Identifying Co-referential Names Across Large Corpora

Levon Lloyd, Andrew Mehler, and Steven Skiena

Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794-4400
{lloyd, mehler, skiena}@cs.sunysb.edu

Abstract. A single logical entity can be referred to by several different names over a large text corpus. We present our algorithm for finding all such co-reference sets in a large corpus. Our algorithm involves three steps: morphological similarity detection, contextual similarity analysis, and clustering. Finally, we present experimental results on over large corpus of real news text to analyze the performance our techniques.

1 Introduction

A single logical entity can be referred to by several different names over a large text corpus. For example, *George Bush* is often referred to as *Bush*, *President Bush*, *George W. Bush*, or *"W"*, even among polite company. However, morphologically-similar names like *George H.W. Bush* can refer to different entities. Accurately identifying the members of the *co-reference set* for a given entity is an important problem in text mining and natural language processing.

Our interest in identifying such co-reference sets arises in the context of our system *Lydia* [1–4], which seeks to build a relational model of people, places, and things through natural language processing of news sources. Indeed, we encourage the reader to visit our website (*http://www.textmap.com*) to study our analysis of recent news obtained from over 500 daily online news sources. In particular, we display the members of each of the 100,000 synsets we reconstruct daily (on a single commodity computer) from the roughly 150,000 entity-names we currently track.

Our algorithm for identifying co-referring name sets accurately and efficiently on a large scale involves optimizing our algorithm's three steps:

1. *Morphological Similarity* – The scale of our problem makes it infeasible to explicitly compare each pair of names for possible co-reference. First, we narrow our search space by identifying candidate pairs for analysis on a strictly syntactic basis via morphologically-sound hashing techniques.
2. *Contextual Similarity* – Next, we determine how similar a pair of names is based on the contexts in which they are used. The scale of our problem makes it infeasible to explicitly analyze all text references associated with each pair of candidate names. Instead, we propose methods using co-occurrence analysis to *other* entities to determine the probability that they are co-referent by context.

3. *Evidence Combination and Clustering* – Finally, we combine our measures of contextual and morphological similarity in order to cluster the names. The problem of clustering names is complicated by the vast difference in the number of references between popular and infrequently-used names. The strength of our contextual evidence is thus substantially weaker for unpopular names. We propose and evaluate methods for dealing with this problem.

Our problem is different from traditional cross-document co-reference analysis (see Section 2.1). In that problem, there is a set of documents that all mention the *same* name and the difficulty is clustering the documents into sets that are mentioning the same entity. In our problem, there is a set of documents that mention the many entities each possibly with multiple names and we want to cluster the names. This difference, combined with our need to manage the daily flow and scale of the news presented challenges that separate us in the following ways: (1) the use of entity co-occurrence lists as the sole feature for contextual analysis, (2) our high-speed dimension reduction techniques (based on k-means clustering and graph partitioning algorithms) to improve the quality of our contextual analysis and the efficiency of our algorithms, (3) our use of morphological similarity hashing techniques to avoid the need for pairwise-similarity testing of all name pairs, and (4) our use of *variable precision phonetic hashing* in order to tune the performance of our morphological similarity phase.

The rest of this paper is organized as follows. Section 2 surveys previous work on this and other problems. Section 3 discusses notions of morphological similarity, while Section 4 shows how we compute the probability that two names are co-referential from their respective co-occurrence lists. Section 5 discusses issues that arise in clustering. Experimental results are given in Section 6. We present our conclusions in Section 7.

2 Related Work

The problem of identifying co-reference sets has been widely studied in a variety of different contexts. In this section, we survey related work.

We now describe work on three related problems in the subsections below, namely, cross-document and in-document co-reference resolution in natural language processing and record linkage in databases.

2.1 Cross Document Co-reference Resolution

The complementary problem of cross-document co-reference has been examined fairly extensively.

Bagga and Baldwin [5] present an algorithm which extracts each sentence in each document that contains an ambiguous name and forms a summary of the document with respect to the entity. They then use the vector space model to compute the similarity of two such summaries. If the similarity of the two documents is above a threshold, then they consider the two documents to be referring the same person. They concluded that good results could be achieved by looking at the context surrounding the occurrences of the name and comparing documents using techniques from information retrieval.

Mann and Yarowsky [6] present a partially supervised algorithm for this problem. The algorithm takes as input either a small set of seed tuples for each of a small set of personal attributes from which it generates extraction patterns or a set of hand-crafted extractions for each of the personal attributes. Next, it uses these values along with other contextual clues as the feature vector for each document before using bottom-up centroid agglomerative clustering.

