The Pennsylvania State University
The Graduate School
College of Information Sciences and Technology

NAME DISAMBIGUATION IN ACADEMIC PUBLICATIONS

A Thesis in
Information Sciences and Technology
by
Pucktada Treeratpituk

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

August 2012

The thesis of Pucktada Treeratpituk was reviewed and approved* by the following:

C. Lee Giles
David Reese Professor of Information Sciences and Technology
Thesis Advisor

Prasenjit Mitra
Associate Professor of Information Sciences and Technology

Dongwon Lee
Associate Professor of Information Sciences and Technology

Madhu Reddy
Associate Professor of Information Sciences and Technology
Graduate Program Director of the College of Information Sciences and Technology

*Signatures are on file in the Graduate School.

Abstract

In digital libraries, author ambiguities arise when an author use multiple aliases and when more than one author shares the same names. Since substantial amount of queries in digital libraries are author related, such ambiguity can be inconvenient for users. Without author disambiguation, users are required to manually go through search result when they want to find all the articles written by a particular author. Author disambiguation also enables better bibliometric analysis by allowing a more accurate counting and grouping of publications and citations. While many disambiguation algorithms have been proposed, the most successful ones are those that apply machine learning techniques, such as decision trees and SVMs, to learn the domain specific rules. While SVM-based disambiguation methods have been shown to work well, they suffer from typical drawback of SVMs such as long-training time and arbitrary kernel functions.

In this thesis, we propose a comprehensive set of similarity profile features to assist in author disambiguation and a novel pair-wise author disambiguation algorithm based on random forests, an ensemble classifier based on decision trees. Our experiments on the Medline and CiteSeer databases show that our random forest method outperforms other previously proposed techniques including the SVM-based approaches. Compared with SVMs, the random forest model is substantially faster to train and requires less parameter tuning to achieve good performance. We also provide detail analysis of interactions between different features and the prediction accuracy in the two databases. Finally, we demonstrate how feature selections can be applied to reduce the complexity of the model with little degradation in the disambiguation accuracy.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First, I would like to expression my deepest gratitude to my thesis committee, Dr. C. Lee Giles, Dr. Prasenjit Mitra, and Dr. Dongwon Lee for their valuable suggestions, time and efforts in guiding me through this thesis. I would also like to specially thanks my advisor, Dr. C. Lee Giles, for his continuing guidance and support ever since my very first day in the Pennsylvania State University.

This thesis also would never have come to fruition without the supports of many individuals. I would like to thank my colleagues in the Intelligent Systems Research Laboratory, Ding Zhou, Isaac Councill, Yang Sun, Jian Huang, Shuyi Zheng, Yang Song, Ziming Zhuang, Huajing Li, Pradeep Teregowda, Qi He, and Juan Pablo Fernandez Ramirez, for their intellectual contribution and support. I also would like to take this opportunity to express my immense gratitude to all my friends, Jing Wang, Kun Chen, Liang Guo, Bi Chen, Haibin Liu, Shaoke Zhang, Sujatha Das and Pimonmart Wankanapon.

Lastly, I would like to thank my parents and my sister, for their unconditional support throughout my academic career.

Chapter 1

Introduction

An entity can be referred to in different ways in multiple records. For instance, two web pages about the same person may provide different name spellings and different addresses. The goal of name disambiguation is to resolve such ambiguities, linking and merging all the records of the same person together. Generally, the ambiguity of a person's name comes in three varieties: (1) the aliasing problem - when a person uses multiple name variations; and (2) the common name problem - when there is more than one person with the same name, which is especially problematic for high frequency names such as many Chinese names; and (3) typographical errors - which could result from human input or automatic extraction systems.

In academic digital libraries, it is often desirable to be able to disambiguate author names for many reasons. First, users who browse and search academic articles are often interested to find articles written by a particular author. However, since most academic digital libraries do not disambiguate between multiple authors of the same name, users are generally required to manually sort through search results to find the correct author. Second, disambiguation of author names also enables better bibliometrics analysis by permitting more accurate counting and grouping of publications and citations, measures often used in academic promotion and grant funding. Third, the resulting disambiguated names can also be used to help improve other data mining techniques such as homepage search, natural language processing, and social network analysis in digital libraries.

A typical name disambiguation algorithm is composed of two main components: a pair-wise linkage function and a clustering algorithm. The pair-wise linkage function determines whether two records refer to the same entity based on some attributes. Output of a linkage function can be either a binary decision (yes or no), or a similarity level value between 0 and 1 (and thus can be interpreted as the probability). The clustering algorithm then clusters records based on their similarities as defined by the linkage function.

Much research has proposed various distance measures for determining whether two records should be linked together. These distance measures range from string-based distance, such as

the Levenshtein distance and Jaro distance, to vector-based distance, such as the cosine distance. However, an actual record often does not have only one attribute, but rather multiple attributes. As an example, an author record has attributes such as the first name, last name, academic affiliation, email address, a list of topical keywords for his research interests, and content of publications, all of which are potentially useful for disambiguation. While one can treat such a multi-field record as a single long string or a single field, it is often advantageous to use different distance measures for different fields, e.g. Jaro distance for names, Jaccard distance for affiliation, etc. In that case, the similarity between records can be represented as a feature vector of similarities instead of one single similarity score. Then, an aggregate function is needed to combine these multiple similarity features into a single binary decision or a probability score when two records or more should be linked. While an aggregate function can be as simple as the maximum, the average, the summation, or some fix functions [1], an adaptive aggregate function, which learns how to optimally combine different similarity features, generally yields a higher performance [2]. In fact, high performance classifiers such as SVM have often been used with good success.

1.1 Scope and Structure of Thesis

In this thesis, I focus on the problem of finding a high-quality adaptive pair-wise linkage function for disambiguating author names in academic digital libraries. More specifically, I propose to use the machine learning method, random forests, which has been successfully applied to various classification problems with comparable results to other top discriminative classifiers such as SVMs and Adaboost. This thesis has the following main contributions:

1. I propose a comprehensive set of similarity features for disambiguating author names in digital libraries.

2. I explore the novel use of random forests in the problem of name disambiguation. I evaluate and compare my model with other disambiguation techniques on two notable digital libraries: Medline and CiteSeer. For Medline, I construct a new author disambiguation testbed that can be used for future research. For CiteSeer, I use the previously published data set for direct comparison. My experiments show that the random forest model achieves significantly better pair-wise disambiguation accuracy over previous results.

3. Through variable analysis and feature selection, I identify a small set of predictive features that can achieve high disambiguation performance. In addition, my analysis provides insights into the difference between the disambiguation tasks in the CiteSeer and Medline data.

The rest of this thesis is organized as follows. Chapter 2 describes related works in author name disambiguation. Chapter 3 gives a brief description of digital libraries, especially the two that I used in my evaluation: the Medline and CiteSeer databases. Chapter 4 describes

the random forests algorithm and feature engineering. Chapter 5 presents and discusses our evaluation results. Chapter 6 provides a conclusion and discusses future directions.

Chapter 2

Related Works

Name disambiguation is an instance of a more general problem called record linkage. The goal of record linkage is to link multiple records that refer to the same entity together. In the case of author name disambiguation, the records are the author names, the entities are the authors, and the goal is to link all the names that refer to the same author together. The record linkage problem is referred to by different names by different research communities. The database community refers to it as record linkage [3], information integration [4], database hardening [5], and duplicate detection [6]. The NLP community refers to it as coreference resolution [7]. Other names include entity resolution and string matching [2]. We can categorize research in record linkage into two groups, those that focus on linkage functions and those that focus on blockage functions.

2.1 Linkage Functions

Much record linkage research has focused on linkage functions, especially on their accuracy [8, 2, 9, 10, 11, 4, 12]. A linkage function determines whether two records semantically belong to the same group. It can use just a single attribute of a record or a combination of multiple attributes. This research ranges from using a simple string editing distance, such as the Jaccard distance, to machine learning techniques, such as decision trees. This thesis is closely related to these works.

Tejada et al. used decision trees to learn the mapping rules based on attribute similarity between records [4]. Christen made a comparison study on author name matching [9]. Ross and Jain integrated three similarity scores based on face, fingerprint, and hand geometry in a biometric user verification system using three methods: the summation, a decision tree, and Linear Discriminant Analysis (LDA) [13]. Han et al. proposed two classifiers, hybrid Naive Bayes and Support Vector Machine (SVM), for disambiguating authors in the DBLP [10]. Huang et al. used a density-based clustering algorithm (DBSCAN) to cluster CiteSeer authors based on metadata such as affiliations, email, addresses, name variations, and URL of the papers [14]. They used SVM to combine multiple metadata into one single similarity score. Bilenko

et al. experimented with multiple string distance matrices. They also experimented with three aggregate functions for combining string similarity from multiple fields: SVM, average, and simple string concatenation [2]. Bekkerman and McCallum used the link structure of web pages to disambiguate between web pages of different people [8]. Song et al. proposed a two-stage topic-based disambiguation algorithm [15]. In the first step, they introduced a new latent variable for authors into two popular document generative models, PLSA (Probabilistic Latent Semantic Analysis) and LDA (Latent Dirichlet Allocation). The generative models were then used to learn the topic distribution for each author name, using the metadata plus the first page of the paper. In stage two, the bottom-up agglomerative clustering algorithm was used to cluster names with similar topic distribution together. Song et al. showed that their method is more effective than other unsupervised learning methods, including spectral clustering and DBSCAN. Since the document generative models and the agglomerative clustering are unsupervised methods, their approach requires no labelled data.

2.2 Blocking Functions

Other research assumed a black-box linkage function and focused on improving efficiency and scalability [16, 17, 18, 19, 20, 21, 22]. Without any blocking function, the record linkage problem requires $O(n^2)$ pair-wise comparisons. If the records are organized into blocks, where each record is only compared with other records in the same block, the number of pair-wise comparisons can be reduced to $O(wb) < O(n^2)$, where w is the block size and b is the number of blocks.

Monge and Elkan used a union-find data structure and assumed transitivity of linkage to efficiently merge linked records [21]. Often, efficiency is achieved by certain types of a blocking scheme that then reduce the number of considered record pairs. Hernandez et al. devised sorted neighborhood methods where records are first sorted based on their most discriminating attributes [18]. Each record is then restricted to only comparisons with its local neighbors within some fixed window. In [19], Jin et al. introduced an idea similar to [18]. Instead of sorting records based on some attributes lexically, which is susceptible to data-entry errors, the records are mapped to multidimensional Euclidean space that preserves domain-specific similarity.

Typically, the choice of the blocking function is arbitrary or is chosen manually by trial and error. Winkler presented a method for evaluating the accuracy of any given blocking function through a capture-recapture model [23]. Recent work by Bilenko et al. proposed a way to automatically learn the optimal blocking function. Given similarity predicates on different record fields, the algorithm tries to learn the optimal combination of those predictions as the blocking function [16]. Yan et al. extended the sorted neighborhood methods from [18] by adaptively changing the block size [24]. McCallum et al. also suggested that efficiency can be accomplished by first clustering/blocking data into smaller partitions with an inexpensive distance measure, and then later using a more expensive distance measure to disambiguate within each partition [25]. Recent research tries to improve efficiency by increasing the parallelism of the record linkage [17, 20].

