Disambiguating Personal Names on the Web using Automatically Extracted Key Phrases

Danushka Bollegala [1] and **Yutaka Matsuo** [2] and **Mitsuru Ishizuka** [3]

Abstract. When you search for information regarding a particular person on the web, a search engine returns many pages. Some of these pages may be for people with the same name. How can we disambiguate these different people with the same name? This paper presents an unsupervised algorithm which produces unique phrases to disambiguate different people with the same name (i.e. namesakes). Our algorithm takes in a personal name and outputs multiple sets of phrases which uniquely identify the different namesakes on the web. These phrases could then be added to the query to narrow down the search to a specific namesake. We evaluated the algorithm on a collection of documents retreived from the Web. Experimental results show a significant improvement over the existing methods proposed for this task.

1 Introduction

The Internet has grown into a collection of billions of web pages. One of the most important interfaces to this vast information are web search engines. We send simple text queries to search engines and retrieve web pages. However, due to the ambiguities in the queries and the documents, search engines return lots of irrelevant pages. In the case of personal names, we may receive web pages to other people with the same name (*namesakes*). However,the the different namesakes appear in quite different contexts. For example if we search for *Michael Jackson* in Google, among the top hundred hits we get a beer expert and a gun dealer along with the famous singer. Although namesakes share a common name, in most of the cases they appear in entirely different contexts. For example, in the case of Michael Jackson, terms such as *music, album, trial* associate with the famous singer, whereas we observe terms *beer, travel, hunter* for the *beer expert* namesake. This paper proposes an unsupervised algorithm which extracts such uniquely identifying key phrases from the Web to disambiguate people with the same name. These automatically extracted key phrases could then be used to modify the original query and narrow down the search.

Disambiguating namesakes on the Web is a challenging task. To begin with, the number of namesakes for a particular name is unknown. Moreover, not all namesakes are equally represented on the Web. In many cases there are two or three famous namesakes which have lots of pages about them and all other namesakes have just one or two pages on them. Another difficulty is identifying the scope of the context of a name. An entire web page/site may be written on a person (for example home pages and fan sites) or the name may appear on passing (for example book reviews on Amazon mentioning an author of a book, conference programs mentioning names of the authors of papers, etc). On the other hand there are cases where an individual has various web appearances. For example the renowned linguist Noam Chomsky appears as a linguist and also as a critic of American foreign policy. It would be interesting to see how a namesake disambiguation method responds to such complications.

Disambiguating namesakes is vital for social network extraction systems [14, 17]. Matsuo et al [14], proposes a social network extraction system in which they measure the connection between two persons *A* and *B* based on the number of hits for the query *A AND B* on a web search engine. However, this method cannot be used to create social networks for namesakes because of the ambiguity of the names. We can easily overcome this limitation by including a phrase in the query, that uniquely identifies the person under consideration from his or her namesakes.

2 Related Work

There is little previous work we know of that directly addresses the problem of extracting key phrases to disambiguate personal names on the web, but some related problems have been studied. Disambiguation of namesakes is similar to tuple matching in databases–the problem of deciding whether multiple relational tuples from heterogeneous sources refer to the same real-world entity [8, 1, 9].

From a natural language perspective, there has been a lot of work on the related problem of co-reference resolution [2, 15]. The goal in co-reference resolution is to link occurrences of noun phrases and pronouns, typically occurring in a close proximity, within a few sentences or a paragraph, based on their appearance and local context. Co-reference resolution is vital for many natural language tasks such as text summarization, question answering and entity extraction [5]. Various algorithms have been proposed for co-reference resolution. Fundamentally, these algorithms map the local information around a pronoun to a set of features and use a machine learning technique to determine whether a given pronoun corresponds to a given noun phrase.

