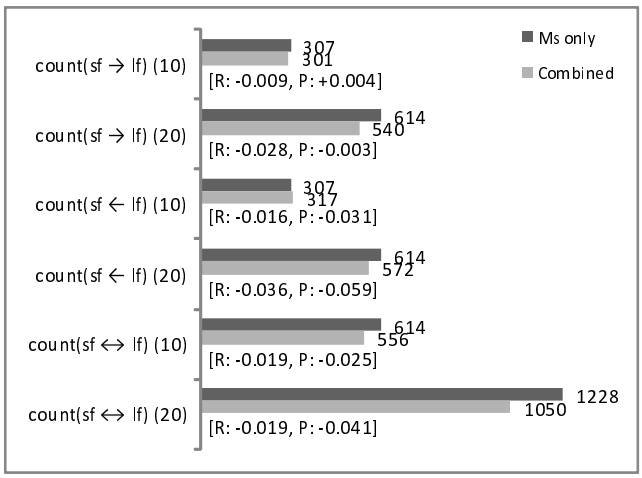
I have shown that the adaptive combination framework can effectively combine the count-based method and the query probing method in the example application of link-



(a) DBLP ($n = 3$, $min\_freq = 3$)



(b) NASDAQ ($n = 2$, $min\_freq = 2$)



(c) GENOMES ($n = 3$, $min\_freq = 2$)

Figure 4.1: Number of search engine calls using M_s alone and adaptively combining M_s with query probing, where numbers in parentheses represent the number of snippets retrieved per query. The parameters n-grams and min_freq are indicated in the captions, and $k_p = 10$. The change in average recall (R) and average ranked precision (P) are indicated in square brackets.

ing short forms to long forms. Along the same lines, Elmacioglu [Elmacioglu, 2008] also reported success in using the adaptive combination method to combine two other methods for the same problem, namely host overlap and web page similarity:

- **Host overlap.** Similar to IHF weight, this method also uses host information. We can obtain the hostnames of the top-10 search engine results for the short form sf and the long form lf respectively. Then the Jaccard similarity between the sf hostnames and the lf hostnames can be used as a scoring function.

- **Web page similarity.** For each query q, we download the web pages at the top-10 results, and concatenate them into a single text document D_q. For a short form sf and a long form lf, the standard tf-idf cosine similarity between D_{sf} and D_{lf} is the scoring function.

Elmacioglu performed his evaluation on the DBLP dataset and reported that host overlap achieves a recall of 0.641 and a ranked precision of 0.545, while web page similarity achieves a recall of 0.802 and ranked precision of 0.714. Using host overlap as the weaker method M_w and web page similarity as the stronger method M_s, he experimented with five different heuristics and selected the best heuristic that gave the best results:

> Let r_1 and r_2 be the top two results for a short form sf. Let h be the common URL between r_1 and sf with the highest IHF weight. If IHF(h) is below a threshold, then web page similarity is used instead.

Using this heuristic, his combined method achieves a recall of 0.788 and a ranked precision of 0.719. The recall of the combined method is close to web page similarity, while its ranked precision beats web page similarity, and the execution time of the combined method is 43.8% faster than web page similarity. Although the reported recall and ranked precision is not better than my count-based methods, it still shows the effectiveness of the adaptive combination framework.

In general, the adaptive combination framework is suitable for combining a more costly accurate stronger method with a cheaper weaker method as long as the cheaper

method produces very few false matches. This premise is satisfied in a very large class of problems, including the short to long form linkage problem involving web resources studied in this chapter.

However, the adaptive combination framework is not without its drawbacks.

- Firstly, it may be difficult to select a good heuristic that decides whether the weaker method M_w actually produces a reasonably good ranking of the long forms. Also, many heuristics involve some cutoff threshold, therefore finding the optimal threshold can be a problem.

- Secondly, some heuristics are specific to the methods that are applied to the specific linkage task, and cannot easily be generalized to arbitrary linkage tasks.

- Thirdly, it is not clear how the adaptive combination framework can be used to combine more than two methods. In particular, selecting the order of application of multiple methods is a problem in itself.

These issues are clearly evident in Elmacioglu's work, and also in my work to a lesser extent. In Chapter 6, I propose a resource acquisition framework that addresses these kinds of issues in a more principled manner. My resource acquisition framework is targeted at general linkage problems, and can combine an arbitrary number of methods. In my framework, methods can be seen as resources, and then the question becomes which resources to acquire.

Chapter 5

Cost-sensitive Attribute Value Acquisition for Support Vector Machines

5.1 Introduction

In this chapter, I consider the supervised classification problem. In many real life supervised classification problems, it is possible to control the quality of the training data but not that of the testing data. In particular, test instances often contain missing attribute values that hinders the classification task, but missing values can be acquired at some cost. As an example, consider the medical diagnosis of a patient. The doctor can order various tests such as blood tests and chest X-rays to acquire more information about the patient to make a more informed diagnosis, but each test comes at some monetary cost. As another example, a supervised classifier may be used to classify whether two records are a match, based on known attribute values derived from the record fields, as well as unknown attribute values to be obtained from the Web by forming queries using strings found inside the records. Acquiring these missing attribute values involves running time costs associated with the Web queries. In such *cost-sensitive attribute value acquisition* or *cost-sensitive acquisition* problems, the objective is to acquire attribute values in such a way to minimize both attribute acquisition and misclassification costs of the test instances.

Here, I propose a new, iterative framework for cost-sensitive attribute value acquisition for the support vector machine based classification. To the best of my knowl-

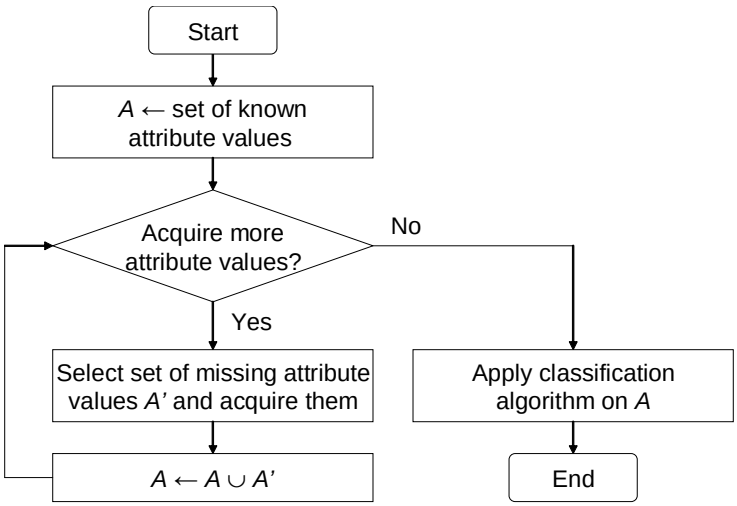


Figure 5.1: General iterative framework for solving classification problems with missing attribute values.

edge, there has been no work on extending support vector machine classification to handle cost-sensitive attribute value acquisition, although prior work has investigated it in other learning models such as decision trees. This is an important contribution as support vector machines show consistently good performance over a wide range of problem classes.

My proposed framework is illustrated in Figure 5.1: a process that acquires attribute values in an iterative manner. Starting with a (possibly empty) set of known attribute values A, I acquire missing attribute values A' and add them iteratively to A. Once a termination condition is satisfied, no more attributes are acquired and a final classification is performed on A. To apply this framework, I need to specify three parts:

1. The classification algorithm.
2. An algorithm for selecting the set A' to acquire.
3. The termination condition.

This acquisition framework generalizes most related work in Section 5.2 and trivially embodies algorithms that do not consider attribute value acquisition – simply by setting the termination condition to be always be true and not looping at all. In this work, I will assume that the classification algorithm is the support vector machine, and focus on the second and third parts of the framework.

To keep the exposition easy to follow, I have purposefully taken on several assumptions that are minor and easily removed from my cost-sensitive acquisition algorithm. First, I assume that this framework is applied separately for each test instance. Second, I apply my algorithm in the context of the standard two-class support vector machine. Finally, I also assume that the attribute acquisition and misclassification costs are known *a priori*, but note that I do not assume uniform costs for misclassifying instances of different classes. Following [Ling et al., 2006], my objective is to minimize the sum of the acquisition and misclassification costs.

My contributions in this chapter are as follows:

1. I provide a realization of this framework for support vector machines, something that has not yet been seen in the published literature as far as I know.

2. I develop an approach for estimating the certainty and misclassification cost of a test instance with missing attribute values, before and after acquiring an *arbitrary* subset of these missing values.

3. My approach can be applied to *arbitrary* kernels, unlike previous work on linear classifiers that deals with missing attribute values in test instances.

In particular, I would like to highlight the second contribution. Firstly, it makes it easy to come out with a cost-sensitive acquisition algorithm for the support vector machine. Just as importantly, it facilitates the construction of benefit functions in Chapter 7, which is required to apply the resource acquisition framework of Chapter 6.

I first review related work in Section 5.2. Section 5.3 is a preliminaries section that gives a brief overview on support vector machines, as well as introduce the notation I will use for the remainder of this work. To build up my cost-sensitive algorithm, I first explain in Section 5.4 how to compute expected misclassification costs for a test instance with missing attribute values as-is, as well as the same instance with a subset of its missing attribute values acquired. Armed with these expected misclassification costs, I propose a cost-sensitive acquisition algorithm for the support vector machine classifier in Section 5.5 that specifies the second and third parts that are needed in the iterative

acquisition framework. I experimentally validate the effectiveness of my acquisition algorithm in Section 5.6, before concluding this chapter in Section 5.7.

5.2 Related Work

There is a significant amount of related work dealing with classification problems involving missing values in data. [Turney, 2000] described various kinds of costs, of which the most pertinent are attribute acquisition costs and (mis)classification costs. For a problem where correct classifications incur a cost as well, [Greiner et al., 2002] showed that it can be transformed into an equivalent problem where correct classifications incur zero cost, hence in this work I assume that no cost is incurred when a test instance is classified correctly. [Elkan, 2001] argued that *cost-sensitive classification*, where unequal misclassification costs are assigned for different classes, should be applied in situations such as when the class distribution is imbalanced. [Ling et al., 2004] extended the notion of cost-sensitive classification to include acquisition of missing attribute values in test instances, resulting in the problem I consider here. In this section, I shall focus more on algorithms handling missing values in test instances, as well as algorithms for support vector machines.

[Saar-Tsechansky and Provost, 2007] treated the underlying classifier as a black box, and proposed training a classification model on each (selected) subset of the input attributes. Test instances would then be classified using the models corresponding to its known attribute values. However, this method does not consider the possibility of acquiring missing attribute values. The active feature-value acquisition framework of [Saar-Tsechansky et al., 2009] may be seen as an extension of [Saar-Tsechansky and Provost, 2007], except that it is targeted at missing values in training data. A special case of active feature-value acquisition is applied to selective acquisition of missing attribute values in test instances [Kanani and Melville, 2008]. [Ji and Carin, 2007] formulated a partial observable Markov decision process to deal with cost-sensitive attribute value acquisition when the classifier is a hidden Markov model. For Naïve Bayes classi-

fiers, [Chai et al., 2004] proposed classifying test instances by simply ignoring missing attribute values.

Cost-sensitive decision tree classifiers have received the most attention by researchers. [Greiner et al., 2002] proposed an algorithm for building a cost-sensitive decision tree and analyzed its theoretical properties under a probably approximately correct learning framework. [Turney, 1995] proposed using genetic algorithms, while [Zubek and Dietterich, 2002] uses Markov decision processes to generate candidate decision trees. On the other hand, [Davis et al., 2006] and [Ling et al., 2004] proposed the building of a single decision tree from the training data. [Ling et al., 2006] proposed a lazy tree formulation that improved on [Ling et al., 2004] in decision tree construction. Some of these methods generate many candidate models and have efficiency issues, while others generate only a single classification model and are efficient.

Relatively few works deal with missing values in the case of support vector machines and other linear classifiers. Training instances of a support vector machine can be weighted by their misclassification costs such that the trained support vector machine is sensitive to different misclassification costs of different classes [Osuna et al., 1997]. [Smola and Vishwanathan, 2005] and [Pelckmans et al., 2005] both builds (kernelized) support vector machines that handles missing values in training data (as opposed to test data), and does not consider the option of acquiring missing values. [Globerson and Roweis, 2006] and [Dekel and Shamir, 2008] proposed linear classifiers that are robust to missing attribute values during classification time, with [Dekel and Shamir, 2008] using a linear programming formulation which improves on [Globerson and Roweis, 2006] and is the current state-of-the-art. However, both linear classifiers 1) require the number of missing attribute values to be supplied during training, 2) does not consider acquisition of missing attribute values, and 3) cannot easily be kernelized. To the best of my knowledge, there is no support vector machine classifier that also considers the cost-sensitive acquisition of missing attribute values during classification time. My aim is to fill in this gap by proposing such an algorithm for the support vector machine, which can be applied to any kernel satisfying Mercer's condition.

5.3 Preliminaries and Notation

In this section, I first give a brief overview of support vector machines and introduce my notation at the same time. For a more detailed introduction to support vector machines, I refer the reader to tutorials such as [Osuna et al., 1997] and [Burges, 1998].

5.3.1 Background on Support Vector Machines

Let us first ignore the issue of missing attribute values and assume all attribute values are known. Consider a classification problem where instances have n numeric attributes. A kernel function (that satisfies the Mercer's condition) is a function $K : \mathbb{R}^n \times \mathbb{R}^n \to R$ that can be expressed as $K(\mathbf{x}_i, \mathbf{x}) = \langle \mathbf{\Phi}(\mathbf{x}_i), \mathbf{\Phi}(\mathbf{x}) \rangle$, where $\mathbf{\Phi} : \mathbb{R}^n \to \mathcal{F}$ is a feature map from the attribute space $\mathbb{R}^n$ to some feature space $\mathcal{F}$. For a user-selected kernel function $K(\cdot, \cdot)$, a support vector machine is a linear classifier in the feature space, *i.e.*, it maps an instance $\mathbf{x} \in \mathbb{R}^n$ to a class in $\{-1, +1\}$ via $y = \text{sign}[f(\mathbf{x})]$, where the decision function $f(\cdot)$ is defined as:

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{\Phi}(\mathbf{x}) \rangle + b \tag{5.1}$$

Training a support vector machine involves learning the weight vector $\mathbf{w} \in \mathcal{F}$ and bias $b \in \mathbb{R}$. Given training data $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where each $\mathbf{x}_i$ is a training instance and y_i being its class, a support vector machine can be trained by solving the following (primal) quadratic optimization problem:

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{i=1}^{N} \varepsilon_i \\ \text{Subject to} \quad & \forall i \in \{1, \ldots, N\} \quad y_i(\langle \mathbf{w}, \mathbf{\Phi}(\mathbf{x}_i) \rangle + b) \geq 1 - \varepsilon_i \\ & \forall i \in \{1, \ldots, N\} \quad \varepsilon_i \geq 0 \end{aligned} \tag{5.2}$$

Geometrically, a support vector machine is designed to maximize the margin $\rho = \frac{2}{\|\mathbf{w}\|_2}$. The constant C specifies the amount of misclassifications allowed for the training in-

stances, and can either be user-specified or determined through cross-validation. Instead of solving the above primal problem directly, it is often more convenient to solve the dual optimization problem instead:

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \mathbf{\Phi}(\mathbf{x}_i), \mathbf{\Phi}(\mathbf{x}_j) \rangle \\ \text{Subject to} \quad & \forall i \in \{1, \ldots, N\} \quad 0 \leq \alpha_i \leq C \\ & \sum_{i=1}^{N} \alpha_i y_i = 0 \end{aligned} \tag{5.3}$$

The optimal solutions of the primal and dual problems are related by:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{\Phi}(\mathbf{x}_i) \tag{5.4}$$

Commonly used kernels include the following:

- Linear kernel: $K(\mathbf{x}_i, \mathbf{x}) = \langle \mathbf{x}_i, \mathbf{x} \rangle$.
- Polynomial kernel: $K(\mathbf{x}_i, \mathbf{x}) = (\langle \mathbf{x}_i, \mathbf{x} \rangle + 1)^d$.
- Radial basis function (RBF) kernel: $K(\mathbf{x}'_i, \mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}'_i - \mathbf{x}'\|_2^2}{2\sigma^2}\right)$.

The linear kernel corresponds to the identity feature map $\mathbf{\Phi}(\mathbf{x}) = \mathbf{x}$. However, when the training data is non-linearly separable, a nonlinear feature map $\mathbf{\Phi}(\cdot)$ whose feature space $\mathcal{F}$ is a high or infinite dimensional space is employed. Examples of nonlinear kernel includes the polynomial kernel and the RBF kernel. In such cases, often the kernel function $K(\cdot, \cdot)$ is specified directly, leaving $\mathbf{\Phi}(\cdot)$ and $\mathcal{F}$ implicitly defined. This leaves the weight vector $\mathbf{w}$ implicitly defined as well. However, this is not a difficulty because we can train the support vector machine by solving the dual formulation (after substituting $\langle \mathbf{\Phi}(\mathbf{x}_i), \mathbf{\Phi}(\mathbf{x}_j) \rangle$ with $K(\mathbf{x}_i, \mathbf{x}_j)$). At the same time, Equation 5.4 implies that we can express the decision function as:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \tag{5.5}$$

Also, Equation 5.4 implies that:

$$\|\mathbf{w}\|_2^2 = \langle \mathbf{w}, \mathbf{w} \rangle = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{5.6}$$

which means that the margin ρ can be computed as usual.

5.3.2 Posterior Probability of Classification

A trained support vector machine can be extended to provide the posterior probability that a test instance belongs to a class [Platt, 2000]. Let $s_+(\mathbf{x})$ and $s_-(\mathbf{x})$ be the probabilities that a test instance $\mathbf{x}$ belongs to the positive and negative class respectively. Then we can fit a sigmoid function [Platt, 2000] as follows:

$$s_+(\mathbf{x}) = \frac{1}{1 + \exp(Af(\mathbf{x}) + B)} \tag{5.7}$$

where the parameters A and B are to be learned from the training data. We can then compute $s_-(\mathbf{x}) = 1 - s_+(\mathbf{x})$.

5.3.3 Classifying an Instance with Missing Attribute Values

I now consider the classification of a test instance $\mathbf{x} \in \mathbb{R}^n$ with missing attribute values and without acquiring any of them. The attribute values of $\mathbf{x}$ can be partitioned into a set $\overline{M}(\mathbf{x})$ of known (or observed) attribute values and a set $M(\mathbf{x})$ of missing attribute values, such that $\{x_1, \ldots, x_n\}$ is the disjoint union of $\overline{M}(\mathbf{x})$ and $M(\mathbf{x})$. I assume that, prior to classification time, a set of fixed constants $x'_1, \ldots, x'_n$ is already defined and computed. When classifying a test instance $\mathbf{x}$ with some of its attribute values missing, and without acquiring any of its missing values, we can impute (*i.e.*, replace) each missing attribute value $x_k \in M(\mathbf{x})$ with the constant x'_k before computing $f(\mathbf{x})$. Here, I assume that these fixed constants are independent of the test instance $\mathbf{x}$, but I do not impose on the exact semantics of these fixed constants. Typically, x'_k can be the mean value of the kth attribute of the training instances, or simply be the zero value. This

simple approach is widely used in related work (*e.g.*, [Globerson and Roweis, 2006] and [Dekel and Shamir, 2008]), as well as in support vector machine implementations (*e.g.*, the sequential minimal optimization (SMO) classifier of Weka [Witten and Frank, 2005]). Although imputing missing attribute values with fixed constants is not necessarily the best approach, imputation methods is not the focus of this work and I leave improvements in this area as future work. Thus, I simply used mean value imputation in my experiments. Similarly, for the computation of the sigmoid function, any missing attribute values of an instance are also replaced by the fixed constants before computing its posterior probability.

5.4 Computing Expected Misclassification Costs

Consider a test instance $\mathbf{x}$ and $A' \subseteq M(\mathbf{x})$, a subset of its missing attribute values. Let $\mathbf{x} + A'$ denote the test instance $\mathbf{x}$ with A' acquired. I define the following:

- $E[mc(\mathbf{x})]$ is the expected misclassification cost of $\mathbf{x}$.
- $E[mc(\mathbf{x} + A')]$ is the expected misclassification cost of $\mathbf{x}$ after acquiring A'.

The aim of this section is to give a way for computing the quantities $E[mc(\mathbf{x})]$ and $E[mc(\mathbf{x} + A')]$ for arbitrary $\mathbf{x}$ and A'. These quantities are two of the key quantities that drive my cost-sensitive acquisition algorithm, which I describe in the next section.

In this section, I first construct the computation of $E[mc(\mathbf{x})]$ and $E[mc(\mathbf{x} + A')]$ for an arbitrary kernel. Then, I give an interpretation for the linear kernel, and explain how these quantities can be computed efficiently in $O(n)$ time, linear in the number of attributes.

I first define $\mathbf{w}'(\mathbf{x})$, the weight vector $\mathbf{w}$ modified by the missing values of $\mathbf{x}$.