Gooi and Allan [7] study statistical techniques for cross-document co-reference resolution. Like Bagga and Baldwin, they use snippets of text around each mention of the ambiguous name. They compare *agglomerative clustering*, repeatedly merging the closest pair of clusters, with *incremental clustering*, either adding each point to an existing cluster or starting a new singleton cluster, and KL-divergence as a distance function with cosine similarity. They conclude that agglomerative clustering performs better than incremental clustering, however incremental clustering is much more time efficient. They also conclude that cosine similarity performs better using KL-divergence.

2.2 Within document co-reference resolution

The natural language processing community has extensively studied the problem of within document co-reference resolution, finding chains of noun phrases that refer to the same things. For example, in a news article, *Dick Cheney* may later be referred to as *Vice President*, *he*, or *Mr. Cheney*.

Ng and Cardie [8] present a supervised machine learning-based algorithm for within document co-reference resolution. They use a decision tree classifier to classify each pair of noun phrases in a document as either co-referring or not and a clustering algorithm to resolve conflicting classifications. They experiment with different feature sets, clustering algorithms, and training set selection algorithms. They conclude that linking a proper noun phrase to its most probable previously occurring co-referring phrase is a better way of clustering, that a training set selection algorithm that is designed for this clustering algorithm is superior, and while adding features can be helpful, too many can degrade performance.

Bean and Riloff [9] present an unsupervised approach to co-reference resolution that uses contextual role knowledge to determine if two noun phrases co-refer. First they identify easy-to-resolve co-referring pairs and use them as training data. Information extraction patterns are then used to generate information about the role each noun phrase plays in the text. The information extracted from the training data is used to help resolve the other pairs in the corpus. They show that this phase increases recall substantially with just a slight decrease in precision.

2.3 Record Linkage

Our co-reference set identification problem is similar to the record linkage problem from data mining. The problem arises when there is no shared, error-free key field to join on across databases. Consider two tables containing information about people from two different databases. Even if both databases used the person's name and address as the primary key, conventions concerning abbreviations and word usage may differ, and

typos and misspellings may appear in either field. The goal is to identify which records correspond to the same entities.

Hernandez and Stolfo [10] present two different techniques for large databases. The first approach sorts the data on some key and only considers two records for a merge if they are in a small neighborhood of each other. The second clusters the records in such a way that two records will be in the same cluster if they are potentially referring to the same entity. Finally, they propose taking the transitive closure of independent runs of the above algorithms, with independent key fields, as the final merge. They show that this multi-pass algorithm is superior to all the other algorithms that they consider.

Cohen and Richman [11] consider two problems: (1) taking in a pair of lists of names and determining which pairs of names in the different lists are the same and (2) taking in a single list of names and partitioning them into clusters that refer to the same entity. They propose adaptive learning-based matching and clustering methods to solve either of these problems. Their feature vector includes whether one string is a substring of the other and the edit distance between the two strings.

3 Morphological Similarity

With hundreds of thousands of names occurring in a large corpus, it is intractable to compare every pair as potentially co-referential. Further, most of these comparisons are clearly spurious, and thus would increase the possibility of false positives. We propose that most pairs of co-referential names result from the following set of morphological transformations:

- *Subsequence Similarity* – Taking a string subsequence of a name is one way of generating aliases of that name. For example, *Ford Motor Co.* is often referred to as *Ford* and *George W. Bush* is also called *George Bush*. To identify these pairs, we examine all 2^n possible string subsequences of each n-word name, hashing the name on each of its subsequences. Note that n, the number of words in a name, is bounded by about 10. Any subsequence matching another name implies potential morphological compatibility.
- *Pronunciation Similarity* – The Metaphone [12] algorithm returns a hash code of a word such that two words share the same hash code if they have similar English pronunciations. Here we say that two names are morphologically-compatible if they have the same metaphone hash code. Metaphone is useful in identifying different spellings of foreign language names (e.g. *Victor Yanukovich* and *Viktor Yanukovych*) as possibly co-referential. In Section 3.1, we detail our methods for tuning the performance of this aspect of morphological similarity using variable precision phonetic hashing
- *Stemming* – We use a Porter stemmer [13] to stem each word of each name and use the stem as a hash code for each name. A hash code collision means that two names have morphologically-compatible names. Stemming can be used to identify pairs like *New York Yankee* and *New York Yankees*.
- *Abbreviations* – If one name is an abbreviation of another, then we say that they are morphologically compatible. For example JFK and John F. Kennedy are both co-referential with John Fitzgerald Kennedy. To find all names that are abbreviations

of an name, we check if any of the 2^n possible abbreviations of the name's n-words are also in our corpus.