A few works address the problem of personal name disambiguation across a collection of documents. Mann, et al [13] considers the problem of distinguishing occurrences of a personal name in different documents. They proposes an unsupervised algorithm which extracts *people-specific* biographical information such as birth date, birth place, occupation etc using a set of regular expressions to cluster the documents to their namesakes. However, such person-specific information is not always available for all the namesakes on the web. Even in cases where such information is available, a set of fixed regular expressions as used by Mann et al [13] is not sufficient to ex-

[1] University of Tokyo, danushka@mi.ci.i.u-tokyo.ac.jp
[2] Japanese National Institute of Advanced Industrial Science and Technology, y.matsuo@aist.go.jp
[3] University of Tokyo, ishizuka@i.u-tokyo.ac.jp

tract them. Bekkerman, et al [4] proposes a link structure model and an agglomerative-conglomerative double clustering (A/CDC) based algorithm to disambiguate a group of people on the web. The algorithm assumes our ability to obtain information regarding the social network (associates) of the person to be disambiguated. The method can be readily used when we have such information. However, in most of the situations we do not know well enough about the associates of the person which we want to disambiguate. Pedersen et. al. [18] proposes a method for discriminating names by clustering the instances of a given name into groups. They extract the context of each instance of the ambiguous name and generate second order context vectors using significant bigrams. The vectors are clustered such that instances that are similar to each other are placed into same clusters. Li, et al [12] suggests an algorithm which could be used to disambiguate not only personal names but other types of named entities such as organizations and locations. They propose a discriminative model based on agglomerative clustering and a generative model which uses a language model combined with EM algorithm. Their experimental results show that the generative model out performs the discriminative model. They do not discuss the topic of extracting key phrases to distinguish the different entities.

3 Method

3.1 Problem Settings and Modeling

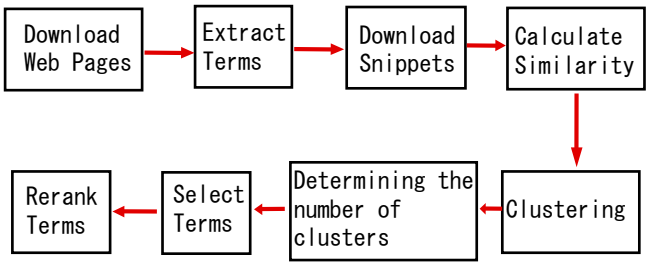


Figure 1. Outline of the method

The outline of our method is illustrated in in figure 1. Our method takes the name to be disambiguated as the input and outputs a list of key phrases for each of the different namesakes. The first step is to collect a set of documents that covers all the namesakes for a given name. This step could be omitted in cases where a particular name is to be disambiguated in a given document collection. Collecting a set of documents from the Web that covers all namesakes of a given name is beyond the scope of this paper. In our system, to collect information regarding a particular name we use Google [4] and download the top 100 web pages for the given name.

We assume each document in the collection to represent exactly one namesake. This assumption allows us to identify each document in the collection with a particular namesake. Thus, the problem of disambiguation becomes a one of document clustering. We cluster the set of documents such that each cluster represent a different namesake, and extract key phrases from each cluster to identify the namesake it represents.

[4] http://www.google.com/apis

3.2 Term Model

Contextual Hypothesis for Senses [21] states that two occurrences of an ambiguous word belong to the same sense to the extent that their contextual representations are similar. According to this hypothesis, if the contexts of which two instances of a name appears are similar, then we could infer that both the instances of the name belong to the same person. In order to achieve this goal, we need a model that represents all the salient knowledge regarding a particular instance of a name. Traditionally, a document is modeled as a bag-of-words and represented by a word vector [19]. However, in our case the context of an instance of a name may vary from few lines to an entire web page. Considering all the words (excluding stop words) adds too much noise. Moreover, a document may not totally focus on the namesake, but also contain lots of other information. The knowledge representation model should be robust enough to capture the salient information for disambiguation.

In this paper, we propose *Term Models* as our knowledge representation model. A Term Model $\mathrm{T(A)} = t_1, \ldots, t_N$, of a personal name A (since we consider each document as representing exactly one namesake, we have exactly one term model for each document) is defined as the set of N terms $t_i, \ldots, t_N$ extracted from the context of a personal name. In our algorithm we consider the context of a name in a document to be the entire document and extract terms from the entire document. We use *C-value* [6, 7], an automatic term recognition algorithm, to extract multi-word terms.

