Disambiguating Authors in Academic Publications using Random Forests

Pucktada Treeratpituk
Information Sciences and Technology
Pennsylvania State University
University Park, PA, 16802, USA
pxt162@ist.psu.edu

C.Lee Giles
Information Sciences and Technology
Computer Science and Engineering
Pennsylvania State University
University Park, PA, 16802, USA
giles@ist.psu.edu

ABSTRACT

Users of digital libraries usually want to know the exact author or authors of an article. But different authors may share the same names, either as full names or as initials and last names (complete name change examples are not considered here). In such a case, the user would like the digital library to differentiate among these authors. Name disambiguation can help in many cases; one being a user in a search of all articles written by a particular author. Disambiguation also enables better bibliometric analysis by allowing a more accurate counting and grouping of publications and citations. In this paper, we describe an algorithm for pairwise disambiguation of author names based on a machine learning classification algorithm, random forests. We define a set of similarity profile features to assist in author disambiguation. Our experiments on the Medline database show that the random forest model outperforms other previously proposed techniques such as those using support-vector machines (SVM). In addition, we demonstrate that the variable importance produced by the random forest model can be used in feature selection with little degradation in the disambiguation accuracy. In particular, the inverse document frequency of author last name and the middle name's similarity alone achieves an accuracy of almost 90%.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation

Keywords

Author Disambiguation, Medline, Random Forests

1. INTRODUCTION

In academic digital libraries, it is often desirable to be able to disambiguate author names for many reasons. First, while browsing and searching academic articles, users would be 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 authors. Second, disambiguation of author names allows 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 be used to improve other data mining tasks such as homepage search and natural language processing. Typically, the ambiguity of person's name comes in three varieties: (1) the aliasing problem - when a person uses multiple name variations such as "Ronald W. Williams" and "R.W. Williams", (2) the common name problem - when there are more than one person with the same name; this is especially problematic for high frequency names likes Chinese names, and (3) the typographic errors.

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. A string matching is a special case of a linkage function, where the number of attribute is one. Output of a linkage function can be either a binary decision (yes or no) or a similarity score between 0 and 1. The linkage function can be either rule-based or distance measure based (e.g. Levenshtein distance). Supervised classifiers such as SVM, and decision trees also have been used as the pair-wise linkage function [14, 23]. The clustering algorithm then clusters entities based on their similarities, as defined by the linkage function.

This paper focuses on the problem of finding a high-quality pair-wise linkage function for disambiguating author names in the Medline database. More specifically, we propose to use the machine learning method, random forests. Random forests have been successfully applied to various classification problems with comparable results to other top discriminative classifiers such as SVMs and Adaboost. In addition, random forests also produce variable importance measures for each predictor variable, which can be used for feature selection. This paper has three main contributions: 1) it proposes a comprehensive feature set for disambiguating author names in Medline. 2) it explores the novel use of random forests in the problem of name disambiguation, and 3) through variable analysis and feature selection, it identifies a small set of predictive features that can achieve high disambiguation performance. The rest of the paper is organized as followed. Section 2 describes related works in author name disambiguation. Section 3 gives a brief description of Medline database. Section 4 describes random forests algorithm and the feature engineering. Section 5 discusses experimental results. Section 6 provides conclusion and discussion of future works.

2. RELATED WORK

Name disambiguation is an instance of a more general problem of 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 names and the entity is an author. Record linkage is well-studied in many different research communities under multiple names. It is referred to as record linkage [10], database hardening [9], and duplicate detection [27] in the database community. The NLP community refers to it as coreference resolution [22]. Other names include entity resolution, and string matching [4].

Much research in this area has focused on the linkage function, especially on its accuracy [1, 4, 8, 11, 12, 23, 24]. This research ranges from using simple string editing distance to machine learning techniques such as decision trees. Tejada et al. used decision tree to learn the mapping rules based on attribute similarity between records [23]. Christen made comparison study on person name matching [8]. Han et al. proposed two classifiers, hybrid Naive Bayes and Support Vector Machine (SVM), for disambiguating authors in the DBLP [11]. Bekkerman and McCallum used the link structure of web pages to disambiguate between web pages of people [1]. Huang et al. used density-based clustering to cluster CiteSeer authors based on metadata such as affiliations, emails, addresses, name variations and URLs of the papers. SVM-based distance function was used as the pair-wise similarity function [14]. Song et al. developed a topic-based disambiguation algorithm based on PLSA and LDA to disambiguate author names based on the content of the articles [21].

