Alias Detection in Link Data Sets

Paul Hsiung and Andrew Moore and Daniel Neill and Jeff Schneider
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
{hsiung+, awm, neill, schneide}@cs.cmu.edu

ABSTRACT

The problem of detecting aliases - multiple text string identifiers corresponding to the same entity - is increasingly important in the domains of biology, intelligence, marketing, and geoinformatics. This paper investigates the extent to which probabilistic methods can help.

Aliases are created from entities who are trying to hide their identities, from a person with multiple names, or from words which are unintentionally or even intentionally misspelled. While purely orthographic methods (e.g. string similarity) can help solve unintentional spelling cases, many types of alias (including those adopted with malicious intent) can fool these methods.

In this paper we try to take advantage of the observation that even if an entity has a changed name in some context, some or all of the set of other entities with which it has relationships can remain stable. We thus try to exploit the local social network by using the relationships as semantic information.

We propose a combined algorithm that takes advantage of both orthographic and semantic information to detect aliases. This combined algorithm is able to take the best of both types of information and outperform algorithms that were built strictly on orthographic or semantic information. We give empirical results on three real world data sets, then describe and demonstrate approaches to make such analyses scalable.

Keywords

link data sets, string-edit distance, social networks

1. INTRODUCTION

The premise of a link data set is that one **entity** represents an unique individual whether they be an actual person, word, or even research paper. However, each entity can have many **names**. If two or more names map to an entity, we call them **aliases**. A link data set consists of a set of names and a set of **links**. Each link contains two or more names and represents an observed or probabilistic relation between the names. The definition of relation is very general but can be specified when we focus on a particular data set. For example, a relationship can be any connection ranging from membership in the same terrorist cell to collaboration on a research paper.

The motivation for link data sets arises from the way information is presented in media such as news articles and emails. Link data sets offer a concise, general purpose, information-filled, and computable form of data as opposed to a raw newspaper article. For example, consider the following web-article [1]:

```
Wanted al-Qaeda terror network chief Osama bin
Laden and his top aide, Ayman al-Zawahri, have
moved out of Pakistan and are believed to have
crossed the mountainous border back into
Afghanistan.
```

Rather than using the entire free text of this article, we can summarize the information in a small number of links. For example, we can have a link describing a terrorist cell:

$$(Osama\ bin\ Laden, Ayman\ al\ Zawahri, al\ Qaeda)$$

Regarding location information of people, we can have links:

$$(Pakistan, Osama\ bin\ Laden)$$

$$(Afghanistan, Osama\ bin\ Laden)$$

For this paper, we do not take into account the types of relationship between names in the links. We just use the statistical information that names appear together, counting the number of times in which they fall into the same link.

On the positive side, link data sets create a way for algorithms to understand human-readable information. But on

[1] http://uk.news.yahoo.com/040225/323/emvp1.html

the negative side, link data is susceptible to noise and often requires human interventions and subjective judgments. The use of link data was previously discussed in these papers: [2] [5].

The way that the intelligence community gathers data is compatible with the way in which link data sets are constructed. Intelligence analysts collect articles (often written in foreign languages) and write down their subjective observations concerning the relations between names inside the articles. In this case, aliases can occur based on misspelled names, cross-lingual transliterations (*Usama* and *Osama*), or even different titles (*Usama* might be called *The Emir*).

The problem of alias detection is very broad. In another variant of this problem, one name corresponds to many entities. For example the name Michael Jordan represents both a statistician and a sports figure as well as many others who share that name. This case was discussed in [7] (as a special case of the word sense disambiguation problem) and [4].

Our focus in this paper is on many names corresponding to a single entity. For example, *Osama* is also known as *Usama*. These two strings are orthographically similar. So it is easy to realize that they are aliases. But there are other more difficult aliases such as *The Prince* or *The Emir*. For these aliases, we have to exploit the local social network structure of these names. We define the **friends** of *Osama* as all the names that have appeared in the same link (have some relationship) with *Osama* (i.e. occur in the same link). To exploit the social network, we compare friends of *Osama* with friends of *The Prince* and see how well these two sets of friends correspond.