The *C-value* approach combines linguistic and statistical information. The linguistic information consists of the part-of-speech tagging of the document being processed, the linguistic filter constraining the type of terms extracted and the stop lits. The statistical part combines statistical features of the candidate string. The linguistic filter contains a predefined set of patterns of nouns, adjectives and prepositions that are likely to be terms. The stop list is a list of words which are not expected to occur as term words in a given domain. The combinations of nouns, adjectives and prepositions that are allowed by the linguistic filter and the stop list are considered as the potential candidates as terms. The *termhood* (likeliness of a candidate to be a term) is evaluated using C-value. C-value is built using statistical characteristics of the candidate string, such as, the total frequency of occurrence of the candidate string in the document, the frequency of the candidate string as part of other longer candidate strings, the number of these longer candidate terms and the length of the candidate string (in number of words). C-value is defined as follows,

$$C - value(a) = \begin{cases} \log_2 |a| \cdot f(a) & a\text{is not nested,} \\ \log_2 |a|(f(a) - \frac{1}{P(T_a)} \sum_{b \in T_a} f(b)) & \text{otherwise} \end{cases} . \quad (1)$$

where, a is the candidate string, $f(a)$ is its frequency of occurrence in the document, $|a|$ is the length of the candidate string, T_a is the set of extracted candidate terms that contain a, $P(T_a)$ is the number of candidate terms.

3.3 Similarity Calculation

In order to cluster the documents using the term model explained in previous section, we need to calculate the similarity between two documents. Exact matches of terms are rare. Therefore, we would require a similarity metric which is capable of comparing the terms at a semantic level. For example, the two terms *George Bush* and

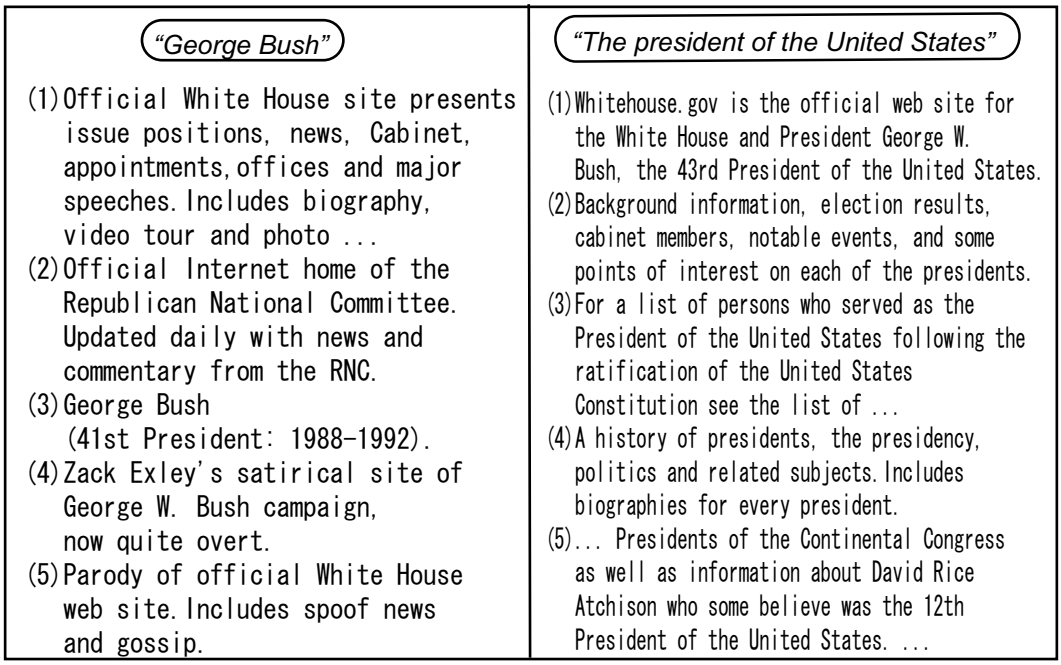


Figure 2. Top five snippets extracted for two terms

The president of the United States are closely related but do not have any words in common. Word Net [5] based similarity metrics have been widely used as semantic similarity measures between words in sense disambiguating tasks [16, 3]. However, personal names are not covered in the Word Net. Sahami et al [20] proposes a method to calculate similarity between terms using snippets retrieved by a web search engine. A Snippet is a small piece of text, containing two or three sentences extracted from the document around the query term. Most web search engines provide snippets as short summaries of the search results.