Other research has focused on improving the efficiency and scalability [3, 2, 13, 15, 20, 18, 19], often assuming a black-box linkage function. Monge and Elkan used a union-find data structure and assumed transitivity of linkage to efficiently merge linked records [18]. Often, the efficiency is achieved by some types of a blocking scheme that reduces the number of considered record pairs. Hernandez et al. devised sorted neighborhood methods where records are first sorted on their most discriminating attributes [13]. Each record is then compared only with its local neighbors within some fixed window. In [15], Jin et al. introduced an idea similar to [13]. But 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 [26]. 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 [3]. McCallum et al. also suggested that the efficiency can be accomplished by first clustering/blocking data into smaller partitions with an inexpensive distance measure, and later used a more expensive distance measure to disambiguate within each partitions [17]. Others have tried to improve the efficiency by increasing the parallelism of the record linkage [2, 20].

3. MEDLINE

Medline is a de facto literature resource for the biomedical research. It has metadata of over 16 millions articles dating from 1965, including 4,500 scientific journals from over 80 countries. In addition to usual metadata (titles, authors, affiliation, journal, abstract), every article in Medline is manually indexed with the Medical Subject Heading (MeSH), a controlled medical vocabulary. 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 1. Notice that the MeSH hierarchy is not a tree, and each node can have more than one parent. On average one article is associated with 10-15 MeSHes. Table 1 shows examples of three articles in Medline written by author with the name “Watson, AJ.”

Disambiguating author names in Medline can be challenging even to a human assessor. On one hand, Medline provides much metadata such as affiliation and the MeSH terms that can be used for author name disambiguation. On the other hand, it lacks much information that have been reported to be helpful in disambiguating author names, such as email, URL, and citations [14, 1]. Furthermore, Medline only consistently recorded 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. 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 uniques. The name “Wang, Y.” alone appeared in more than 7,000 articles.

Consider the three articles in Table 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 MeshHeading). For the articles (1) and (3), they are highly

Table 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, ...

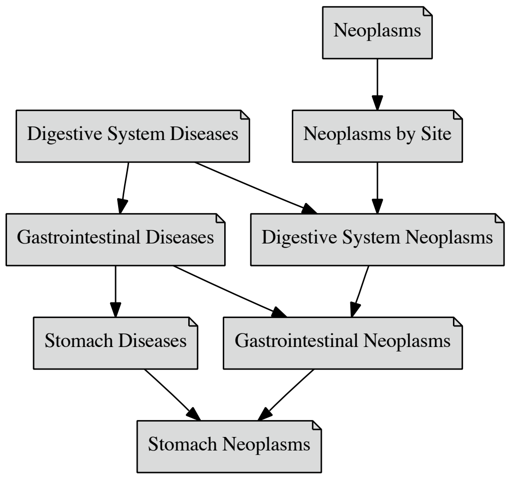
Neoplasms
Digestive System Diseases
Neoplasms by Site
Gastrointestinal Diseases
Digestive System Neoplasms
Stomach Diseases
Gastrointestinal Neoplasms
Stomach Neoplasms

Figure 1: An example of Medical Subject Headings (MeSH)'s structure.

likely to be written by the same "Watson, AJ." Not only that *University of Calgary* is 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," because (2) and (3) are both co-authored by "Schultz, GA." In fact, if one has access to the article (2) not just its metadata in Medline, one will discover that the affiliation of "Watson" and "Schultz" indeed are *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 that the accurate disambiguation becomes possible.