Definition 5.1 (Modified weight vector). Let $\mathbf{x}$ be a test instance. Let $\mathbf{x}'_i$ be the training instance $\mathbf{x}_i$ with the attributes that are missing in the test instance $\mathbf{x}$ replaced by fixed constants, *i.e.*,

$$x'_{ik} = \begin{cases} x_{ik} & \text{if } x_k \in \overline{M}(\mathbf{x}) \\ x'_{ik} & \text{otherwise} \end{cases} \tag{5.8}$$

The *modified weight vector* $\mathbf{w}'(\mathbf{x})$ is defined such that:

$$\|\mathbf{w}'(\mathbf{x})\|_2^2 = \langle \mathbf{w}'(\mathbf{x}), \mathbf{w}'(\mathbf{x}) \rangle = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}'_i, \mathbf{x}'_j) \tag{5.9}$$

For the instance $\mathbf{x} + A'$, I define its modified weight vector $\mathbf{w}'(\mathbf{x} + A')$ by taking $\overline{M}(\mathbf{x} + A') = \overline{M}(\mathbf{x}) \cup A'$. Also, notice that Equation 5.9 is defined in a manner that is analogous to Equation 5.6.

The quantity $\|\mathbf{w}'(\mathbf{x})\|_2$ can be interpreted as the amount of information that is remaining from $\|\mathbf{w}'\|_2$ after taking into account of information that is lost due to the missing attribute values. This aspect will be made obvious when an interpretation for $\mathbf{w}'(\mathbf{x})$ for the linear kernel is made later.

Definition 5.2 (Uncertainty factor of test instance). The *uncertainty factor* for a test instance $\mathbf{x}$ is defined by:

$$u(\mathbf{x}) = \frac{\|\mathbf{w}'(\mathbf{x})\|_2}{\|\mathbf{w}\|_2} \tag{5.10}$$

Empirically, I found that with very high probability, $\|\mathbf{w}'(\mathbf{x})\|_2 \leq \|\mathbf{w}'\|_2$ is satisfied, which implies $0 \leq u(\mathbf{x}) \leq 1$. These inequalities are always satisfied when the linear kernel is used.

I now explain the computation of the certainty of a test instance with missing attribute values. Let mc_+ and mc_- be the misclassification cost for misclassifying a positive and negative instance, respectively. Suppose the support vector machine classifies a test instance $\mathbf{x}$ to be positive, *i.e.*, $f(\mathbf{x}) \geq 0$. Then according to the sigmoid function of Equation 5.7, the posterior probability of $\mathbf{x}$ belonging to the positive class is $s_+(\mathbf{x})$. However, the computation of $s_+(\mathbf{x})$ does not take missing attribute values into account. Hence, the certainty of the classification is reduced by the uncertainty factor $u(\mathbf{x})$, *i.e.*, meaning that the certainty of is now $p_+(\mathbf{x}) = u(\mathbf{x}) \cdot s_+(\mathbf{x})$. The certainty of a test instance that is classified as negative is derived in a similar manner.

Definition 5.3 (Certainty of test instance). The *certainty* of a test instance $\mathbf{x}$ is de-

fined as:

$$p_+(\mathbf{x}) = u(\mathbf{x}) \cdot s_+(\mathbf{x}) \quad \text{if } f(\mathbf{x}) \geq 0 \tag{5.11}$$

$$p_-(\mathbf{x}) = u(\mathbf{x}) \cdot s_-(\mathbf{x}) \quad \text{otherwise} \tag{5.12}$$

The values of $p_+(\mathbf{x} + A')$ and $p_-(\mathbf{x} + A')$ can be computed by the above definition, with $\mathbf{x}$ substituted by $\mathbf{x} + A'$.

Using the certainty values, I now explain the computation of the expected misclassification cost of a test instance with missing attribute values. For a test instance $\mathbf{x}$ classified as positive, its uncertainty is $1 - p_+(\mathbf{x})$. Therefore, its expected misclassification cost can be computed by $(1 - p_+(\mathbf{x})) \cdot mc_-$. The expected misclassification cost for a test instance that is classified as negative is defined similarly.

Definition 5.4 (Expected misclassification cost). Let mc_+ and mc_- be the misclassification cost for misclassifying a positive and negative instance, respectively. The *expected misclassification cost* of a test instance $\mathbf{x}$ is defined as:

$$E[mc(\mathbf{x})] = \begin{cases} (1 - p_+(\mathbf{x})) \cdot mc_- & \text{if } f(\mathbf{x}) \geq 0 \\ (1 - p_-(\mathbf{x})) \cdot mc_+ & \text{otherwise} \end{cases} \tag{5.13}$$

The value of $E[mc(\mathbf{x} + A')]$ can be calculated using the above definition, with $\mathbf{x}$ substituted by $\mathbf{x} + A'$.

The inequality $\|\mathbf{w}'(\mathbf{x})\|_2 \leq \|\mathbf{w}'(\mathbf{x} + A')\|_2$ always holds for the linear kernel and holds with high probability for nonlinear kernels. If this inequality holds, then we also have $u(\mathbf{x}) \geq u(\mathbf{x} + A')$, and hence $E[mc(\mathbf{x})] \geq E[mc(\mathbf{x} + A')]$. In other words, the acquisition of missing attribute values should lead to a decrease in expected misclassification cost. By this token, the difference of $E[mc(\mathbf{x})]$ and $E[mc(\mathbf{x} + A')]$ can be exploited to determine how useful it is to acquire A'. This difference is the key element for constructing the cost-sensitive acquisition algorithm in the next section.

Earlier, I mentioned that the quantity $\|\mathbf{w}'(\mathbf{x})\|_2$ can be interpreted as the amount of information remaining after taking into account the information that is lost from the

missing attribute values. In the following, I will make this interpretation obvious in the context of the linear kernel. As this quantity will be computed many times in the cost-sensitive acquisition algorithm, I also suggest a way to compute it efficiently when a non-linear kernel is used.

5.4.1 Modified Weight Vector for Linear Kernel

Recall that the linear kernel is $K(\mathbf{x}_i, \mathbf{x}) = \langle \mathbf{x}_i, \mathbf{x} \rangle$, whose feature map is $\mathbf{\Phi}(\mathbf{x}) = \mathbf{x}$. For a support vector machine that uses a linear kernel, its decision function can be simplified as follows:

$$f(\mathbf{x}) = \sum_{k=1}^{n} w_k x_k + b \tag{5.14}$$

If we assume that the attribute values in both training and test instances have been appropriately normalized to the same scale, then the magnitude of the weight w_k of each attribute x_k indicates the relevance or importance of that attribute. Thus, if the missing attribute values of a test instance has weights of large magnitude, then its classification would be highly uncertain.

For a test instance $\mathbf{x}$, I can define its modified weight vector explicitly as $\mathbf{w}'(\mathbf{x}) = (w'_1(\mathbf{x}), \ldots, w'_n(\mathbf{x}))$, where

$$w'_k(\mathbf{x}) = \begin{cases} w_k & \text{if } x_k \in \overline{M}(\mathbf{x}) \\ 0 & \text{otherwise} \end{cases} \tag{5.15}$$

Theorem 5.1. When the linear kernel is used, both the modified weight vectors $\mathbf{w}'(\mathbf{x})$ as defined by Equation 5.9 and Equation 5.15 compute the same value for $\|\mathbf{w}'(\mathbf{x})\|_2$, *i.e.*,

$$\|\mathbf{w}'(\mathbf{x})\|_2^2 = \sum_{k:x_k \in \overline{M}(\mathbf{x})} w_k^2 \tag{5.16}$$

Proof. When the linear kernel is used, we have that for two training instances $\mathbf{x}_i$ and $\mathbf{x}_j$, $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle = \sum_{k:x_k \in M(\mathbf{x})} x'_{ik} + \sum_{k:x_k \in \overline{M}(\mathbf{x})} x_{ik}$. Note that the first term on the right hand side is a constant.

For $\mathbf{w}'(\mathbf{x})$ as defined by Equation 5.9, we have

$$\|\mathbf{w}'(\mathbf{x})\|_2^2 = \sum_{k:x_k \in M(\mathbf{x})} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x'_{ik} x'_{jk} + \sum_{k:x_k \in \overline{M}(\mathbf{x})} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_{ik} x_{jk} \tag{5.17}$$

From the dual formulation of the support vector machine, $\sum_{i=1}^{N} \alpha_i y_i = 0$, and thus the first term on the right hand side of Equation 5.17 is zero. Also, when the linear kernel is used, we have $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$, which implies $w_k = \sum_{i=1}^{N} \alpha_i y_i x_{ik}$. Therefore, the second term on the right hand side of Equation 5.17 becomes $\sum_{k:x_k \in \overline{M}(\mathbf{x})} w_k^2$. Hence, Equation 5.16 is satisfied.

For $\mathbf{w}'(\mathbf{x})$ as defined by Equation 5.15, it is easy to see that Equation 5.16 is satisfied as well.

The above theorem states that I can compute $\|\mathbf{w}'(\mathbf{x})\|_2$ using the form of $\mathbf{w}'(\mathbf{x})$ that is explicitly defined in Equation 5.15. As this explicit form of $\mathbf{w}'(\mathbf{x})$ is formed by zeroing the weights of $\mathbf{w}$ corresponding to the missing attribute values of $\mathbf{x}$, the quantity $\|\mathbf{w}'(\mathbf{x})\|_2 = \sum_{k:x_k \in \overline{M}(\mathbf{x})} w_k^2$ has a particularly intuitive interpretation. It is less than $\|\mathbf{w}\|_2 = \sum_{k=1}^{N} w_k^2$ and represents the amount of useful information remaining after taking into account the missing attribute values of $\mathbf{x}$.

Given the explicit representation of the modified weight vector $\mathbf{w}'(\mathbf{x})$ for the linear kernel, we can see that computing $\|\mathbf{w}'(\mathbf{x})\|$, and hence $E[mc(\mathbf{x})]$, incurs $O(n)$ time linear in the number of attributes. The same can be said for the computation of $E[mc(\mathbf{x} + A')]$.

A final remark I make is that this method for computing $E[mc(\mathbf{x})]$ and $E[mc(\mathbf{x} + A')]$ can be applied to any linear classifier whose decision function is of the form Equation 5.14. It is not necessary for the linear classifier to be a support vector machine.

5.4.2 Modified Weight Vector for Nonlinear Kernel

The main challenge for the computation of $\|\mathbf{w}'(\mathbf{x})\|_2$ for a nonlinear kernel is that it requires $O(M^2 n)$ time to compute using Equation 5.6, where M is the number of support vectors in the support vector machine. This makes my proposed acquisition algorithm

of computationally expensive during classification time as many computations of expected misclassification costs may be required. This is especially so when M is large and when the test instance $\mathbf{x}$ has many missing attribute values, which are typical characteristics for large datasets. Using Equation 5.6 to compute $\|\mathbf{w}'(\mathbf{x})\|_2$ will make my proposed acquisition algorithm impractical for these cases.

One approach around the problem of computational complexity is to reduce the number of support vectors in the trained support vector machine. This approach has been studied in papers such as [Burges, 1996] [Lin and Lin, 2003], [Apolloni et al., 2004], and [Nguyen and Ho, 2005]. However, I choose to use another approach that allows us to compute $\|\mathbf{w}'(\mathbf{x})\|_2$ in $O(n)$ time, linear in the number of attributes. This approach approximates the computation of the kernel function $K(\mathbf{x}'_i, \mathbf{x}'_j)$ using "weak kernels" [Fung et al., 2007]. Let $K_k(\mathbf{x}'_i, \mathbf{x}'_j)$ be the kernel function computed using only the kth attribute of $\mathbf{x}'_i$ and $\mathbf{x}'_j$. For example, if $K(\mathbf{x}'_i, \mathbf{x}'_j)$ is the radial basis function (RBF) kernel defined by $K(\mathbf{x}'_i, \mathbf{x}'_j) = \exp\left(-\frac{\|\mathbf{x}'_i - \mathbf{x}'_j\|_2^2}{2\sigma^2}\right)$, then $K_k(\mathbf{x}'_i, \mathbf{x}'_j) = \exp\left(-\frac{\|x'_{ik} - x'_{jk}\|_2^2}{2\sigma^2}\right)$. Then I approximate the computation of $K(\mathbf{x}'_i, \mathbf{x}'_j)$ by:

$$K(\mathbf{x}'_i, \mathbf{x}'_j) \approx \sum_{k=1}^{n} \gamma_k K_k(\mathbf{x}'_i, \mathbf{x}'_j) \tag{5.18}$$

where the constants $\gamma_1, \ldots, \gamma_n$ are to be determined. Thus, Equation 5.9 becomes:

$$\|\mathbf{w}'(\mathbf{x})\|_2^2 = \langle \mathbf{w}'(\mathbf{x}), \mathbf{w}'(\mathbf{x}) \rangle \approx \sum_{k=1}^{n} \gamma_k t_k(\mathbf{x}) \tag{5.19}$$

where:

$$t_k(\mathbf{x}) = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K_k(\mathbf{x}'_i, \mathbf{x}'_j) \tag{5.20}$$

Note that each of the $t_k(\mathbf{x})$ terms can take on only one of two possible values, which depends on whether $x_k \in M(\mathbf{x})$, *i.e.*, whether the kth attribute in the test instance $\mathbf{x}$ is known or missing. This allows us to precompute these terms during training, by selecting a subset of missing patterns out of the 2^n possible patterns and computing their corresponding actual $\|\mathbf{w}'(\mathbf{x})\|_2^2$ values, and determining the values of $\gamma_1, \ldots, \gamma_n$

using a standard linear regression algorithm. Subsequently, during classification time, when I need to estimate $\|\mathbf{w}'(\mathbf{x})\|_2$ for a test instance $\mathbf{x}$ with missing values, I use these precomputed approximations. The same precomputed approximations can also be used to compute $\|\mathbf{w}'(\mathbf{x} + A')\|_2$ for any A'. In this way, each computation of $\|\mathbf{w}'(\mathbf{x})\|_2$ or $\|\mathbf{w}'(\mathbf{x} + A')\|_2$, and hence $E[mc(\mathbf{x})]$ or $E[mc(\mathbf{x} + A')]$, takes only $O(n)$ time, linear in the number of attributes. In this work, I always use this approximation when computing expected misclassification costs for a support vector machine using a nonlinear kernel.

5.5 A Cost-sensitive Attribute Value Acquisition Algorithm

Given a method for computing $E[mc(\mathbf{x})]$ and $E[mc(\mathbf{x} + A')]$ efficiently, I can now describe an algorithm that uses a trained support vector machine to perform cost-sensitive attribute value acquisition under my framework, described by Figure 5.1. This algorithm defines the attribute value selection algorithm and the termination condition, the second and third parts in the iterative framework.

Note that the expression $E[mc(\mathbf{x})] - E[mc(\mathbf{x} + A')]$ gives the expected decrease in misclassification cost, and the aim here is to minimize the total cost of acquisitions and misclassifications. Hence, I define the following.

Definition 5.5 (Expected reduction in total cost). Consider a test instance $\mathbf{x}$ and $A' \subseteq M(\mathbf{x})$, a subset of its missing attribute values. Let $ac(A')$ be the acquisition cost of the attribute values A', which is assumed to be given in the cost-sensitive acquisition problem. The *expected reduction in total cost* for acquiring A' is defined by:

$$E[rc(A')] = (E[mc(\mathbf{x})] - E[mc(\mathbf{x} + A')]) - ac(A') \tag{5.21}$$

My cost-sensitive attribute value acquisition algorithm is driven by the expected reduction in total cost that can be achieved by acquiring missing attribute values in test instances. This approach follows that of [Ling et al., 2006]. Here, I consider a strategy

Algorithm 5.1 Cost-sensitive attribute value acquisition algorithm.

Input: A test instance $\mathbf{x}$

1: **loop**
2: Select the attribute value $x_k \in M(\mathbf{x})$ with the maximum $E[rc(\{x_k\})]$
3: **if** $E[rc(\{x_k\})] \leq 0$ **then**
4: **break**
5: $\mathbf{x} \leftarrow \mathbf{x} + \{x_k\}$ {Acquire the attribute value x_k}
6: Classify $\mathbf{x}$

of sequentially acquiring one attribute value in each iteration, until there is no more attribute value x_k with positive $E[rc(\{x_k\})]$. This acquisition algorithm is shown in Algorithm 5.1.

I note that it is possible to employ batch acquisition strategies, where multiple attributes are acquired in one go. However, as my focus in this work is the computation of $E[mc(\mathbf{x})]$ and $E[mc(\mathbf{x}+A')]$, thereby facilitating the construction of a cost-sensitive attribute value acquisition algorithm for a trained support vector machine, therefore I leave batch acquisition strategies as future work.

5.6 Evaluation

In this section, I empirically establish the effectiveness of my acquisition algorithm. I compared my cost-sensitive acquisition algorithm against a baseline that randomly acquires a proportion (25%, 50%, or 75%) of the missing attribute values in each test instance. This evaluation is performed separately on SVMs with three different kernels: linear, polynomial, and radial basis function (RBF). The classifiers are made sensitive to imbalanced misclassification costs by modifying the primal optimization problem, by replacing the $\sum_{i=1}^{N} \varepsilon_i$ part of the objective function with $\sum_{i=1}^{N} \lambda_i \varepsilon_i$, where λ_i is a weight proportional to the misclassification cost of the training instance $\mathbf{x}_i$ [Osuna et al., 1997].

I performed the experiments using ten commonly used datasets from the UCI machine learning repository [Asuncion and Newman, 2007], all of which consists of entirely numeric or binary attributes. All datasets with more than two classes were converted into two-class problems by merging classes. By convention, the minority class

Dataset	Description	Attributes	Instances	Distribution
ECOLI	Protein localization sites	8	336	143 / 193
GLASS	Glass identification	9	214	51 / 163
HEART	Cleveland heart disease	13	303	139 / 164
HEPATITIS	Hepatitis domain	34	351	126 / 225
IONOSPHERE	Ionosphere radar data	34	351	126 / 225
PIMA	Pima Indians diabetes	8	768	268 / 500
SONAR	Sonar signals, mines versus rocks	60	208	97 / 111
SPAM	Emails spam detection	20	4601	1813 / 2788
THYROID	Thyroid disease	20	3772	284 / 3488
WDBC	Wisconsin diagnostic breast cancer	30	569	212 / 357

Table 5.1: Summary of the datasets.

is set as the positive class. A summary of the datasets is given in Table 5.1. For each dataset, I normalized the instances such that each attribute is supported on $[-1, +1]$ with mean 0, and split each dataset into equal-sized training and testing sets with the same class distribution in each set. Following [Ling et al., 2006], I used the training set as-is, and randomly mark a proportion (20%, 40%, 60%, 80%, or 100%) of the attribute values in the testing set as missing. At classification time, all unacquired missing attribute values of a (normalized) test instance are replaced by the zero value before its class is determined. I follow the protocol of [Ling et al., 2006] and set the misclassification costs of a positive instance and a negative instance to be 600 and 200 respectively. Where available, I used the acquisition costs supplied in [Turney, 1995]; and for the remainder of the datasets, I set the acquisition cost of each attribute to be a random number between 1 and 100. The evaluation measure is the total cost of acquisitions and misclassifications.

In my experiments, I use the LIBSVM implementation for support vector machines [Chang and Lin, 2001]. When training a classifier, five-fold cross validation is used to perform parameter tuning. For each acquisition algorithm and classifier combination, I repeat the training and testing process 20 times, and record the average total cost per test instance. The results of my experiments are shown in Figure 5.2 for the linear kernel,

Acquisition algorithm	Proportion of missing attribute values				
	20%	40%	60%	80%	100%
Linear kernel					
Random (25%)	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$
Random (50%)	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$
Random (75%)	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$
Polynomial kernel					
Random (25%)	$p < 0.05$	$p < 0.05$	$p < 0.01$	$p < 0.01$	$p < 0.01$
Random (50%)	$p < 0.01$	$p < 0.05$	$p < 0.01$	$p < 0.01$	$p < 0.01$
Random (75%)	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$
RBF kernel					
Random (25%)	$p < 0.10$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$
Random (50%)	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$
Random (75%)	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$

Table 5.2: p-values of the one-tail Wilcoxon signed-rank tests between my cost-sensitive acquisition algorithm and random acquisition.

Figure 5.3 for the polynomial kernel, and Figure 5.4 for the RBF kernel.