We observe that there is a notion of degree of morphological similarity. For example, *George Bush* is more likely to be co-referential with *George W. Bush* than *U.S.* is with *Assistant U.S. Attorney Richard Convertino*. For each of our notions of morphological similarity we have a different measure of the degree of similarity. For example, for pronunciation similarity, we model the generation of aliases as a stochastic "typing" process where the probability of a mis-type is a constant. Then we compute the probability that one name was "typed-in" when the other was intended.

3.1 Variable Precision Phonetic Hashing

Several (e.g. [12], [14], [15]) phonetic hashing schemes have been developed to work well on a specific data set or for specific performance levels. No methods exist that allow the hashing scheme to be parameterized to give different precision/recall tradeoffs. In this section we investigate phonetic hashing schemes that have an adjustable parameter giving a range of operating points with different precision/recall tradeoffs.

Given a query string, we envision a sequence of transformations from the query string to an empty or null string, where each transformation is a new version of the string that has had some tokenization or weakening applied to it. We can model the space of transformations on the universe of strings as a graph. For example the name 'Wright', is shown in Figure 1, with a possible transformation sequence.

The weight of each change is determined by how drastic it is. So the distance from 'Writ' to 'Rit' should be relatively small when compared with the distance from 'Rt' to 'R'. This tokenization path gives us different versions of the query name to use in different tolerances of the hashing function. We also see that the path for the name 'Rite' eventually joins the path of 'Wright'. The name 'Reston' similarly joins the path, but lower down; suggesting that 'Rite' and 'Wright' are closer to each other then to 'Reston'.

$$\begin{aligned} &\text{Reston} \rightarrow \text{Restn} \rightsquigarrow \text{Rst} \searrow \\ &\text{Wright} \rightarrow \text{Writ} \rightarrow \text{Rit} \rightarrow \text{Rt} \rightarrow \text{R} \rightarrow \\ &\qquad\qquad\quad \text{Rite} \nearrow \end{aligned}$$

Fig. 1. Tokenization Path of the Name 'Wright'.

A particular tokenizer in this scheme specifies a set of n-gram substitution rules, along with weights for the rules. The rules are applied in a lowest cost rule first order. An example set of rules that could have generated Figure 1 is shown below. This table says the cheapest rule is substituting a 't' for 'ght'. The next cheapest is substituting an

Code Weight	Precision	Recall
0	0.002	1
120	0.150	0.909
121	0.139	0.818
141	0.157	0.727
146	0.293	0.636
167	0.360	0.545
172	0.442	0.454
187	0.662	0.363
229	1.000	0.090
Metaphone	0.715	0.732
Soundex	0.468	0.797
NYSIIS	0.814	0.672

Table 1. Precision and Recall for our Variable Precision Phonetic Hashing and fixed precision hashing

'r' for 'wr' only if at the start of a query. Finally there are three deletion rules. The vowel deletion is considered less destructive, and is given a lower weight then the consonant deletion.

- ght $\rightarrow$ t;0.2
- _wr $\rightarrow$ r;0.3
- (a|e|i|o|u) $\rightarrow$;1
- (t|r) $\rightarrow$;5

To complete the definition of the hash function we must specify how to select the point on the tokenization path to operate at. Among the many candidates for these scoring methods, our experimentation showed that selecting the code that is a fixed distance from the null string works best.

Table 1 shows how we can vary the precision and recall of our hashing algorithm to get different tradeoffs. For a hand-created set of names extracted from our test set (see Section 6), we measured the precision and recall of our hashing algorithm at a range of its operating points. For comparison, we also show the precision and recall of three other phonetic hashing algorithms. It shows how we can use our algorithm to dial in the precision and recall of our notion of pronunciation similarity.

4 Contextual Similarity

Our mental model of where an entity fits into the world depends largely upon how it relates to other entities.

We predict that the co-occurrences associated with two co-referential names (say *Martin Luther King* and *MLK*) would be far more similar than those of morphologically-similar but not co-referential pairs (say *Martin Luther King* and *Martin Luther*). Thus

we use the vector of co-occurrence counts for each name as our feature space for contextual similarity.

We identified two primary technical issues in determining contextual similarity using this feature space: (1) dimension reduction and (2) functions for computing the similarity of two co-occurrence lists. Each of these will be described in the following subsections.