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 total number of instances, $M$ is the total number of predictors. $x_{ki}$ refers the $k$th attribute (predictor variable) of the $i$th instance, and $y_i$ is the class label (response variable) of the $i$th 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.
```

4. DISAMBIGUATION ALGORITHM

In this section, we first explain the random forest classifier, define variable importance, and how variable importance can be inferred from the model. Then, we define the feature set for disambiguating author names in Medline.

4.1 Random Forest

Random forest is an ensemble classifier proposed by Breiman that combines a collection of decision trees [5]. Each decision tree within the forests is built with a different bootstrap sample drawn from the original data set. 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. It 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 achieve performance that is even better than that of SVMs over a wide range of classification problems [5].

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 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 some 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 [16]. Also, since only small randomly selected subset of features ($m << M$) are 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 seperate 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 from that tree's training sample and thus can be used for error estimation. For every data point x in the original data, define the out-of-bag (OOB) trees of x as the set of trees where x is not included in their bootstrap samples. Then, calculate the error rate of the random forest on the entire original data, where the classification for each data point is done only by its out-of-bag trees. We call this error estimate the out-of-bag (OOB) error, which was shown to be an unbiased estimate of the test set error [5]. This 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, random forest can achieve good accuarcy even for the problem with many weak variables (each variable conveys only a small amount of information). Furthermore, since each classifier in the forest is just a decision tree, 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 an example, "if the affiliations are matched, and both are the first author, then ...". In addition, a random forest is very fast both in the training and 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.

4.2 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 forest attractive compared to other classifiers such as kernel-based SVM, which though often achieves good classification error rates, 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 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 criteria for selecting the split at each node in the decision tree construction; the split that yeilds 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. Then, 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 criteria for decision tree based on information theory. Since it has been shown that Gini index and Information Gain produce almost indistinguishable splits in most problems [6], we will not discuss Information Gain here.

The other measure of variable importance is the permutation accuracy importance. To compute the permutation accuracy importance of a variable X_i, for each tree that contains variable X_i, first randomly permute the 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. Then, the variable importance of X_i is this decrease in accuracy averaged over all trees that contains X_i. The rationale is that if the variable X_i is strongly correlated with the response, by permuting the value of X_i, the prediction accuracy should decrease significantly. Both permuation importance and Gini importance can be used to measure the relevance of each variable in the feature selection.

4.3 Similarity Profile

Here, we 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. We want to disambigute whether they refer to the same person. We 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 the 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 CiteSeerX, where author names are automatically extracted, this assumption shoud be tested. The following metadata of $paper_A$ and $paper_B$ are used to construct the similarity profile:

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