The results show that generally, in all datasets, my cost-sensitive acquisition algorithm can achieve a much lower total cost compared to randomly acquiring a proportion of attribute values, especially when a greater proportion of attribute values is missing. I found that my cost-sensitive acquisition algorithm may incur acquisition costs greater or lesser than that of random acquisition, but my acquisition algorithm generally makes suitable trade-offs between acquisitions and misclassifications, allowing total costs to be minimized. I performed one-tail Wilcoxon signed-rank tests comparing the average total costs of my cost-sensitive acquisition algorithm versus each of the three random acquisition algorithms. The statistical significance tests were performed separately for each of the three kernels (linear, polynomial, and RBF) and each of the five proportions of missing attribute values (20%, 40%, 60%, 80%, and 100%). The results of the significance tests are shown in Table 5.2, which indicates that the differences are statistically significant in all classifiers. For the vast majority of the forty-five cases, I record a significant difference at the $p < 0.01$ level, although four of them were significant only at the $p < 0.05$ or $p < 0.10$ level. The statistical significance is seen for all of the linear,

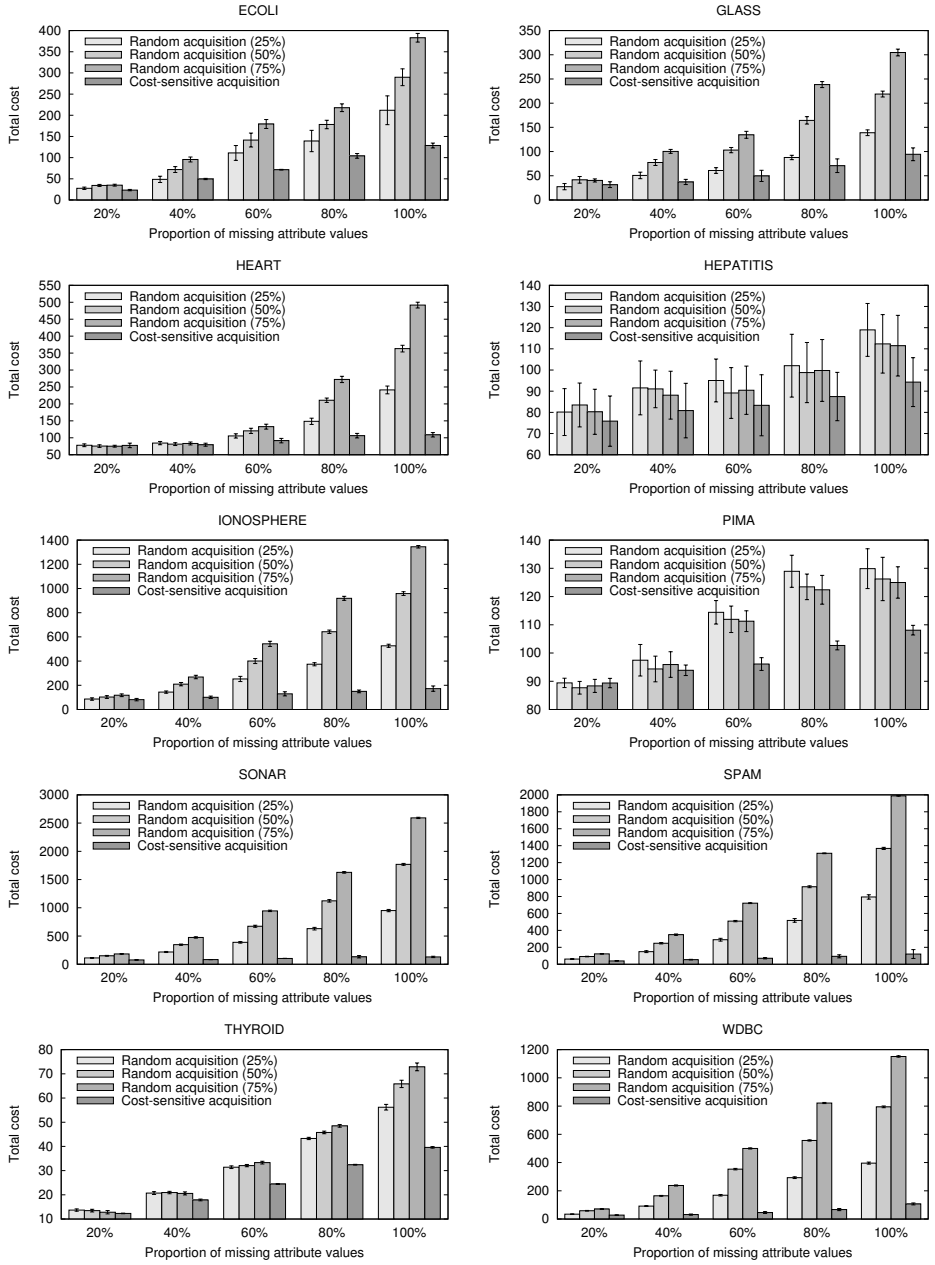


Figure 5.2: Average total cost per test instance for the linear kernel. Error bars indicate one standard deviation.

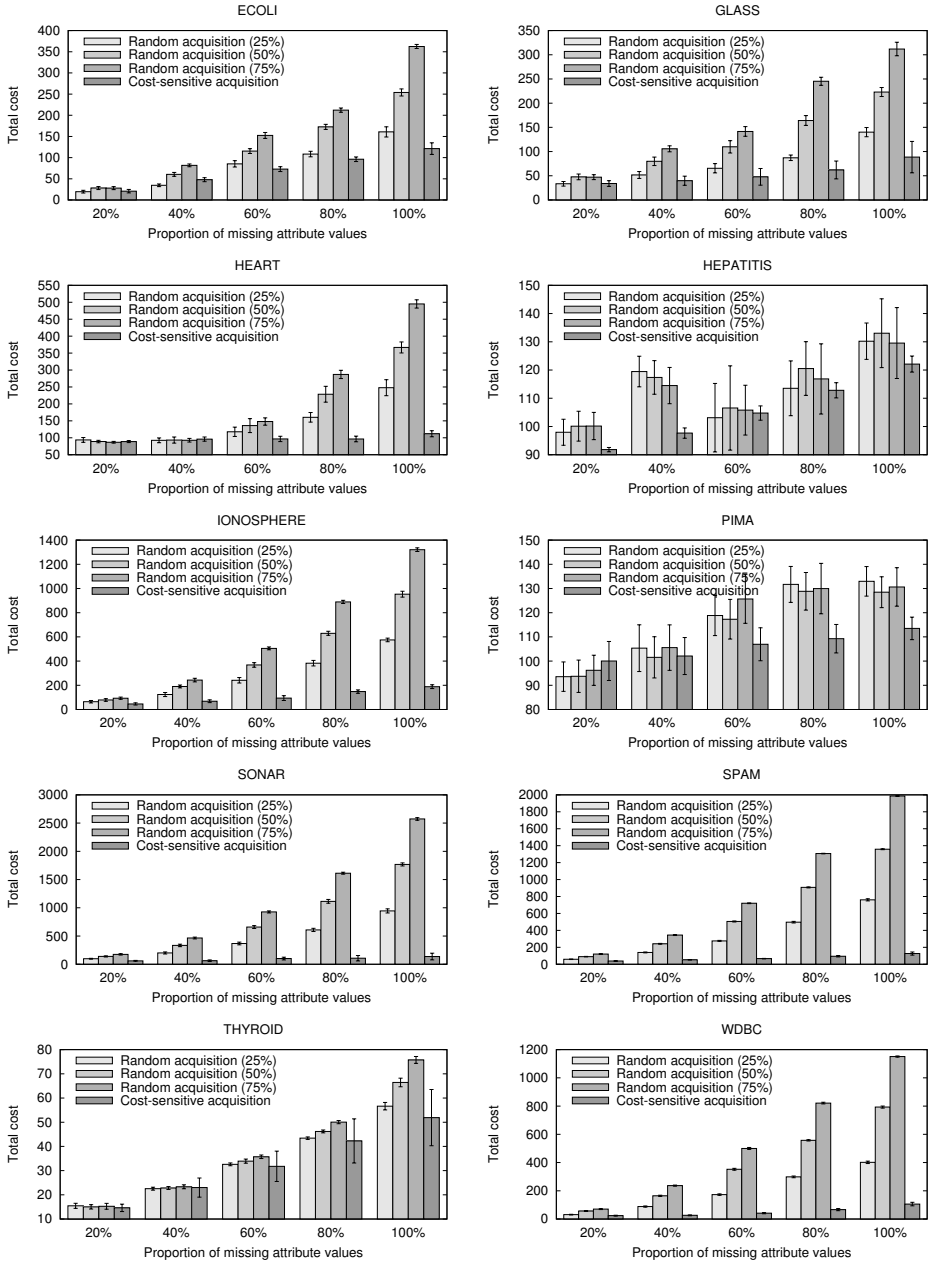


Figure 5.3: Average total cost per test instance for the polynomial kernel. Error bars indicate one standard deviation.

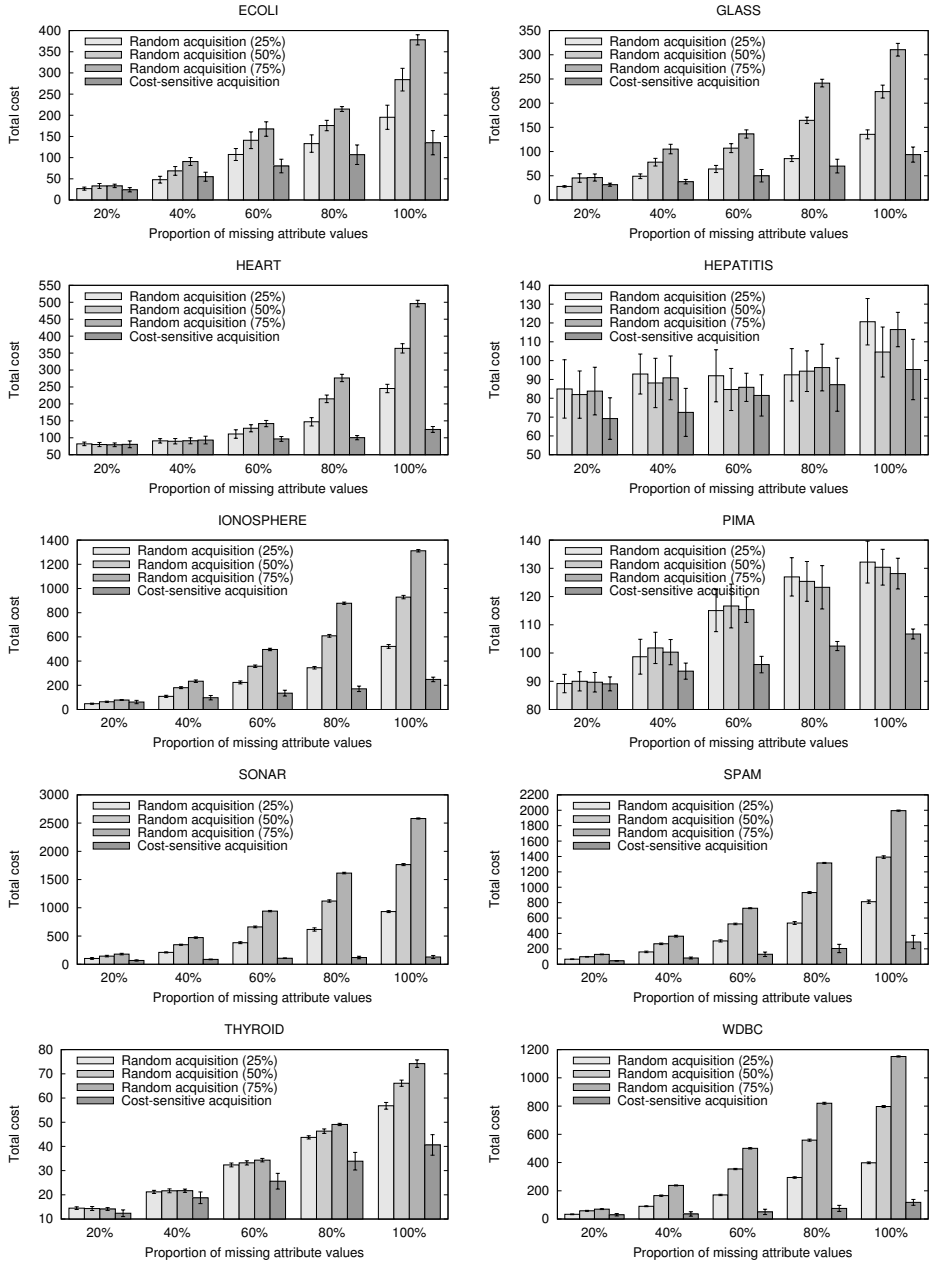


Figure 5.4: Average total cost per test instance for the RBF kernel. Error bars indicate one standard deviation.

polynomial, and RBF kernels.

In summary, my experiments have established that my acquisition algorithm is effective, when compared to random acquisition.

5.7 Conclusion and Discussion

In this chapter, I described a general iterative framework for solving the cost-sensitive attribute value acquisition problem. I proposed a realization of the framework when the classifier is the support vector machine by computing the expected reduction in total costs for acquiring attribute values. A key part in my algorithm is a method for estimating the classification certainty (hence the expected misclassification cost of a test instance) with missing attribute values using the weight vector of the support vector machine. Another key aspect in this work is the generalization of my algorithms such that they can be applied to support vector machines with arbitrary kernels, and not merely limited to linear support vector machines as in others' previous work. My experiments show that my cost-sensitive acquisition algorithm is effective on the linear, polynomial, and RBF kernels. The one-tail Wilcoxon signed-rank test shows that my improvements are statistically significant against random acquisition.

This work assumes that the each attribute value in each test instance may be acquired independently. However, when using Web-based resources to solve record matching problems, where each test instance is a pair of records, this assumption is often not true. Firstly, different test instances may share the same attribute value. For example, all test instances that compare a record a against another record can contain the hit count of a as an attribute value, therefore it needs to be acquired only once for all these test instances. Secondly, different attribute values may be derived from a common higher-level resource. For example, the search engine results of a query a may be used to generate a multitude of different attribute values, such as the hit count of a, term frequencies from the titles and snippets, as well as URL-related attribute values. In Chapter 6, I present a hierarchical cost-sensitive resource acquisition framework that

removes this limitation. However, the work presented in this chapter is used as a basis in Chapter 7, where I construct benefit functions that are needed for the application of my hierarchical resource acquisition framework.

Chapter 6

A Framework for Hierarchical Cost-sensitive Web Resource Acquisition

6.1 Introduction

As seen in Chapter 3, acquiring web resources through a search engine can help record matching algorithms improve their matching effectiveness. Such success can be seen in problems across many domains, such as the related work described in Chapter 2. However, whether explicitly stated in these works or not, obtaining search engine results and downloading web page are time consuming processes. Further, search engines often perform rate limiting. As of 2009 June, Google Search API limits the number of queries one can make in a day to 1000, while Yahoo! Search API imposes a daily limit of 5000. This means that for two lists of only 100 items each, if we send all 10,000 pairwise queries of the form $a \wedge b$ to the Yahoo! Search API, then it would take two days to execute. Thus, slow web accesses and rate limiting make acquiring such web information a very time consuming process. This can cause significant bottlenecks unless selective acquisition of web resources is done.

One of the most common approach to limit the cost of resource acquisitions while achieving reasonable task performance is through *blocking*, which filters out obvious mismatches and only performs matching on non-obvious record pairs (see *e.g.*, [Winkler, 2006]). Successful applications of blocking can reduce the number of required pairwise comparisons by a few orders of magnitude while achieving only a small re-

duction in task performance [Goiser and Christen, 2006], and hence provide a significant reduction in the required resource acquisitions. However, blocking techniques are often ad-hoc domain-specific techniques and can be difficult to define properly; poor blocking decisions can result in degraded performance through a large number of false positives or false negatives [Goiser and Christen, 2006].

On the other hand, there is also a large pool of work that formulates such resource acquisition problems into selective and cost-sensitive acquisition of missing attribute values in a classification model (*e.g.*, [Ling et al., 2006], [Saar-Tsechansky et al., 2009]). While such works provide more principled acquisition algorithms, they generally do not consider dependencies in resource acquisitions, and assume that resource acquisitions are independent events across different instances. However, such assumptions are often not true in the web resource context: in order to compute similarity metrics based on results from search engines, one must acquire the search engine results first; and search engine results for query q are available for all pairwise instances that compares q with another query. Therefore, acquisition of some resources may enable the acquisition of others, and an acquired resource may possibly be shared among multiple instances. Put another way, resource acquisitions can have hierarchical dependencies.

Earlier, in Chapter 4, I proposed an adaptive querying framework where two algorithms of different strengths, one stronger in matching performance, and one faster in running time, are combined together into one with good performance and with reasonable runtime. In this context, we can cast these two algorithms as producing two different attributes for test instances, to be used by a classifier to determine whether these instances represent matches or mismatches. Thus, another line of related work is in the cost-sensitive acquisition of missing attribute values in instances, such as [Ling et al., 2006; Saar-Tsechansky et al., 2009]. However, none of these works considered the possibility of different instances sharing the same attribute values, or attribute values in different instances being derived from the same resource, as commonly found in the structure of record matching problems. In [Kanani and McCallum, 2007], the authors solved a clustering problem expressed as a complete graph, and where additional web

resources can be acquired and added as additional vertices in the graph. However, their work only considered one kind of additional resource (search engine results) and did not account for varying acquisition costs that differing resources may incur. Further, another common issue in all of these cost-sensitive acquisition papers is that none of them considered the hierarchical dependency of resources, such as the fact that web page downloads depend on search engine results. To the best of our knowledge, our work is the first to generalize the resource acquisition problem to handle the conditions modeled in our framework.

In this chapter, my contributions are as follows:

1. I propose a framework for performing cost-sensitive resource acquisition problems with hierarchical dependences, through the use of a *resource dependency graph*. My framework abstracts away from any specific problem instance, and I show that a number of problems involving selective resource acquisitions can be formulated using resource dependency graphs.

2. Given a resource dependency graph of a record matching problem, as well as a benefit function for determining the benefit value for making particular resource acquisitions, I propose an algorithm for making suitable resource acquisitions that take the unique characteristics of such graphs into account. I cast the problem as a combinatorial search problem, and my algorithm is an application of the widely-studied Tabu search algorithm [Glover, 1990].

This chapter is organized as follows. In Section 6.2, I describe my cost-sensitive resource acquisition framework, as well as its possible applications. In Section 6.3, I describe an algorithm for solving the generic resource acquisition problem in the context of my framework. In Section 6.4, I conclude this chapter with a discussion.

This chapter deals with the framework itself. Two benefit functions that are suitable for use in my framework when solving record matching problems is proposed in Chapter 7. Evaluation of my framework in conjunction with the proposed benefit framework is also performed in Chapter 7.

6.2 Resource Acquisition Framework

My cost-sensitive resource acquisition framework is designed to be versatile, and can handle a large variety of resource acquisition problems. Central to my framework is the notion of a *resource dependency graph*. To aid my description, I use a simple record matching example involving web-based resources as my running example. I then expand on this example to illustrate how other record matching problems can be mapped into my resource acquisition framework, by showing how the resource dependency graphs can be constructed. I define our framework in this section, and propose an algorithm for solving the acquisition problem in Section 6.3.

6.2.1 My Framework

I use a particular kind of record matching problem known as record linkage as my running example. Recall that in record linkage, the input is two lists A and B, and the aim is to determine which record pairs $(a, b) \in A \times B$ are matches. For example, list A may be a list of short forms, and list B may be a list of long forms. As illustrated earlier, we may choose to query a search engine with the query strings a, b, or $a \wedge b$, and obtain raw search engine results which I denote as $search(a)$, $search(b)$, and $search(a \wedge b)$ respectively. We can then form a test instance $\mathbf{x}_{a,b}$ whose attribute values are information extracted from these search engine results, and have a classifier determine whether $\mathbf{x}_{a,b}$ is a match or mismatch. For this running example, I do not assume that blocking is performed, hence all instances $\mathbf{x}_{a,b}$ for all $a \in A$ and $b \in B$ will need to be classified. For concreteness, let us say that each instance $\mathbf{x}_{a,b}$ consists of four attribute values. The first three are $hitcount(a)$, $hitcount(b)$, and $hitcount(a \wedge b)$, representing the hit counts (the number of web pages matching a query) returned by the search engine in $search(a)$, $search(b)$, and $search(a \wedge b)$ for the respective queries. The fourth is the Dice coefficient computed by $dice(a, b) = \frac{2 \cdot hitcount(a \wedge b)}{hitcount(a) + hitcount(b)}$.

In this example, we can consider each of $search(a)$, $search(b)$, $search(a \wedge b)$, $hitcount(a)$, $hitcount(b)$, $hitcount(a \wedge b)$, and $dice(a, b)$ as a different type of resource

that may be acquired. The acquisition costs of different resources are not uniform, in the sense that querying a search engine to obtain $search(a)$ takes significantly longer than extracting $hitcount(b)$ after $search(a)$ is acquired. Also, we may not want to acquire all resources, as the classifier may be able to classify an instance correctly if we only acquire a subset of the attribute values in an instance. Further, some resource acquisitions have dependencies: for example, acquiring $hitcount(a)$ requires $search(a)$ to be acquired first. These dependencies can be captured in a resource dependency graph. I now formalize the resource dependency graph and the resource acquisition problem. Note that throughout this chapter and the next chapter, the terms *vertex* and *resource* will be used interchangeably.

Definition 6.1 (Resource dependency graph). A *resource dependency graph* is a directed acyclic graph $G = (V, E)$, where each vertex in the vertex set V represents a resource, and an edge $u \rightarrow v$ exists in the edge set E if resource u must be acquired before v can be acquired. In other words, E encodes the dependencies between resource acquisitions: a vertex may be acquired only after all its ancestor vertices have been acquired.

Definition 6.2 (Feasible vertex set). Given a *resource dependency graph* $G = (V, E)$, a set of vertices $V' \subseteq V$ is called a *feasible vertex set* if for all vertices $v \in V'$, we have $u \rightarrow v$ implies that $u \in V'$. That is, a feasible vertex set can be acquired without violating any acquisition dependencies. I denote the set of all feasible vertex sets in the graph G by $F(G)$, with $F(G) \subseteq 2^V$.