Chapter 3

Digital Libraries

This chapter provides a brief description of two prominent digital libraries, Medline and CiteSeer.

3.1 Medline

Medline is the de facto literature resource for biomedical research. It contains bibliographic references to all biomedical articles, editorials, and letters to the editor in approximately 4,500 scientific journals from over 80 countries. In total, Medline holds over 16 million articles dating back to 1965. Each article in Medline contains up to 49 metadata fields. In addition to the usual metadata (titles, authors, affiliation, journal, abstract), each is manually indexed with the Medical Subject Heading (MeSH), the controlled medical vocabulary created and maintained by NLM. An example of a citation in Medline is shown in Figure 3.1. On average, each article is associated with 10-15 MeSHes. Each MeSH belongs to the MeSH hierarchy. Each heading can belong to more than one part of the hierarchy tree. A part of the MeSH hierarchy related to a MeSH, "Stomach Neoplasms" is shown in Figure 3.2. Notice that the MeSH hierarchy is not a tree and that each node can have more than one parent. Table 3.1 shows examples of three articles in Medline written by an author with the name "Watson, AJ."

Disambiguating author names in Medline can be challenging even to a human assessor. On one hand, Medline provides metadata such as affiliation and MeSH terms that can be used for author name disambiguation. On the other hand, it lacks much information that has been reported to be helpful in disambiguating author names, such as email, URL, and citations [14, 8]. Furthermore, Medline only consistently records the affiliation of the first author for each article after 1988. Even now, no affiliations other than that of the first author are recorded. In addition, the policy of how authors are recorded in Medline has changed over time. Between 1966 and 1984, and between 2000 and present, every author of each article is included. Between 1984 and 1995, only the first ten authors were included, and only the first twenty-five authors were included between 1995 and 2000. Moreover, not until 2002 was the author's first name recorded.

```
<MedlineCitation>
    <Owner> NLM </Owner>
    <PMID> 11281430 </PMID>
    <ArticleTitle> Olanzapine may be an effective adjunctive therapy in
the management of acne excoriée: a case report. </ArticleTitle>
    <AbstractText>
    BACKGROUND: The self-inflicted dermatoses such as acne excorieé and
neurotic excoriations are often chronic, recurring, and resistant to
standard dermatologic therapies. OBJECTIVE: We present a 28-year-old
woman with longstanding acne excorieé, whose acne started at age 14 years
and was followed by acne excorieé at age 16 years. The patient reported
that her acne and self-excoriative behavior were exacerbated by
psychological stress. The previously treatment-resistant acne excorieé
responded favorably to treatment with the atypical antipsychotic agent
olanzapine. METHODS: The patient was started on olanzapine 2.5 mg at
bedtime. RESULTS: After 4 weeks of therapy with olanzapine she reported a
significant decline in her self-excoriative behavior which was associated
with an improvement in her acne excorieé. The patient used the olanzapine
2.5 mg for 6 months, during which time she also entered psychotherapy in
order to deal with some psychosocial stressors that were exacerbating her
self-excoriative behavior. The patient has not experienced a recurrence
in her self-excoriative behavior or acne excorieé for 4 months after
discontinuing the olanzapine. CONCLUSION: Olanzapine may prove to be a
useful adjunctive therapy in some self-induced dermatoses including acne
excorieé.
    </AbstractText>
    <Language> eng </Language>
    <Country> Canada </Country>
    <Affiliation> Department of Psychiatry, University of Western
Ontario, London, Canada. </Affiliation>
    <CompleteAuthors> Y </CompleteAuthors>
    <Author> Gupta, M A (MA) </Author>
    <Author> Gupta, A K (AK) </Author>
    <MeshHeading> Acne Vulgaris [N] (drug therapy [Y]; psychology [Y];
therapy [N]) </MeshHeading>
    <MeshHeading> Adult [N] </MeshHeading>
    <MeshHeading> Antipsychotic Agents [N] (therapeutic use [Y])
</MeshHeading>
    <MeshHeading> Benzodiazepines [N] </MeshHeading>
    <MeshHeading> Female [N] </MeshHeading>
    <MeshHeading> Humans [N] </MeshHeading>
    <MeshHeading> Pirenzepine [N] (analogs & derivatives [Y]; therapeutic
use [Y]) </MeshHeading>
    <MeshHeading> Self-Injurious Behavior [Y] </MeshHeading>
    <MeshHeading> Stress, Psychological [N] (complications [Y])
</MeshHeading>
    <JournalTitle> Journal of cutaneous medicine and surgery
</JournalTitle>
    <PubDate> 2001 Jan-Feb </PubDate>
</MedlineCitation>
```

Figure 3.1. An example of a Medline Citation

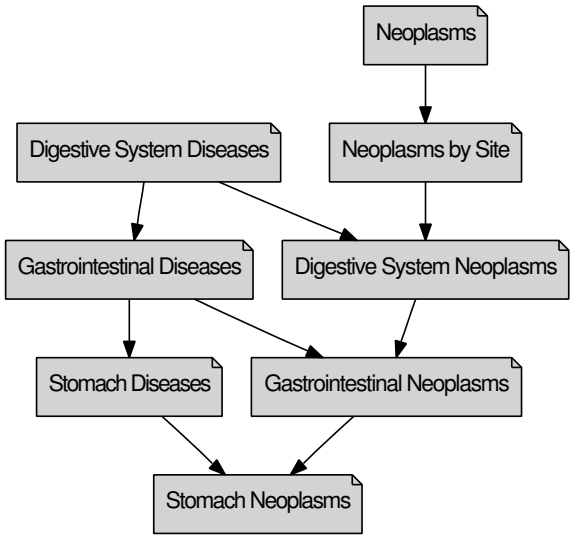

Figure 3.2. An example of Medical Subject Headings (MeSH)'s structure.

Previously, most authors are listed with only their last names and first initials. The size of Medline itself also results in high ambiguity. For example, out of three million articles published in Medline after 1999, there are almost twelve million author names. Of those twelve million names, only approximately 3.5 million names are unique. The name "Wang, Y." alone appeared in more than 7,000 articles.

Consider the three articles in Table 3.1 to illustrate how one would possibly disambiguate authors in Medline. All three articles contain the author name "Watson, AJ." For articles (1) and (2), it is difficult to determine whether they are both written by the same "Watson, AJ" because (1)'s affiliation is *University of Calgary* while (2)'s affiliation is *University of Missouri*. The departments listed in both papers are also different, *Medical Biochemistry* vs. *Animal Sciences*. There are clues, however, that suggest the possibility of (1) and (2) being a match. Both venues are related to "reproduction," and both articles are related to Animals (based on the MeSH). For articles (1) and (3), they are highly likely to be written by the same "Watson, AJ." Not only is *University of Calgary* the affiliation of both first authors, but both articles are also about *Blastocyst* (which is related to "embryo" and "reproduction"). Given that (1) and (3) are matched, then all three articles are likely to be written by the same "Watson, AJ" because (2) and (3) are both co-authored by "Schultz, GA." In fact, if one has access to article (2), not just its metadata in Medline, one will discover that the affiliation of "Watson" and "Schultz" is indeed *University of Calgary*. This example highlights the difficulty of disambiguating author names in Medline where some essential information is missing. It also illustrates some limitations of pair-wise disambiguations. For some article pairs, such as (1) and (2), it is difficult to disambiguate based solely on the pair and, only when an additional instance, in this case the article (3), is included does accurate disambiguation become possible.

The only attempt at disambiguation of author names in Medline that I am aware of is the work

Table 3.1. Examples of metadata in three Medline papers authored by a unique author

(1)	**ArticleTitle**	The cell biology of **blastocyst** development.
	Affiliation	Dept of Medical Biochemistry, University of Calgary, Health Science Center, Alberta
	Authors	**Watson, AJ**
	JournalTitle	Molecular reproduction and development [ENG]
	PubDate	1992 Dec
	MeshHeading	Animals, **Blastocyst**, Embryonic and Fetal Development, ...
(2)	**ArticleTitle**	Expression of bovine trophoblast interferon in conceptuses derived by in vitro techniques.
	Affiliation	Dept of Animal Sciences, University of Missouri, Columbia, 65211
	Authors	Hernandez-Ledezma, JJ, Sikes, JD, Murphy, CN, **Watson, AJ, Schultz, GA**, Roberts, RM
	JournalTitle	Biology of reproduction [ENG]
	PubDate	1992 Sep
	MeshHeading	Animals, Base Sequence, Cattle, Culture Techniques, ...
(3)	**ArticleTitle**	How to make a **blastocyst**.
	Affiliation	Department of Medical Biochemistry, University of Calgary, Alta., Canada
	Authors	**Watson, AJ**, Kidder, GM, **Schultz, GA**
	JournalTitle	Biochemistry and cell biology = Biochimie et biologie cellulaire [ENG]
	PubDate	1992 Oct-Nov
	MeshHeading	Animals, Animals & Domestic, **Blastocyst**, DNA & Recombinant, ...

of Torvik et al [12]. They proposed a model that estimates the probability that a pair of author names that share the same last name and first initial appearing on two different Medline articles refer to the same individual. Their model can also be categorized as a pair-wise comparison method. However, their model uses simplistic features, most of which are categorical variables. They did not incorporate any advances in distance function research and information retrieval, such as TFIDF, which could have significantly improved the accuracy.

3.2 CiteSeer

CiteSeer is a popular scientific literature digital library and search engine. Its main coverage is academic publications in Computer Science, Mathematics, and Physics. Unlike other digital libraries, CiteSeer automatically acquires and indexes scientific documents available on the World Wide Web. Once a document is acquired, CiteSeer automatically extracts metadata information from the document, including authors, titles, abstracts, venues, keywords, and citations.

While CiteSeer provides many types of metadata information, in this thesis, I will only utilize two types of metadata: author information and article title. The experiments in Chapter 5 will use only article titles, authors' names, emails, and affiliations in the disambiguation. Figure 3.3 shows an example of two records in the CiteSeer database that need to be disambiguated. Both

documents were written by the same author, “Ashish Gupta.”

Author disambiguation in CiteSeer presents a different set of challenges than those in Medline. On one hand, CiteSeer provides more metadata for each author in a document. Affiliation information is available for all authors, not just the first author. On the other hand, since metadata in CiteSeer is automatically extracted, it contains many typological and extraction errors. However, since much record linkage research has used CiteSeer data in their evaluations [14, 2, 15], using CiteSeer data for evaluation allows us to easily compare the results with other methods.

```
<Doc id="429160">
  <Title>ADAPTING MATERIALIZED VIEWS AFTER REDEFINITIONS:
TECHNIQUES
        AND A PERFORMANCE STUDY</Title>
  <Author id="957775">
    <Name>Ashish Gupta</Name>
    <Affil>IBM Almaden Research Center</Affil>
    <Email>ashishgupta@stanfordalumni.org</Email>
  </Author>
  <Author id="957776">
    <Name>Inderpal S. Mumick</Name>
    <Email>mumick@mumick.com</Email>
  </Author>
  <Author id="957777">
    <Name>Jun Rao</Name>
    <Affil>IBM Almaden Research Center</Affil>
    <Email>junrao@almaden.ibm.com</Email>
  </Author>
  <Author id="957778">
    <Name>Kennth A. Ross</Name>
    <Affil>Columbia University</Affil>
    <Email>kar@cs.columbia.edu</Email>
  </Author>
</Doc>
<Doc id="15161">
  <Title>Constraint Management on Distributed Design
Configurations</Title>
  <Author id="32872">
    <Name>Ashish Gupta</Name>
    <Affil>Department of Computer Science, Stanford
University</Affil>
    <Addr>Stanford, CA 943052140</Addr>
    <Email>agupta@cs.stanford.edu</Email>
  </Author>
  <Author id="32873">
    <Name>Sanjai Tiwari</Name>
    <Affil>Department of Civil Engineering, Stanford
University</Affil>
    <Addr>Stanford, CA 943054020</Addr>
    <Email>tiwari@cive.stanford.edu</Email>
  </Author>
</Doc>
```

Figure 3.3. Examples of two CiteSeer records with the author name "Ashish Gupta" to be disambiguated

Chapter 4

Random Forest

In this chapter, I first explain the random forest classifier, define variable importance, and explain how variable importance can be inferred from the model. Then, I define the comprehensive feature set that can be used in disambiguating author names.