We restrict our problem to the test of whether a pair of names are aliases based on orthographic information and/or semantic information. All the following models and measures are based on analysis of pairs of names in a link data set.

2. PROBABILISTIC ORTHOGRAPHIC MODEL

The most natural measure of likelihood that two names are aliases is orthographic similarity. If two names are very similar, they are likely to be two versions of the same name. One of the most popular measures of similarity is string edit distance: the minimum number of insertions, deletions, and substitutions required to transform one string into the other (see section 5.6 of [4]).

There are many possible string edit distance functions that measure similarity. The central question is either, how to choose the best one of these measures, or, how to effectively combine multiple orthographic measures? This will be discussed in Section 4.

2.1 Orthographic Measures

We will now describe a suite of potential orthographic measures.

String Edit Distance (SED) Again, this is the minimum number of insertions, deletions, and substitutions required to transform one string into the other.

Normalized String Edit Distance (NSED) This is computed by dividing *SED* with the max length of the two string we are comparing.

$$NSED(s_1, s_2) = \frac{SED(s_1, s_2)}{max(length(s_1), length(s_2))}$$

Discretized String Edit Distance (DSED) We binarized *NSED* by manually setting a threshold of 0.7 based on general observations. If the *NSED* is less than 0.7, we set *DSED* to 0, and we set *DSED* to 1 otherwise.

Exponential String Edit Distance (ESED)

$$ESED(s_1, s_2) = exp(SED(s_1, s_2))$$

There are many other possible measures mentioned in [9] and [10].

Table 1: Hypothetical Example of Friends Lists

	Osama	The Prince
# Occurrences with AlQaeda	10	2
# Occurrences with CNN	2	8
# Occurrences with Music	0	50
# Occurrences with Islam	5	0

Table 2: Normalized Friends Lists

	Osama	The Prince
Normalized AlQaeda	0.588	0.033
Normalized CNN	0.118	0.133
Normalized Music	0.000	0.833
Normalized Islam	0.294	0.000

3. PROBABILISTIC SEMANTIC MODEL

We believe that the set of links that names have appeared in offers additional secondary evidence for this problem. For example, if two people have almost exactly the same set of friends (i.e. people they have historically occurred in links with) in the same historical proportions, then it is more likely (though by no means certain) that the two people are in fact aliases for one person. We have implemented a battery of such semantic similarity tests. Again the question of how to combine the measures is discussed in Section 4.

3.1 Semantic Measures

Dot Product (DP) For each name we can represent its friends list as a vector of occurrences with other names it appears with. *Dot Product* is just the dot product of two vectors from two names. For example, in Table 1, we calculate

$$DP(Osama, ThePrince) = 10 * 2 + 2 * 8 = 36$$

Normalized Dot Product (NDP) This is almost the same except we normalize each vector by dividing by its magnitude before taking the dot product. In Table 2, we have

$$NDP(Osama, ThePrince) =$$

$$0.588 * 0.033 + 0.118 * 0.133 = 0.0351$$

Common Friends (CF) We define **common friends** as friends that co-occur with both names. In our example, *AlQaeda* and *CNN* are common friends of *Osama* and *The Prince*. In this measure, we record the number of common friends, which is 2 in our example.

KL Distance (KL) We can treat normalized friends lists as probability vectors. Then we can use *KL* to measure the similarity between these two vectors. We use add-one smoothing (i.e. add one to each value of the vector before normalization) to deal with cases where two entities do not co-occur in a link. The KL distance is given:

$$\sum_i O_i log(\frac{O_i}{P_i}) + P_i log(\frac{P_i}{O_i})$$

where P_i is the ith entry in probability vector of *The Prince* and O_i is the ith entry of *Osama*.

4. COMBINED MODEL

We have reviewed and introduced a number of measures of similarity between a pair of names that could help identify if they are aliases. The main question addressed in this paper is how they should be combined. We consider four approaches below.