For this running example, the resource dependency graph corresponding to two lists of 2 records each is depicted in Figure 6.1. Observe how some resources are common to multiple instances, *e.g.*, $hitcount(a_1)$ is a common resource in the instances $\mathbf{x}_{a_1,b_1}$ and $\mathbf{x}_{a_1,b_2}$. Even in this simple case, the resource dependency graph already starts to get complex. However, the structure of the resource dependency graph contains many repeated patterns. Hence I represent the resource dependency graph for input lists of any size using Figure 6.2, using a variant form of the plate notation introduced in [Buntine,

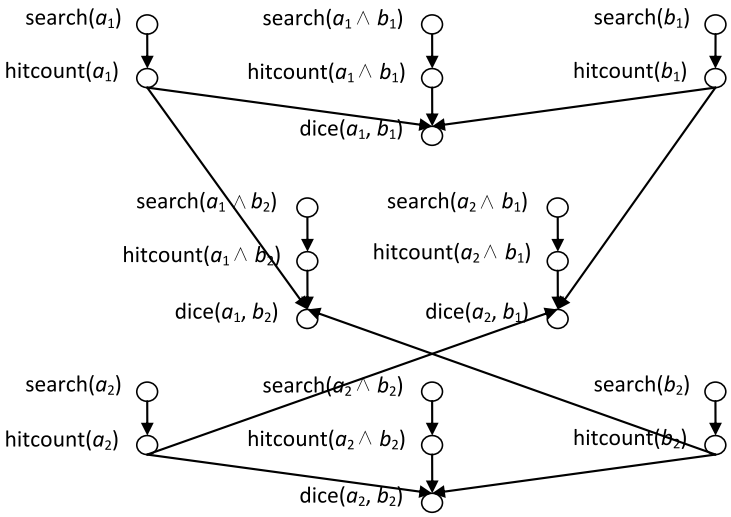


Figure 6.1: Example resource dependency graph for two lists of 2 records each.

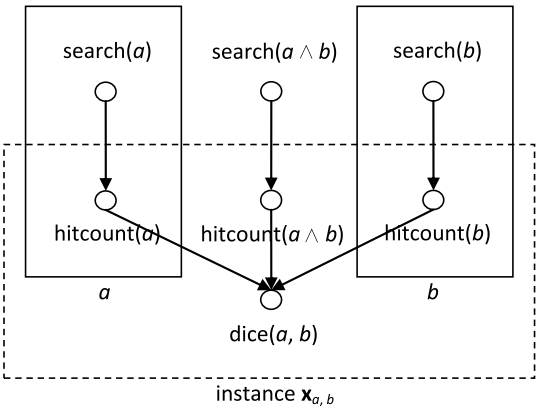


Figure 6.2: Structure of example resource dependency graph.

1994]. Here, the solid box labeled a means that the vertices inside the box are common to all instances involving a, and likewise for the box labeled b. Hence, there are $|A|$ vertices of the form $search(a)$ as opposed to $|A|\times|B|$ vertices of the form $search(a\wedge b)$. The dashed box indicates the vertices that form the attribute values of the test instance $\mathbf{x}_{a,b}$.

Definition 6.3 (Resource acquisition problem). A *resource acquisition problem* takes in the following as input:

- A resource dependency graph $G = (V, E)$.
- An *acquisition cost* function $cost : V \to \mathbb{R}^+ \cup \{0\}$. I assume that $cost(v) = 0$ for any acquired vertex v.

- A *benefit* function $benefit : F(G) \rightarrow \mathbb{R}$. I assume that $benefit(\emptyset) = 0$ and $benefit(V') = benefit(V'_u)$, where V'_u is the set of unacquired vertices in V'.
- A *vertex type* function $type : V \rightarrow T$, where T is a finite set of vertex types.
- A *budget*, denoted by $budget$.

The aim of the resource acquisition problem is to acquire a feasible subset of vertices $V' \in F(G)$, with $cost(V') \leq budget$, such that the objective function $obj(cost(V'), benefit(V'))$, expressed as a function of $cost(V')$ and $benefit(V')$, is maximized.

Two possible objective functions are $obj(cost(V'), benefit(V')) = benefit(V') - cost(V')$ and $obj(cost(V'), benefit(V')) = \frac{benefit(V')}{cost(V')}$. When the context is clear, I will denote $obj(cost(V'), benefit(V'))$ by $obj(V')$. For convenience, I also define the following shorthand notations for the cost and benefit functions.

- For any feasible set of vertices $V' \subseteq F(G)$, $cost(V') = \sum_{v \in V'} cost(v)$.
- For any vertex v, $benefit(v) = benefit(\{v\})$.

In this running example, there are six vertex types, corresponding to $search(a)$, $search(b)$, $search(a \wedge b)$, $hitcount(a)$, $hitcount(b)$, $hitcount(a \wedge b)$, and $dice(a, b)$. Depending on problem requirements, we can scale the acquisition costs and benefit values appropriately. We may (empirically or otherwise) set the acquisition cost of each $search(\cdot)$ vertex to 10 and the acquisition cost of all other vertices to 1, because search engine results take significantly longer to obtain compared to extracting hit counts. Also, as the classifier can only understand and process the attribute values in test instances, $search(a)$, $search(b)$, and $search(a \wedge b)$ will have zero benefit value since these vertices are not part of any attribute values. More generally, a classification model can accept values that are numeric but not raw search engine results. Also, if we use $obj(cost(V'), benefit(V')) = benefit(V') - cost(V')$ as the objective function, we can scale the acquisition costs and benefit values appropriately based on problem requirements. Thus we would like to acquire a suitable subset of vertices that maximizes the benefit of acquiring these vertices while minimizing the total cost of acquiring them.

In most problems, we can assume that the resource dependency graph and the acquisition cost function are known and fixed. However, the benefit function is typically only a heuristic that approximates to the true performance metric of the problem, because knowing the true benefit function entails knowing the solution set. For example, in a record matching problem, we may want to minimize the total misclassification cost of all instances or to maximize the F-measure of the matching instances, but the benefit function may just return the estimated increase in classification confidence (scaled appropriately) instead. Here, when I consider algorithms for solving the resource acquisition problem, I will treat the benefit function to be an oracle whose inner workings are unknown. However, good benefit functions should be monotonic or mostly monotonic, *i.e.*, $benefit(V_1 \cup V_2) \geq benefit(V_1)$ for most, if not all, of V_1 and V_2 that are feasible subsets of the vertex set V, facilitating algorithms that acquires vertices in an incremental manner. In Chapter 7, I will define benefit functions in the context of classification problems where missing attribute values in test instances may be acquired.

6.2.2 Applications

By modeling resource acquisition problems as resource dependency graphs, I have defined my framework so it can be applied to various problem and solution settings. My framework does not insist on having a classifier to classify test instances, as long as we have a benefit function that indicates how useful vertices are. In the following, I suggest how resource dependency graphs can be constructed for record matching problems when additional types of resources and attributes are added, and when a variant of the record linkage problem is encountered.

Term frequencies obtained from web pages. Term frequencies and TF-IDF values from documents such as web pages are information retrieval concepts that are commonly used in record matching applications. In this situation, for a query q, we can add two types of resources to the resource dependency graph of the running example: $webpage(q)$, representing the set of web pages that may be downloaded from the URLs in the top-k search engine results of $search(q)$; and $tf(q)$, representing the term fre-

quencies of the terms found in the downloaded web pages $webpage(q)$. Of course, $webpage(q)$ depends on $search(q)$, and $tf(q)$ depends on $webpage(q)$. The term frequencies in $tf(\cdot)$ can then be used as attribute values in the test instances. If TF-IDF values are desired instead of term frequencies, then the $tf(q)$ vertices can be replaced by $tfidf(q)$ vertices, with the document frequencies pre-calculated during a precomputation phase when the classification model is trained. Another alternative to using such frequencies directly is to compute cosine similarity between the term frequency or TF-IDF vectors; such $cos(a, b)$ vertices can be added into the resource dependency graph easily. For acquisition costs, if we assume that each web access has a cost of 10, then we can assign a cost of $10k$ for each $webpage(q)$ vertex and a cost of 1 to each of the $tf(q)$, $tfidf(q)$, and $cos(a, b)$ vertices.

Common n-grams. In Section 3.3, I observed that it is possible that the queries formed from a list of records contain many common contiguous subsequences of tokens, also known as n-grams. If a particular n-gram is shared by at least k of the queries, then it is possible that querying this n-gram may produce results for all these queries, and hence we can save on the number of queries if we query on these n-grams using a query probing approach instead of the individual queries. As such, we may construct a vertex $search(g)$ for each n-gram g that occurs in at least k of the original queries, with additional vertices of the form $value(q, g)$ for information extracted from $search(g)$ pertaining to the original query q.

Clustering. Another common form of record matching takes in a list L as input and determines which record pairs $(a, b) \in L \times L$ are matches. The building of the resource dependency graph is analogous to that of record linkage. However, instances in clustering problems are symmetric, *i.e.*, $\mathbf{x}_{a,b}$ and $\mathbf{x}_{b,a}$ are equivalent. Hence, when constructing an instance $\mathbf{x}_{a,b}$, $hitcount(a)$ and $hitcount(b)$ make poor attribute values because the equivalent instance $\mathbf{x}_{b,a}$ would have the positions of these two attribute values interchanged, confusing classifiers such as linear support vector machines. Instead, we can create two new vertices for the smaller and larger of $hitcount(a)$ and $hitcount(b)$, and use them as attribute values instead.

Blocking. It is possible to apply my record acquisition framework after blocking is performed on the input. Effectively, blocking causes some vertices and edges to be pruned from the original resource dependency graph. For each test instance $\mathbf{x}_{a,b}$ that is pruned, we can delete the vertices $search(a \wedge b)$, $hitcount(a \wedge b)$, and $dice(a \wedge b)$ and their associated edges from the resource dependency graph. Further, if all instances involving a record $a \in A$ are pruned, then we can delete $search(a)$ and $hitcount(a)$ as well.

6.2.3 Observations on Graph Structure of Record Matching Problems

The careful reader may discern some regularity in the structure of resource dependency graphs that model web resource acquisition. I make some observations about this structure that informs my proposed acquisition algorithm in the next section.

The first observation I make is that the depth of a typical resource dependency graph of a web resource acquisition problem is shallow, usually of depth 2 to 4. (The depth of a directed acyclic graph G is defined as the number of vertices in the longest directed path in G with no repeated vertices.) Further, the vertices at or near the roots are often the results of costly web queries but with zero benefit value (*e.g.*, $search(\cdot)$ and $webpage(\cdot)$), while the vertices at or near the leaves are typically attribute values that can be extracted from the raw information at negligible costs (*e.g.*, $hitcount(\cdot)$). Given this structure, many resource dependency graphs can be restructured as hierarchies with depth 2, where the root level mainly consists of vertices that are costly to acquire and the leaf level mainly consists of vertices that can be acquired very cheaply. For example, if the original graph contains the edges $search(a) \rightarrow webpage(a)$ and $webpage(a) \rightarrow tfidf(a)$, then we can restructure the edges as $search(a) \rightarrow tfidf(a)$ and $webpage(a) \rightarrow tfidf(a)$. This does not change the problem at all because acquiring $tfidf(a)$ still requires acquiring both $search(a)$ and $webpage(a)$, and merely acquiring $webpage(a)$ without $tfidf(a)$ is useless and produces a suboptimal solution. As another example, consider the original running example of Figure 6.2. Because

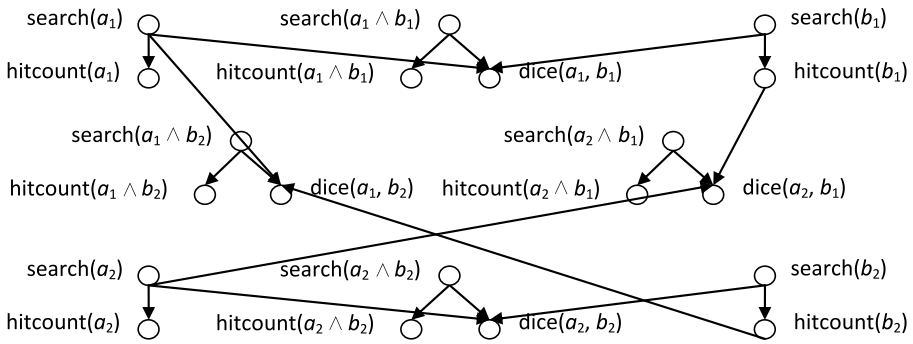


Figure 6.3: Example resource dependency graph with alternative structure for two lists of 2 records each.

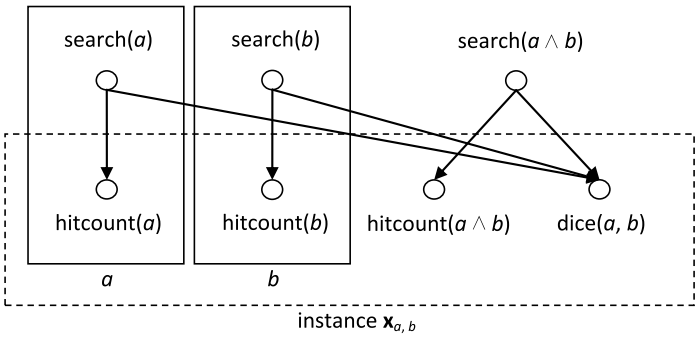


Figure 6.4: Alternative structure of example resource dependency graph.

all of $hitcount(a)$, $hitcount(b)$, $hitcount(a \wedge b)$, and $dice(a, b)$ can be acquired very cheaply given the search engine results, we can collapse these vertices yielding a two-layer graph as shown in Figure 6.4 that approximates the original with little difference, since the acquisition costs are dominated by $search(a)$, $search(b)$, and $search(a \wedge b)$.

The next observation is that the vertex out-degree may vary widely. To illustrate this observation we again consider the resource dependency graph of Figure 6.4. A vertex for $search(a \wedge b)$ for $a \in A$ and $b \in B$ typically only have very few child vertices, such as $hitcount(a \wedge b)$. On the other hand, a vertex for $search(a)$ can have at least $O(|B|)$ descendant vertices such as $dice(a, b)$. Therefore, resource dependency graphs have bushy topologies.

6.3 Solving the Resource Acquisition Problem for Record Matching

The resource acquisition problem aims to find a subset of vertices $V' \subseteq V$ to acquire such that the objective function $obj(V')$ is maximized. This is essentially a combinatorial search problem over all subsets of V. However, a brute-force enumeration of all feasible subsets of V is prohibitively expensive even for a moderate-sized graph G. Even for the simple resource dependency graph for linking two lists of 2 records each as shown in Figure 6.3, if we consider only the leaf vertices that are attribute values in the test instances, there are already 2^{12} possible acquisition decisions (there are 12 leaf vertices and each vertex may either be acquired or not acquired). More generally, this resource dependency graph has the structure of Figure 6.4, meaning that $2^{|A|} \times 2^{|B|} \times 4^{|A|\times|B|}$ possible acquisition decisions if we consider only the leaf vertices. Owing to the exponential number of possible acquisition decisions, it is completely infeasible to perform exhaustive search. Therefore, I need to employ some form of heuristic search.

In my heuristic search algorithm, I use $F(G)$, the set of feasible subsets of the vertex set V, as my state space. We denote the current state by V', and set $V' = \emptyset$ as my initial state. For the set of legal moves that can be applied to the current state to bring it to another state, I restrict them to functions that add or remove vertices to reduce the search space, which I describe later. However, the unique structure of the resource dependency graph and the benefit function in record matching problems poses special difficulty as it is easy for the search algorithm to get stuck in local maxima. As an example consider the resource dependency graph of Figure 6.4, where only the leaf vertices have positive benefit value and each of them has benefit greater than cost. Starting from an initial, empty state $V' = \emptyset$, suppose I only add or remove a single vertex such that the new state is a feasible vertex set. Then this empty state V' is a local maximum since only $search(\cdot)$ vertices can be added, which only incurs cost but no benefit. Further, when the search algorithm allows visiting states with such lower objective values, there is no

guidance on which $search(\cdot)$ vertex to add. Let us say $search(a)$ (for some $a \in A$) was added to the current state and one of the best vertex to acquire is $hitcount(b)$ (for some $b \in B$). Then it is easy for the algorithm to add $hitcount(a)$ into the current state and get stuck in another local maxima. With a large number of local maxima, it is easy for search algorithms to keep revisiting the states around local maxima.

A solution must be able to address these topological issues in our web acquisition dependency graphs. I tackle these issues in the following ways:

1. I apply a widely-studied metaheuristic search algorithm known as Tabu search [Glover, 1990]. This search algorithm avoids revisiting the same states, allowing a better exploration of the state space.

2. I define a set of legal moves that are more intelligent. These moves allow the search algorithm to reach a state that includes the leaf vertices faster.

3. I introduce a surrogate benefit function that propagates the benefit values from the leaf vertices upwards to the root vertices. This guides the search algorithm towards vertices with the highest benefit values.

6.3.1 Application of Tabu Search

I first describe my application of Tabu search [Glover, 1990]. Tabu search can be seen as a modification of simple hill climbing with a mechanism to avoid revisiting states and to escape local maxima. It has been applied successfully in a number of combinatorial search problems. Applications of Tabu search in recent years include clique partitioning problems [Kochenberger et al., 2005], training multilayer feed-forward neural networks [Glover and Martí, 2006], vehicle routing problems [Archetti et al., 2006], generalized minimum spanning tree problems, [Öncan et al., 2008], and three-dimensional bin packing problems [Crainic et al., 2009].

In simple hill climbing, one has the notion of the current state and a set of legal moves. A move, when applied to the current state, generates a new state that is a neighbor of the current state. In each iteration, simple hill climbing chooses the move that

leads to the neighboring state with the highest objective value, and sets that neighboring state to be the new current state. However, simple hill climbing is prone to getting stuck in local maxima and plateaus. Tabu search extends on the idea of simple hill climbing by adding in Tabu list, which stores the moves that are prohibited, or marked as Tabu. More specifically, when a particular move is applied at some iteration, the move(s) that undo the effect of the applied move will be placed in the Tabu list for a set number of iterations known as the Tabu tenure. This prevents the search process to repeatedly visit the same states and encourages exploration of the state space. However, as an exception, Tabu search allows a Tabu move to be applied to the current state if the resultant state has an objective value that is better than the states that have been visited so far. This *aspiration criterion* is applied because the resultant state is obviously an unexplored state. I stop the Tabu search when a certain number of consecutive iterations have occurred without improvement in the best objective value. I then acquire the vertices in the state with the highest objective value.

6.3.2 Legal Moves

The next key aspect in my design is to provide a relatively small number of potentially useful legal moves from the current state V' to reduce the search space in such large resource dependency graphs, because an improper set of legal moves can lead to a very large search space, possibly exponential in size. For $V' \neq \emptyset$, let $D(V')$ be the set of children vertices of vertices in V' that are also not in V'. For example, in Figure 6.4, $D(\{search(a)\})$ contain the $hitcount(a)$ vertex and all $dice(a, b)$ vertices. For $V' = \emptyset$, let $D(V')$ to be the set of root vertices in G instead. The legal moves in my framework are:

- $Add(v)$: For a vertex $v \in D(V')$, add v and all its ancestors to V'.
- $Remove(v)$: For a vertex $v \in V'$, remove v and all its descendants from V'.
- $AddType(t)$: For a vertex type $t \in T$, construct an arbitrary maximum subset of vertices $U \subseteq D(V')$ with vertex type t, such that the acquisition cost of V', U,

and the ancestors of U does not exceed $budget$. Then add all the vertices in U and their ancestors to V'.

Note that all of these moves have the potential to batch add or remove multiple vertices in one operation. The aim for defining the $Add(v)$ and $Remove(v)$ this way is to allow the search algorithm reach the vertices with high benefit values more easily. For example, in Figure 6.4, if the current state V' has a vertex $search(a)$, it might be worthwhile to add $dice(a, b)$ as well, but this move would not be possible if I define the legal moves to add only one vertex at a time unless the current state already contains $search(b)$ and $search(a \wedge b)$ as well. Finally, $AddType(t)$ allows as many descendant vertices of the same type to be added as possible, because there are cases where the true benefit comes from acquiring one costly root vertex and many cheap leaf vertices rather than one costly vertex and only one cheap leaf vertex. As for the Tabu moves, whenever a move adds a set of vertices V'' to the current state, I add any move that removes any vertex in V'' to the Tabu list; and whenever a move removes a set of vertices V'' from the current state, I add any move that adds any vertex in V'' to the Tabu list.

6.3.3 Surrogate Benefit Function

Having defined the set of legal moves, I now turn to the question of how to choose a move from the current state V'. Many of the root vertices share the same acquisition cost and have zero benefit, giving no guidance on which root vertices lead to the best leaf vertices. Therefore, I propagate benefit values upwards from the leaf vertices to the root vertices, by repeatedly propagating from vertices to edges and then from edges to vertices. This allows me to define a surrogate benefit function $surr\text{-}benefit(V', V'')$ for the case when a subset of vertices V'' is added to the current state V' that takes into account the benefit values propagated from the descendant vertices of V''. Using this surrogate benefit function, I can then define a surrogate objective function $surr\text{-}obj(V', V'')$ that I use to select the best legal move to reach the next state.