4.1 Random Forest

Random forests are ensemble classifiers proposed by Breiman that combine a collection of decision trees [26]. Each decision tree within the forest is built with a different bootstrap sample drawn from the original dataset. Each tree is then constructed to the maximum size without any pruning. The variable selection for each split in the tree is conducted on a randomly selected subset of features, instead of on the full feature set as is usually done in the traditional decision tree (see Algorithm 1). Once the forest is built, the classification can be done by simply aggregating the votes of all trees.

Consider the building block of a random forest, a decision tree. While it has low bias, it generally suffers from high variance, leading to a high error rate. It is usually susceptible to noise in the data. A slight change in the training data often affects the structure of the tree dramatically. Random forests gain their performance improvement over decision trees by achieving both low bias and low variance. They accomplishes this by aggregating a large number of low-correlated decision trees, each of which has low bias and high variance. The low bias of a forest is achieved by growing each tree without any pruning. The low variance is obtained from bagging (bootstrap aggregating) and random variable selection. Random forests have been reported to perform better than that of SVMs over a wide range of classification problems [26].

There are only two parameters to tune in random forests: T, the number of trees to grow, and m, the number of features to consider when splitting each node. The error rate of a random forest depends on two factors: the correlation between trees in the forest and the strength of each individual tree. The more correlated each tree is, the higher the error rate becomes. The stronger

Algorithm 1 Constructing a random forest

Given a set of instances, $S = \{(x_{1i}, x_{2i}, ..., x_{Mi}, y_i)\}$, $i \in 1...N$, where N is the total number of instances, M is the total number of predictors. x_{ki} refers to the kth attribute (predictor variable) of the ith instance, and y_i is the class label (response variable) of the ith instance.

Choose T, the number of trees to grow, and $m << M$, the number of predictors to be considered when splitting each node.

When growing a tree t
1: Construct a bootstrap sample (with replacement) S_t from S. Use S_t to construct the tree t.
2: At each node, randomly select m variables out of M. Select the best split for that node out of these m variables.
3: Grow the tree to the maximum extent without pruning.

each individual tree is (high accuracy), the lower the error rate becomes. By increasing m, the number of features selected, both the correlation and the strength of each tree increases. By lowering m, each tree becomes more independent (less correlated), but also becomes weaker at the same time. Thus, there exists optimal values of m that provide the optimal balance between the correlation and the strength to get the best error rate. Breiman suggested the default value for m to be $log_2(M + 1)$, where M is the total number of features. This default value has been reported to work well in practice [27]. Also, since only a small, randomly selected subset of features ($m << M$) is used at each split, random forests tend to work well even for data with high dimensionality where there are more variables than observations.

From its use of bagging, the test error of a random forest can be estimated internally without a cross-validation or a separate test set. Since each tree in the forest is constructed on a different bootstrap sample, about one-third of the original data is left out of that tree's training sample and thus can be used for error estimation. For every data point x in the original data, we can define the out-of-bag (OOB) trees of x as the set of trees where x is not included in their bootstrap samples. Then, it is possible to calculate the error rate of the random forest on all the original data, where the classification for each data point is done only by its out-of-bag trees. This error estimate is called the out-of-bag (OOB) error, which was shown to be an unbiased estimate of the test set error [26]. The OOB error estimate is also used later in the computation of variable importance.

A random forest has many nice characteristics that make it promising for the problem of name disambiguation. First, a random forest can achieve good accuracy even for a problem with many weak variables (where each variable conveys only a small amount of information). Furthermore, since each classifier in the forest is just a decision tree, a random forest can model interactions and dependencies between features in making predictions. This is useful because users generally use such rules to disambiguate names; for example, "if the affiliations are matched and both are the first author, then ..." In addition, a random forest is very fast both in training and in making predictions, thus making it ideal for a large-scale problem such as name disambiguation.

The voting of the trees in the forests can also naturally be used as the similarity distance in any clustering algorithm.

4.1.1 Variable Importance

Since a random forest is an ensemble of trees, it cannot be as easily interpreted as a decision tree. However, a random forest offers a simple way to measure variable importance for each of its features, giving insights into the interaction between each feature and the prediction accuracy. This makes random forests attractive compared to other classifiers such as kernel-based SVM, which often achieves good classification error rates but is hard to interpret.

Variable importance is a measurement of how much influence an attribute has on the prediction accuracy. There are two methods of measuring variable importance in a random forest: by Gini importance and by permutation importance. Gini importance is calculated based on the Gini index (or Gini Impurity), which is the measure of class distribution within a node. Gini index of a node i, $I_G(i)$, is defined as:

$$I_G(i) = \sum_{j=1}^{K} p_j(1 - p_j) = 1 - \sum_{j=1}^{K} p_j^2$$

where p_j is the proportion of instances of class j in the node i, and K is the number of classes. $I_G(i)$ is minimum ($= 0$) when the node is pure. Gini index is used as the criterion for selecting the split at each node in the decision tree construction; the split that yields the biggest reduction in Gini index is selected. Therefore, in a decision tree, Gini impurity of the two descendent nodes is always less than that of the parent. Gini importance of a variable can be computed by averaging the Gini decreases for that variable over all trees in the forest. A variable with high Gini importance is the one that on average provides informative partitioning of data. The Gini importance measure is related to the concept of Information Gain, which is another popular splitting criterion for decision trees based on information theory. Since it has been shown that the Gini index and Information Gain produce almost indistinguishable splits in most problems [28], I will not discuss Information Gain here.

The other measure of variable importance is the permutation accuracy importance. To compute the permutation accuracy importance of variable X_i, for each tree that contains variable X_i, first randomly permute feature X_i in its out-of-bag data and then calculate the new OOB accuracy for this permuted input compared with its original OOB accuracy. The variable importance of X_i is this decrease in accuracy averaged over all trees that contain X_i. The rationale is that if variable X_i is strongly correlated with the response, by permuting the value of X_i, the prediction accuracy should decrease significantly. Both permutation importance and Gini importance can be used to measure the relevance of each variable in the feature selection.

4.2 Similarity Profile

Here, I first give the formal formulation of the author name disambiguation problem and then define the set of attributes, called the similarity profile, that will be used by random forest for disambiguation. Given two papers, $paper_A$ and $paper_B$, both containing an author with the name "$lname$, $init$" where $lname$ refers to the last name and $init$, the first initial, the goal is to disambiguate whether they refer to the same person. I use a naive blocking function that blocks author name based on the last name and the first initial. Thus, only two papers that shared an author with the same last name and first initial will be disambiguated. This is a reasonable assumption for a manually created database such as Medline, where the errors in the author names are low. For databases such as CiteSeer, where author names are automatically extracted, this assumption should be tested. However, if that is the case, soft blocking strategies such as McCallum's canopy, which puts each author record into multiple candidate blocks, can be applied instead [25]. The following metadata of $paper_A$ and $paper_B$ are used to construct the similarity profile:

$paper_A = (lname_A, fname_A, init_A, mid_A, suf_A, coauth_A, aff_A, email_A, title_A, jour_A, lang_A, year_A, mesh_A)$
$paper_B = (lname_B, fname_B, init_B, mid_B, suf_B, coauth_B, aff_B, email_A, title_B, jour_B, lang_B, year_B, mesh_B)$

where

$lname_i$ = the author's last name in $paper_i$
$fname_i$ = the author's first name in $paper_i$
$init_i$ = the author's first initial in $paper_i$
mid_i = the author's middle name in $paper_i$, if given
suf_i = the author's suffix in $paper_i$, if given, e.g. "Jr", "Sr"
$coauth_i$ = set of coauthors' last names in $paper_i$
aff_i = the author's affiliation in the $paper_i$. In the case that only the affiliation of the 1st author is available (such as in Medline), use the 1st author's affiliation instead.
$email_i$ = email of the $paper_i$'s 1st author
$title_i$ = $paper_i$'s title
$jour_i$ = $paper_i$'s journal name
$lang_i$ = $paper_i$'s journal language, e.g. English, Chinese
$year_i$ = $paper_i$'s year of publication
$mesh_i$ = set of MeSH terms in the $paper_i$

The similarity profile between author name "$lname, init_A$" in $paper_A$ and $paper_B$ consists of 22 features, which can be grouped into six categories based on the metadata they are calculated

from: author similarity, affiliation similarity, coauthor similarity, concept similarity, journal similarity, and title similarity. However, not every feature is available in both the Medline CiteSeer datasets. For example, the MeSH terms and journal years are not available in the CiteSeer data. Also, while the affiliation of the 1st author in Medline sometimes contains email information, they were not extracted because they are not included frequently enough. Thus, email similarity is not used in Medline data. Therefore, in total, the Medline dataset has 21 out of 22 features, and the CiteSeer dataset contains 12 features. All 22 features are defined as follows:

4.2.1 Author Similarity

1) $auth\_fst$: the first name similarity. $auth\_fst$ is highest when both first names are provided and are perfectly matched. $auth\_fst$ is 0 if both first names are provided and do not match. In the case that one of the first names is not provided, the first initials are used in calculating the similarity.

$$auth\_fst = \begin{cases} 0 & \text{if } fname_A \neq fname_B, \text{ both are given} \\ 1 & \text{if one of the first names is missing, and } init_A \neq init_B \\ 2 & \text{if one of the first names is missing, and } init_A = init_B \\ 3 & \text{if } fname_A = fname_B, \text{ both are given} \end{cases}$$

2) $auth\_mid$: similarity between mid_A and mid_B.

$$auth\_mid = \begin{cases} 0 & \text{if } mid_A, mid_B \text{ are given, } mid_A \neq mid_B \\ 1 & \text{if both } mid_A, mid_B \text{ are not given} \\ 2 & \text{if only one of } mid_A, mid_B \text{ are given} \\ 3 & \text{if } mid_A, mid_B \text{ are given, } mid_A = mid_B \end{cases}$$

3) $auth\_suf$: similarity between suf_A and suf_B. If suffixes are given in both papers and they are equal, $auth\_suf$ similarity is 1, otherwise it is zero.

$$auth\_suf = \begin{cases} 1 & \text{if } suf_A, suf_B \text{ are given, } suf_A = suf_B \\ 0 & \text{otherwise} \end{cases}$$

4) $auth\_ord$: similarity between the orders of each author. $auth\_ord$ reflects the similarity between the orders that the two authors appear in both papers. $auth\_ord$ is highest when both authors appear as the 1st author, is medium when both authors appear as the last author, and zero otherwise. This feature is designed to work in conjunction with the affiliation similarity, especially in Medline data. Because only the affiliation of the 1st author is provided in Medline, if both authors are the 1st author, the affiliation similarity in Medline data should be given more weight. On the other hand, if none of the authors is the 1st author, then the affiliation similarity should be discounted.

$$auth\_ord = \begin{cases} 2 & \text{if both authors are the 1st author} \\ 1 & \text{if both authors are the last author} \\ 0 & \text{otherwise} \end{cases}$$

5) $auth\_lname\_idf$: IDF weight of the author's last name. IDF is the inverse of the fraction of names in the corpus.

$$auth\_lname\_idf = log(IDF(lnameA))$$

where

$$IDF(lnameA) = IDF(lnameB) = \frac{L}{DF_{lname_A}}$$

and L is the total number of articles in Medline and DF_{lname_A} is the total number of $lname_A$ in Medline. A high value of $IDF(lname)$ means that $lname$ is rare, and thus is less likely to be ambiguous.