1. **Manually pick the best single measure.** This "feature selection" approach does not work well because of the fact that both orthographic and semantic similarity convey different and useful information. This is backed up in the results section, where we demonstrate that our combined model outperforms either only using orthographic or only using semantic measures.

2. **Hand designed formulas** This method is based on subjective judgments made by human beings. On the one hand, we might be able to understand what the model represents, but this might not produce the optimal combination. Worse, the best combination is likely to change from domain to domain. Examples of this are Baroni et al. [1], who used weighted sum, and Hernandez et al. [3], who manually created rules.

3. **Probabilistic model** Another approach to this problem would be a full probabilistic generative model of links, names, and the string corruption process. Under the assumption that it would be possible to model such a complex set of link phenomena and under the assumption that it would be computationally tractable to solve such models for more than a few hundred names, this would be a very promising approach. Marthi et al. [6] gave a very good detailed description of what such a generative model would be like in the related area of bibliometrics analysis. Future work in this area is likely to continue and be productive though both the representational and computational challenges will be severe.

4. **Semi-supervised learning** This approach requires a human to provide a relatively small set of positive and negative examples of whether a pair of names are aliases. These examples are used to build a classifier that tests if two names are aliases. This is the approach taken in this paper.

5. ALIAS CLASSIFIER

5.1 Training

In this combined semi-supervised model, the goal is to train a classifier that tests whether two names in a link data set are aliases. For training the classifier, the user gathers a collection of name pairs that includes positive and negative examples. Positive examples come from human-selected aliases. Because our pool of names is so large and the number of true aliases is small, the user can, with a high degree of safety, pick negative examples by randomly selecting pairs of names among the link data set with a very low chance of picking a true pair of aliases. For each pair of names, we can calculate all of the measures in both the semantic and orthographic models and add them as attribute into our training set. We also keep an attribute called *Alias?* to label the positive and negative examples. Each pair of names with all its attributes represents a row in the training set. For example, see Table 3.

Having framed the problem this way, we are back in the more familiar world of straightforward classification. We performed a series of experiments using a suite of common classification algorithms including Decision Trees, KNN, Naive Bayes, SVM, and Logistic Regression. Logistic Regression had slightly better general performance, and so that is used in subsequent experiments in this paper. However, we believe that it would be possible, in follow-up work, to investigate a wider range of classifiers. The space of possible classifiers is somewhat constrained by the requirement that they can rank order their predictions. So they need some measure of conditional class probability in addition to a straight classification.

5.2 Prediction

For the task of prediction, we first pick a query name, for instance, *Osama*. We pair *Osama* up with all of the other names in the link data set. Then we compute the attributes for all the pairs we created. We run the classifier on all the pairs and rank them by the class conditional scores from the classifier. For an example prediction data set, see Table 4.

6. EMPIRICAL EVALUATION

In order to evaluate our algorithms, we need to use a large link data set filled with alias-rich information. Because of the large amount of data needed, we decided to use information that can be gathered on a day-to-day basis such as newspaper articles and emails. However, the type of people who use aliases are the ones that tend to hide from the general public. Bearing this in mind, two obvious candidates for our link data sets are terrorists and spam. We can obtain terrorist information from newspaper articles. And of course, spam is readily available.

We chose spam also because spammers will often try to create aliases through intentional misspelling to confuse the

Table 3: Hypothetical training set

		Orthographic Measures		Semantic Measures		
name1	name2	SED	...	DP	...	Alias?
Osama	The Prince	15	...	36	...	yes
Usama	Osama	2	...	24	...	yes
...	...	...	...	...	...	...
Bob	Sid	6	...	2	...	no
John	The Prince	10	...	5	...	no
...	...	...	...	...	...	...