Two questions need to be addressed when defining the surrogate benefit function. Firstly, how much benefit value should we propagate upwards? While propagating

nothing results in the aforementioned problem, propagating everything results in the algorithm believing that it is sufficient to acquire only the root vertices, and not any of the leaf vertices where the attribute values are. Secondly, when a vertex v has multiple children and each child vertex propagates benefit values to v, which of these propagated benefit values should be aggregated into the surrogate benefit value of v? This poses a problem if v has many children, but either none of them or only a portion of the child vertices are useful. If we aggregate the propagated benefit values from all of them, then we overestimate the actual benefit value because not all child vertices should be acquired. On the other hand, if we aggregate only from one child vertex and ignore the rest, then we run the risk of not acquiring v if it actually contains many useful children. To control both aspects, I use a *propagation factor* λ and an *aggregation factor* β, with $0 \leq \lambda \leq 1$ and $0 \leq \beta \leq 1$.

Definition 6.4 (Propagated benefit function). Let $0 \leq \lambda \leq 1$ be a *propagation factor* and $0 \leq \beta \leq 1$ be an *aggregation factor*. For a vertex $v \in V$, let $pa(v)$ and $ch(v)$ be the parent and child vertices of v respectively. I define the following recursively:

- The *propagated benefit* of a vertex $v \in V$ is:

$$prop\text{-}benefit(v) = benefit(v) + max\text{-}pb(v) + \beta \cdot rest\text{-}pb(v)$$

  where:

$$max\text{-}pb(v) = \max_{u \in ch(v)} prop\text{-}benefit(v \to u)$$
$$rest\text{-}pb(v) = \sum_{u \in ch(v)} prop\text{-}benefit(v \to u) - max\text{-}pb(v)$$

- The *propagated benefit* of an edge $u \to v \in E$ is:

$$prop\text{-}benefit(u \to v) = \frac{\lambda \cdot (prop\text{-}benefit(v) - cost(v))}{|pa(v)|}$$

For a leaf vertex v, we have $prop\text{-}benefit(v) = benefit(v)$. This allows us to

precompute all the propagated benefits as an efficient preprocessing step, using dynamic programming, computing from the leaves towards the root.

Definition 6.5 (Surrogate benefit function). Suppose the current state is $V' \in F(G)$. Let $V'' \subset V \setminus V'$ be a set of vertices such that $V' \cup V'' \in F(G)$ is a feasible subset of G. Let $E'(V', V'')$ be the set of edges $v \to u$ with $v \in V''$ and $u \in V \setminus (V' \cup V'')$. Then the *surrogate benefit* function of adding V'' to the current state V' is:

$$surr\text{-}benefit(V', V'') = \sum_{v \to u \in E'(V', V'')} prop\text{-}benefit(v \to u)$$

My surrogate benefit function $surr\text{-}benefit(V', V'')$ has two arguments V' and V'', as opposed to using just a single V' argument. This allows the function to "forget" the propagated benefit values of V'' when the Tabu search moves to the new state $V' \cup V''$ when the actual benefit value of V'' is zero.

Definition 6.6 (Surrogate objective function). Suppose the current state is $V' \in F(G)$.

- The *surrogate objective* function of adding a set of vertices $V'' \subset V \setminus V'$ to the current state V', such that the new state $V' \cup V'' \in F(G)$ is also a feasible subset of vertices in G, is defined as $surr\text{-}obj(V', V'') = obj(cost(V''), surr\text{-}benefit(V', V''))$.

- The *surrogate objective* function of removing a set of vertices $V'' \subset V'$ from the current state V', such that the new state $V' \setminus V'' \in F(G)$ is also a feasible subset of vertices in G, is defined as $surr\text{-}obj(V', V'') = -obj(cost(V''), benefit(V''))$.

Thus, in the current state V', the Tabu search algorithm selects a legal move that adds or removes the vertices V'' with the maximum surrogate objective value $surr\text{-}obj(V', V'')$ from the moves that are either non-Tabu or which pass the aspiration criterion. To allow greater exploration of the state space, I allow the Tabu search to enter a state V' whose total cost exceeds the budget, but I subtract from the surrogate objective value and objective value a large penalty proportional to $budget - cost(V')$. When the search process terminates, I acquire the vertex set V' with the maximum objective value $obj(V', V'')$.

6.4 Conclusion and Discussion

In this chapter, I introduced the hierarchical cost-sensitive resource acquisition framework. Key in this framework is the notion of the resource dependency graph, which captures the hierarchical dependencies of resource acquisitions typically seen in a record matching problem involving Web based resources. My framework is versatile and can be applied to many kinds of problems. Next, I characterized the challenging nature of resource dependency graphs seen in record matching problem involving Web based resources. To solve these challenging topological issues, I applied the metaheuristic Tabu search algorithm to allow a better exploration of the search space. To guide the search algorithm to select resources to acquire, I also defined a set of legal moves, as well as a surrogate benefit function that propagates the benefit values from the leaf vertices upwards to the root vertices.

In Chapter 7, I will introduce two benefit functions that can be used in my resource acquisition framework, when applied to record matching problems involving Web based resources. Evaluation of my resource acquisition framework will also be performed in the context of these two benefit functions.

Depending on the dataset, the resource dependency graph can be large. In my experiments, I have a resource dependency graph with over a million vertices and a million edges for an author disambiguation problem involving a few hundred ambiguous author names. With respect to large graphs, it has been suggested that the graph be generated incrementally as the algorithm progresses, thereby avoiding a potentially quadratic cost in constructing the graph. However, graph generation is very computationally cheap when compared to the amount of time required to perform search engine queries. For the largest dataset in my experiments, generating the resource dependency graph is only in the order of minutes. This renders the incremental generation of graphs unnecessary in my experiments, although of potential interest in future work if dealing with input resource dependency graphs that are very much larger than used here.

Chapter 7

Benefit Functions for Record Matching in the Resource Acquisition Framework

7.1 Introduction

To use the hierarchical cost-sensitive resource acquisition framework of Chapter 6, a cost function and a benefit function need to be supplied. Usually, the cost function is already provided, or a suitable one can be easily estimated or engineered. However, coming out with an appropriate benefit function can be a challenge.

The design of an appropriate benefit function will typically depend on the evaluation measure that is used. For record matching problems, a common practice is to construct test instances for some or all of the record pairs, and classify each test instance as a match or a mismatch. As such, evaluation measures that measure the quality of (binary) classifications is often used. Such measures include total misclassification cost of the test instances and the F_1 measure of the match class. However, the classes of the test instances are unknown during test time. Hence, the benefit function will be an estimated improvement in the evaluation measure for resource acquisitions, such as the expected decrease in total misclassification cost, or the expected increase in F_1 measure.

By considering the problem at the instance level, it becomes very related to *cost-sensitive attribute value acquisition* for classification problems. Unfortunately, while there is a significant pool of work in cost-sensitive selective acquisition, much of these works use decision trees as their classifier (*e.g.*, [Ling et al., 2006], [Davis et al., 2006]),

and these works only consider acquiring only a *single* attribute value per iteration and dictates the order which these values may be acquired. These make cost-sensitive decision trees unsuitable for my problem, since I needed a method to return an estimate of the reduction in misclassification cost if an *arbitrary* subset of attribute values in a test instance is acquired, which is not provided by any prior work I am aware of. At the same time, the related work in Chapter 2 demonstrated the effectiveness of the support vector machine classifier in many record matching problems, which motivated my work on cost-sensitive support vector machine in Chapter 5. The way I constructed my cost-sensitive support vector machine is specifically crafted to be a natural building block for the construction of one of the benefit functions in this chapter.

In this chapter, I propose two different benefit functions for two different evaluation measures.

1. The first benefit function is catered for the support vector machine classifier when the evaluation measure is the total cost of misclassifications.

2. The second benefit function is designed for the case when the evaluation measure is the F_1 measure. It requires the classifier to be able to return classification certainty values for before and after acquiring some subset of resources.

The initial work focused on the first benefit function, which is easier because the benefit function for acquiring vertices can easily be decomposed into separate benefit functions, each for one test instance. However, the F_1 measure is more commonly used compared to the total misclassification cost. To make my overall framework applicable for the F_1 evaluation measure, I expanded my work to include the second benefit function. Designing the second benefit function is more challenging, because the F_1 measure is computed from all test instances in the testing set, as opposed to simple summation for total misclassification cost.

This chapter is organized as follows. In Section 7.2, I describe the first benefit function. In Section 7.3, I describe the second benefit function. In Section 7.4, I evaluate my hierarchical cost-sensitive resource acquisition framework of Chapter 6 in the context of these two benefit functions. In Section 7.5, I conclude this chapter.

7.2 A Support Vector Machine based Benefit Function for Total Misclassification Cost

I first consider the case when the evaluation measure is the total misclassification costs of the test instances. In this setting, it is very straightforward to adapt the cost-sensitive support vector machine of Chapter 5 to construct a benefit function for record matching problems in my resource acquisition framework.

Again, I start by considering a test instance $\mathbf{x}$ with missing attribute values, as well as considering the acquisition of a subset $A' \subseteq M(\mathbf{x})$ of its missing attribute values. In Chapter 5, I have described how the expected misclassification costs $E[mc(\mathbf{x})]$ and $E[mc(\mathbf{x}+A')]$ can be computed when the classifier is a support vector machine. Using these quantities, I can follow the approach of [Ling et al., 2006] and define the benefit of A' to be the expected decrease in misclassification cost of $\mathbf{x}$ for acquiring A'.

Definition 7.1 (Benefit of acquiring attribute values). Let $\mathbf{x}$ be a test instance and let $A' \subseteq M(\mathbf{x})$ be a subset of its missing attribute values. The *benefit* of acquiring A' is defined as the expected decrease in misclassification cost of $\mathbf{x}$ for acquiring A':

$$benefit(\mathbf{x}, A') = E[mc(\mathbf{x})] - E[mc(\mathbf{x}+A')] \tag{7.1}$$

Now, I return to my resource acquisition framework as applied to record matching problems, and recall that the aim is to come out with a benefit function for the acquisition of a subset of vertices V' in the resource dependency graph.

Firstly, I assume that the classification of test instances are independent of each other. This assumption importantly allows me to decompose the benefit function for a set of vertices V' into separate benefit functions for V' with respect to different test instances $\mathbf{x}$.

Definition 7.2 (Benefit function). Let G be a resource dependency graph and V' be a feasible subset of vertices in G. Let $I(V')$ be the set of test instances that contain one or more vertices in V' as attribute values, and let $A(V', \mathbf{x})$ denote the attribute values of

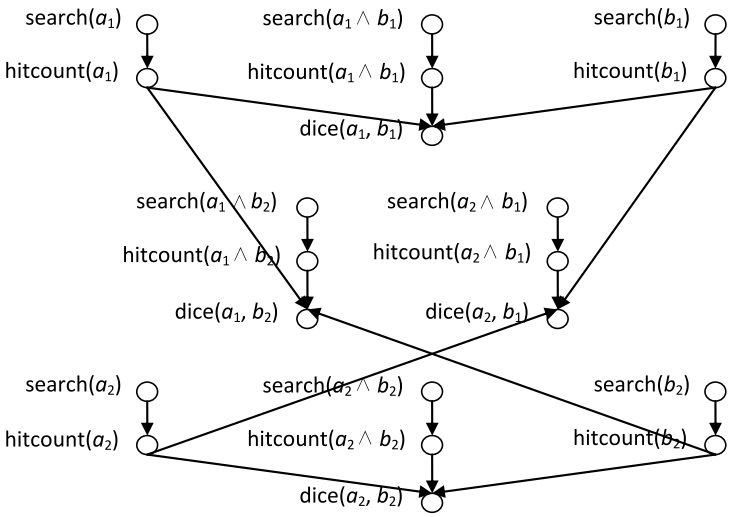


Figure 7.1: Example resource dependency graph for two lists of 2 records each.

the test instance $\mathbf{x}$ that are contained in V'. Then the *benefit* of V' is defined as:

$$benefit(V') = \sum_{\mathbf{x} \in I(V')} benefit(\mathbf{x}, A(V', \mathbf{x})) \tag{7.2}$$

To illustrate the above definition, consider the resource dependency graph of Figure 6.1 in the running example of Chapter 6. For convenience, it is reproduced here as Figure 7.1. Here, the attribute values for each test instance $\mathbf{x}_{a_i,b_j}$ are $hitcount(a_i)$, $hitcount(b_j)$, $hitcount(a_i \wedge b_j)$, and $dice(a_i, b_j)$, for $1 \leq i, j \leq 2$. Suppose $search(a_1)$ and $search(b_1)$ have been acquired, and now we want to compute $benefit(V')$ for $V' = \{hitcount(a_1), hitcount(b_1)\}$. Then $I(V') = \{\mathbf{x}_{a_1,b_1}, \mathbf{x}_{a_1,b_2}, \mathbf{x}_{a_2,b_1}\}$, and we have $A(V', \mathbf{x}_{a_1,b_1}) = \{hitcount(a_1), hitcount(b_1)\}$, $A(V', \mathbf{x}_{a_1,b_2}) = \{hitcount(a_1)\}$, and $A(V', \mathbf{x}_{a_2,b_1}) = \{hitcount(b_1)\}$. Thus, the benefit of V' is:

$$\begin{aligned} benefit(V') =& benefit(\mathbf{x}_{a_1,b_1}, \{hitcount(a_1), hitcount(b_1)\}) \\ &+ benefit(\mathbf{x}_{a_1,b_2}, \{hitcount(a_1)\}) \\ &+ benefit(\mathbf{x}_{a_2,b_1}, \{hitcount(b_1)\}) \end{aligned} \tag{7.3}$$

This completes the application of my resource acquisition framework to record matching problems where a support vector machine is used to classify whether record pairs are matches or otherwise, and when the evaluation measure is the total misclassi-

fication cost.

7.3 A Benefit Function for the F_1 Evaluation Measure

Here, I consider the F_1 measure of the match class as the evaluation measure. The formulation of the F_1 measure is sufficiently non-trivial that we cannot simply decompose the benefit function for a set of vertices by itself into separate benefit functions, each for a test instance. There are other such non-trivial evaluation measures, such as B-cubed, and I believe that the ideas in this section can be ported to other evaluation measures of such nature. Unlike the support vector machine based benefit function, which assumed the availability of expected misclassification costs, here I assume the availability of the probability of classifications, which is available in a large number of different classifiers.

For ease of reference, I repeat the definition of the F_1 measure here. Suppose there are N test instances, of which a instances are true matches that are classified correctly, b instances are true matches that are misclassified as mismatches, c instances are true mismatches that are classified correctly, and d instances are true mismatches that are misclassified as matches, and thus $N = a + b + c + d$. The precision P and recall R of the match class is then:

$$P = \frac{a}{a+d} \tag{7.4}$$

$$R = \frac{a}{a+b} \tag{7.5}$$

The F_1 measure is then the harmonic mean of the precision and recall:

$$F_1 = \frac{2 \cdot P \cdot R}{P + R} \tag{7.6}$$

The key problem here is then to estimate the expected F_1 measure when the true matches and mismatches are unknown, before and after the acquisition of attribute values. Here, I assume that the classifier is able to estimate the certainty of its classifi-

cations in the form of probability values, before and after acquisition of an arbitrary subset of its attribute values. Let $p_+(\mathbf{x})$ and $p_-(\mathbf{x})$ be the respective certainties of a test instance $\mathbf{x}$ being a match and mismatch, with $p_+(\mathbf{x}) + p_-(\mathbf{x}) = 1$. As in Section 7.2, I assume that the classification of test instances are independent of each other. I now use $p_+(\cdot)$ and $p_-(\cdot)$ as a proxy to estimate precision and recall, and hence the F_1 measure.

Definition 7.3 (Expected F_1 measure). Let X be the collection of all the test instances in the test set, divided into the subsets X_+ and X_- of test instances which are classified as matches and mismatches respectively. Define $E[a(X)]$, $E[b(X)]$, $E[c(X)]$, and $E[d(X)]$ to be the respective expected number of test instances that are (a) true matches that are classified correctly, (b) true matches that are misclassified as mismatches, (c) true mismatches that are classified correctly, and (d) true mismatches that are misclassified as matches. That is:

$$E[a(X)] = \sum_{\mathbf{x} \in X_+} p_+(\mathbf{x}) \tag{7.7}$$

$$E[b(X)] = \sum_{\mathbf{x} \in X_-} p_+(\mathbf{x}) \tag{7.8}$$

$$E[c(X)] = \sum_{\mathbf{x} \in X_-} p_-(\mathbf{x}) \tag{7.9}$$

$$E[d(X)] = \sum_{\mathbf{x} \in X_+} p_-(\mathbf{x}) \tag{7.10}$$

The expected precision and expected recall are defined as follows:

$$E[P(X)] = \frac{E[a(X)]}{E[a(X)] + E[d(X)]} \tag{7.11}$$

$$E[R(X)] = \frac{E[a(X)]}{E[a(X)] + E[b(X)]} \tag{7.12}$$

The expected F_1 measure is defined by:

$$E[F_1(X)] = \frac{2 \cdot E[P(X)] \cdot E[R(X)]}{E[P(X)] + E[R(X)]} \tag{7.13}$$

Definition 7.4 (Expected F_1 measure after resource acquisitions). Let G be a resource

dependency graph and V' be a feasible subset of vertices in G. Let X be the collection of all the test instances in the test set, and let $X + V'$ denote the test instances with V' acquired. Let $A(V', \mathbf{x})$ denote the attribute values of the test instance $\mathbf{x}$ that are contained in V'. I define the following in an analogous manner:

$$E[a(X + V')] = \sum_{\mathbf{x} \in X_+} p_+(\mathbf{x} + A(V', \mathbf{x})) \tag{7.14}$$

$$E[b(X + V')] = \sum_{\mathbf{x} \in X_-} p_+(\mathbf{x} + A(V', \mathbf{x})) \tag{7.15}$$

$$E[c(X + V')] = \sum_{\mathbf{x} \in X_-} p_-(\mathbf{x} + A(V', \mathbf{x})) \tag{7.16}$$

$$E[d(X + V')] = \sum_{\mathbf{x} \in X_+} p_-(\mathbf{x} + A(V', \mathbf{x})) \tag{7.17}$$

The expected precision and expected recall after acquiring V' are defined as follows:

$$E[P(X + V')] = \frac{E[a(X + V')]}{E[a(X + V')] + E[d(X + V')]} \tag{7.18}$$

$$E[R(X + V')] = \frac{E[a(X + V')]}{E[a(X + V')] + E[b(X + V')]} \tag{7.19}$$

The expected F_1 measure after acquiring V' is defined by:

$$E[F_1(X + V')] = \frac{2 \cdot E[P(X + V')] \cdot E[R(X + V')]}{E[P(X + V')] + E[R(X + V')]} \tag{7.20}$$

Definition 7.5 (Benefit function). The *benefit* of V' is defined as the expected increase in F_1 measure for acquiring V':

$$benefit(V') = E[F_1(X + V')] - E[F_1(X)] \tag{7.21}$$

This benefit function allows me to apply my resource acquisition framework to record matching problems where the evaluation measure is the F_1 measure of the matches.

As an alternative to the F_1 measure of the match class, we can also consider the average F_1 measure of the match class and the mismatch class. In this case, the benefit

function just defined can easily be modified to compute the expected increase in average F_1 measure instead of the expected increase in F_1 measure of only the match class.

In the evaluation that follows, I used a support vector machine as my classifier, both for convenience as well as making results more comparable. The support vector machine can output expected classification certainties in the form of posterior probabilities, as explained in Section 5.3. However, it is important to note that any other classifier that is able to predict expected classification certainties can be used as well.

7.4 Evaluation

Having proposed these two benefit functions, I can now evaluate my proposed hierarchical cost-sensitive resource acquisition algorithm.