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 name in $paper_i$
aff_i = affiliation 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 21 features, which can be grouped into six categories based on the metadata, they are calculated from: author similarity, affiliation similarity, coauthors similarity, concept similarity, journal similarity, and title similarity. The 21 features are defined as followed:

Author Similarity

1) $auth\_fst$: the first name similarity.

$$auth\_fst = \begin{cases} 0 & \text{if } fname_A \neq fname_B \text{ and both are fullname} \\ 1 & \text{if } init_A \neq init_B \text{ and are not both fullname} \\ 2 & \text{if } init_A = init_B \text{ and are not both fullname} \\ 3 & \text{if } fname_A = fname_B \text{ and both are fullname} \end{cases}$$

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

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

3) $auth\_suf$: similarity between suf_A and suf_B.

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

4) $auth\_ord$: similarity between the orders of author.

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

5) $auth\_lname\_idf$: IDF weight of the author 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, DF_{lname_A} is the total numbers of $lname_A$ in Medline. High value of $IDF(lname)$ means that $lname$ is rare, thus is less likely to be ambiguous.

Affiliation Similarity

6) aff_jac: the jaccard similarity between aff_A and aff_B

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

7) 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.

8) $aff\_softtfidf$: 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. 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 [4]. Here, we use Jaro-Winkler distance as the measurement whether two terms are similar [25].

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

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,

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

Coauthors Similarity

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

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

10) $coauth\_lname\_idf$: the sum of IDF values of all shared coauthor last names.

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

11) $coauth\_lname\_jac$: the jaccard similarity between $coauth_A$ and $coauth_B$.

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

Concept Similarity

12) $mesh\_shared$: the number of shared mesh terms between the two papers.

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

13) $mesh\_shared\_idf$: the sum of IDF values of all shared mesh terms.

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

14) $mesh\_tree\_shared$: Define a set $T(mesh_i)$ to be the set of all ancestor concepts of $mesh_i$ in MeSH hierarchy. $c \in T(mesh_i)$ 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 to $mesh_i$. The feature takes into account the tree structure of the MeSH hierarchy. 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 $mesh\_tree\_shared$.

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

15) $mesh\_tree\_shared\_idf$: similar to $mesh\_tree\_shared$, but instead of the number of shared MeSH headings in the hierarchy, use the IDF weight sum.

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

Journal Similarity

16) $jour\_shared\_idf$: IDF value of the shared journal, if both are published in the same journal. The less common the journal is, the more informative this feature should be.

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

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

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

18) $jour\_lang\_idf$: IDF value of the shared journal's language.

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

19) $jour\_year$: categorical variable reflecting 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.

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

20) $jour\_year\_diff$: difference in publication years.

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

Title Similarity

21) $title\_shared$: the jaccard similarity between $title_A$ and $title_B$.

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

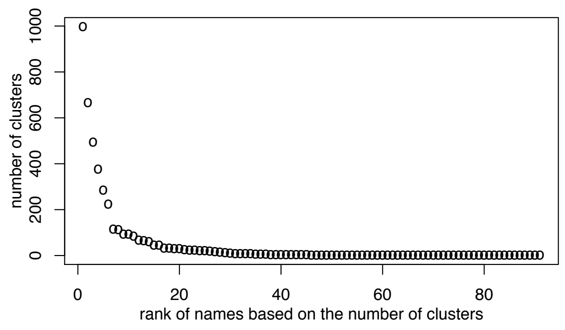

Figure 2: Number of unique author clusters for each of 91 sampled author names.

5. EXPERIMENTS

5.1 Comparative Studies