Table 4: Hypothetical prediction data set

name1	name2	SED	...	DP	...	Classifier's Estimate $\hat{P}(alias \mid measures)$
Osama	The Prince	15	...	36	...	0.70102
Osama	Usama	2	...	24	...	0.69283
Osama	Bob	6	...	6	...	0.11451
Osama	Sid	6	...	4	...	0.02315
Osama	John	7	...	12	...	0.01204
...	...	...	...	...	...	...

spam filter. For example, instead of using *mortgage* in a email soliciting a loan, the spammers might use the word *m0rtg@ge* instead. While the spam filter might be confused, any human being will recognize the similarity between *mortgage* and *m0rtg@ge*.

We choose to represent each spam email as a separate link, with the names in that link being the word-tokens appearing in the subject header and the main body of the email. However this requires several steps of pre-processing on each email message:

1. We parsed out all the HTML tags.
2. All the words are converted to tokens with the boundary being white space or new lines. All the tokens are unique, so multiple occurrences of a single word in an email will be treated as a single occurrence.
3. The tokens are filtered through a stop list of about 120 common words.
4. If a token does not appear in more than 2 emails, we eliminate it.

For example, given a spam email that looks like this

```
Subject:Mortgage rates as low as 2.95%

Ref<suyzvigcffl>ina<swwvvcobadtbo>nce
to<shecpgkgffa>day to as low as
2.<sppyjukbywvbqc>95% Sa<scqzxytdcua>ve
thou<sdzkltzcyry>sa<sefaioubryxkpl>nds
of dol<scarqdscpvibyw>l<sklhxmxbvdr>ars
or b<skaavzibaenix>uy the <br>
ho<solbbdcqoxpdxcr>me of yo<svesxhobppoy>ur
dr<sxjsfyvhhejoldl>eams!<br>
```

Our final processed link might look like this (given that all these tokens appeared in more than two other emails):

$$(mortgage, rates, low, refinance, today,$$

$$save, thousands, dollars, home, dreams)$$

Table 5: Data sets and their size

Data set	Links	Names
Terrorists	5581	4088
HsiungSpam	373	2452
ArchiveSpam	5601	8451

6.1 Link Data Sets

In this section we describe the three link data sets we used to evaluate our algorithm.

Terrorists This link data set was manually extracted from a set of public web pages and news stories (often written/hosted by various governments and news organizations) related to terrorism. The names mentioned in the articles were linked subjectively upon reading the information. This data was used in Kubica et al[5].

HsiungSpam is a collection of spam emails from the author's mailbox. See Table 5 for size information.

ArchiveSpam is a collection of spam emails from the website *www.spamarchive.org*. The setup is the same as *HsiungSpam*. Also see Table 5.

6.2 Alias Selection

To gather positive training examples for the semi-supervised learning, we had to find all the aliases for a particular entity. Then we can generate alias pairs by exhaustively matching up all aliases that belong to that entity. Each alias pair corresponds to a positive example in the training set.

We now describe how we collect our aliases from the three link data sets.

Table 6: Data sets and alias ground truth

Data set	Source	Ground Truth Entities	Alias Pairs
Terrorists	FBI Website	20	919
HsiungSpam	Hand Labeled	21	89
ArchiveSpam	Hand Labeled	10	47

Terrorists We collected the aliases from the FBI website of the top 20 most wanted terrorists. So we have 20 entities, each having two to 14 aliases each. The entity with two aliases only generates one alias pair where as the entity with 14 aliases generates $\binom{14}{2}$ or 91 pairs of aliases. In this training set, there are 919 possible alias pairs (positive training examples).

HsiungSpam We manually searched for aliases. See Table 6. Here is a subset of the aliases:

1. add added increase plus
2. big huge large massive
3. pill pills drugs
4. viagra v1a*ra v1agra
5. fr—ee free freee

ArchiveSpam We also manually searched for aliases. See Table 6. Here is a subset:

1. brilliant smart intelligent
2. exercise exercises exercising
3. small little tiny mini micro

7. EMPIRICAL RESULTS

7.1 ROC Curve Analysis

Our algorithm is the combined classifier. We will test the combined classifier as compared to a classifier that is built strictly from orthographic attributes and a classifier that is built strictly from semantic attributes.