7.4.1 Datasets

I evaluated my algorithm on three datasets in two different problems with important real-world applications. The first two datasets, SL-GENOMES and SL-DBLP, are datasets for the task of matching a list of short forms SF to a list of long forms LF, such as "KDD" to "Knowledge Discovery and Data Mining" and "WI-IAT" to "International Conference on Web Intelligence and International Conference on Intelligent Agent Technology". These two datasets were described in Section 3.3 and have very different characteristics. The third dataset, AUTHOR-DBLP, is a dataset for author name disambiguation. This dataset is created by randomly selecting 352 author names from DBLP, and is essentially a much larger version of the dataset used in Section 3.2. The task can be seen as clustering the publication records of each ambiguous author name. More specifically, the input is $L_1, \ldots, L_n$, where each L_i is a list of publication records containing the author name s_i which may represent one or more individuals. The clustering-style task here is to determine which pairs of publication records $(a, b) \in L_i$ belongs to the same individual. One important aspect of this author name disambiguation dataset is that test instances in this dataset are symmetric, in the

SL-DBLP Description: DBLP conference and workshop titles Source: http://www.informatik.uni-trier.de/~ley/db/conf/indexa.html Query: "$\langle title \rangle$" conference OR conferences OR workshop OR workshops
SL-GENOMES Description: Human genome acronym list Source: http://www.ornl.gov/sci/techresources/Human_Genome/acronym.shtml Query: "$\langle title \rangle$" genome OR genomes
AUTHOR-DBLP Description: Author name disambiguation of publications in DBLP Source: http://dblp.uni-trier.de/xml/ Query: "$\langle title \rangle$"

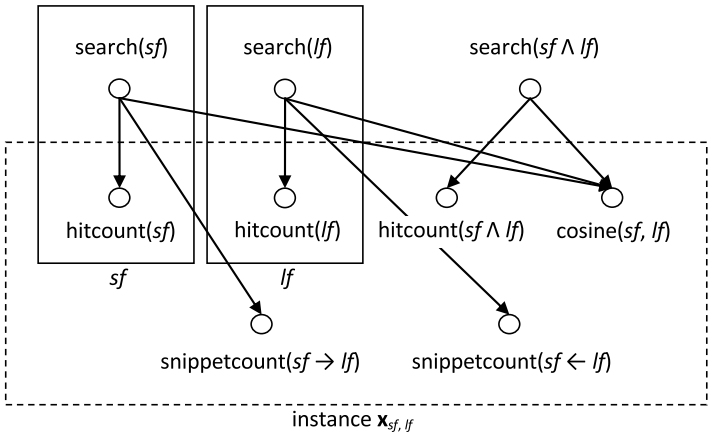
Table 7.1: Descriptions and sources of the evaluation datasets, as well as the form of queries made.

Dataset	SL-GENOMES	SL-DBLP	AUTHOR-DBLP
Test instances	23,409	213,444	27,046
Matching pairs	307	926	20,582
Vertices	117,657	1,069,068	302,266
Edges	140,760	1,281,588	595,012
Misclassification cost	50 / 500	50 / 1,500	1,000 / 200

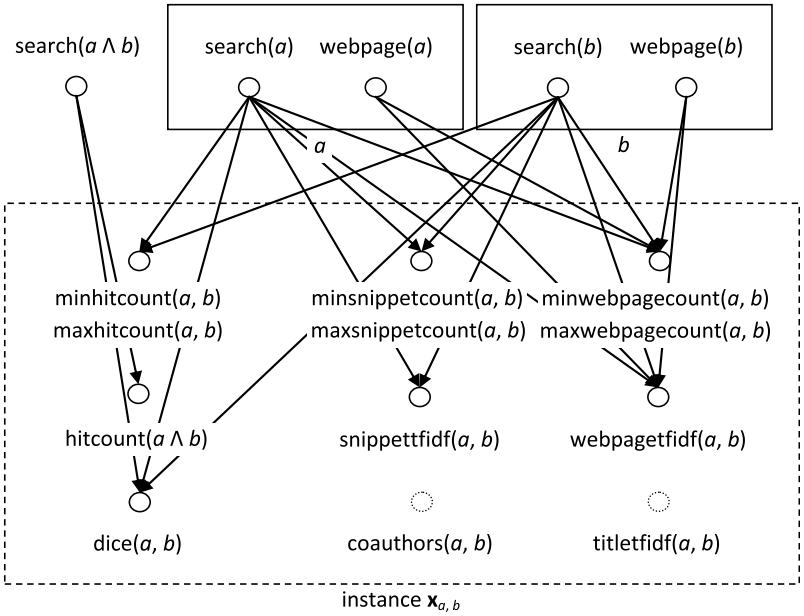
Table 7.2: Statistics of the evaluation datasets.

sense that $\mathbf{x}_{a,b}$ is the same test instance as $\mathbf{x}_{b,a}$. This is due to this problem being of a clustering-style problem, which is very different from the record linkage problem of matching short forms to long forms.

The sources and the form of queries used for web acquisitions for each of these three datasets are shown in Table 7.1, with their characteristics summarized in Table 7.2. The resource dependency graphs of these datasets are shown in Figure 7.2, with the vertex labels described in Table 7.3. The attribute values in the test instances reflect typical attribute values used in related work. I used Google SOAP Search API and retrieved the top-10 results for each query. For SL-GENOMES and SL-DBLP, each query is made up of a short form or a long form plus a few domain specific keywords to promote results of that domain. For AUTHOR-DBLP, each query is the title of a publication record. Each dataset is split into equal training and testing halves; the training half is used to train a linear support vector machine and the resource dependency graph is built on the



(a) Structure of resource dependency graph for SL-GENOMES and SL-DBLP.



(b) Structure of resource dependency graph for AUTHOR-DBLP.

Figure 7.2: Structure of resource dependency graphs. Dotted vertices indicate pre-acquired vertices before starting the acquisition algorithm.

testing half and evaluated. During training, five-fold cross-validation is performed on the training instances to find the optimal regularization parameter of the support vector machine, and the same trained support vector machine classifier is used to classify test instances for all the algorithms I evaluated. For AUTHOR-DBLP, all $coauthor(\cdot, \cdot)$ and $titletfidf(\cdot, \cdot)$ vertices were pre-acquired prior to executing the acquisition algorithm. Note that SL-DBLP has a large resource dependency graph with over 1 million vertices and 1 million edges.

For my experiments, I set the acquisition costs of each $search(\cdot)$, $webpage(\cdot)$, and

Label	Description
$search(a)$	Top-10 search engine results for the query a.
$webpage(a)$	Web pages at URLs in $search(a)$.
$hitcount(a)$	Hit count returned by $search(a)$.
$minhitcount(a, b)$	Minimum of $hitcount(a)$ and $hitcount(b)$.
$maxhitcount(a, b)$	Maximum of $hitcount(a)$ and $hitcount(b)$.
$snippetcount(a \rightarrow b)$	Number of snippets in $search(a)$ containing the term b.
$snippetcount(a \leftarrow b)$	Number of snippets in $search(b)$ containing the term a.
$minsnippetcount(a, b)$	Minimum of $snippetcount(a \rightarrow b)$ and $snippetcount(a \leftarrow b)$.
$maxsnippetcount(a, b)$	Maximum of $snippetcount(a \rightarrow b)$ and $snippetcount(a \leftarrow b)$.
$cosine(a, b)$	$\frac{hitcount(a \wedge b)}{\sqrt{hitcount(a)hitcount(b)}}$.
$dice(a, b)$	$\frac{2 \cdot hitcount(a \wedge b)}{hitcount(a)+hitcount(b)}$.
$snippettfidf(a, b)$	TF-IDF cosine similarity between the tokens of the snippets in $search(a)$ and $search(b)$.
$webpagetfidf(a, b)$	TF-IDF cosine similarity between the tokens of the web pages in $search(a)$ and $search(b)$.
$titletfidf(a, b)$	TF-IDF cosine similarity between the tokens of the titles of the publication records a and b.
$coauthors(a, b)$	Number of common coauthors between the publication records a and b.

Table 7.3: Description of vertex labels in resource dependency graphs.

other unacquired vertex to 10, 100, and 1 respectively. This is because search engine queries take significantly longer to execute than extracting an attribute value, and downloading the top-10 web pages requires ten web accesses compared to a search engine query. Note that with this setting of acquisition costs, all vertices of the same type have the same acquisition cost.

7.4.2 Experimental Setup

For consistency in my experiments within each dataset, I used the same trained linear support vector machine model for all algorithms I evaluated. Also, the same trained model is used for the two benefit functions I developed in this chapter. As my focus is to validate my acquisition framework and not on parameter tuning, I simply selected

parameter values found to be effective for my algorithm. Additionally, I observed that my algorithm is not sensitive to changes in parameter values. For my Tabu search based algorithm, I use a static Tabu tenure of 7 (used in many other Tabu search applications), terminate the algorithm when 20 consecutive iterations have occurred without an increase in the objective value, use a propagation factor λ of 0.5, and an aggregation factor β of 0.1.

I evaluated my algorithm against five baselines:

- **Random.** This algorithm acquires vertices at random until the budget is reached.
- **Least cost.** This algorithm acquires vertices with the least cost until the budget is reached.
- **Best benefit.** This algorithm acquires the vertices with the best benefit until the budget is reached.
- **Best cost-benefit ratio.** This algorithm acquires the vertices with the best benefit to cost ratio until the budget is reached.
- **Best type.** This algorithm acquires the maximum number of vertices of one type not exceeding the budget that maximizes the objective value.

I evaluated each algorithm by starting with a resource dependency graph with no vertices acquired, and then I execute the algorithm for a number of iterations. In each iteration, Each time, the algorithm will be executed with a budget of 100 for SL-GENOMES and SL-DBLP, and a budget of $1{,}000$ for AUTHOR-DBLP. At the end of each execution, I recorded the cumulative acquisition cost of all the vertices acquired so far, as well as its benefit value, giving one data point. All data points for each algorithm are then plotted on a chart of total misclassification cost against total acquisition cost. These charts allow us to compare the performance of the different algorithms.

I also include in these charts a line labeled **Manual**, which estimates the best possible performance. This manual process is performed with the knowledge of which attribute value acquisitions will allow which test instances to be correctly classified by

the classifier. I manually acquired vertices that allowed all instances to be correctly classified while keeping acquisition costs as low as possible, taking advantage of an observation that the required acquisitions for many of the instances fall into a few patterns. This produces one data point, which I joined with the data point where no resource acquisitions have taken place. Naturally, this manual process, guided by expert domain knowledge, will likely outperform any automated algorithm.

7.4.3 Results

The various acquisition algorithms are evaluated on both benefit functions defined in this chapter.

I first perform experiments using the total misclassification cost as the evaluation measure. Here, the objective function used is $obj(cost(V'), benefit(V')) = benefit(V') - cost(V')$, with the benefit function defined in Section 7.2. The misclassification costs of test instances are given in Table 7.2, which are assigned based on the relative ratios of matches to mismatches in these datasets.

The results are shown in Figure 7.3. It is desirable for an algorithm to incur less misclassification cost using less acquisition cost. From these charts, it can be seen that my algorithm significantly outperforms all the baseline algorithms I evaluated against. For the purposes of comparison, I consider the second best algorithm to be the baseline algorithm that gives the least total misclassification cost at any amount of total acquisition cost. In each of the three datasets, both the one-tail Wilcoxon signed-rank test and the one-tail sign test indicates a significant difference between my algorithm and the second best algorithm at the $p < 0.0001$ level. My algorithm is able to reduce the total cost of misclassifications as resource acquisitions are made.

I now consider how much my algorithm improves over the second best algorithm, by considering (a) the difference in misclassification cost between the second best algorithm and the manual process, and (b) the difference in misclassification cost between my algorithm and the manual process. I compute the average improvement in the difference when my algorithm is used instead of the second best. This average is computed

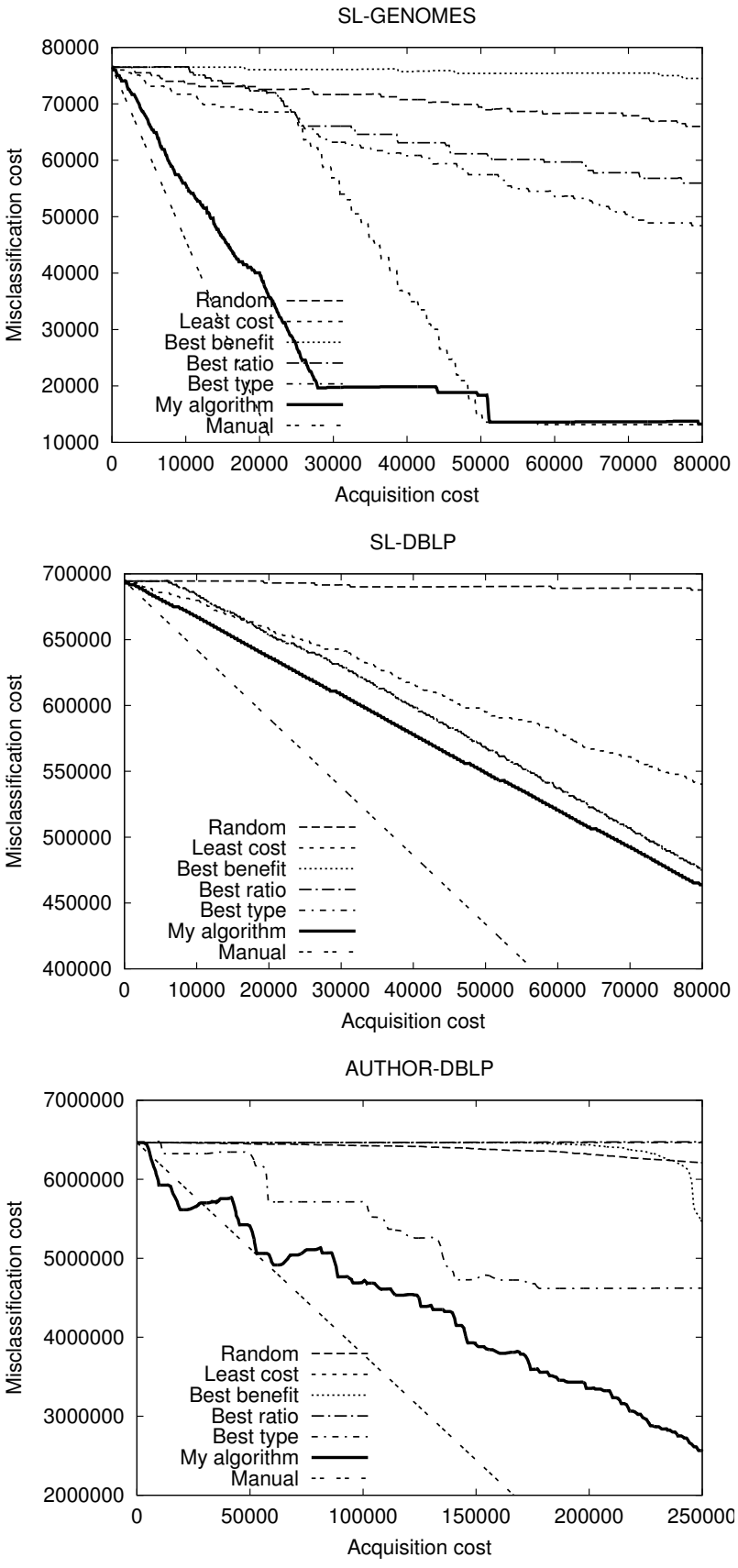


Figure 7.3: Results for using total misclassification cost as the evaluation measure.

using interpolated misclassification costs for different acquisition costs, for acquisition costs at or above 1000 in intervals of 100. For the SL-GENOMES dataset, my algorithm makes an average improvement of 57.1% over the best cost algorithm, which is the second best. For the SL-DBLP dataset, the least cost algorithm is the second best when the total acquisition cost is slightly less than $20{,}000$. When the total acquisition cost is higher, the best benefit, best ratio, and best type algorithms are very tightly clustered together, and can be considered to be the second best algorithm. On average, my algorithm improves over the second best algorithm by 18.9%. For the AUTHOR-DBLP dataset, my algorithm makes an average improvement of 52.2% over the best type algorithm, which is the second best. Importantly, my algorithm consistently holds as the best algorithm over the entire range of acquisition costs plotted in the charts for SL-DBLP and AUTHOR-DBLP, both of which are larger datasets compared to SL-GENOMES. For SL-GENOMES, my algorithm is the best algorithm when the total acquisition cost is less than $48{,}200$, and then closely follows the best cost algorithm when the total acquisition cost is greater than $51{,}200$. The significant aspect of the result for SL-GENOMES is that my algorithm outperforms all the baselines when the total acquisition cost is less, a very desirable outcome.

As can be seen in the charts of these three datasets, as well as the above narrative, the baselines perform very differently. For example, the best benefit algorithm is one of the second best algorithms in SL-DBLP, but is the worst performing algorithm in SL-GENOMES and is one the worst in AUTHOR-DBLP. On the other hand, the best cost algorithm is the second best in SL-GENOMES as well as the beginning of SL-DBLP, but performs very poorly in AUTHOR-DBLP. These results are strong indicators that the three datasets have very different characteristics. The results show that my algorithm significantly outperforms all the baseline algorithms in all three datasets, which are of very different characteristics.

I now examine the search engine queries (*i.e.*, acquisition of $search(\cdot)$ vertices) issued by my algorithm. For this, I consider the test instances that are correctly classified when my acquisition algorithm is used, and compare it with acquiring all the

attribute values of these instances. For the SL-GENOMES dataset, less than 13% of the $search(\cdot)$ vertices were acquired. For the SL-DBLP dataset, less than 1% of the $search(\cdot)$ vertices were acquired. For the AUTHOR-DBLP dataset, less than 15% of the $search(\cdot)$ vertices were acquired. In all three datasets, a large proportion of the search engine queries were saved. Note that each $search(a)$ or $search(b)$ vertex services a large number of test instances, whereas each $search(a \wedge b)$ vertex services only a single test instance. As most of the instances do not require the information in $search(a \wedge b)$ to be classified correctly, my algorithm mostly made search engine calls of the form $search(a)$ or $search(b)$, resulting in significantly less queries being made. This is important not only for saving time, but also allows more work to be done with the search engine in the face of daily quotas or rate limiting.

Next, I perform experiments using the average F_1 measure of the match class and the mismatch class as the evaluation measure. Here, the objective function used is $obj(cost(V'), benefit(V')) = \frac{benefit(V')}{cost(V')}$. The benefit function used is a simple variant of that defined in Section 7.3 – instead of calculating the expected increase in F_1 measure over one class, I calculate the expected increase in F_1 measure averaged over both classes.

The results are shown in Figure 7.4. It is desirable for an algorithm to obtain a higher value for F_1 measure using less acquisition cost. From these charts, it can be seen that my algorithm significantly outperforms all the baseline algorithms I evaluated against. In each of the three datasets, both the one-tail Wilcoxon signed-rank test and the one-tail sign test indicates a significant difference between my algorithm and the second best algorithm at the $p < 0.0001$ level. My algorithm is able to obtain higher F_1 values as resource acquisitions are made.

Again, I consider how much my algorithm improves on the difference in average F_1 measure with the manual process, compared to the second best algorithm. For the SL-GENOMES dataset, the improvement is a whopping 89.8% over the least cost algorithm, which is the second best. For the SL-DBLP dataset, the improvement is 28.7% over the best type algorithm. For the AUTHOR-DBLP dataset, the improvement is

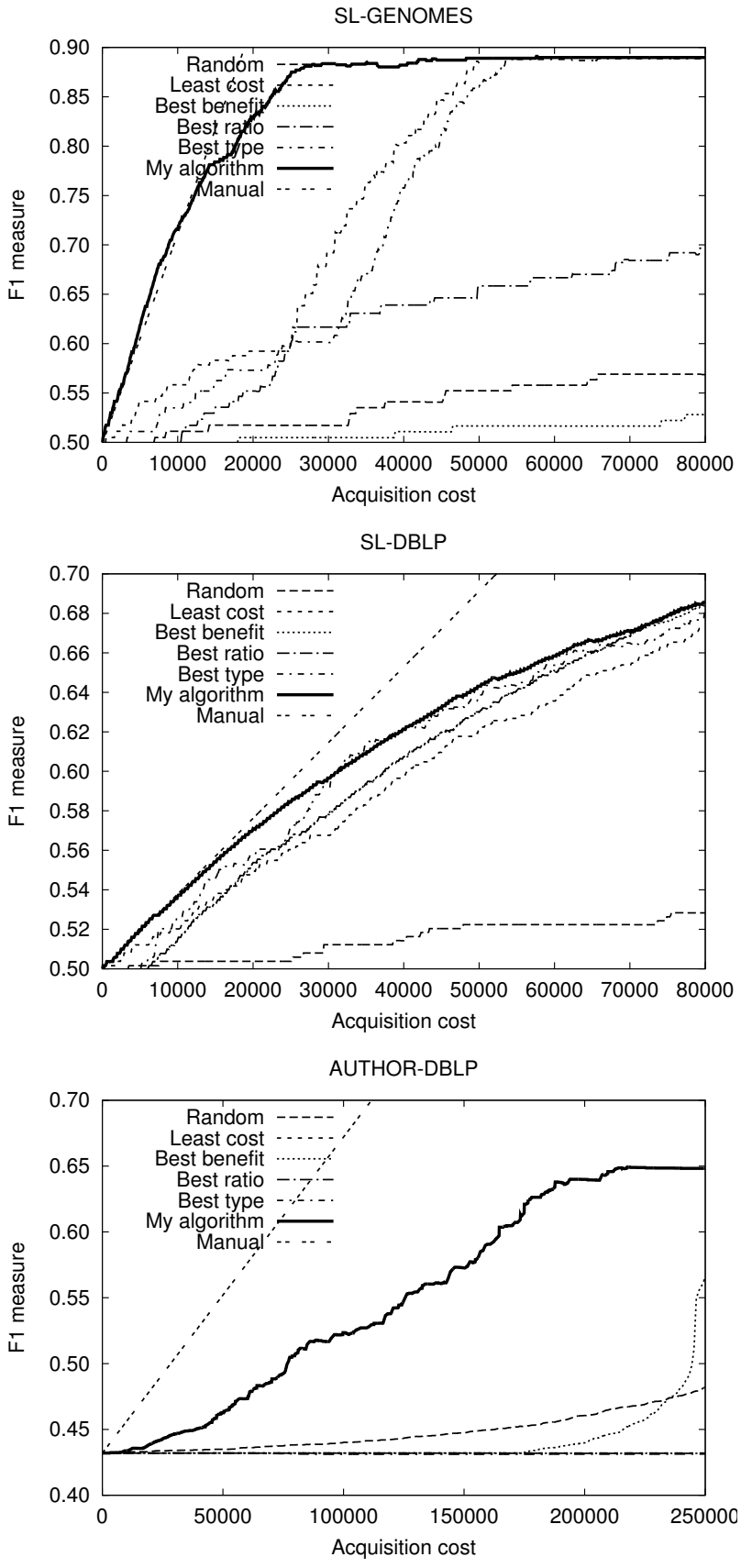


Figure 7.4: Results for using average F_1 measure as the evaluation measure.