To evaluate the performance of the random forest for disambiguation, we first randomly select 91 unique author names (as defined by the last name and the first initial) from Medline database. For each selected name, we then manually cluster all the 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 the training and the evaluation. Figure 2 shows the number of unique authors 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. We also utilize 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, we will look at the full affiliations 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), we assume each was written by a unique author. We then randomly sample without replacement 100 article-article pairs from each of the 91 names gold standard, and aggregate them all together to create the evaluation set called S100. S100 contains the total of

Table 2: 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

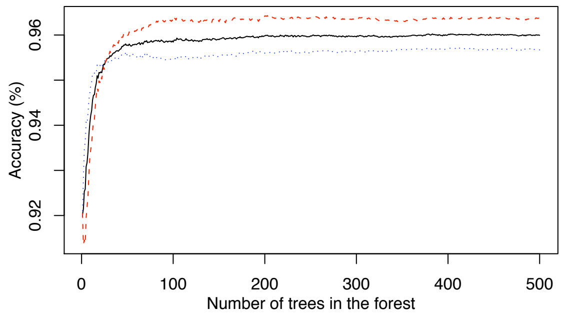

Figure 3: Accuarcy of the random forest on S500. Black=the total accuracy, Red=class 1's accuracy Blue=class 0's accuracy.

8,064 instances (article-article pairs). Similarly, we also create S200, S300, S400, S500 evaluation set with the sample size = 200, 300, 400, 500 respectively. The sampling is 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 similarity profile, we collect 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*, 2008 version of the MeSH hierarchy is used.

We compared the performance of our random forest model (RF) with four other traditional classifiers, logistic regression, Naive-Bayes, a decision tree, and a SVM (Table 2). We 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, we use 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. We use the libsvm implementation of SVM [7].

Table 3: Training time (minutes & seconds) on different sample sizes

set	S100	S200	S300	S400	S500
SVM	25.83s	1m59s	4m31s	7m49s	12m39s
RF	21.39s	46.80s	1m16s	1m47s	2m20s

The decision tree (DT) is built using C4.5 algorithm with pruning. In our experiment, the random forest is built with 500 trees ($T = 500$), and $m = int(log2(21 + 1)) = 4$. The number reported in Table 2 are computed using 10-folds cross-validation. Each fold is stratified so that it contains approximately the same proportions of class distribution as the original dataset. Random forest consistently outperforms all other classifiers for every data set, achieving almost 96% accuracy for the S500 data. Figure 3 shows the accuracy on S500 data, as the trees were grown in the random forest. There is small change from 100 to 500 trees, suggesting that 100 trees might be sufficient to get a reasonable result. 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 poor 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 our NaiveBayes is also not valid. It is interesting to note that logistic regression, decision tree, SVM and random forests all can achieve over 90% accuracy. This is actually not bad considering that none of the non-first author's affiliations is available to these classifiers.

We also look at the effect of training data size on the accuracy and training time. Each classifier achieves similar accuracy across all five data set. The random forests and the decision tree seem to perform slightly better on the bigger data set. Further experiments with larger data set are needed to see whether such a improvement will persist. Table 3 compares the training time between SVM and random forests (all other classifiers take under four seconds to train on each data set). The training time of random forests increases linearly with the size of the training data, while SVM takes noticeably longer time as the size of the training data increases. Additionally, parameter tuning in SVM takes up substantial amount of time, for instance, the grid search on S200 took over 6 hrs.

5.2 Features Selection

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 [17]. Here, we investigate the relevance of each features to the prediction, and then look at the effectiveness of the feature selection.

First, we examine the correlation between features, and their class distribution. The correlation between each features and the class label are shown in Figure 5. The shape and the color of each circle indicates the strength of the correlation. The green circle indicates that there is little or no correlation between features. The yellow ellipse indicates the positive correlation between the two features, while the blue ellipse indicates the 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 group-