K-fold cross-validation was used to evaluate all three classifiers. For each "fold", we remove an entity and all related alias pairs from the training set. Then each classifier is trained on the remaining training set. Since our classifiers test whether two names are aliases, we perform a query with one name of the removed entity against all possible names in the link data set (this is the prediction set). From the classifier scores of all possible names, we can sort the scores, identify the ranks of the correct aliases (other names of the removed entity) and produce a ROC curve. For the next "fold," we perform another query on another name of the same removed entity, produce another ROC curve, and repeat until we run out of names. When that happens, we move on to the next entity.

We represented all ROC curves by first normalizing all the axes from zero to one and then averaging them together. The legends of these charts contain the area under the ROC curve (AUC).

We produce three average ROC curves. The dash-dot curve is based on the classifier that only has attributes from the probabilistic semantic model. The dotted curve is based on the orthographic model. And the solid line curve is the combined model.

Table 7: Training Set Degradation

Percentage of Original	Combine AUC	Semantic AUC	Orthographic AUC
100	0.805	0.689	0.721
75	0.803	0.705	0.721
50	0.796	0.698	0.718
25	0.777	0.682	0.709

Terrorists The combined classifier outperformed the other two. Since the links were produced manually (although subjectively), there is relatively little noise. Under this circumstance, the semantic classifier performed very well. But still the combined classifier was able to take advantage of the extra information given by the orthographic measures and outperform the semantic classifier. See Figure 1.

HsiungSpam Both spam data sets contain noise due to the nature of spam. In the presence of noise, the semantic classifier did not do as well. But the combined classifier still takes advantage of both models and produces a significantly better AUC than both. See Figure 2.

ArchiveSpam Again the combined classifier was able to obtain the highest AUC. This is almost the same as *HsiungSpam*, except the semantic classifier was able to outperform the combined classifier on the latter part of the curve. However, the combined classifier performed very well at the first part of the curve, which for many applications should be more important than the latter part. See Figure 3.

In all three combined performances, you can see the spike at the initial part of the graph. This means that in most cases, the combined algorithm is able to distinguish one alias from everyone else.

7.2 Training Set Degradation

To get an idea of how well our training set performs under a data shortage, we randomly remove training data and see how it affects the AUC for K-fold test on *HsiungSpam*. See Table 7 for results.

Surprisingly, the performance degrades smoothly when we degrade our training set. This is especially true in the orthographic case, where we do not need a large training set to train the classifier.