31.1% over the random acquisition algorithm. Just like the previous set of experiments when the evaluation measure is the total misclassification cost, my algorithm consistently achieves an average F_1 measure that is at least as good as the baselines, for the whole range of total misclassification costs.

Similar to the case when total misclassification cost is used, the baseline algorithms exhibit very different behaviour across different datasets when the evaluation measure is the average F_1 measure of the match and mismatch classes. Again, the best cost algorithm is the second best algorithm in SL-GENOMES and is among the worst performing in AUTHOR-DBLP. However, there are also differences in the performances of individual baselines in a given dataset when the evaluation measure is changed. The most prominent difference shows up in AUTHOR-DBLP, where the second best algorithm is best type when total misclassification cost is used, but is changed to random acquisition when average F_1 measure is used. Also, the algorithms whose performance are almost indistinguishable in SL-DBLP when total misclassification cost is used becomes not so tightly clustered together when average F_1 measure is used. Despite the differences in characteristics caused by the application of different evaluation measures, my algorithm still outperforms the baselines consistently and significantly across all three datasets. This shows that my algorithm can be applied in a large variety of different problem scenarios.

I now examine the search engine queries issued by my algorithm for the correctly classified instances as before. For the SL-GENOMES dataset, less than 12% of the $search(\cdot)$ vertices were acquired. For the SL-DBLP dataset, less than 1% of the $search(\cdot)$ vertices were acquired. For the AUTHOR-DBLP dataset, less than 23% of the $search(\cdot)$ vertices were acquired. Just like the case when total misclassification cost is used, my algorithm favours the acquisition of the $search(a)$ and the $search(b)$ vertices, over the $search(a \wedge b)$ vertices acquired by making conjunctive queries. Again, this results in significantly less queries being made compared to acquiring all the $search(\cdot)$ vertices in the correctly classified instances.

From these results, I conclude that my algorithm is able to optimize on two differ-

ent kinds of evaluation measures, namely total misclassification cost and average F_1 measure, and outperform all the baseline algorithms in all three datasets.

7.5 Conclusion

In this chapter, I proposed two benefit functions that can be used in conjunction with my hierarchical cost-sensitive resource acquisition framework. These two benefit functions are catered for different evaluation measures: total cost of misclassification costs and average F_1 measure of the test instances. My evaluation on three datasets of very different characteristics, as well as using two different evaluation measures, shows that my acquisition algorithm significantly outperforms a number of baseline algorithms. The evaluation results highlights the versatility of both my resource acquisition framework as well as my resource acquisition algorithm.

One limitation of this work is that evaluation is performed only on the support vector machine classifier. It would be desirable to validate my proposed cost-sensitive acquisition algorithm, together with the benefit functions, on other classifiers such as decision trees and Naïve Bayes. Such validation is left for future work.

In addition, the benefit functions proposed in this chapter are mainly geared towards matching of record pairs. On the other hand, record matching is sometimes part of a larger task, such as clustering (in the traditional sense, where records are divided into different groups). I believe it is also possible to come out with good benefit functions that cater to the overall larger task. For example, for the bottom-up hierarchical agglomerative clustering algorithm as used in Section 3.2, a benefit function might involve estimating the expected change in cluster-to-cluster similarity values for acquiring resources, and hence determine whether resource acquisition is likely to alter the final clustering of the input records. Regardless of the solution, the benefit function must be efficient since it will be called a large number of times.

Chapter 8

Conclusion

8.1 Goals Revisited

Record matching datasets often contain insufficient or ambiguous information that make the matching task difficult. As such, acquiring additional information from external resources can improve record matching solutions. In this thesis, I examined Web-based resources; in particular, results from search engine queries and web page downloads. However, web page downloads are slow, and search engines typically throttle the amount of queries one can make in a day. As such, making a conjunctive query for each pair of records in the dataset can be prohibitively expensive even for a mid-sized dataset.

Before summarizing the work presented in this thesis, it is worth restating the two major goals of this thesis:

- To establish that acquisition of Web-based resources can benefit the task performance of record matching tasks, and

- To propose an algorithm for selective acquisition of web based resources for record matching tasks. It should balance acquisition costs and acquisition benefits, while taking acquisition dependencies between resources into account.

In the remainder, I first recap the contributions of this thesis. Next, I discuss the limitations of my work. Finally, I outline possible directions for future work.

8.2 Contributions

In this thesis, I have made several contributions towards these two goals. The major contributions are the usage of web resources for record matching, followed by a framework for hierarchical cost-sensitive web resource acquisition, with the latter being the main contribution.

8.2.1 Using Web Resources for Record Matching

I demonstrated that Web based resources can be useful for performing record matching tasks for in three different problems: author name disambiguation, linkage of short forms to long forms, and web people search.

For author name disambiguation, I proposed an algorithm that queries the titles of the citations on a search engine, weighing the returned URLs by their inverse host frequencies, and then applying single-link hierarchical agglomerative clustering to disambiguate the different authors sharing the same name. Using cosine similarity on inverse host frequency vectors is simple yet effective for solving this mixed citation problem. Additionally, I also considered a similarity measure using coauthor linkage on a coauthorship graph. This method is complementary to the inverse host frequency method, and can be combined together to form a clustering algorithm with even better disambiguation accuracy.

For linkage of short forms to long forms, I proposed exploiting the external knowledge from the Web to obtain the required contextual information that is typically missing from such data. In the process, I unify similar threads of research (as applied to other tasks) by viewing the task as two related facets: a) query composition, and b) search engine evidence analysis. In my investigation of the linkage of short forms to long forms, I proposed a simple yet effective count-based method that utilizes the snippets of the returned search engine results. My method experimentally outperforms a number of other methods consistently across three datasets of different domains.

For web people search, I presented a brief summary of our PSNUS system that par-

ticipated in the Web People Search Task in SemEval 2007. Our system simply clustered the input web pages using TF-IDF cosine similarity of the named entity tokens, and our system is placed in third position out of sixteen participating systems. Therefore, our conclusion is that named entities can be an important source of information for the Web People Search Task.

8.2.2 A Framework for Adaptively Combining Two Methods for Record Matching

In this work, I continued my investigation of the problem of linking short forms to long forms. First, I proposed a query probing method that makes use of the observation that many long forms in the datasets share common n-grams, or subsequences of tokens. Therefore, querying a common n-gram often yield sufficient information for multiple linkages.

A key aspect of this work is an adaptive combination framework for combining two methods, such that the combined method has the better aspect of each method. In the context of linking short forms to long forms, I combined the count-based method with the query probing method. The resultant method significantly reduced the number of search engine queries required while maintaining linkage performance.

8.2.3 Cost-sensitive Attribute Value Acquisition for Support Vector Machines

In this work, I consider supervised classification of test instances which contain missing attribute values which may individually be acquired at some cost. Therefore, the aim is to achieve a balance between acquisition costs and misclassification costs. While cost-sensitive selective acquisition of attribute values have been applied on other classifiers, I have not seen a realization of this framework for the support vector machine classifier in the published literature, despite it being an effective classifier for a large number of different tasks.

Within this framework, I proposed an algorithm that uses the weight vector of a trained support vector machine to compute, for a test instance and after taking into account its missing attribute values, (a) the certainty of its classification, and (b) its expected misclassification cost. Both quantities can be computed before and after acquiring a subset of its missing attribute values. There are two key contributions in my proposed algorithm. Firstly, my algorithm can be applied to arbitrary subsets of missing attribute values, which is unlike some of the other acquisition algorithms which dictate the order in which dictate the order of acquisition. Secondly, my algorithm is generalizable to arbitrary kernels; it is not limited to just the linear kernel. Armed with a way to compute the expected decrease in misclassification cost for attribute value acquisitions, it is easy to come out with a simple cost-sensitive acquisition algorithm with an appropriate termination condition.

I would like to emphasize the key contribution of the computation of the expected decrease in misclassification cost for arbitrary subsets of missing attribute values. This is the key feature that enables the definition of a benefit function for my subsequent work on hierarchical Web resource acquisition.

8.2.4 A Framework for Hierarchical Cost-sensitive Web Resource Acquisition

In this work, I presented a general framework for performing cost-sensitive resource acquisition problems with hierarchical dependences. This problem framework takes in as input a resource dependency graph, an acquisition cost function (for determining the cost of making particular resource acquisitions), and a benefit function (for determining the benefit value of making particular resource acquisitions), and selects resources to acquire to maximize the benefit of acquisitions while minimizing the acquisition costs. This framework abstracts away from any specific problem instance, and can be applied to many problems involving selective resource acquisitions. This framework may be seen as a generalization of the adaptive combination framework.

The resource acquisition framework can be applied to record matching problems.

However, I identified some unique characteristics of resource dependency graphs for record matching problems that make the cost-sensitive selective acquisition of resources a challenging task. To overcome these problems, I applied a widely-studied metaheuristic search algorithm known as Tabu search. This search algorithm avoids revisiting the same states, allowing a better exploration of the state space. Next, I defined a set of legal moves that are more intelligent. These moves allow the search algorithm to reach a state that includes the leaf vertices faster. Finally, I also introduced a surrogate benefit function that propagates the benefit values from the leaf vertices upwards to the root vertices. This guides the search algorithm towards vertices with the highest benefit values.

8.2.5 Benefit Functions for Record Matching

While an acquisition cost function can be easily obtained or engineered, designing an appropriate benefit function can be a challenge. To complete a resource acquisition algorithm for record matching problems in my framework, this component is crucial. In this setting, I proposed breaking down the problem of finding the benefit of acquiring vertices into the related problem of finding the benefit of acquiring missing attribute values in classification problems. Here, I proposed two benefit functions.

The first benefit function is a straightforward adaptation of work on cost-sensitive acquisition of attribute values in support vector machines. The benefit function for acquiring missing attribute values in test instances is then the difference between the expected decrease in misclassification cost for its acquisition and its acquisition cost. Therefore, this benefit function is suited when the evaluation measure employed is the total misclassification cost of the test instances.

The second benefit function is targeted towards more general evaluation measures. While I focused on the pairwise F_1 measure of the match class, other evaluation measures such as B-cubed could be substituted instead. Here, I made use of the change in classification confidence for making attribute value acquisitions, and used it to estimate the change in precision and recall, and hence the expected increase in F_1 measure.

8.3 Limitations

While this thesis made advances to cost-sensitive selective acquisition of Web-based resources for record matching problems, I readily acknowledge that the work is not without its limitations. Owing to the fairly broad scope of this thesis, some of the aspects have not been fully investigated. For example, it would be beneficial to investigate in more kinds of Web based resources, as well as datasets of a greater variety of domains.

Datasets. To ensure repeatability of the experiments involving search engine results and web page downloads, I elected to pre-download all Web-based resources that are considered and save them. This also has the advantage of running experiments that simulates the Web resource acquisitions without actually spending time downloading them, and also allow me to have all the search engine results before Google terminated its SOAP Search API. However, while a number of record matching datasets are readily available, they do not come with search engine results. Therefore, I had to acquire these Web-based resources, which takes more than a week for the smaller datasets, and more than a month for the larger datasets. This time lag placed a severe limitation on the number of datasets I am able to work with, even though I acknowledge that having more datasets from varying domains would benefit my thesis work. Note that this time lag refers to the acquisition of resources, and not the overhead in training and running my proposed algorithm, which is quite fast.

Estimation of misclassification costs. Perhaps a biggest obstacle to applying my resource acquisition framework is in estimating expected benefit in an efficient manner when the evaluation measure involved is not a simple linear combination of the number of classifications and misclassifications. While accuracy is widely known not be a good evaluation measure for unbalanced datasets, varying misclassification costs have also been criticized for being hard to interpret. This prompted me to perform work when the evaluation measure is pairwise F_1. However as it is not a simple linear combination, I found that a fairly substantial amount of engineering work is needed to allow it to be computed as efficiently as possible, without having to make a pass through every

single instance each time it needs to be computed. With more sophisticated evaluation measures, such as B-cubed, I anticipate that the amount of engineering work required to estimate it would be formidable.

Statistics from downloaded corpora. Some Web-based techniques download a collection of documents from the Web, either through query probing or otherwise, and treat it as a corpus where statistics are gathered, and then these statistics are then used to solve the task in hand. However, the resource acquisition framework in its current form does not make it easy to estimate the amount of error that would be incurred if only a subset of the collection is downloaded rather than the full collection. This prevents the resource acquisition framework from being applied to such kinds of Web-based resources.

8.4 Future Work

My thesis can benefit from additional work that addresses its limitations. Here, I outline more possible directions for future work.

More complex acquisition cost models. In my proposed resource acquisition framework, I assumed that the cost of acquiring different web resources are independent of each other. However, some search engines support the bundling of multiple queries into a single web request, such as the Yahoo! Query Language from Yahoo!. As such, the acquisition cost of running a number of queries individually one after another can be quite different from that of running them as a single bundle. Therefore, an interesting direction for future work is to generalize my resource acquisition framework to such more complex acquisition cost models. While there is some preliminary work that takes such more complex cost models in collecting search engine data into account, such as [Nuray-Turan, 2011] and [Kothari, 2011], the work is done in a different context and does not directly apply in this thesis.

Parallel or distributed algorithms for resource acquisition. Throughout this thesis, I have considered only sequential algorithms that runs on a single machine. It is

noted that the time spent on downloading web pages is dominated by network I/O costs rather than CPU computational costs. As such, it would be beneficial to consider parallel or distributed algorithms that may be run on multiple machines. However, search engines are known to impose daily quotas based on IP addresses or application keys issued by search engine providers, making parallelism of limited usefulness for search engine resources. Nevertheless, parallel or distributed algorithms can be useful as we consider the conjunction of different kinds of Web-based resources, such as downloading web pages while waiting for the results of a search engine query, or when we consider even more kinds of Web-based resources. Still, there is a limit on the amount of network transfer the internet connection allows at any point of time.

Relating the work to existing work on set coverage and multi-objective optimization. Another direction for future work is to relate my algorithms to existing work on set coverage and multi-objective optimization [Papadimitriou and Yannakakis, 2001], such as the approach used in [Hore et al., 2004]. This can allow me to establish some theoretical properties of my algorithms.

Bibliography

[Aizawa and Oyama, 2005] Aizawa, A. and Oyama, K. (2005). A fast linkage detection scheme for multi-source information integration. In *International Workshop on Challenges in Web Information Retrieval and Integration (WIRI)*, pages 30–39.

[Ao and Takagi, 2005] Ao, H. and Takagi, T. (2005). ALICE: An algorithm to extract abbreviations from MEDLINE. *Journal of the American Medical Informatics Association*, 12(5):576–586.

[Apolloni et al., 2004] Apolloni, B., Marinaro, M., and Tagliaferri, R. (2004). An algorithm for reducing the number of support vectors. In *Italian Workshop on Neural Nets (WIRN VIETRI)*, pages 99–105.

[Archetti et al., 2006] Archetti, C., Speranza, M. G., and Hertz, A. (2006). A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science*, 40(1):64–73.

[Artiles et al., 2007] Artiles, J., Gonzalo, J., and Sekine, S. (2007). The SemEval-2007 WePS evaluation: Establishing a benchmark for the Web People Search Task. In *International Workshop on Semantic Evaluations (SemEval)*, pages 64–69.

[Asuncion and Newman, 2007] Asuncion, A. and Newman, D. J. (2007). UCI machine learning repository. Available at http://archive.ics.uci.edu/ml/.

[Aumüller, 2009] Aumüller, D. (2009). Towards web supported identification of top affiliations from scholarly papers. In *Datenbanksysteme in Business, Technologie und Web (BTW)*, pages 237–246.

[Aumüller and Rahm, 2009] Aumüller, D. and Rahm, E. (2009). Web-based affiliation matching. In *International Conference on Information Quality (ICIQ)*.

[Bell and Dravis, 2006] Bell, R. and Dravis, F. (2006). Is your data dirty? (and does that matter?). Accenture White Paper.

[Bhattacharya and Getoor, 2004] Bhattacharya, I. and Getoor, L. (2004). Iterative record linkage for cleaning and integration. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD)*, pages 11–18.

[Bhattacharya and Getoor, 2006] Bhattacharya, I. and Getoor, L. (2006). A latent dirichlet model for unsupervised entity resolution. In *SIAM International Conference on Data Mining (SDM)*, pages 47–58.

[Bilenko et al., 2006] Bilenko, M., Kamath, B., and Mooney, R. J. (2006). Adaptive blocking: Learning to scale up record linkage and clustering. In *IEEE International Conference on Data Mining (ICDM)*, pages 87–96.

[Bilenko and Mooney, 2003] Bilenko, M. and Mooney, R. J. (2003). Adaptive duplicate detection using learnable string similarity measures. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 39–48.

[Bilenko et al., 2003] Bilenko, M., Mooney, R. J., Cohen, W. W., Ravikumar, P., and Fienberg, S. E. (2003). Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23.

[Bollegala et al., 2006a] Bollegala, D., Matsuo, Y., and Ishizuka, M. (2006a). Disambiguating personal names on the web using automatically extracted key phrases. In *European Conference on Artificial Intelligence (ECAI)*, pages 553–557.

[Bollegala et al., 2006b] Bollegala, D., Matsuo, Y., and Ishizuka, M. (2006b). Extracting key phrases to disambiguate personal names on the web. In *International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, pages 223–234.

[Bollegala et al., 2007] Bollegala, D., Matsuo, Y., and Ishizuka, M. (2007). Measuring semantic similarity between words using web search engines. In *International Conference on World Wide Web (WWW)*, pages 757–766.

[Broadbent and Iwig, 1999] Broadbent, K. and Iwig, B. (1999). Record linkage at NASS using AutoMatch. In *Federal Committee on Statistical Methodology Research Conference*.

[Buntine, 1994] Buntine, W. L. (1994). Operations for learning with graphical models. *Journal of Artificial Intelligence Research (JAIR)*, 2(1):159–225.

[Burges, 1996] Burges, C. J. C. (1996). Simplified support vector decision rules. In *International Conference on Machine Learning (ICML)*, pages 71–77.

[Burges, 1998] Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.

[Buscaldi and Rosso, 2006] Buscaldi, D. and Rosso, P. (2006). Mining knowledge from Wikipedia from the question answering task. In *International Conference on Language Resources and Evaluation (LREC)*, pages 727–730.

[Callan and Connell, 2001] Callan, J. P. and Connell, M. E. (2001). Query-based sampling of text databases. *ACM Transactions on Information Systems (TOIS)*, 19(2):97–130.

[Chai et al., 2004] Chai, X., Deng, L., Yang, Q., and Ling, C. X. (2004). Test-cost sensitive naive bayes classification. In *IEEE International Conference on Data Mining (ICDM)*, pages 51–58.

[Chang and Lin, 2001] Chang, C.-C. and Lin, C.-J. (2001). *LIBSVM: a library for support vector machines*. Available at http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

[Chang et al., 2002] Chang, J. T., Schütze, H., and Altman, R. B. (2002). Creating an online dictionary of abbreviations from MEDLINE. *Journal of the American Medical Informatics Association*, 9(6):612–620.

[Chen and Martin, 2007] Chen, Y. and Martin, J. H. (2007). Cu-comsem: Exploring rich features for unsupervised web personal name disambiguation. In *International Workshop on Semantic Evaluations (SemEval)*, pages 125–128.

[Chen et al., 2009] Chen, Z., Kalashnikov, D. V., and Mehrotra, S. (2009). Exploiting context analysis for combining multiple entity resolution systems. In *ACM SIGMOD International Conference on Management of Data*, pages 207–218.

[Christen, 2007] Christen, P. (2007). Towards parameter-free blocking for scalable record linkage. Technical Report TR-CS-07-03, Australian National University.

[Cimiano et al., 2004] Cimiano, P., Handschuh, S., and Staab, S. (2004). Towards the self-annotating web. In *International Conference on World Wide Web (WWW)*, pages 462–471.

[Cimiano et al., 2005] Cimiano, P., Ladwig, G., and Staab, S. (2005). Gimme' the context: Context-driven automatic semantic annotation with C-PANKOW. In *International Conference on World Wide Web (WWW)*, pages 332–341.

[Cohen et al., 2003] Cohen, W. W., Ravikumar, P., and Fienberg, S. E. (2003). A comparison of string distance metrics for name-matching tasks. In *Information Integration on the Web (IIWeb)*, pages 73–78.