For example, consider the first five snippets returned by Google for *George Bush* and *The president of the United States* in figure 2. Even among the first five snippets for these two terms, we find many common terms such as *President, White House, Official, and, site*, etc. For a given term we collect its snippets and construct the distribution of words in the snippet. The frequency of each word in the collection of snippets is divided by the total number of words. We compute Kullback-Liebler (KL) divergence, as a measure of dissimilarity of the two terms. For two probability distributions $p(x)$ and $q(x)$, which are defined over a random variable $x \in X$, their KL-divergence $D(p||q)$ is defined as follows,

$$D(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}. \quad (2)$$

Therein, X is the vocabulary. KL-divergence becomes undefined when there are words with zero probabilities. Skew divergence is used to overcome this problem [11]. Skew divergence $S_\alpha(p, q)$ is defined as follows,

$$S_\alpha(p, q) = D(q||\alpha p + (1 - \alpha)q). \quad (3)$$

Therein: $\alpha \in [0, 1]$ is the degree of skewness between the two distributions p and q. In order to transform the asymmetrical skew divergence to symmetric similarity measure $\mathrm{sim}(p, q)$, we take the average divergence as follows,

$$\mathrm{sim}(p, q) = \exp(-\frac{1}{2}(S_\alpha(p, q) + S_\alpha(q, p))). \quad (4)$$

In our implementation, we considered the top 100 snippets from Google and set $\alpha = 0.99$.

Let $T(A) = \{a_1, \ldots, a_n\}$ be the term model of document A and $T(B) = \{b_1, \ldots, b_m\}$ the term model of document B. Here, $a_1, \ldots, a_n$ are n terms extracted from document A and $b_1, \ldots, b_m$ are m terms extracted from document B. We define the similarity, $\mathrm{DocSim}(A, B)$, between two documents A and B using their term models $T(A)$,$T(B)$ as follows,

$$\mathrm{DocSim}(A, B) = \frac{1}{mn} \sum_{a_i \in A; b_j \in B} \mathrm{sim}(a_i, b_j). \quad (5)$$

Here, $\mathrm{sim}(a_i, b_j)$ is calculated using equation 4 based on the probability distributions.

3.4 Clustering

We use group-average agglomerative clustering (GAAC) to cluster the documents to their namesakes. Initially, each document is assigned to a separate cluster. GAAC in each iteration executes the merger that gives rise to the cluster Γ with the largest average correlation $C(\Gamma)$ where,

$$C(\Gamma) = \begin{cases} 1 & |\Gamma| = 1, \\ \frac{1}{2} \frac{1}{|\Gamma|(|\Gamma|-1)} \sum_{u \in \Gamma} \sum_{v \in \Gamma} \mathrm{DocSim}(u, v) & \text{otherwise.} \end{cases} \quad (6)$$

Therein: $|\Gamma|$ denotes the number of documents in the merged cluster Γ; u and v are two documents in Γ and $\mathrm{DocSim}(u, v)$ is given by equation 5.

[5] http://wordnet.princeton.edu/perl/webwn

3.5 Cluster Quality

Ideally, clustering process should terminate when the number of formed clusters matches the number of different namesakes in the document collection. However, in real world problems the number of namesakes for a given name is not known. Clustering in general can be considered as an optimizing problem. In clustering we try to;

1. maximize the similarity of documents within a cluster,
2. minimize the similarity of documents between clusters.