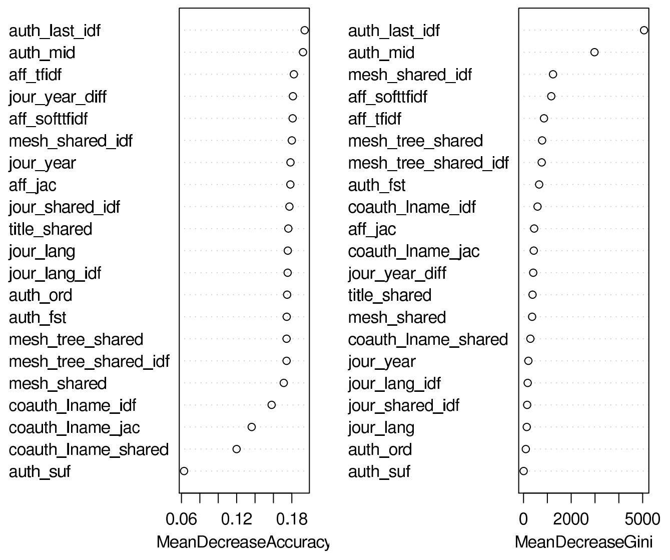

Figure 4: Variable importance according to permutation and Gini importance for the random forests.

ing of related features, whether they are affiliation-based or coauthors-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*. Figure 6 shows the distribution for each feature in each of the two classes, 1 for same person and 0 for different person. The plots clearly show that the normality assumption used in the NaiveBayes is violated, partly explain the poor performance by NaiveBayes. Also, the two classes are not well-separated along any one feature.

We ranked all twenty-one features by permutation importance and Gini importance (Figure 4). *auth_last_idf* and *auth_mid* have the highest importance for both measures with around 0.2 mean decrease accuracy. For permutation importance, the features ranked third (*aff_tfidf*) till seventeenth (*mesh_shared*) have approximately the same mean decrease accuracy. All top six variables with high permutation importance show the strong correlation with the class label in the correlation chart. The top permutation importance attributes, *auth_last_idf* and *auth_mid*, indeed show good seperation between two classes in the boxplot (Figure 6). The range between the 1st quartile and the 3rd quartile for both attributes are almost disjointed. Affiliation similarity also have smaller importance than we expected. Even though all three affiliation similarities still rank high on permutation importance, their effect did not stand out as we thought they would. This is probably because Medline does not provide affiliations for every authors. If the author name in question is not the first author, then the affiliation similarities might not help much. For other data set, such as CiteSeerX, we do expect affiliation similarity to have more influences. 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 [24] which reported common coauthor names to be the most discriminative feature. We think that this is because coauthors similarities, especially *coauth_lname_jac*, are highly correlated with affiliation similarities (Figure 5). Thus, when other features are included, the permutation of coauthors 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. We think that it is 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 sharp drop off after the top two features, and shows flat line for lower rank features. We believe that this is because features other than the top two are weak, in a sense that each is not very informative by itself. They need to be considered in conjunction with other features. Thus, on average, their splits lower the Gini index by only small amount.

Table 4: Accuracy based on the top 6 features with the highest variables importance ranked by permutation importance and Gini importance respectively (ie. *auth_last_idf*, *auth_mid*, aff_tfidf, $jour\_year\_diff$, $aff\_softtfidf$, *mesh_shared_idf* for RF-P)

	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

The accuracy of random forest with feature selections is shown in Table 4. RF-P and RF-G are the random forests with feature selection based on permutation importance, and Gini 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 all 21 features. With the top four features, RF-P already outperforms SVM (93.68% in Table 2). 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 high level of disambiguation accuracy can be achieved with only 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.

6. CONCLUSIONS

We demonstrate how random forests can be adapted to the problem of author name disambiguation in scientific databases. We propose a wide-range of features for computing similarity profiles based on metadata available in Medline. These features can be easily adapted to other digital libraries. Our experiments show that the random forest model outperforms other previously used classifiers, such as SVM, in the pair-wise disambiguation task. By analyzing variable importance in the random forest, we were able to identify a small number of predictive features for disambiguating author names. We found that the inverse document frequency value of author's last name and the similar-

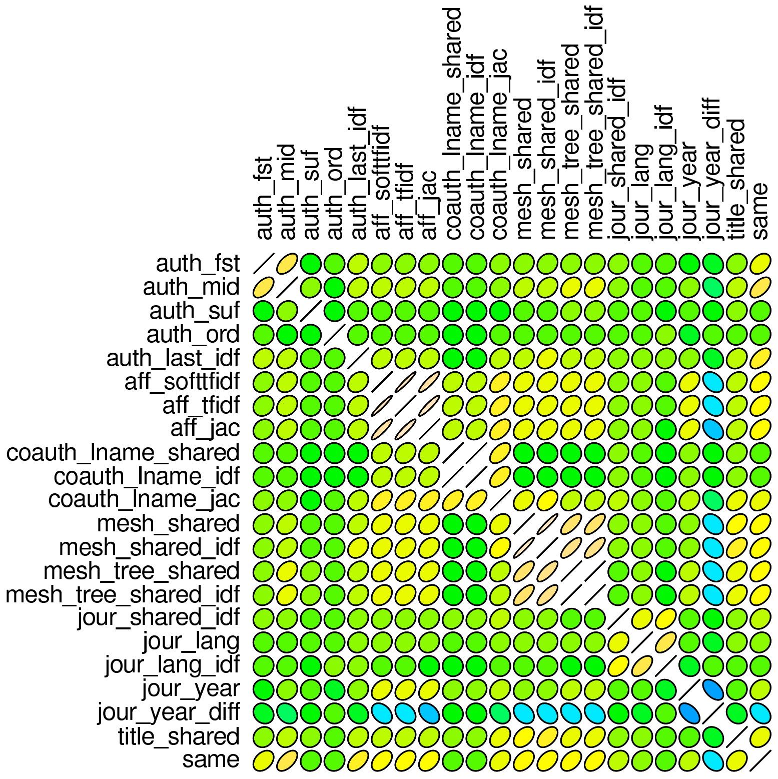

Figure 5: Correlations between different features and the class label. The shape and the color of the circle indicate the strength of the correlations.