6) $auth\_email\_jac$: the Jaccard similarity between $email\_A$ and $email\_B$. In calculating $auth\_email\_jac$, I simply tokenize each email into two tokens, the username and the domain name, i.e. "akumar@psu.edu" into "akumar" and "psu.edu" because some authors use the same usernames with two different domains, i.e. "psu.edu" and "ist.psu.edu." The other way around is also true: some authors might have two aliases on the same domain. I use the token-based Jaccard distance because it is simple to calculate. The drawback of the token-based Jaccard similarity is that it does not take partially matched strings into account, i.e. "akumar" and "kumar" would be considered a totally different string. A more complicated distance measure such as character editing distance might help in this regard but would also be slower.

$$email\_jac = \frac{|email_A \bigcap email_B|}{|email_A| + |email_B|}$$

4.2.2 Affiliation Similarity

7) aff_jac: the Jaccard similarity between aff_A and aff_B. The more terms both affiliations share, the bigger aff_jac is.

$$aff\_jac = \frac{|aff_A \bigcap aff_B|}{|aff_A| + |aff_B|}$$

8) aff_tfidf: the sum of TFIDF weights of shared terms in aff_A and aff_B.

$$aff\_tfidf = \sum_{t \in aff_A \bigcap aff_B} TFIDF(t, aff_A) \times TFIDF(t, aff_B)$$

where $TFIDF(t, S) = log(TF_{t,S} + 1) \times log(IDF(t))$, and $TF_{t,S}$ is the frequency of t in S. aff_tfidf gives more weights to the agreement on rare terms than the agreement on common terms. This makes aff_tfidf more attractive compared to aff_jac, which is insensitive to the

rarity of terms.

9) $aff\_softtfidf$: the soft-TFIDF distance between aff_A and aff_B. The soft-TFIDF distance is a hybrid distance that combines a string-based distance with the TFIDF distance. The soft-TFIDF does not only account for TFIDF of terms that occur in both strings, but also of a term that occurs in one string and has a similar term appearing in the other string [2]. Here, I use the Jaro-Winkler distance as the measurement of whether two terms are similar [29].

$$Jaro(t,v) = \frac{1}{3} \times (\frac{|CC_{t,v}|}{|C_t|} + \frac{|CC_{v,t}|}{|C_v|} + \frac{|CC_{t,v}| - T_{CC_{t,v},CC_{v,t}}}{2|CC_{t,v}|})$$

where C_i is all characters in i, $CC_{i,j}$ is all characters in i that appear in j, and $T_{p,q}$ is the number of transpositions of characters in p relative to q. And

$$JaroWinkler(t,v) = Jaro(t,v) + \frac{max(L,4)}{10} \times (1 - Jaro(t,v))$$

where L is the length of the longest common prefix of t and v. Then, define a set $C(aff_A, aff_B)$ to be the set of term $t \in aff_A$ such that $\exists v \in aff_B, JaroWinkler(t,v) < 0.8$. And $\forall t \in C(aff_A, aff_B)$, define $N(t, aff_B) = max_{v \in aff_B}\ JaroWinkler(t,v)$. Then,

$$aff\_softtfidf = \sum_{t \in C(aff_A, aff_B)} TFIDF(t, aff_A) \times TFIDF(t, aff_B) \times N(t, aff_B)$$

4.2.3 Coauthor Similarity

10) $coauth\_lname\_shared$: the number of shared coauthor last names between the two papers.

$$coauth\_lname\_shared = |coauth_A \bigcap coauth_B|$$

11) $coauth\_lname\_idf$: the sum of IDF values of all shared coauthor last names. This is similar to *coauth_lname_shared* but gives more weight to shared coauthors with rare names.

$$coauth\_lname\_idf = \sum_{ln \in coauth_A \bigcap coauth_B} log(IDF(ln))$$

12) $coauth\_lname\_jac$: the Jaccard similarity between $coauth_A$ and $coauth_B$. This is similar to *coauth_lname_shared*, but with normalization.

$$coauth\_lname\_jac = \frac{|coauth_A \bigcap coauth_B|}{|coauth_A| + |coauth_B|}$$

4.2.4 Concept Similarity

These features are only applicable to Medline data, since keywords are not available in the CiteSeer dataset.

13) $mesh\_shared$: the number of shared MeSH terms between the two papers. If two papers share multiple MeSH concepts, it is more likely that they were written by the same author.

$$mesh\_shared = |mesh_A \bigcap mesh_B|$$

14) $mesh\_shared\_idf$: the sum of IDF values of all shared MeSH terms. This is similar to $mesh\_shared$ but gives more weight to the agreement of rare MeSH concepts. The frequency of each MeSH concept is collected from the entire Medline database.

$$mesh\_shared\_idf = \sum_{t \in mesh_A \bigcap mesh_B} log(IDF(t))$$

15) $mesh\_tree\_shared$: This feature is similar to $mesh\_shared$ but also takes the tree structure of the MeSH hierarchy into consideration. It incorporates the indirect relationship between MeSH concepts. For instance, "Breast Neoplasms" and "Lactation Disorders" would be thought of as two different concepts in $mesh\_shared$, thus have zero similarity. However, since they share a parental concept, "Breast Diseases," they will have positive similarity according to $mesh\_tree\_shared$.

$$mesh\_tree\_shared = |T(mesh_A) \bigcap T(mesh_B)|$$

where a set $T(mesh_i)$ is defined as the set of all ancestor concepts of $mesh_i$ in the MeSH hierarchy. For a MeSH concept c, $c \in T(mesh_i)$ if c is in a path from the root to $mesh_i$. Note that each MeSH concept can have multiple parents, thus there could be multiple paths from the root of the hierarchy to $mesh_i$.

16) $mesh\_tree\_shared\_idf$: similar to $mesh\_tree\_shared$, but instead of the number of shared MeSH concepts in the hierarchy, it uses the IDF weight sum. It gives more weight to the agreement of rare concepts. Since the IDF value of a parental concept is always less than or equal to that of its children, it can also be thought of as discounting the weight of the indirect relationship between concepts.

$$mesh\_tree\_shared = \sum_{c \in T(mesh_A) \bigcap T(mesh_B)} log(IDF(c))$$

4.2.5 Journal Similarity

17) $jour\_shared\_idf$: IDF value of the shared journal. In the case that both are published in the same journal, if the journal is rare, in the sense that it contains a small number of papers,

then the two papers are more similar than when the journal is common.

$$jour\_shared\_idf = \begin{cases} log(IDF(jour_A)) & \text{if } jour_A = jour_B \\ 0 & \text{otherwise} \end{cases}$$

18) $jour\_lang$: categorical similarity between language of both journals.

$$jour\_lang = \begin{cases} 0 & \text{if } lang_A \neq lang_B, \text{ both are non-English} \\ 1 & \text{if } lang_A \neq lang_B, \text{ one is English} \\ 2 & \text{if } lang_A = lang_B, \text{ both are English} \\ 3 & \text{if } lang_A = lang_B, \text{ both are non-English} \end{cases}$$

19) $jour\_lang\_idf$: IDF value of the shared journal's language. This is similar to $jour\_lang$ but also takes the rarity of language into consideration. $jour\_lang\_idf$ gives more weight to the agreement of a language that is rare. For example, if both papers are in Maori, it gets more weight than if both papers are in Chinese.

$$jour\_lang\_idf = \begin{cases} log(IDF(lang_A)) & \text{if } jour_A = jour_B \\ 0 & \text{otherwise} \end{cases}$$

20) $jour\_year$: categorical variable reflecting a change in Medline's policy. 1988 is the first year that affiliation of the 1st author is included, and 2002 is the first year that author full names are included.

$$jour\_year = \begin{cases} 0 & \text{if both are before 1988} \\ 1 & \text{if one is before 1988, one is after 1988} \\ 2 & \text{if both are between 1988 and 2002} \\ 3 & \text{if both are after 1988, one is after 2002} \\ 4 & \text{if both are after 2002} \end{cases}$$

21) $jour\_year\_diff$: difference in publication years. The idea is that if two papers are published far apart in time, e.g. 10 years, it might be unlikely that they are written by the same author.

$$jour\_year\_diff = |year_A - year_B|$$

4.2.6 Title Similarity

22) $title\_shared$: the Jaccard similarity between $title_A$ and $title_B$.

$$title\_shared = \frac{|title_A \bigcap title_B|}{|title_A| + |title_B|}$$

4.3 Summary

In this chapter, I gave a brief account of the random forest algorithm and its rationale. I argued why random forests are suitable for author name disambiguation. I describe the built-in mechanisms in random forests for measuring the importance of each features. While, like other ensemble classifiers, random forests are harder to interpret than their building block classifier, the decision tree, such measurements offer insight into the model. In the rest of the chapter, I provided a detailed definition for the similarity profile between two author records.

The following chapter is devoted to the evaluation of the random forest model and the similarity profile proposed in this chapter. The model is evaluated across two datasets, one from Medline and one from CiteSeer digital libraries. I shall attempt to thoroughly explore how the features in the similarity profile interact with each other and how they affect the disambiguation performance.

Chapter 5

Evaluation

The evaluation in this chapter is structured into two sets of experiments on two different datasets, Medline and CiteSeer. The goal of these experiments is to evaluate how well random forests perform in the author disambiguation task compared to other techniques. I also want to understand how the task of disambiguating author names in Medline is different from that in CiteSeer. Furthermore, I analyze the interaction between different features in the similarity profile proposed in the previous chapter.