We prefer our clusters to be well correlated internally and each of the clusters to be different among themselves. The quality (goodness) of the formed clusters can be evaluated based on how well the clusters satisfy these two conditions [10]. We define *internal correlation* as a measure of how well the first condition is satisfied (i.e. the degree of similarity of documents within clusters). Internal correlation, $\mathrm{IntCor}(\Lambda)$, of a set Λ of n clusters $c_1, c_2, \ldots, c_n$ is defined as follows,

$$\mathrm{IntCor}(\Lambda) = \frac{1}{n}\sum_{\Gamma\in\Lambda} C(\Gamma). \tag{7}$$

Where, $C(\Gamma)$ is the average correlation defined in equation 6. We define *external correlation* as a measure of how well the second condition is satisfied. Using the above notation, external correlation, $\mathrm{ExtCor}(\Lambda)$, is defined as the dis-similarity between the two most similar clusters in Λ as follows,

$$\mathrm{ExtCor}(\Lambda) = 1 - \frac{1}{|\Gamma_a||\Gamma_b|}\sum_{u\in\Gamma_a}\sum_{v\in\Gamma_b}\mathrm{DocSim}(u,v). \tag{8}$$

Where,

$$(\Gamma_a, \Gamma_b) = \arg_{\Gamma_i,\Gamma_j\in\Lambda}\min C(\Gamma_i \oplus \Gamma_j) \tag{9}$$

and the operator $\oplus$ denotes the merging operation between two clusters. Using equations 7 and 8 we define *Cluster Quality*, $Q(\Lambda)$, as follows,

$$Q(\Lambda) = \frac{1}{2}(\mathrm{IntCor}(\Lambda) + \mathrm{ExtCor}(\Lambda)). \tag{10}$$

To label the clusters we select all the terms that appear in a cluster for a certain namesake but do not appear in other clusters.

4 Results and Discussion

4.1 Test Data

We evaluated our algorithm on pseudo names as well as naturally ambiguous names. For automated pseudo name evaluation purposes, we collected 150 (50 per person) documents from the web for three different people for conflation. Our collection contains documents for *Maria Sharapova* the tennis player, *Bill Gates* chairman Microsoft and *Bill Clinton* former president of the United Sates. We then replace every occurrence of these names in the documents with *person-x*. We evaluate the algorithm on naturally ambiguous names such as *Jim Clark*, *William Cohen*, *Tom Mitchell* and *Michael Jackson* [6]. To evaluate our algorithm on people with different web appearances we tested for *Noam Chomsky*.

[6] This collection contains 50 documents per ambiguous name

4.2 Disambiguation Accuracy

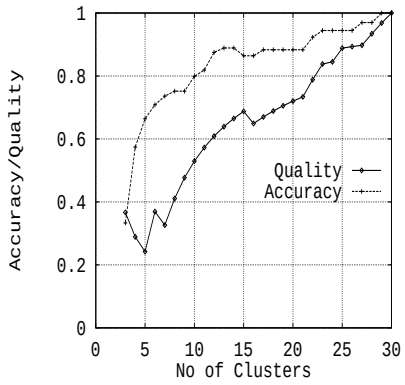
We assign each cluster to the namesake that appears most in that cluster (*holder* of the cluster). Precision, $P(C)$, of cluster C is calculated as follows,

$$P(C) = \frac{\text{No of docs in C for its holder}}{\text{Total No of docs in C}}. \tag{11}$$

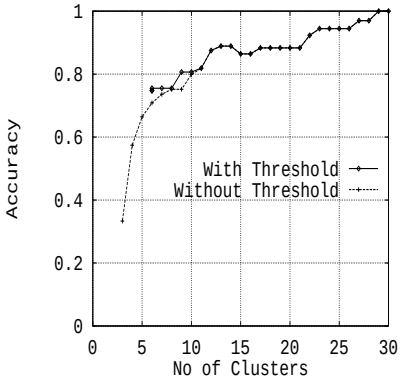
However, some namesakes have lots of documents on them, where as others are mentioned only in few documents. To reflect this fact in our evaluation metric we define *Disambiguation Accuracy*, as the weighted sum of each cluster's precision. Disambiguation accuracy (Accuracy) is defined as follows,

$$\text{Accuracy} = \sum_{C\in\Lambda} P(C)\frac{\text{No of docs in the collection for the holder of C}}{\text{Total No of docs in the collection}}. \tag{12}$$

Where, Λ is the set of clusters and $P(C)$ is given by equation 11.