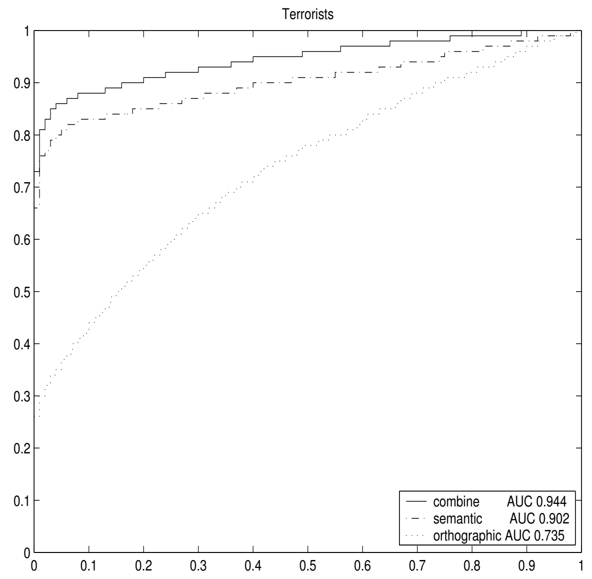

Figure 1: Average ROC for *Terrorists*

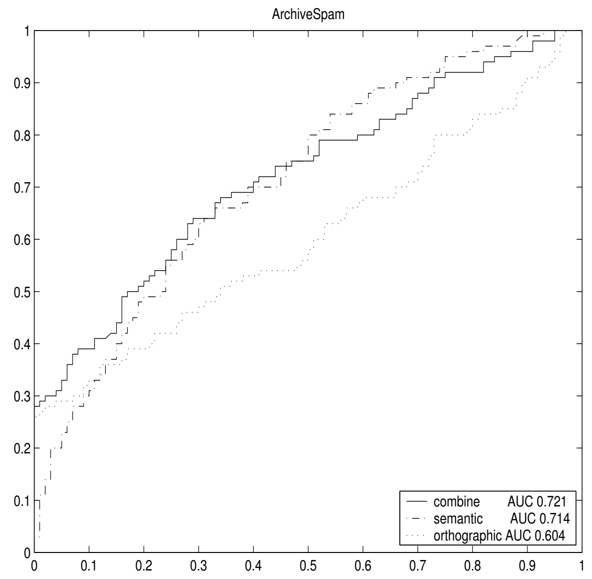

Figure 3: Average ROC for *ArchiveSpam*

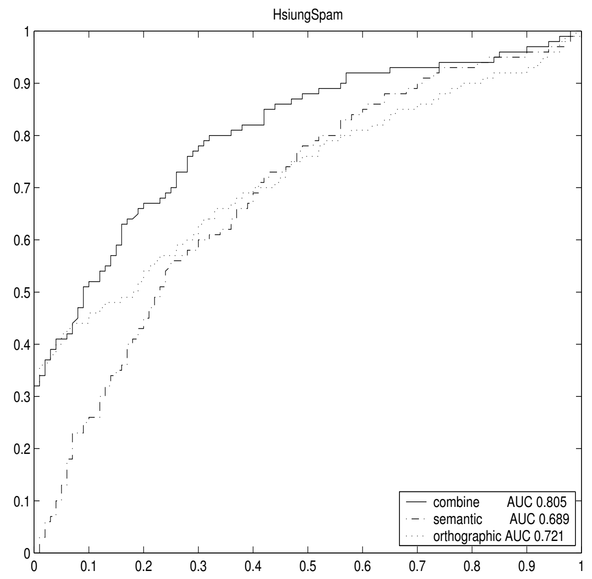

Figure 2: Average ROC for *HsiungSpam*

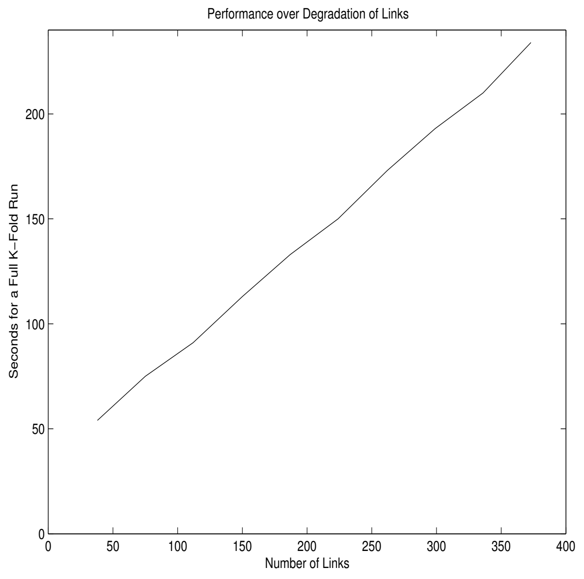

Figure 4: Scalability

Table 8: AUC Degradation with Attribute Elimination

Attribute(s) Deleted	AUC Degradation
String Edit Distance (SED)	no degradation
Normalized SED	-0.01
Exponential SED	no degradation
Discretized SED	-0.058
Normalized SED and Discretized SED	-0.083
Dot Product	-0.001
Normalized Dot Product	-0.026
Common Friends	-0.002
KL Distance	no degradation

7.3 Run time

To see how our algorithm scales, we randomly removed links. Figure 4 shows the length of each run given the reduced number of links. These runs were performed on *HsiungSpam*. As one can see, our algorithm is linear with respect to number of links.