5.1 Medline Experiments

5.1.1 Data Sampling

To evaluate the performance of the random forest for disambiguation in Medline, I first randomly selected 91 unique author names (as defined by the last name and the first initial) from the Medline database. For each selected name, I then manually clustered all articles in Medline written by that name. Each resulting cluster corresponds to the set of articles written by one unique author. These are used as the gold standard in training and evaluation. Figure 5.1 shows the number of unique author clusters for each of the 91 names. The most ambiguous name is "Park, J.," with 997 unique authors for 6,803 total articles; 46 names out of 91 contain only one or two unique authors. The clusters are constructed by manually examining metadata provided in Medline in deciding whether two articles were written by the same author. I also utilized information not provided in Medline in the manual disambiguation. For instance, if two articles seem likely to be written by the same author but their first author's affiliations in Medline are different, I looked at the full affiliation of every author (through other data source such as the internet, since it is not available in Medline) in the disambiguation process. For articles that have little information (e.g. empty affiliation), I assumed each was written by a unique author. I then randomly sampled without replacement 100 article-article pairs from each of the 91 names gold standard and aggregated them to create the evaluation set called S100. S100 contains a total

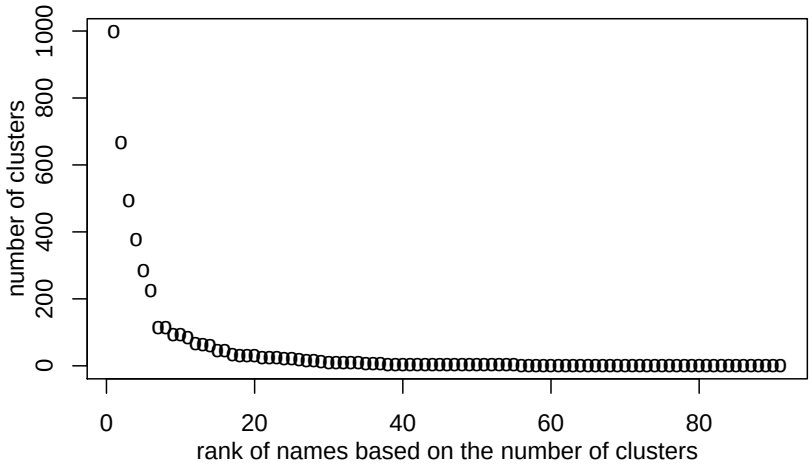

Figure 5.1. Number of unique author clusters for each of 91 sampled author names.

of 8,064 instances of article-article pairs. Similarly, I also created the S200, S300, S400, S500 evaluation sets with a sample size equal to 200, 300, 400, and 500 respectively. The sampling was done to make the evaluation feasible, since SVM takes a long time to train and to investigate the effect of size of the training data on the accuracy. In order to calculate IDF and TFIDF for different features in the similarity profile, I collected statistics such as name frequency and number of articles in each journal from the entire Medline database. For the computation of *mesh_tree_shared* and *mesh_tree_shared_idf*, the 2008 version of the MeSH hierarchy was used.

5.1.2 Comparative Studies

I compared the performance of our random forest model (RF) with four other traditional classifiers: logistic regression, NaiveBayes, a decision tree, and an SVM (Table 5.1). I also included a classifier that simply predicts the majority class in the training data as the baseline (Majority). For NaiveBayes, the normal distributions are assumed for the numerical features such as *coauth_shared* in calculating the probability, $P(X_k = x_{ki}|Y = y_i)$. For SVM, I used the RBF (Radial Basis Function) kernel, where $RBF(x, y) = e^{\gamma||x-y||2}$. The parameters and C (the penalty of errors) in the SVM are then tuned using grid-search with 10-fold cross-validation. I used the libsvm implementation of SVM [30]. The decision tree (DT) was built using C4.5 algorithm with pruning. In this experiment, the random forest is built with 500 trees ($T = 500$), and $m = int(log2(21 + 1)) = 4$.

The numbers reported in Table 5.1 are computed using 10-fold cross-validation. Each fold is stratified so that it contains approximately the same proportion of class distribution as the original dataset. Random forest consistently outperforms all other classifiers for every dataset, achieving almost 96% accuracy for the S500 data. Figure 5.2 shows the accuracy on S500 data, as the trees were grown in the random forest. There is a small accuracy change from 100 to 500

Table 5.1. Classification Accuracy (%) for various classifiers on different sample sizes

	Accuracy (%)				
set	S100	S200	S300	S400	S500
#instances	8,064	15,643	22,860	29,612	35,732
Majority	56.69	55.33	55.10	54.15	53.52
NaiveBayes	77.79	78.32	78.22	78.30	77.66
Logistic	90.29	90.04	90.55	90.65	90.87
DT	91.15	92.73	93.37	93.43	94.15
SVM	92.78	93.10	93.39	93.66	93.68
RF	94.87	95.35	95.55	95.85	95.99

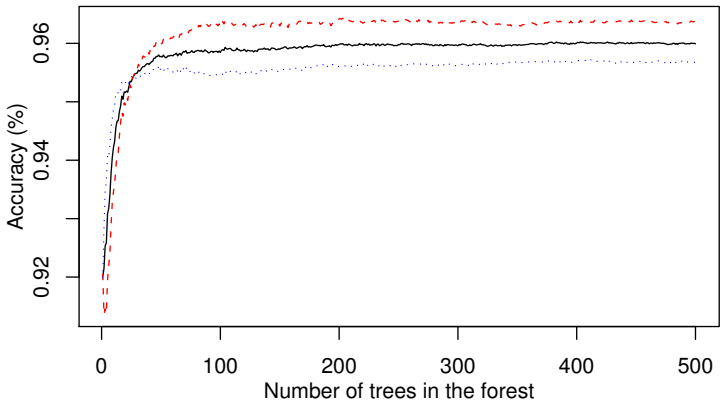

Figure 5.2. Accuracy of the random forest on S500. Black=total accuracy, Red=class 1's accuracy Blue=class 0's accuracy.

trees, suggesting that 100 trees might be sufficient to get a reasonable result. Also, I observed that class 0's accuracy (for unmatched authors) is higher than class 1's accuracy in the beginning when the number of trees is small. The reverse is true when the number of trees increases. The next best classifiers are SVM and the decision tree, which achieved around 94% accuracy. In comparing SVM and random forests, the ANOVA shows that the difference in accuracy between the two are significant. Logistic regression performs reasonably well with around 90% accuracy. NaiveBayes performs noticeably poorly compared to the other classifiers. This might be because high correlations between features violate the independence assumption of NaiveBayes. The normal distribution assumption of numerical features used in NaiveBayes is also not valid. It is interesting to note that logistic regression, decision tree, SVM, and random forests can all achieve over 90% accuracy. This is actually not bad considering that none of the non-first authors' affiliations is available to these classifiers.

I also looked at the effect of training data size on the accuracy and training time. Each classifier achieves similar accuracy across all five datasets. The random forests and the decision tree seem to perform slightly better on the bigger dataset. Further experiments with larger datasets are needed to see whether such an improvement will persist. Table 5.2 compares the training time between SVM and random forests (all other classifiers take under four seconds to

Table 5.2. Training time (minutes & seconds) for different sample sizes

set	S100	S200	S300	S400	S500
Majority	0.02s	0.02s	0.03s	0.04s	0.04s
NaiveBayes	0.13s	0.24s	0.38s	0.48s	0.59s
Logistic	1.39s	1.66s	2.28s	3.28s	3.81s
DT	0.36s	0.81s	1.5s	1.83s	2.48s
SVM	25.83s	1m59s	4m31s	7m49s	12m39s
RF	21.39s	46.80s	1m16s	1m47s	2m20s

train on each dataset). The training time of random forests increases linearly with the size of the training data, while SVM takes a noticeably longer time as the size of the training data increases. Additionally, parameter tuning in SVM takes up a substantial amount of time; for instance, the grid search on S200 took over 6 hours.

5.1.3 Analysis of Features

Since name disambiguation is often run on a large amount of ambiguous names, scalability is often a concern. The problem of name disambiguation in its most naive form involves $N \times N$ pair-wise comparisons, where N is the number of instances. For a digital library with millions of articles and author names, a cheaper distance measure that can be used to partition data into smaller subsets without sacrificing much performance is desirable [25]. Here, I investigate the relevance of each feature to the prediction and then look at the effectiveness of the feature selection.

First, I examined the correlation between features and their class label. The correlation between each feature and the class label is shown in Figure 5.3. The shape and the color of each circle indicate the strength of the correlation. The green circle indicates that there is little or no correlation between features. The yellow ellipse indicates a positive correlation between the two features, while the blue ellipse indicates a strong negative correlation. The stronger the correlation, the thinner the ellipse. Since each attribute has the correlation of 1 with itself, each diagonal entry is a thin ellipse. One can quickly see the grouping of related features, whether they are affiliation-based or coauthor-based, which show up as the yellow square blocks. The correlation plot also shows that the class label, "same," is strongly positively correlated with *auth_mid*, *auth_last_idf*, *aff_softtfidf*, *aff_tfidf*, *aff_jac*, *coauth_lname_jac*, *mesh_shared*, *mesh_shared_idf*, *mesh_tree_shared*, and *mesh_tree_shared_idf*. It is also negatively correlated with *jour_year_diff*, which confirms our hypothesis that the further apart two papers are published, the less likely that they will be written by the same author. Also in general, it seems that every affiliation similarity and concept similarity is quite indicative of the class label, more so than coauthor similarity and journal similarity. For coauthor similarity, the IDF values of the shared coauthors' last names do not seem to help much in the disambiguation. The simple Jaccard distance of coauthors seems to be adequate. Figure 5.4 shows the distribution for each feature in each of the two classes, 1 for the same person and 0 for a different person. The plots

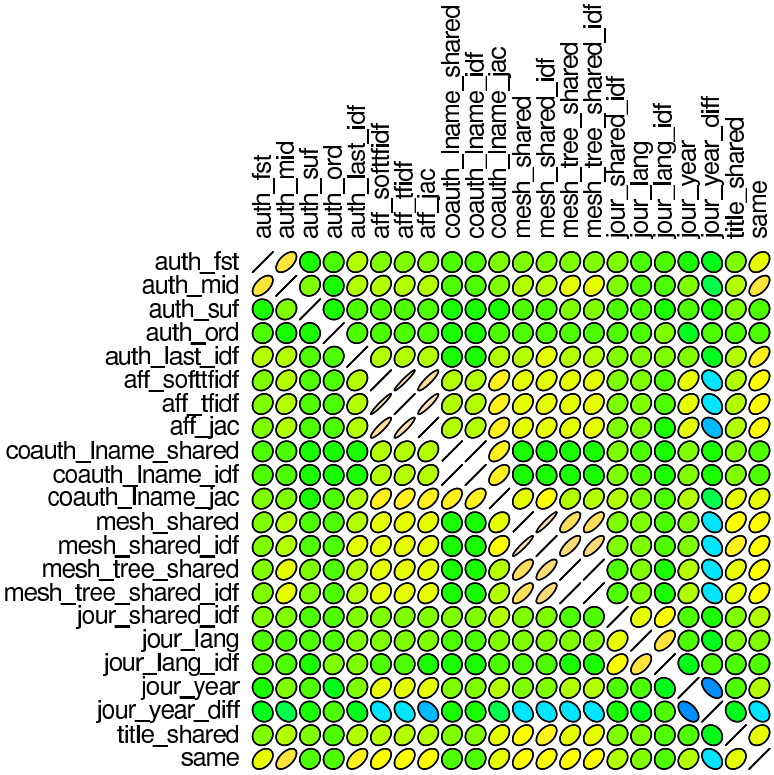

Figure 5.3. Correlations between different features and the class label in the Medline dataset. The shape and the color of the circle indicate the strength of the correlations. Yellow ellipses indicate strong positive correlations, while blue ellipses indicate strong negative correlations. The narrower the ellipse, the stronger the correlation. The green circle indicates small or no correlations.

clearly show that the normality assumption used in the NaiveBayes is violated, partly explaining the poor performance by NaiveBayes. Also, the two classes are not well-separated along any one feature.