4.1 Dimension Reduction

In the experimental run of $88,097$ newspaper days of text we used throughout our experiments (details in Section 6), we encountered $174,191$ different names that occurred more than 5 times. This implies an extremely sparse, high-dimensional feature space – large because each additional entity name represents a new dimension, and sparse because a typical entity only interacts with a few hundred or so other entities even in a large text corpus.

Our experiments show that simple techniques which hunted for identical terms among the 100 or so most significant entries on each co-occurrence list failed, because the most significantly co-occurring terms for an name were highly unstable, particularly for low frequency names. Much more consistent were "themes" of co-occurring terms. In other words, while the most significant associations of *George Bush* and *"W"* might have relatively few names in common, both will be strongly associated with "Republican" and "Texas" themes.

Dimension-reduction techniques provide a way to capture such themes, and can improve both recognition accuracy and the computational efficiency of co-reference set construction. We examined two different dimension-reduction techniques based on creating crude clusters of names, then project our co-occurrence lists onto this smaller space.

- *K-means clustering* – This widely-used clustering method is simple and performs well in practice. Beginning with k randomly selected names as initial cluster centroids, we assign each name to its closest centroid (using cosine similarity of co-occurrence lists) and recompute centroids. After repeating for a given number of iterations (5, in our case) we assign each name to its closest centroid and take this as our final clustering.
- *Graph partitioning* – The problem of graph partitioning seeks to partition the vertices of a graph into a small number of large components by cutting a small number of edges. Such components in a graph of co-occurrences should correspond to "themes", subsets of terms which more strongly associate with themselves than the world at large. Thus we propose graph partitioning as a potential dimension reduction technique for such relational data – the names in each component will collapse to a single dimension.

  Although graph partitioning is NP-complete [16], reasonable heuristics exist. In particular, we used METIS[17], a well-known program for efficiently partitioning large weighted graphs into k high-weight subgraphs, with k being a user-specified parameter. Our graph contains a node for every name and an edge between every pair of nodes (x, y) if they co-occur with each other at least once. The weight

assigned to each edge is the cosine similarity between the co-occurrence lists of x and y.

4.2 Measuring Contextual Similarity

Given two names, with their co-occurrence lists projected onto our reduced dimensional space, we now want a measure of how similar they are. We consider two different approaches: (1) they can be viewed as probability distributions and be compared by *KL-divergence* or (2) they can be viewed as vectors and compared by the cosine of the angle between the vectors. We detail each of these potential measures here.

KL-Divergence The KL-Divergence is an information theoretic measure of the dissimilarity between two probability distributions. Given two distributions, the KL-Divergence of them is defined by

$$KL(p,q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}$$

To use this measure, we turn each co-occurrence list into a probability distribution for each name i,

$$\hat{p}_i(j) = \frac{\text{number of co-occurences between i and j}}{\text{total number of co-occurrences for i}}$$

As a discounting method for probability-0 pairs, we do linear smoothing of all probabilities with the background distribution setting

$$p_i(j) = \alpha \hat{p}_i(j) + (1-\alpha) bg(j)$$

where

$$bg(j) = \frac{\text{total occurrences of names in cluster j}}{\text{total number of entity occurrences in corpus}}$$

Cosine Similarity A standard way of comparing contexts views the two contexts as vectors in a high dimensional space and computes the cosine of the angle between them. [5] proposed this technique for the similar problem of personal name disambiguation. We use the *term frequency-inverse document frequency* of each vector position, we weight each term in the vector by the inverse of the number of occurrences it has in the corpus. Letting N be the number of sentences in the corpus, our score is

$$d(x,y) = \sum_{i=0}^{k} jp_x^*(i) \cdot jp_y^*(i)$$

where $jp_x(i)$ the number of co-occurrences between i and x, weighted by $\log(\frac{N}{\text{number of occurrences of i}})$, and

$$jp_x^*(i) = \frac{jp_x(i)}{\|jp_x\|}$$

5 Issues in Clustering

Now that we know which pairs of names are morphologically-similar and their degrees of morphological and contextual similarity, we need: (1) a way of combining morphological and contextual similarities into a single probability that two names are co-referential and (2) a method to cluster names into co-reference sets. We discuss each problem below.

5.1 Combining Notions of Similarity

For each pair of morphologically-related names, we have measures of their morphological and contextual similarities. We need a way to combine them into a meaningful probability that the two names are co-referential.