[Crainic et al., 2009] Crainic, T. G., Perboli, G., and Tadei, R. (2009). TS^2PACK: A two-level tabu search for the three-dimensional bin packing problem. *European Journal of Operational Research*, 195(3):744–760.

[Davis et al., 2006] Davis, J. V., Ha, J., Rossbach, C. J., Ramadan, H. E., and Witchel, E. (2006). Cost-sensitive decision tree learning for forensic classification. In *European Conference on Machine Learning (ECML)*, pages 622–629.

[de Kunder, 2006] de Kunder, M. (2006). Geschatte grootte van het geïndexeerde world wide web. Master's thesis, Universiteit van Tilburg.

[de Vries et al., 2009] de Vries, T., Ke, H., Chawla, S., and Christen, P. (2009). Robust record linkage blocking using suffix arrays. In *ACM Conference on Information and Knowledge Management (CIKM)*, pages 305–314.

[Dekel and Shamir, 2008] Dekel, O. and Shamir, O. (2008). Learning to classify with missing and corrupted features. In *International Conference on Machine Learning (ICML)*, pages 216–223.

[Dong et al., 2005] Dong, X., Halevy, A., and Madhavan, J. (2005). Reference reconciliation in complex information spaces. In *ACM SIGMOD International Conference on Management of Data*, pages 85–96.

[Dunn, 1946] Dunn, H. L. (1946). Record linkage. *American Journal of Public Health*, 36(12):1412–1416.

[Elkan, 2001] Elkan, C. (2001). The foundations of cost-sensitive learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 973–978.

[Elmacioglu, 2008] Elmacioglu, E. (2008). *Effective Solutions for Name Linkage and their Applications*. PhD thesis, The Pennsylvania State University.

[Elmacioglu et al., 2007a] Elmacioglu, E., Kan, M.-Y., Lee, D., and Zhang, Y. (2007a). Web based linkage. In *ACM International Workshop on Web Information and Data Management (WIDM)*, pages 121–128.

[Elmacioglu et al., 2007b] Elmacioglu, E., Tan, Y. F., Yan, S., Kan, M.-Y., and Lee, D. (2007b). PSNUS: Web people name disambiguation by simple clustering with rich features. In *International Workshop on Semantic Evaluations (SemEval)*, pages 268–271.

[Elmagarmid et al., 2007] Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S. (2007). Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 19(1):1–16.

[Fazly et al., 2005] Fazly, A., North, R., and Stevenson, S. (2005). Automatically distinguishing literal and figurative usages of highly polysemous verbs. In *ACL-SIGLEX Workshop on Deep Lexical Acquisition*, pages 38–47.

[Feitelson, 2004] Feitelson, D. G. (2004). On identifying name equivalences in digital libraries. *Information Research*, 9(4).

[Fellbaum, 1998] Fellbaum, C., editor (1998). *WordNet: An Electronic Lexical Database*. MIT Press.

[Fellegi and Sunter, 1969] Fellegi, I. P. and Sunter, A. B. (1969). A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210.

[Fung et al., 2007] Fung, G., Rosales, R., and Rao, R. B. (2007). Feature selection and kernel design via linear programming. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 786–791.

[Garfield, 1994] Garfield, E. (1994). The impact factor. *Current Contents*, 25:3–7.

[Giles et al., 1998] Giles, C. L., Bollacker, K. D., and Lawrence, S. (1998). CiteSeer: An automatic citation indexing system. In *ACM Conference on Digital libraries*, pages 89–98.

[Globerson and Roweis, 2006] Globerson, A. and Roweis, S. (2006). Nightmare at test time: robust learning by feature deletion. In *International Conference on Machine Learning (ICML)*, pages 353–360.

[Glover, 1990] Glover, F. (1990). Tabu search: A tutorial. *Interfaces*, 20(4):74–94.

[Glover and Martí, 2006] Glover, F. and Martí, R. (2006). *Metaheuristic Procedures for Training Neural Networks*, chapter Tabu Search, pages 53–69. Springer.

[Goiser and Christen, 2006] Goiser, K. and Christen, P. (2006). Towards automated record linkage. In *Australasian Conference on Data Mining and Analystics (AusDM)*, pages 23–31.

[Gravano et al., 2003] Gravano, L., Ipeirotis, P. G., and Sahami, M. (2003). QProber: A system for automatic classification of hidden-web databases. *ACM Transactions on Information Systems (TOIS)*, 21(1):1–41.

[Grefenstette and Nioche, 2000] Grefenstette, G. and Nioche, J. (2000). Estimation of English and non-English language use on the WWW. In *Recherche d'Information Assistée par Ordinateur (RIAO)*, pages 237–246.

[Greiner et al., 2002] Greiner, R., Grove, A. J., and Roth, D. (2002). Learning cost-sensitive active classifiers. *Artificial Intelligence*, 139(2):137–174.

[Gu et al., 2003] Gu, L., Baxter, R., Vickers, D., and Rainsford, C. (2003). Record linkage: Current practice and future directions. Technical Report 03/83, CSIRO Mathematical and Information Sciences.

[Gu and Baxter, 2004] Gu, L. and Baxter, R. A. (2004). Adaptive filtering for efficient record linkage. In *SIAM International Conference on Data Mining (SDM)*, pages 477–481.

[Gulli and Signorini, 2005] Gulli, A. and Signorini, A. (2005). The indexable web is more than 11.5 billion pages. In *International Conference on World Wide Web (WWW)*, pages 902–903.

[Han et al., 2005] Han, H., Zha, H., and Giles, C. L. (2005). Name disambiguation in author citations using a K-way spectral clustering method. In *ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 334–343.

[Hearst, 1992] Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *Conference on Computational Linguistics (COLING)*, pages 539–545.

[Hernández, 1996] Hernández, M. A. (1996). *A Generalization of Band Joins and The Merge/Purge Problem*. PhD thesis, Columbia University.

[Hirsch, 2005] Hirsch, J. E. (2005). An index to quantify an individual's scientific research output. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 102(46):16569–16572.

[Hore et al., 2004] Hore, B., Hacigümüs, H., Iyer, B. R., and Mehrotra, S. (2004). Indexing text data under space constraints. In *ACM International Conference on Information and Knowledge Management (CIKM)*, pages 198–207.

[Hull, 1993] Hull, D. (1993). Using statistical testing in the evaluation of retrieval experiments. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 329–338.

[Ipeirotis et al., 2006] Ipeirotis, P. G., Agichtein, E., Jain, P., and Gravano, L. (2006). To search or to crawl? Towards a query optimizer for text-centric tasks. In *ACM SIGMOD International Conference on Management of Data*, pages 265–276.

[Jain et al., 2007] Jain, A., Cucerzan, S., and Azzam, S. (2007). Acronym-expansion recognition and ranking on the web. In *IEEE International Conference on Information Reuse and Integration (IRI)*, pages 209–214.

[Jaro, 1989] Jaro, M. A. (1989). Advances in record-linkage methodology as applied to matching the 1985 Census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406):414–420.

[Ji and Carin, 2007] Ji, S. and Carin, L. (2007). Cost-sensitive feature acquisition and classification. *Pattern Recognition*, 40(5):1474–1485.

[Jin et al., 2003] Jin, L., Li, C., and Mehrotra, S. (2003). Efficient record linkage in large data sets. In *International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 137–146.

[Kalashnikov and Mehrotra, 2006] Kalashnikov, D. V. and Mehrotra, S. (2006). Domain-independent data cleaning via analysis of entity-relationship graph. *ACM Transactions on Database Systems (TODS)*, 31(2):716–767.

[Kalashnikov et al., 2008] Kalashnikov, D. V., Nuray-Turan, R., and Mehrotra, S. (2008). Towards breaking the quality curse: a web-querying approach to web people search. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 27–34.

[Kan and Nguyen Thi, 2005] Kan, M.-Y. and Nguyen Thi, H. O. (2005). Fast webpage classification using URL features. In *ACM Conference on Information and Knowledge Management (CIKM)*, pages 325–326.

[Kan and Tan, 2008] Kan, M.-Y. and Tan, Y. F. (2008). Record matching in digital library metadata. *Communications of the ACM (CACM)*, 51(2):91–94.

[Kanani and McCallum, 2007] Kanani, P. and McCallum, A. (2007). Resource-bounded information gathering for correlation clustering. In *Computational Learning Theory (COLT)*, pages 625–627.

[Kanani and Melville, 2008] Kanani, P. and Melville, P. (2008). Prediction-time active feature-value acquisition for customer targeting. In *NIPS Workshop on Cost Sensitive Learning*.

[Kautz et al., 1997] Kautz, H. A., Selman, B., and Shah, M. A. (1997). The hidden web. *AI Magazine*, 18(2):27–36.

[Kochenberger et al., 2005] Kochenberger, G. A., Glover, F., Alidaee, B., and Wang, H. (2005). Clustering of microarray data via clique partitioning. *Journal of Combinatorial Optimization*, 10(1):77–92.

[Kothari, 2011] Kothari, V. (2011). Exploring client side adaptations for optimizing web search applications. Master's thesis, University of California, Irvine.

[Kwok et al., 2007] Kwok, K.-L., Grunfelda, L., and Deng, P. (2007). Employing web mining and data fusion to improve weak ad hoc retrieval. *Information Processing and Management*.

[Larsen and Rubin, 2001] Larsen, M. D. and Rubin, D. B. (2001). Iterative automated record linkage using mixture models. *Journal of the American Statistical Association*, 96(10):32–41.

[Lee et al., 2005] Lee, D., On, B.-W., Kang, J., and Park, S. (2005). Effective and scalable solutions for mixed and split citation problems in digital libraries. In *ACM SIGMOD Workshop on Information Quality in Information Systems (IQIS)*, pages 69–76.

[Lee et al., 2004] Lee, M. L., Hsu, W., and Kothari, V. (2004). Cleaning the spurious links in data. *IEEE Intelligent Systems*, 19(2):28–33.

[Levenshtein, 1966] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

[Ley, 2002] Ley, M. (2002). The DBLP computer science bibliography: Evolution, research issues, perspectives. In *International Symposium on String Processing and Information Retrieval (SPIRE)*, pages 1–10.

[Li et al., 2006] Li, C., Jin, L., and Mehrotra, S. (2006). Supporting efficient record linkage for large data sets using mapping techniques. *World Wide Web*, 9(4):557–584.

[Li et al., 1995] Li, X., Szpakowicz, S., and Matwin, S. (1995). A WordNet-based algorithm for word sense disambiguation. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1368–1374.

[Lin and Lin, 2003] Lin, K.-M. and Lin, C.-J. (2003). A study on reduced support vector machines. *IEEE Transactions on Neural Networks*, 14(6):1449–1559.

[Ling et al., 2006] Ling, C. X., Sheng, V. S., and Yang, Q. (2006). Test strategies for cost-sensitive decision trees. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 18(8):1055–1067.

[Ling et al., 2004] Ling, C. X., Yang, Q., Wang, J., and Zhang, S. (2004). Decision trees with minimal costs. In *International Conference on Machine Learning (ICML)*, page 69.

[Low et al., 2001] Low, W. L., Lee, M. L., and Ling, T. W. (2001). A knowledge-based approach for duplicate elimination in data cleaning. *Information Systems: Special Issue on Data Extraction, Cleaning and Reconciliation*, 26(8):585–606.

[Malin et al., 2005] Malin, B., Airoldi, E., and Carley, K. M. (2005). A network analysis model for disambiguation of names in lists. *Computational and Mathematical Organization Theory*, 11(2):119–139.

[Mani and Sundaram, 2007] Mani, A. and Sundaram, H. (2007). Modeling user context with applications to media retrieval. *Multimedia Systems*, 12(4-5):339–353.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

[Marshall, 1947] Marshall, J. T. (1947). Canada's national vital statistics index. *Population Studies*, 1(2):204–211.

[Marzal and Vidal, 1993] Marzal, A. and Vidal, E. (1993). Computation of normalized edit distance and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):926–932.

[Matsuo et al., 2006] Matsuo, Y., Mori, J., Hamasaki, M., Ishida, K., Nishimura, T., Takeda, H., Hasida, K., and Ishizuka, M. (2006). POLYPHONET: An advanced social network extraction system from the web. In *International Conference on World Wide Web (WWW)*, pages 397–406.

[McCallum et al., 2000] McCallum, A., Nigam, K., and Ungar, L. (2000). Efficient clustering of high-dimensional data sets with application to reference matching. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 169–178.

[Michelson and Knoblock, 2006] Michelson, M. and Knoblock, C. A. (2006). Learning blocking schemes for record linkage. In *National Conference on Artificial Intelligence (AAAI)*.

[Mihalcea and Moldovan, 1999] Mihalcea, R. and Moldovan, D. I. (1999). A method for word sense disambiguation of unrestricted text. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 152–158.

[Nasraoui and Krishnapuram, 2002] Nasraoui, O. and Krishnapuram, R. (2002). One step evolutionary mining of context sensitive associations and web navigation patterns. In *SIAM International Conference on Data Mining (SDM)*, pages 531–547.

[Needleman and Wunsch, 1970] Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 148(3):443–453.

[Newcombe et al., 1959] Newcombe, H. B., Kennedy, J. M., Axford, S. J., and James, A. P. (1959). Automatic linkage of vital records. *Science*, 130(3381):954–959.

[Ng et al., 1999] Ng, K. W., Wang, Z., Muntz, R. R., and Nittel, S. (1999). Dynamic query re-optimization. In *International Conference on Statistical and Scientific Database Management (SSDBM)*, pages 264–273.

[Nguyen and Ho, 2005] Nguyen, D. and Ho, T. (2005). An efficient method for simplifying support vector machines. In *International Conference on Machine Learning (ICML)*, pages 617–624.

[Nuray-Turan, 2011] Nuray-Turan, R. (2011). *High Quality Entity Resolution with Adaptive Similarity Functions*. PhD thesis, University of California, Irvine.

[Oh and Isahara, 2008] Oh, J.-H. and Isahara, H. (2008). Hypothesis selection in machine transliteration: A web mining approach. In *International Joint Conference on Natural Language Processing (IJCNLP)*, pages 233–240.

[Okazaki and Ananiadou, 2006] Okazaki, N. and Ananiadou, S. (2006). Building an abbreviation dictionary using a term recognition approach. *Bioinformatics*, 22(24):3089–3095.

[On et al., 2005] On, B.-W., Lee, D., Kang, J., and Mitra, P. (2005). Comparative study of name disambiguation problem using a scalable blocking-based framework. In *ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 344–353.

[Öncan et al., 2008] Öncan, T., Cordeau, J.-F., and Laporte, G. (2008). A tabu search heuristic for the generalized minimum spanning tree problem. *European Journal of Operational Research*, 191(2):306–319.

[Osuna et al., 1997] Osuna, E. E., Freund, R., and Girosi, F. (1997). Support vector machines: Training and applications. Technical Report AIM-1602, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

[Papadimitriou and Yannakakis, 2001] Papadimitriou, C. H. and Yannakakis, M. (2001). Multiobjective query optimization. In *ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 52–59.

[Pedersen et al., 2004] Pedersen, T., Patwardhan, S., and Michelizzi, J. (2004). WordNet::Similarity - measuring the relatedness of concepts. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 38–41.

[Pei et al., 2005] Pei, J., Jiang, D., and Zhang, A. (2005). On mining cross-graph quasi-cliques. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 228–238.

[Pelckmans et al., 2005] Pelckmans, K., De Brabanter, J., Suykens, J. A. K., and De Moor, B. (2005). Handling missing values in support vector machine classifiers. *Neural Networks*, 18(5-6):684–692.

[Pereira et al., 2009] Pereira, D. A., Ribeiro-Neto, B., Ziviani, N., Laender, A. H. F., Gonçalves, M. A., and Ferreira, A. A. (2009). Using web information for author name disambiguation. In *ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 49–58.

[Petricek et al., 2005] Petricek, V., Cox, I. J., Han, H., Councill, I., and Giles, C. L. (2005). A comparison of on-line computer science citation databases. In *European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, pages 438–449.

[Platt, 2000] Platt, J. (2000). Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74.

[Popescu and Magnini, 2007] Popescu, O. and Magnini, B. (2007). Irst-bp: Web people search using name entities. In *International Workshop on Semantic Evaluations (SemEval)*, pages 195–198.

[Reuther, 2006] Reuther, P. (2006). Personal name matching: New test collections and a social network based approach. Technical Report Mathematics/Computer Science 06-01, University of Trier.

[Saar-Tsechansky et al., 2009] Saar-Tsechansky, M., Melville, P., and Provost, F. (2009). Active feature-value acquisition. *Management Science*, 55(4):664–684.

[Saar-Tsechansky and Provost, 2007] Saar-Tsechansky, M. and Provost, F. (2007). Handling missing values when applying classification models. *Journal of Machine Learning Research*, 8:1623–1657.

[Sahami and Heilman, 2006] Sahami, M. and Heilman, T. D. (2006). A web-based kernel function for measuring the similarity of short text snippets. In *International Conference on World Wide Web (WWW)*, pages 377–386.

[Schwartz and Hearst, 2003] Schwartz, A. S. and Hearst, M. A. (2003). A simple algorithm for identifying abbreviation definitions in biomedical text. In *Pacific Symposium on Biocomputing (PSB)*, pages 451–462.

[Sharoff, 2006] Sharoff, S. (2006). Creating general-purpose corpora using automated search engine queries. In *WaCky! Working papers on the Web as Corpus*, pages 63–98.

[Smalheiser and Torvik, 2009] Smalheiser, N. R. and Torvik, V. I. (2009). Author name disambiguation. *Annual Review of Information Science and Technology (ARIST)*, 43.

[Smith and Waterman, 1981] Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197.

[Smola and Vishwanathan, 2005] Smola, A. J. and Vishwanathan, S. V. N. (2005). Kernel methods for missing variables. In *International Workshop on Artificial Intelligence and Statistics (AISTATS)*, pages 325–332.

[Snae, 2007] Snae, C. (2007). A comparison and analysis of name matching algorithms. *World Academy of Science, Engineering and Technology (WASET)*, 19:252–257.

[Sun et al., 2006] Sun, R., Ong, C.-H., and Chua, T.-S. (2006). Mining dependency relations for query expansion in passage retrieval. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 382–389.

[Tan et al., 2008] Tan, Y. F., Elmacioglu, E., Kan, M.-Y., and Lee, D. (2008). Efficient web-based linkage of short to long forms. In *International Workshop on the Web and Databases (WebDB)*.

[Tan and Kan, 2010] Tan, Y. F. and Kan, M.-Y. (2010). Hierarchical cost-sensitive web resource acquisition for record matching. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 382–389.

[Tan et al., 2006] Tan, Y. F., Kan, M.-Y., and Lee, D. (2006). Search engine driven author disambiguation. In *ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 314–315.

[Tang et al., 2008] Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., and Su, Z. (2008). ArnetMiner: Extraction and mining of academic social networks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 990–998.

[Torii et al., 2006] Torii, M., Liu, H., Hu, Z., and Wu, C. (2006). A comparison study of biomedical short form definition detection algorithms. In *International Workshop on Text Mining in Bioinformatics (TMBIO)*, pages 52–59.

[Turney, 1995] Turney, P. D. (1995). Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research (JAIR)*, 2:369–409.

[Turney, 2000] Turney, P. D. (2000). Types of cost in inductive concept learning. In *ICML 2000 Workshop on Cost-Sensitive Learning*, pages 15–21.

[Wellner et al., 2004] Wellner, B., McCallum, A., Peng, F., and Hay, M. (2004). An integrated, conditional model of information extraction and coreference with application to citation matching. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 593–601.

[Wheatley, 2004] Wheatley, M. (2004). Operation clean data. *CIO Magazine*.

[Winkler, 2005] Winkler, W. E. (2005). Approximate string comparator search strategies for very large administrative lists. Technical Report RRS2005/02, U.S. Bureau of the Census.

[Winkler, 2006] Winkler, W. E. (2006). Overview of record linkage and current research directions. Technical Report RRS2006/02, U.S. Bureau of the Census.

[Winkler and Thibaudeau, 1991] Winkler, W. E. and Thibaudeau, Y. (1991). An application of the Fellegi-Sunter Model of record linkage to the 1990 U.S. Decennial Census. Technical Report RR91/09, U.S. Bureau of the Census.

[Witten and Frank, 2005] Witten, I. H. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, second edition.

[Yan et al., 2007] Yan, S., Lee, D., Kan, M.-Y., and Giles, C. L. (2007). Adaptive sorted neighborhood methods for efficient record linkage. In *ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 185–194.

[Zhao and Ram, 2005] Zhao, H. and Ram, S. (2005). Entity identification for heterogeneous database integration – a multiple classifier system approach and empirical evaluation. *Information Systems*, 30(2):119–132.

[Zhu and Wu, 2004] Zhu, X. and Wu, X. (2004). Data acquisition with active and impact-sensitive instance selection. In *IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 721–726.

[Zhu et al., 2004] Zhu, Y., Rundensteiner, E. A., and Heineman, G. T. (2004). Dynamic plan migration for continuous queries over data streams. In *ACM SIGMOD International Conference on Management of Data*, pages 431–442.

[Zubek and Dietterich, 2002] Zubek, V. B. and Dietterich, T. G. (2002). Pruning improves heuristic search for cost-sensitive learning. In *International Conference on Machine Learning (ICML)*, pages 19–26.