5.1.4 Feature Selection

In figure 5.5, I ranked all twenty-one features by permutation importance and Gini importance. *auth_last_idf* and *auth_mid* have the highest importance for both measures, with around 0.2 mean decrease in accuracy. For permutation importance, the features ranked third (*aff_tfidf*) to seventeenth (*mesh_shared*) have approximately the same mean decrease in accuracy. All top six variables with high permutation importance show a strong correlation with the class label in the correlation chart. The top permutation importance attributes, *auth_last_idf* and *auth_mid*, indeed show good separation between two classes in the boxplot (Figure 5.4). The range between the 1st quartile and the 3rd quartile for both attributes is almost disjointed. Affiliation similarity is also of smaller importance than I expected. Even though all three affiliation similarities still rank high on permutation importance, their effect did not stand out as I thought they would, probably because Medline does not provide affiliations for every author. If the author

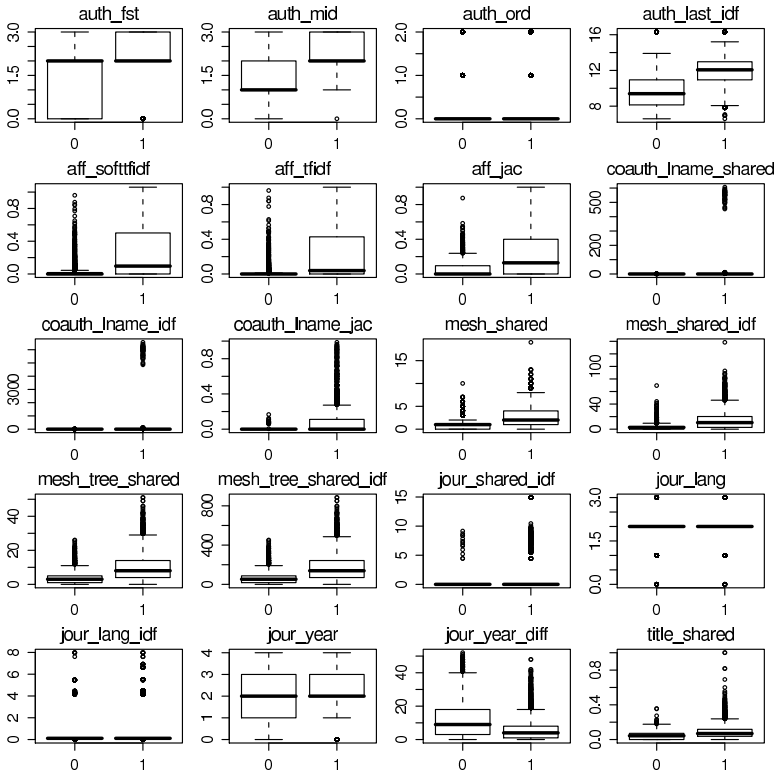

Figure 5.4. Box plot of each feature in the Medline dataset with the exception of $auth\_suf$, which has the smallest variable importance; for each class, 1: same entity, and 0: otherwise.

name in question is not the first author, then the affiliation similarities might not help much. For other datasets, such as CiteSeer, I do expect affiliation similarity to have more influence. Surprisingly, all the coauthor similarity features have low permutation importance, even though *coauth_lname_jac* is highly positively correlated with the class label. This contradicts [12], which reported common coauthor names to be the most discriminative feature. I think that this is because coauthor similarities, especially *coauth_lname_jac*, are highly correlated with affiliation similarities (Figure 5.3). Thus, when other features are included, the permutation of coauthor similarities do not have much effect on the predicted accuracy. The permutation variable importance curve is quite flat with a sharp drop-off for the low rank features, probably because there are multiple features that convey similar information (high correlation among features). Thus, when the value of one variable is permuted, the accuracy of the random forest does not decrease much. In contrast, the Gini variable importance curve shows a sharp drop-off after the top two features and shows a flat line for lower rank features. I believe that this is because features other than the top two are weak, in the sense that each is not very informative by itself. These features need to be considered in conjunction with other features. Thus, on average, their splits lower the Gini index by only a small amount.

The accuracy of random forest with feature selection is shown in Table 5.3. RF-P and RF-G are the random forests with feature selection based on permutation importance and Gini

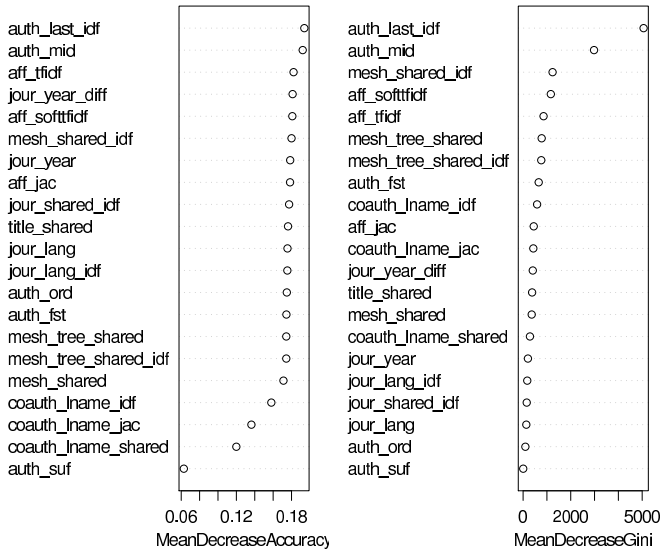

Figure 5.5. Variable importance according to permutation and Gini importance for the random forests in the Medline dataset.

Table 5.3. Accuracy based on the top 6 features with the highest variables importance ranked by permutation importance and Gini importance respectively. The features for RF-P are $auth\_last\_idf$, $auth\_mid$, aff_tfidf, $jour\_year\_diff$, $aff\_softtfidf$ and $mesh\_shared\_idf$. The features for RF-G are $auth\_last\_idf$, $auth\_mid$, $mesh\_shared\_idf$, $aff\_softtfidf$, aff_tfidf and $mesh\_tree\_shared$.

	Accuracy (%)						
#features	1	2	3	4	5	6	full
RF-P	86.1	89.4	91.7	95.1	94.9	95.5	95.9
RF-G	86.1	89.4	91.4	95.5	95.3	95.2	95.9

importance, respectively. For instance, RF-P with $\#features = 3$ refers to the random forest constructed with only the top three features ranked according to the permutation importance. With only the top variable $auth\_lname\_idf$, RF-P already performs with 86.1% accuracy. With the top six features, RF-P can achieve 95.5% accuracy, compared to 95.9% with the full model of all 21 features. With the top four features, RF-P already outperforms SVM (93.68% for S500 in Table 5.1). The performance of RF-G is almost identical to that of RF-P. The accuracy for both RF-P and RF-G seems to converge with four features. This result shows that a high level of disambiguation accuracy can be achieved with only a small subset of features. In disambiguating a pair of author names, computing the similarity profile for the reduced model is much cheaper than for the full model, and this can be done without much deterioration in the accuracy.

5.2 CiteSeer Experiments

5.2.1 Data Sampling

For the CiteSeer data, I evaluated my model on the same dataset previously used in [10, 11, 14, 15] for direct comparison. The dataset contains 12 of the most ambiguous names sampled from the entire CiteSeer database. The names, their number of records, and their number of unique authors are shown in Table 5.5. Each name contains at least 20 unique authors. For instance, C. Chen contains 536 name records corresponding to 103 unique authors. These names are chosen to be geographically diverse in order to cover names of different ethnic groups. In total, the dataset contains 3,727 names corresponding to 578 unique authors. Unlike the Medline data in Section 5.1, where each name was randomly selected, CiteSeer data only contains high frequency names. Therefore, it represents the worst case scenario, not the overall name distribution in the digital library. Because 10 of the 12 names, with the exception of J. Martin and M. Brown, were also used in Huang et al., I can directly compare my model with their results [14].

5.2.2 Comparative Studies

For every experiment in this section, I used three names, D. Johnson, J. Anderson, and M. Jones, as the training data because they were used as such in Huang et al. Each similarity profile in the CiteSeer dataset contains 12 features as defined in Section 4.2, which includes author similarity (without $auth\_suf$), coauthor similarity, affiliation similarity, and title similarity. First, I compared my random forest model (RF) with the same four classifiers in the Medline experiment: logistic regression, NaiveBayes, decision tree, and SVM, with Majority classifier as the baseline. The parameters of SVM are tuned to the training data. The random forest model is built with 200 tress ($T = 200$), and $m = int(log2(12 + 1)) = 3$.

Table 5.4 shows the baseline performance to be 88.55% on average, which is already quite high. The highest average accuracy of 97.10% is achieved by the random forest and the decision tree. The logistic regression has the second highest average accuracy of 96.72%, which is comparable to the random forest and the decision tree. The NaiveBayes (N-Bayes) has an unexpectedly good performance of 94.92%. On the other hand, SVM's performance is surprisingly low, given the fact that SVM performs quite well on the Medline dataset. Its accuracy is only 93.26%, which is even lower than that of NaiveBayes and logistic regression. SVM, however, has the highest average accuracy on the training data, which seems to suggest that it suffers from some overfittings. Overall, I find this result to be quite surprising. While the random forest still gives the best disambiguation accuracy, simple classifiers such as the NaiveBayes, the decision tree, and the logistic regression all perform very well, even better than SVM. I hypothesize that the CiteSeer dataset could in fact be much easier to disambiguate than I originally thought.

Figure 5.6 shows the training accuracy of the CiteSeer dataset, as the trees in the random forest were grown. The disambiguation accuracy of RF increases a little bit in the beginning but seems to stabilize once the number of trees reaches 25. This means that a small number of trees is

Table 5.4. Classification accuracy (%) of various classifiers on different names. The greyed-out rows were used as the training data (D. Johnson, J. Anderson, and M. Jones). The highest accuracy for each name is in bold.