For each measure of contextual similarity and for edit distance, we computed the precision curve on our experimental corpus (see Section 6). Since the precision at a measure of similarity is the probability that a pair from the test set with this amount of similarity were co-referential, we use these curves to turn each of our notions of similarity into a probability. Assuming that these two probabilities are independent, we now can compute the probability that these two names are co-referential by multiplying the probabilities given by their morphological and contextual similarities.

5.2 Clustering Algorithms

Once we have probabilities associated with each pair of morphologically related names, we need to group them into co-reference sets. Because our system must be able to handle large numbers of names, we must be careful of what kind of clustering algorithm we choose. We experimented with two algorithms:

- *Single link* – Here we merge the clusters that two names are in if the probability that they are co-referential is above a threshold.
- *Average link* – Our algorithm merges two clusters if the weighted average probability between names in each of the clusters is above a threshold.

6 Experimental Results

In order to optimize various parameters, decide which methods work best, and verify our techniques, we ran a set of experiments against the same test set that was used to produce the precision curves described in section 5.1. Each of these experiments is described below.

All of the experiments in this paper where conducted on a test set of $88,097$ newspaper-days worth of text, partitioned among 604 distinct publications. These were taken from spidering that was performed between April 11, 2005 and November 5, 2005. We used a hand-crafted set of roughly 320 co-reference sets from the entities in this corpus.

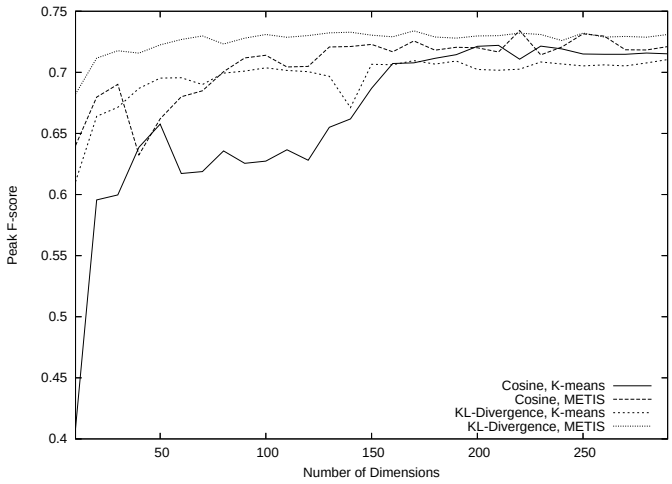


Fig. 2. Number of Clusters vs. Peak F-score for our dimension reduction algorithms and distance measures

In Section 6.1, precision is given by $\frac{tp}{tp+fp}$, recall by $\frac{tp}{tp+fn}$, and f-score by $\frac{1}{\alpha\frac{1}{P}+(1-\alpha)\frac{1}{R}}$ where $tp =$ true positives, $fp =$ false positives, and $fn =$ false negatives. In Section 6.2 these measures are given by the B-cubed algorithm introduced in [5]. For each name

$$\text{Precision} = \frac{\|\text{intersection of propsed set and true set}\|}{\|\text{proposed set}\|}$$

$$\text{Recall} = \frac{\|\text{intersection of proposed set and true set}\|}{\|\text{true set}\|}$$

and overall precision and recall are the averages of these values.

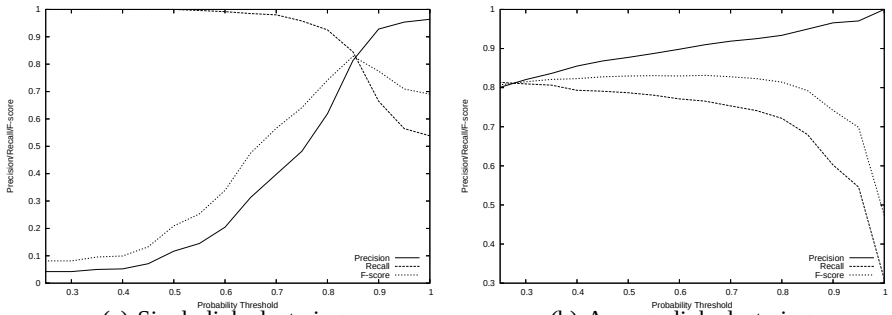
6.1 Optimizing Contextual Similarity Measure

Optimizing our contextual similarity phase involves the proper choice of (1) dimension reduction algorithm, (2) number of dimensions, and (3) contextual similarity measure. For both of the dimension reduction algorithms(k-means, METIS) and both of the distance measures(KL-Divergence, Cosine similarity), we recorded the peak F-score as a function of number of dimensions from 10 to 290.