(a) Accuracy/Quality Vs No of Clusters without Quality Threshold



(b) Accuracy Vs No of Clusters, with and without Quality Threshold

Figure 3. Effect of the quality threshold

Figure 3(a) depicts the accuracy/quality vs the number of clusters in the experiment with pseudo names. It shows that accuracy approximately correlates with cluster quality. This relationship enables us to

guide the clustering process based on unsupervisedly calculated cluster quality. Moreover, figure 3(a) shows that accuracy drops steadily as the number of clusters decreases. This is due to the outliers that get attached to the otherwise pure (representing the dominant namesakes) clusters. To avoid this, we terminate clustering when cluster quality drops below a fixed threshold and classify the remaining documents to the clusters. As seen from figure 3(b), this procedure prevents ill-formed clusters and yields high accuracy values. We experimentally set the cluster quality threshold to 0.6.

Table 1. Accuracy for ambiguous names

Name	Number of namesakes	Proposed	TF IDF
Jim Clark	8	71.95	59.20
Michael Jackson	3	94.96	88.76
William Cohen	10	72.71	57.96
person-X	3	81.10	39.88
Noam Chomsky	2	94.19	82.79

Table 1 shows results of our experiments. We implemented a TF-IDF based clustering algorithm as the baseline for comparison. In the baseline method, each document in the collection is represented by a word vector. We consider all the words (except a fixed list of stop words) in the document and calculate their term frequencies (TF: Term Frequency) in the document. For each word we find the number of documents containing it (DF: document frequency). Using TF,DF values we represent each document as a vector of words, with TF-IDF weights. Similarity between two documents is calculated as the cosine similarity of their word vectors. We then use GAAC to cluster the documents. Base line method utilizes the same cluster quality threshold as the proposed method. However, note that the TF-IDF based method does not produce any key phrases. Table 1 reports higher accuracy values for the proposed method over the baseline. There is a clear advantage to the proposed method over the baseline for person-X collection. Considering the fact that person-X collection contains three very different personalities, their term models are sufficiently discriminative to clearly separate the three personalities. However, for highly ambiguous names table 1 reports comparatively low accuracy values. One reason for this behavior is that, the distribution of documents among namesakes is not being even. Some of the namesakes are not sufficiently represented on the web to build a descriptive term model.

Our algorithm finds key phrases such as *racing driver Jim Clark, Formula One World Championships and motor racing* for the racing car driver-Jim Clark and *Silicon Valley, netscape* for the founder of netscape -Jim Clark. In the case of Michael Jackson, the top ranking terms for the singer are *Fan Club, World network, news, Micheal Jackson case* and *pop star*. The proposed method had the lowest accuracy for william cohen as it found only three out of the ten namesakes in the collection. In the person-X experiments, we find key phrases such as *first set, US open, Wimbledon, Venus Williams* and *Grand Slam title* for Maria Sharapova, *wealthiest person, Microsoft* for Bill Gates and *White house, former president* for Bill Clinton. Although, Noam Chomsky is not an ambiguous name, we tested on it to evaluate the algorithm on individuals with different web appearances. Interestingly, the algorithm produces key phrases such as *preventive war, government complicity, George Bush, Tony Blair* in the Chomskey the critic cluster and *universal grammar, linguistic theory* in the Chomskey the linguist cluster.

5 Conclusion

We proposed and evaluated an algorithm to extract key phrases from the web, to disambiguate personal names. The algorithm is unsupervised and uses a cluster quality metric to determine the number of namesakes. Our experiments show encouraging results. In future, we intend to explore the possibilities to extend the proposed method to disambiguate other types of named entities.