	Data	Majority	N-Bayes	Logistic	DT	SVM	RF
1	A. Gupta	90.73	91.19	**96.08**	95.07	90.73	94.70
2	A. Kumar	92.93	95.86	96.54	97.25	92.93	**97.30**
3	C. Chen	88.21	96.52	**99.19**	98.49	88.21	98.18
4	D. Johnson	79.46	92.56	96.57	98.16	98.60	**98.63**
5	J. Anderson	79.19	95.95	97.79	98.77	**99.42**	99.38
6	J. Martin	91.70	94.37	96.01	**96.31**	91.86	95.28
7	J. Robinson	93.90	**98.25**	97.30	98.03	93.90	97.94
8	J. Smith	87.20	94.80	94.80	**97.21**	87.20	97.14
9	K. Tanaka	89.48	94.41	**95.72**	95.65	91.15	94.85
10	M. Brown	92.85	98.38	**98.81**	97.96	95.74	98.66
11	M. Jones	90.65	95.54	96.78	97.72	**98.50**	98.44
12	M. Miller	86.24	91.21	**95.11**	94.60	90.82	94.71
	Average	88.55	94.92	96.72	97.10	93.26	97.10

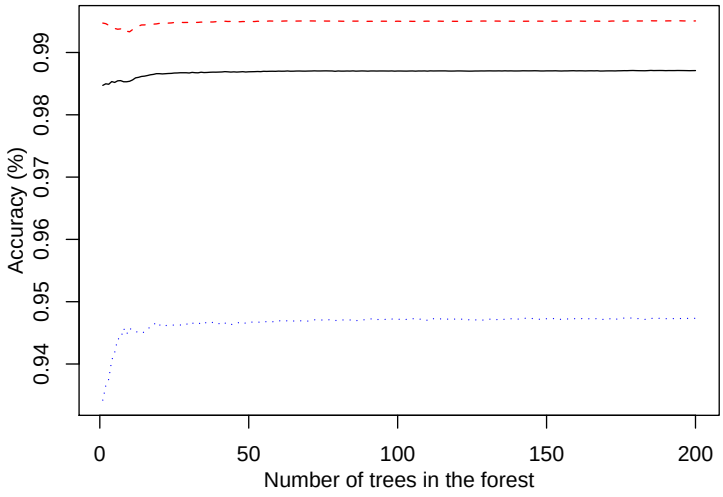

Figure 5.6. Accuracy of the random forest on the CiteSeer dataset. Black=total accuracy, Red=class 1's accuracy Blue=class 0's accuracy.

sufficient to achieve the optimal accuracy for the CiteSeer dataset. This supports my hypothesis that the CiteSeer dataset is easier to disambiguate than previously thought. However, because the similarity profile for the CiteSeer data only contains 12 features, many of which are highly correlated with each other, each tree in the random forest is also highly correlated with each other as well. As the result, adding a new tree to the forest after a certain point has little or no impact on its performance. So it is possible that the performance will increase if more features are introduced. I also observed that the class 1 accuracy is always higher than class 0 accuracy, above 99%. Thus, the trained random forest has already achieved quite a high precision (or high confidence when it thinks that two author records refer to the same person).

Table 5.5. Comparison of classification accuracy (%) between various classifiers on different names. LIBSVM and SVM_0 are the accuracy reported in Huang et al. [14]. SVM and RF are my implementation of support vector machines and random forests using my similarity profile. The greyed-out rows were used as the training data (D. Johnson, J. Anderson, and M. Jones). The average was calculated excluding rows 4, 5, 6, 10 and 11.

	Data	#Rec	#Cluster	SVM_0	LASVM	SVM	RF
1	A. Gupta	506	44	80.66	82.01	90.73	**94.70**
2	A. Kumar	143	36	93.51	93.85	92.93	**97.30**
3	C. Chen	536	103	95.12	95.12	88.21	**98.18**
4	D. Johnson	350	41	-	-	98.63	**98.63**
5	J. Anderson	327	43	-	-	99.42	**99.38**
6	J. Martin	149	40	-	-	91.86	**95.28**
7	J. Robinson	115	30	97.19	97.67	93.90	**97.94**
8	J. Smith	743	86	90.28	90.83	87.20	**97.14**
9	K. Tanaka	53	20	89.00	89.49	91.15	**94.85**
10	M. Brown	223	48	-	-	95.74	**98.66**
11	M. Jones	352	53	-	-	98.50	**98.44**
12	M. Miller	230	34	78.69	81.65	90.82	**94.71**
	Average	-	-	89.21	90.09	90.71	**96.40**

Table 5.5 compares the accuracy of SVM and random forests (RF) in this study with two models studied in Huang et al. [14]. Huang et al. proposed a pair-wise author disambiguation algorithm based on the online active learning SVM (LASVM). They evaluated their LASVM against the traditional support vector machine, which I denote SVM_0, as the baseline. SVM_0 is essentially the same as our SVM implementation, but with a different set of similarity profiles. Both SVM_0 and LASVM calculate the similarity profile based on the combination of author names, emails, affiliations, coauthors, venues, topics, and URLs. My models, on the other hand, do not use any information based on venues, topics, or URLs. Table 5.5 shows that my random forest outperforms both SVM_0 and $LASVM$ significantly on every name. Furthermore, SVM with my similarity profile also outperforms both SVM_0 and LASVM, even though my similarity profile utilizes less metadata (no information on venues, topics, or URLs). This shows that my similarity profile can be used to effectively disambiguate author names.

5.2.3 Analysis of Features

I conducted the same analysis of features for the CiteSeer dataset as I did for the Medline dataset. First, I examined the correlation between features and the class label (Figure 5.7). With the exception of $auth\_last\_idf$, every feature is positively correlated with the class label. In particular, five features, $auth\_fst$, $auth\_mid$, $aff\_softtfidf$, aff_tfidf, and aff_jac have strong positive correlations with the likelihood that two names refer to the same author. Out of the five features, $auth\_mid$ seems to be the most indicative one. Since the CiteSeer dataset only contains high frequency names, the feature $auth\_last\_idf$ (the IDF value of the last name), which was the best indicative feature in the Medline dataset, has no correlation with the class label in CiteSeer. The title similarity ($title\_share$) shows high positive correlations with the coauthor similarity,

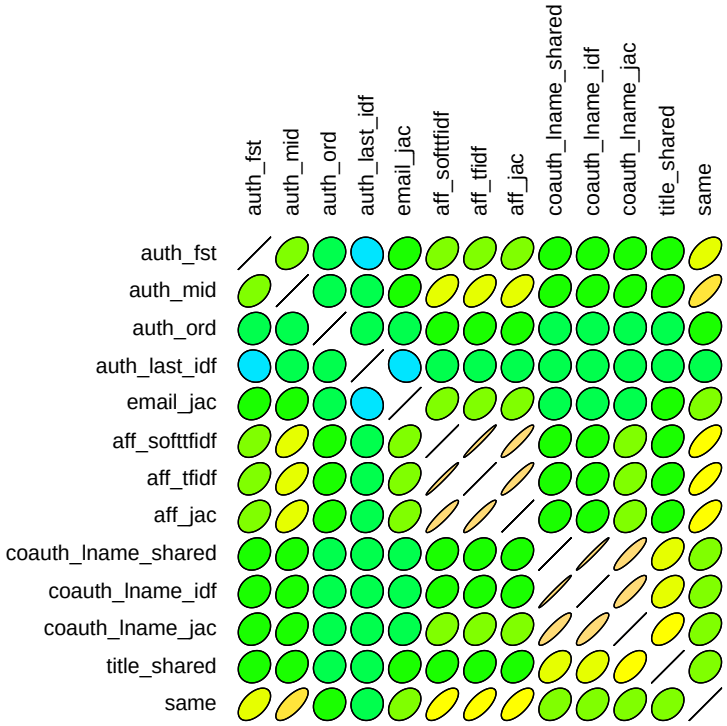

Figure 5.7. Correlations between different features and the class label in the CiteSeer dataset. The shape and the color of the circle indicate the strength of the correlations. Yellow ellipses indicate strong positive correlations, while blue ellipses indicate strong negative correlations. The narrower the ellipse, the stronger the correlation. The green circle indicates small or no correlations.

implying that the titles of papers written by the same set of coauthors are often similar. Note that *email_jac* is positively correlated with all affiliation-based features, which is not surprising.

Figure 5.8 shows the distribution for each feature in each of the two classes. All three affiliation features show great separation between classes ($aff\_softtfidf$, aff_tfidf, and aff_jac). Their values for class 0 mostly lie below their class 1's first quantile. Out of the three, aff_tfidf has the best class separation, while $affjac$ has the worst class separation. The first and middle names ($auth\_fst$ and $auth\_mid$) also have good class separation. The distributions of $auth\_last\_idf$ in the two classes are identical. The title similarity ($title\_shared$) shows some separation between classes, but not much. Coauthor similarity ($coauth\_lname\_shared$, $coauth\_lname\_idf$, and $coauth\_lname\_jac$) is mostly zero for both classes. However, there are higher variance of outliers (outside the 3rd quantile) in class 1. The same is true for $email\_jac$. When comparing Figure 5.8 with Figure 5.4, I observed that the affiliation features show greater class separation in the CiteSeer dataset. The distributions of coauthor features is similar in both datasets. The outliers of coauthor features have higher variance in the CiteSeer dataset. The distribution of the title similarity is almost identical in both datasets.

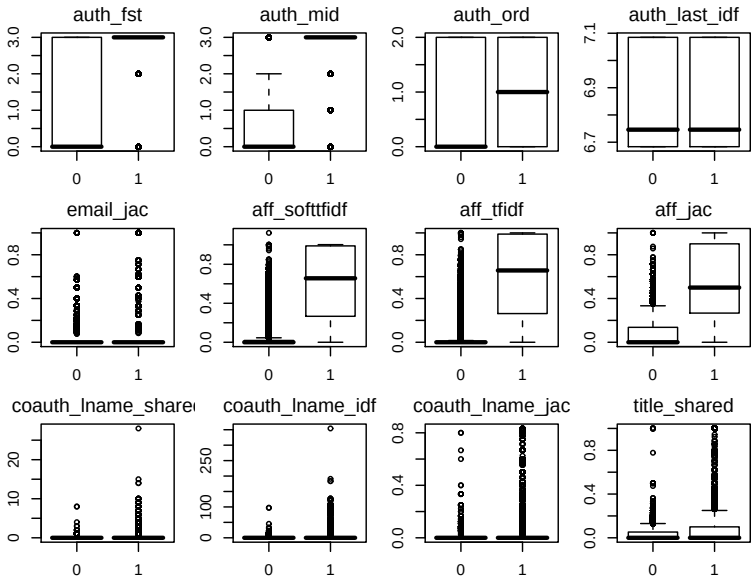

Figure 5.8. Correlations between different features in the CiteSeer dataset and the class label.

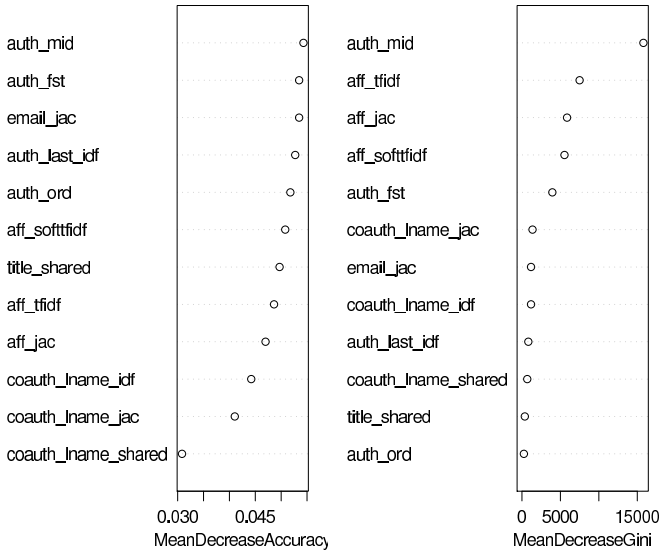

Figure 5.9. Variable importance according to permutation and Gini importance for the random forests with the CiteSeer dataset.