Figures 2 shows this plot. It shows that while the peak performance of all four combinations is to be comparable, KL-Divergence with METIS dimension reduction is to be the most robust to changes in k. For the rest of the analysis in this paper, we used KL-divergence, METIS dimension reduction, and 150 dimensions.

6.2 Clustering Methods

The first clustering algorithm that we tried was simple single link clustering. Figure 3(a) shows that it has decent peak performance, but is not very robust to the setting



(a) Single-link clustering (b) Average-link clustering

Fig. 3. Threshold vs. Precision, Recall, and F-score for our clustering algorithms

of the threshold. Further, manual evaluation of the clusters that are produced shows that it tends to create very long clusters, putting many things into the same cluster that should not even be considered. For example, the sequence *George Bush* → *Bush* → *Bush-Cheney* → *Cheney* → *Dick Cheney* leads to *George Bush* and *Dick Cheney* being called co-referential.

The next clustering algorithm that we tried was weighted-average link. Figure 3(b) shows that this has slightly better peak performance than single-link clustering, but is much more robust in the setting of the threshold.

7 Conclusion

In this paper we present an algorithm to find sets of co-referential names. We introduce the idea of morphological similarity, the notion that two names are potentially co-referential based on the text that comprises the name. Then we discuss the issues surrounding computing the contextual similarity of two names and give two different measures. Clustering names given their morphological and contextual similarities was discussed and we presented experimental results for our system.

References

1. Lloyd, L., Kechagias, D., Skiena, S.: Lydia: A system for large-scale news analysis. In: String Processing and Information Retrieval (SPIRE 2005). Volume Lecture Notes in Computer Science, 3772. (2005) 161–166
2. Lloyd, L., Kaulgud, P., Skiena, S.: Newspapers vs. blogs: Who gets the scoop? In: Computational Approaches to Analyzing Weblogs (AAAI-CAAW 2006). Volume AAAI Press, Technical Report SS-06-03. (2006) 117–124
3. Kil, J., Lloyd, L., Skiena, S.: Question answering with lydia. 14th Text REtrieval Conference (TREC 2005) (2005)
4. Mehler, A., Bao, Y., Li, X., Wang, Y., Skiena, S.: Spatial analysis of news sources. submitted for publication (2006)

5. Bagga, A., Baldwin, B.: Entity-based cross-document coreferencing using the vector space model. In Boitet, C., Whitelock, P., eds.: Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics, San Francisco, California, Morgan Kaufmann Publishers (1998) 79–85
6. Mann, G., D.Yarowsky: Unsupervised personal name disambiguation. In: CoNLL, Edmonton, Alberta, Canada (2003) 33–40
7. Gooi, C., Allan, J.: Cross-document coreference on a large scale corpus. In: Human Language Technology Conf. North American Chapter Association for Computational Linguistics, Boston, Massachusetts, USA (2004) 9–16
8. Ng, V., Cardie, C.: Improving machine learning approaches to coreference resolution. In: 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, Pennsylvania, USA (2002) 104–111
9. Bean, D., Riloff, E.: Unsupervised learning of contextual role knowledge for coreference resolution. In: Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, Boston, Massachusetts, USA (2004) 297–304
10. Hernandez, M., Stolfo, S.: The merge/purge problem for large databases. In: Proceedings of the 1995 ACM SIGMOD International Conference on the Management of Data, San Jose, California, USA (1995) 127–138
11. Cohen, W., Richman, J.: Learning to match and cluster large high-dimensional data sets for data integration. In: Eighth ACM SIGKDD Conf. Knowledge Discovery and Data Mining. (2002) 475–480
12. Philips, L.: Hanging on the Metaphone. Computer Language **7**(12) (1990) 39–43
13. Porter, M.: An algorithm for suffix stripping. http://www.tartarus.org/∼martin/PorterStemmer/def.txt (1980)
14. Taft, R.: Name search techniques. New York State Identification and Intelligence Systems, Special Report No. 1, Albany, New York. (1970)
15. Borgman, C., Siegfried, S.: Getty's synoname and its cousins: A survey of applications of personal name-matching algorithms. JASIS **43**(7) (1992) 459–476
16. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the theory of NP-completeness. W. H. Freeman, San Francisco (1979)
17. Karypis, G., Kumar, V.: METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. http://www-users.cs.umn.edu/ karypis/metis (2003)