This is consistent with our expectations for computational complexity where the dominant cost for finding the aliases of the query name E are iterating over all the other names in the link data set and computing the suite of similarity measures with the other names. Each computation of similarity measure for a pair of names takes $O(F)$ where F is the typical number of friends of a name, and where computations are arranged to take advantage of the sparseness of the link graph. Thus computing all the measures between E and all other names is $O(N\ F)$ where N is the number of names in the link data. At the end, we sort the names by alias probability so the overall cost is

$$O(N\ F + N\ log(N))$$

but the former term is dominant in practice.

7.4 Attributes Importance

To get an informal idea of how well each of our measures performs, we remove one or two attributes from the model and see what the AUC score is. See Table 8.

On the orthographic side, the most important attribute is probably *Discretized String Edit Distance* and *Normalized String Edit Distance*. On the semantic side, the most important is probably *Normalized Dot Product*.

8. RELATED WORK

Very similar to our work, Baroni et al. [1] has discussed and implemented an unsupervised algorithm that detects aliases (morphologically related words) in a text corpus using both orthographic and semantic information. On the orthographic side, they used string edit distance and on the semantic side, they used mutual information. However, to combine the two, they arbitrarily choose a function of the orthographic and semantic scores (weighting the two by hand). This is in contrast to our semi-supervised learning which involves both a much larger pool of similarity measures and which leaves the combination task to the classifier (see Section 4).

Hernandez et al. [3] addressed the problem of detecting repeated records in databases. These records might result from spelling mistakes or wrong digits for ID numbers. Their solution involved scanning through the database in a particular user-defined sorted order and detecting repeated records based upon manually predefined rules, such as:

```
Given two records, r1 and r2.
IF the last name of r1 equals the last name of r2,
       AND the first names differ slightly,
       AND the address of r1 equals the address of r2
THEN
       r1 is equivalent to r2.
```

differ slightly means they compare strings orthographically through string edit distance. Sixty lines of C code were needed to implement their rule set. The key difference between us is that our classifier automatically learns decision policy to determine whether two names are aliases. This learning can be application-dependent.

Pasula et al. [8] showed a very promising probabilistic approach to resolving multiple citations of the same paper. They built a Bayesian network to represent each citation. Their solution was a well tailored domain-specific system (as it relied heavily on relationships which were specific to publications) as opposed to the more general, self tuning, system described in this paper.

9. REFERENCES

[1] M. Baroni, J. Matiasek, and H. Trost. Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In *Proceedings of the Workshop on Morphological and Phonological Learning of ACL/SIGPHON-2002*, 2002.

[2] A. Goldenberg, J. Kubica, P. Komarek, A. Moore, and J. Schneider. A comparison of statistical and machine learning algorithms on the task of link completion. In *KDD Workshop on Link Analysis for Detecting Complex Behavior*, August 2003.

[3] M. Hernandez and S. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. In *Journal of Data Mining and Knowledge Discovery*, 1997.

[4] D. Jurafsky and J. H. Martin. *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice-Hall, 2000.

[5] J. Kubica, A. Moore, D. Cohn, and J. Schneider. cgraph: A fast graph-based method for link analysis and queries. In *Proceedings of the 2003 IJCAI Text-Mining & Link-Analysis Workshop*, August 2003.

[6] B. Marthi, B. Milch, and S. Russell. First-order probabilistic models for information extraction. In *IJCAI 2003 Workshop on Learning Statistical Models from Relational Data*, 2003.

[7] D. B. Neill. Fully Automatic Word Sense Induction by Semantic Clustering. M.Phil Thesis, Cambridge University, 2002.

[8] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *Advances in Neural Information Processing (NIPS)*, 2002.

[9] J. Zobel and P. W. Dart. Finding approximate matches in large lexicons. *Software — Practice and Experience*, 25(3):331–345, 1995.

[10] J. Zobel and P. W. Dart. Phonetic string matching: Lessons from information retrieval. In H.-P. Frei, D. Harman, P. Schäble, and R. Wilkinson, editors, *Proceedings of the 19th International Conference on Research and Development in Information Retrieval*, pages 166–172, Zurich, Switzerland, 1996. ACM Press.