5.2.4 Feature Selection

All twelve features in the CiteSeer dataset are ranked by their permutation importance and Gini importance in Figure 5.9. According to the plots, *auth_mid* is the most informative feature and also the most sensitive feature. All three affiliation features are also very informative, which supports my analysis of features. Another feature that seems to be more informative than the rest is $auth\_fst$. According to the permutation importance plot, the accuracy is quite sensitive to

Table 5.6. Accuracy (%) of random forests based on the top 5 features according to Gini importance compared with that of the full model. The features are *auth_mid*, aff_tfidf, aff_jac, $aff\_softtfidf$, and *auth_fst*, respectively. The greyed-out rows were used as the training data.

	Data	Majority	2F	3F	4F	5F	Full
1	A. Gupta	90.73	91.02	91.14	89.66	92.1	94.70
2	A. Kumar	92.93	94.86	95.03	93.35	95.17	97.30
3	C. Chen	88.21	97.2	97.18	96.41	97.45	98.18
4	D. Johnson	79.46	93.86	94.18	95.57	98.11	98.63
5	J. Anderson	79.19	96.45	97.03	98.93	99.17	99.38
6	J. Martin	91.70	93.22	93.32	92.35	93.54	95.28
7	J. Robinson	93.90	97.04	97.06	96.87	97.59	97.94
8	J. Smith	87.20	95.27	95.27	94.89	97	97.14
9	K. Tanaka	89.48	93.4	93.32	91.8	93.25	94.85
10	M. Brown	92.85	97.52	97.63	96.97	97.75	98.66
11	M. Jones	90.65	95.93	96.12	97.76	97.82	98.44
12	M. Miller	86.24	93.39	93.43	92.65	92.59	94.71
	Average	90.36	94.77	94.82	93.88	95.16	96.53

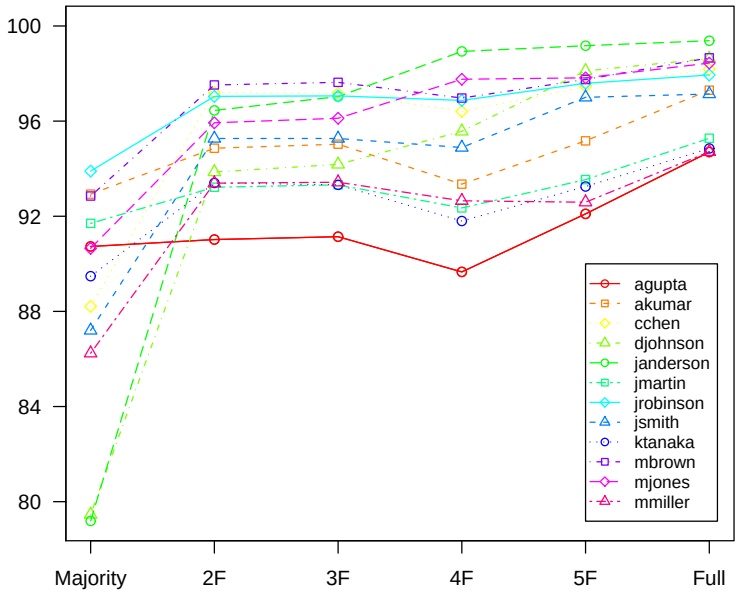

Figure 5.10. Accuracy (%) of random forests with feature selections same as in Table 5.6.

the middle name, the first name, and email. Overall, both the CiteSeer and the Medline datasets produce similar permutation importance plots and Gini importance plots. Gini importance plots, in both CiteSeer and Medline, have a sharp drop-off at the top and are flat at the tail. The main difference seems to be that for the CiteSeer data, the importance of *auth_last_idf* goes down while the importance of affiliation features goes up. The permutation importance plots are also similar in both CiteSeer and Medline.

The accuracy of random forests with feature selections is shown in Table 5.6. With only the top two features, $auth\_mid$ and aff_tfidf, the random forest can already obtain 94.77% average accuracy, which is higher than the accuracies achieved by both LASVM and SVM_0 in the prior studies. As more features are added to the model, the accuracy of the training data consistently increases (the greyed-out rows). The accuracy of the test data also generally increases as more features are added. There is a slight drop in accuracy going from the top three features to the top four features in the test data. Since the second, the third, and the fourth best features are all affiliation-related with high inter-correlations, adding the third best and the fourth best features expectedly will not improve the performance much and might even lower it a bit. Random forests work best when there is little correlation between trees.

Chapter 6

Conclusion

6.1 Summary

I demonstrated how random forests can be adapted to the problem of author name disambiguation in scientific databases. I proposed a comprehensive set of features for computing similarity profiles based on metadata in digital libraries. My experiments showed that random forests can be used to disambiguate author names effectively and efficiently. In my experiments, random forests consistently yielded the highest disambiguation accuracy on both CiteSeer and Medline data when compared with other well-known classifiers such as naive Bayes, logistic regressions, decision trees, and SVMs. My random forest models also significantly outperformed prior author disambiguation techniques in the literature in direct comparisons. Random forests are also much more scalable than SVM in the name disambiguation task, with almost a five-fold reduction in the training time on a large dataset. Random forests also are not sensitive to parameter tuning. SVM, on the other hand, needs parameter tunings in order to get good performance, which requires a substantial amount of time.

My experiments also showed that my proposed similarity profile is effective for the name disambiguation task; even the standard SVM model could gain a performance increase using my similarity profile. Furthermore, I also did an extensive analysis of features in order to better understand the interactions between features in the similarity profile. My analysis gives some insight into the difference between the disambiguation task in CiteSeer and Medline.

By analyzing variable importance in the random forest, I was able to identify a small number of predictive features for disambiguating author names. I found that the inverse document frequency value of an author's last name and the similarity between an author's middle name are the most predictive variables for disambiguating names in Medline, while authors' middle names and affiliations are the most predictive features for CiteSeer. I showed that my reduced model with only the two most predictive features could achieve almost 90% accuracy in Medline and over 94% accuracy for CiteSeer.

6.2 Future Research

For future work, there are multiple things that I would like to do. First, I plan to perform rule extraction from the random forests. Second, I would like to explore the possibility of incorporating this inexpensive classifier within a clustering algorithm to cluster entities on large scale problems such as the entire Medline database, especially in an iterative fashion.

Bibliography

[1] Li, Y., Z. Bandar, and D. McLean (2003) "An approach for measuring semantic similarity between words using multiple information sources," *IEEE Transactions on knowledge and data engineering*, **15**(4), pp. 871–882.

[2] Bilenko, M., R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg (2003) "Adaptive name matching in information integration," *Intelligent Systems*.

[3] Fellegi, I. and A. Sunter (1969) "A theory for record linkage," *Journal of the American Statistical Association*.

[4] Tejada, S., C. Knoblock, and S. Minton (2001) "Learning object identification rules for information integration," *Information Systems*.

[5] Cohen, W., H. Kautz, and D. McAllester (2000) "Hardening soft information sources," *In Proceedings of Conference on Knowledge Discovery and Data Mining*.

[6] Yang, H. and J. Callan (2005) "Near-duplicate detection for eRulemaking," in *In Proceedings of the National Conference on Digital government research*.

[7] Soon, W., H. Ng, and D. Lim (2001) "A Machine Learning Approach to Coreference Resolution of Noun Phrases," *Computational Linguistics*, **27**(4), pp. 521–544.

[8] Bekkerman, R. and A. McCallum (2005) "Disambiguating Web appearances of people in a social network," *In Proceedings of International World Wide Web Conference (WWW)*.

[9] Christen, P. (2006) "A comparison of personal name matching: Techniques and practical issues," *Workshop on Mining Complex Data (MCD)*.

[10] Han, H., C. L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis (2004) "Two supervised learning approaches for name disambiguation in author citations," *In Proceedings of the Joint Conference on Digital Libraries*.

[11] Han, H., H. Zha, and C. L. Giles (2005) "Name disambiguation in author citations using a K-way spectral clustering method," *In Proceedings of the Joint Conference on Digital Libraries*.

[12] Torvik, V., M. Weeber, D. Swanson, and N. Smalheiser (2005) "A probabilistic similarity metric for Medline records: A model for author name disambiguation," *Journal of the American Society for Information Science and Technology*.

[13] Ross, A. and A. Jain (2003) "Information Fusion in Biometrics," *Pattern Recognition Letters*, (13), pp. 2115–2125.
URL `http://linkinghub.elsevier.com/retrieve/pii/S0167865503000795`

[14] Huang, J., S. Ertekin, and C. L. Giles (2006) "Efficient Name Disambiguation for Large-Scale Databases," *In Proceedings of The European Conference on Principles and Practice of Knowledge Discovery in Databases*.

[15] Song, Y., J. Huang, I. Councill, J. Li, and C. Giles (2007) "Efficient topic-based unsupervised name disambiguation," *In Proceedings of the Joint Conference on Digital Libraries*.

[16] Bilenko, M., B. Kamath, and R. Mooney (2006) "Adaptive Blocking: Learning to Scale Up Record Linkage," *IEEE International Conference on Data Mining (ICDM'06)*.

[17] Benjelloun, O., H. Garcia-Molina, H. Kawai, and T. Larson (2007) "D-Swoosh: A Family of Algorithms for Generic, Distributed Entity Resolution," *In Proceedings of the 27th International Conference on Distributed Computing Systems*.

[18] Hernández, M. and S. Stolfo (1995) "The merge/purge problem for large databases," *In Proceedings of the 1995 ACM SIGMOD*.

[19] Jin, L., C. Li, and S. Mehrotra (2003) "Efficient record linkage in large data sets," *Database Systems for Advanced Applications (DASFAA)*.

[20] sik Kim, H. and D. Lee (2007) "Parallel Linkage," *In Proceedings of the 16th ACM Conference on Information and Knowledge Management*.

[21] Monge, A. and C. Elkan (1997) "An efficient domain-independent algorithm for detecting approximately duplicate database records," *In Proceedings of SIGMOD*.

[22] On, B., D. Lee, J. Kang, and P. Mitra (2005) "Comparative study of name disambiguation problem using a scalable blocking-based framework," *In Proceedings of the Joint Conference on Digital Libraries*.

[23] Winkler, W. (2004) "Approximate string comparator search strategies for very large administrative lists," *In Proceedings of the Section on Survey Research Methods*.

[24] Yan, S., D. Lee, M. Kan, and L. Giles (2007) "Adaptive sorted neighborhood methods for efficient record linkage," *In Proceedings of the 7th ACM/IEEE-CS joint Conference on Digital Libraries*, pp. 185–194.

[25] McCallum, A., K. Nigam, and L. Ungar (2000) "Efficient clustering of high-dimensional data sets with application to reference matching," *In Proceedings of the International Conference on Knowledge Discovery and Data Mining*.

[26] Breiman, L. (2001) "Random Forests," *Machine Learning*.

[27] Liaw, A. and M. Wiener "Classification and Regression by randomForest," *R News*.

[28] Buntine, W. and T. Niblett (1992) "A further comparison of splitting rules for decision-tree induction," *Machine Learning*, **8**(1), pp. 75–85.

[29] Winkler, W. (1999) "The state of record linkage and current research problems," *Statistics of Income Division*.

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