REFERENCES

[1] P. Andritsos, R.J. Miller, and P. Tsapars, 'Information-theoretic tools for mining database structure from large data sets', in *Proceedings of the ACM SIGMOD Conference*, (2004).

[2] A. Bagga and B. Baldwin, 'Entity-based cross-document coreferencing using the vector space model', in *Proceedings of COLING*, pp. 79–85, (1998).

[3] Satanjeev Banerjee and Ted Pedersen, 'An adapted lesk algorithm for word sense disambiguation using word net', in *Proceedings of the third international conference on computational linguistics and intelligent text processing*, pp. 136–145, (2002).

[4] Ron Bekkerman and Andrew McCallum, 'Disambiguating web appearances of people in a social network', in *Proceedings of the 14th international conference on World Wide Web*, pp. 463–470, (2005).

[5] Matthias Blume, 'Automatic entity disambiguation: Benifits to ner, relation extraction, link analysis, and inference', in *Proceedings of International Conference on Intelligence Analysis*, (2005).

[6] K.T. Frantzi and S. Ananiadou, 'Extracting nested collocations', in *16th Conference on Computational Lingustics*, pp. 41–46, (1996).

[7] K.T. Frantzi and S. Ananiadou, 'The c-value/nc-value domain independent method for multi-word term extraction', *Journal of Natural Language Processing*, **6(3)**, 145–179, (1999).

[8] M. Hernandez and S. Stolfo, 'The merge/purge problem for large databases', in *SIGMOD Conference*, pp. 127–138, (1995).

[9] Dmitri V. Kalashnikov, Sharad Mehrotra, and Zhaoqi Chen, 'Exploiting relationships for domain-independent data cleaning', in *SIAM International Conference on Data Mining (SIAM SDM)*, Newport Beach, CA, USA, (April 21–23 2005).

[10] Ravi Kannan, Santosh Vempala, and Adrian Vetta, 'On clustering: Good, bad and spectral', *Computer Science*, (2000).

[11] Lillian Lee, 'On the effectiveness of the skew divergence for statistical language analysis', *Artificial Intelligence and Statistics*, 65–5, (2001).

[12] Xin Li, Paul Morie, and Dan Roth, 'Semantic integration in text, from ambiguous names to identifiable entities', *AI Magazine, American Association for Artificial Intelligence*, **Spring**, 45–58, (2005).

[13] Gideon S. Mann and David Yarowsky, 'Unsupervised personal name disambiguation', in *Proceedings of CoNLL-2003*, pp. 33–40, (2003).

[14] Yutaka Matsuo, Junichiro Mori, and Masahiro Hamasaki, 'Polyphonet: An advanved social network extraction system from the web', in *Proceedings of the World Wide Web Conference*, (to appear in 2006).

[15] A. McCallum and B. Wellner, 'Toward conditional models of identity uncertainty with application to proper noun coreference', in *IJCAI Workshop on Information Integration on the Web, 2003*, (2003).

[16] D. McCarthy, R. Koeling, J. Weeds, and J. Carroll, 'Finding predominant word senses in untagged text', in *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04)*, pp. 279–286, (2004).

[17] P. Mika, 'Bootstrapping the foaf-web: and experiment in social networking network minning', in *Proceedings of 1st Workshop on Friend of a Friend, Social Networking and the Semantic Web*, (2004).

[18] Ted Pedersen, Amruta Purandare, and Anagha Kulkarni, 'Name discrimination by clustering similar contexts', in *Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics*, (2005).

[19] V.V. Raghavan and S.K.M. Wong, 'A critical analysis of vector space model for information retreival', *Journal of the American Society for Information Science*, **37**(5), 279–287, (1986).

[20] Mehran Sahami and Tim Heilman, 'A web-based kernel function for matching short text snippets', in *International Workshop located at the 22nd International Conference on Machine Learning (ICML 2005)*, (2005).

[21] Hinrich Schutze, 'Automatic word sense discrimination', *Computational Linguistics*, **24**(1), 97–123, (1998).