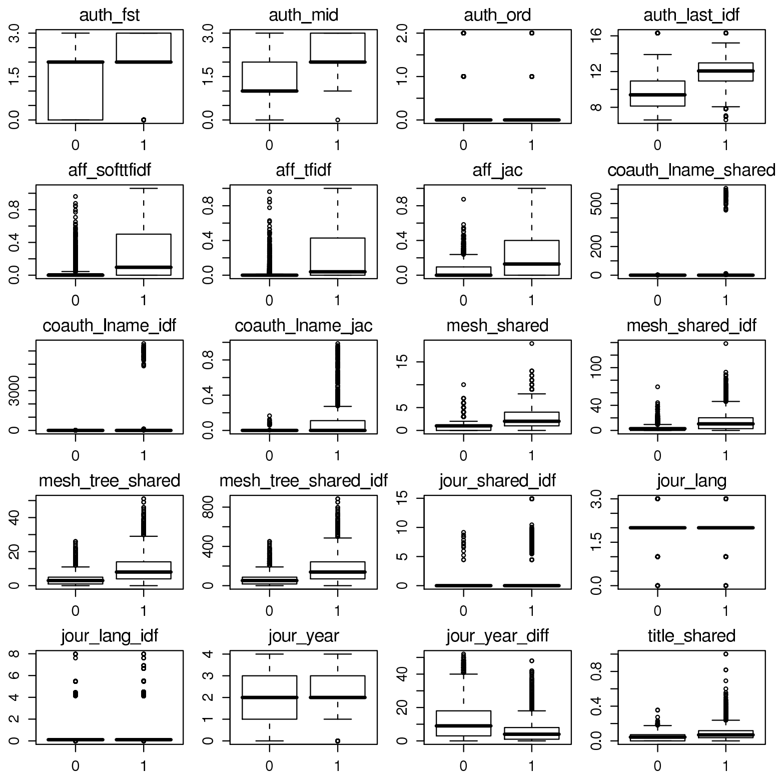

Figure 6: Box plot of each features with the exception of auth suf, which has the smallest variable importance, for each class, 1: same entity, and 0: otherwise.

ity between author's middle name are the most predictive variables for disambiguating names in Medline. The reduced random forest model using just those two variables can attain almost 90% accuracy. Our experiments with feature selections also demonstrate that near-optimal accuracy can be achieved with just four variables, the inverse document frequency value of author's last name and the similarity between author's middle name, their affiliations' tfidf similarity, and the difference in publication years. For future work, we would like to explore the possiblity of incorporating this inexpensive classifier with a clustering algorithm to cluster entities on large scale problems such as the entire Medline database.

7. ACKNOWLEDGMENTS

The authors would like to acknowledge partial support from the National Science Foundation and Raytheon. We gratefully acknowledge Prasenjit Mitra for his help with the Medline data, and he, Jian Huang, Zheng Shuyi, Pradeep Teregowda, and B.J. Simpson for their useful suggestions.

8. REFERENCES

[1] R. Bekkerman and A. McCallum. Disambiguating web appearances of people in a social network. *Proc of Int'l World Wide Web Conf (WWW)*, 2005.

[2] O. Benjelloun, H. Garcia-Molina, H. Kawai, and T. Larson. D-swoosh: A family of algorithms for generic, distributed entity resolution. *Proc of the 27th Int'l Conf on Distributed Computing Systems*, 2007.

[3] M. Bilenko, B. Kamath, and R. Mooney. Adaptive blocking: Learning to scale up record linkage. *IEEE Int'l Conf on Data Mining (ICDM'06)*, 2006.

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

[5] L. Breiman. Random forests. *Machine Learning*, 2001.

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

[7] C. Chang and C. Lin. Libsvm: a library for support vector machines. *http://www. csie. ntu. edu. tw/cjlin/libsvm*, 2001.

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

[9] W. Cohen, H. Kautz, and D. McAllester. Hardening soft information sources. *Proc of Conf on Knowledge Discovery and Data Mining*, 2000.

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

[11] H. Han, C. L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis. Two supervised learning approaches for name disambiguation in author citations. *Proc of the Joint Conf on Digital Libraries*, 2004.

[12] H. Han, H. Zha, and C. L. Giles. Name disambiguation in author citations using a k-way spectral clustering method. *Proc of the Joint Conf on Digital Libraries*, 2005.

[13] M. Hernández and S. Stolfo. The merge/purge problem for large databases. *Proc of the 1995 ACM SIGMOD*, 1995.

[14] J. Huang, S. Ertekin, and C. L. Giles. Efficient name disambiguation for large-scale databases. *Proc of The European Conf on Principles and Practice of Knowledge Discovery in Databases*, 2006.

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

[16] A. Liaw and M. Wiener. Classification and regression by randomforest. *R News*.

[17] A. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. *Proc of the Int'l Conf on Knowledge Discovery and Data Mining*, 2000.

[18] A. Monge and C. Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. *Proc of SIGMOD*, 1997.

[19] B. On, D. Lee, J. Kang, and P. Mitra. Comparative study of name disambiguation problem using a scalable blocking-based framework. *Proc of the Joint Conf on Digital Libraries*, 2005.

[20] H. sik Kim and D. Lee. Parallel linkage. *Proc of the 16th ACM Conf on Information and Knowledge Management*, 2007.

[21] Y. Song, J. Huang, I. Councill, J. Li, and C. Giles. Efficient topic-based unsupervised name disambiguation. *Proc of the Joint Conf on Digital Libraries*, 2007.

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

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

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

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

[26] W. Winkler. Approximate string comparator search strategies for very large administrative lists. *Proc of the Section on Survey Research Methods*, 2004.

[27] H. Yang and J. Callan. Near-duplicate detection for eRulemaking. In *Proc of the National Conf on Digital government research*, 2005.