Extracting Key Phrases to Disambiguate Personal Names on the Web

Danushka Bollegala[1], Yutaka Matsuo[2], and Mitsuru Ishizuka[1]

[1] University of Tokyo
{danushka,ishizuka}@miv.t.u-tokyo.ac.jp
[2] AIST
y.matsuo@carc.aist.go.jp

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 key phrases for the different people with the same name. These key phrases could be used to further narrow down the search, leading to more person specific unambiguous information. The algorithm we propose does not require any biographical or social information regarding the person. Although there are some previous work in personal name disambiguation on the web, to our knowledge, this is the first attempt to extract key phrases to disambiguate the different persons with the same name. To evaluate our algorithm, we collected and hand labeled a dataset of over 1000 Web pages retrieved from Google using personal name queries. Our experimental results shows an improvement over the existing methods for namesake disambiguation.

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. However, the context in which the singer appears is quite different from his namesakes. However, context associated with a personal name is difficult to identify. In cases where the entire web page is about the person under consideration, the context could be the complete page. On the other hand the context could be few sentences having the specified name. In this paper we explore a method which uses terms extracted from web pages to represent the context of namesakes. For example, in the case of Michael Jackson, terms such as *music, album, trial* associate with the famous singer, whereas we get *beer, travel, hunter* as terms for the other (beer expert) namesake of Michael Jackson. These term sets appear to be defining different contexts. We could use this difference in context to discriminate the namesakes.

Disambiguating namesakes on the Web is a difficult task due to the diversity of web pages. We do not know in advance the exact number of namesakes for a name on the Web. In many cases there are two or three famous namesakes which have lots of pages regarding them and all other namesakes have just one or two pages on them. Some of the web pages are not exclusively about a person, but just mention the name on passing (ex: book reviews on Amazon mentioning an author of a book, conference programs mentioning names of the authors of papers, etc). This paper presents an unsupervised clustering framework, which uses a robust similarity metric to overcome these difficulties.

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 disambiguiation method responds to such complications. In Chomsky's example one would like to extract terms such as *Generative Grammar, Linguistic Theory, Transformational Grammar, etc* from pages which describe Chomsky's linguistic work whereas *American foreign policy, Iraq, critic, etc* from pages which describe Chomsky's political views. Although our main focus is on disambiguating people with one specific web appearance, we also explore the posibilities of our algorithm to identify the different web apperances of individuals.

This paper is structured as follows. First we give an overview of the related work in this area. Then we explain the different components in our system. Namely; term extraction, similarity calculation, clustering, determining the number of namesakes and term ranking. Finally we show experimental results for the proposed method and conclude this paper.

2 Related Work and Problem Setting

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 [7, 1].

From a natural language perspective, there has been a lot of work on the related problem of co-reference resolution [2, 11]. 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 and question answering. 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 [10] 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 [10] is not sufficient to extract 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. [13] 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 [9] suggests an algorithm which could be used to disambiguate not only personal names but other 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. However, they do not discuss the topic of extracting key phrases to distinguish the different entities. In this paper we try to extract key phrases to distinguish each of the different namesakes in our document collection. In this paper we will assume that each document in the collection represents only one of the namesakes (i.e. no document covers two or more namesakes). We will first cluster the set of documents and then select key phrases from these clusters to distinguish the different namesakes.

3 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. As shown in figure 1, the algorithm we propose for this task consists of eight steps. We first introduce each of the steps in our method and details are left for the sections to follow.

First we send the name to be disambiguated to a web search engine and download a set of web pages. We used Google [3] and download the top 100

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

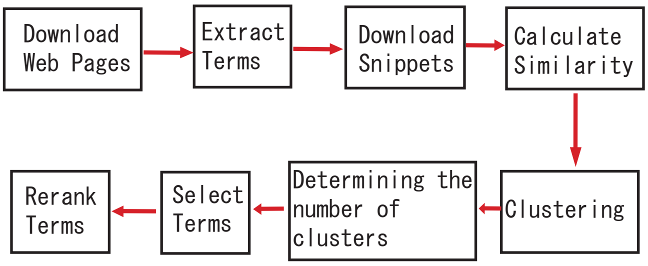


Fig. 1. Outline of the method

pages for the given name. These pages will be processed in the next steps in our method. Downloaded pages are not required to be exclusively on a certain person. However, we assume one page to be associated with only one of the namesakes. We extract a set of terms from each one of the pages in our document collection (which was downloaded in the previous step). The term extraction algorithm we use for this task is explained in section 3.1. We then cluster the document collection based on the terms we extracted. To cluster documents we define a pairwise similarity measure. We use *Snippet Similarity* to measure the similarity between two terms. Section 3.2 explains snippet similarity. We utilize an agglomerative clustering method to cluster the document collection as described in section 3.3. Ideally, the clusters yielded by the clustering algorithm should represent a different namesake. However, in reality we do not know the exact number of namesakes for a given name in advance. Therefore, we define a measure which we will call *Cluster Quality* in this paper based on the internal and external correlation of the clusters, and decide the number of clusters. Finally, we select representative terms from each of the clusters and rank them according to their relevance to the name under consideration.

3.1 Term Extraction

Our method is based on the fact that different namesakes appear under different contexts on the web. We assume that each document in our downloaded web page collection represents some namesake of the given name. **Contextual Hypothesis for Senses** [15] 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 two pages are similar in context, then we could assume that these pages are likely to be on the same namesake. However, a document may not totally focus on the namesake, but also contain lots of irrelevant information. Therefore, we need to represent the documents in a model that captures the essence of the document and ignores the irrelevant facts. We use *C-value* [5, 6], an automatic term recognition algorithm, to extract multi-word terms from the documents and represent each document by the set of terms extracted from it.

The *C-value* approach combines linguistic and statistical information, emphasis being placed on the statistical part. 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. Having a stop list improves the precision. However, in our experiments we did not use a stop list because it is not possible to determine in advance the domains which a namesakes belongs to.

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,

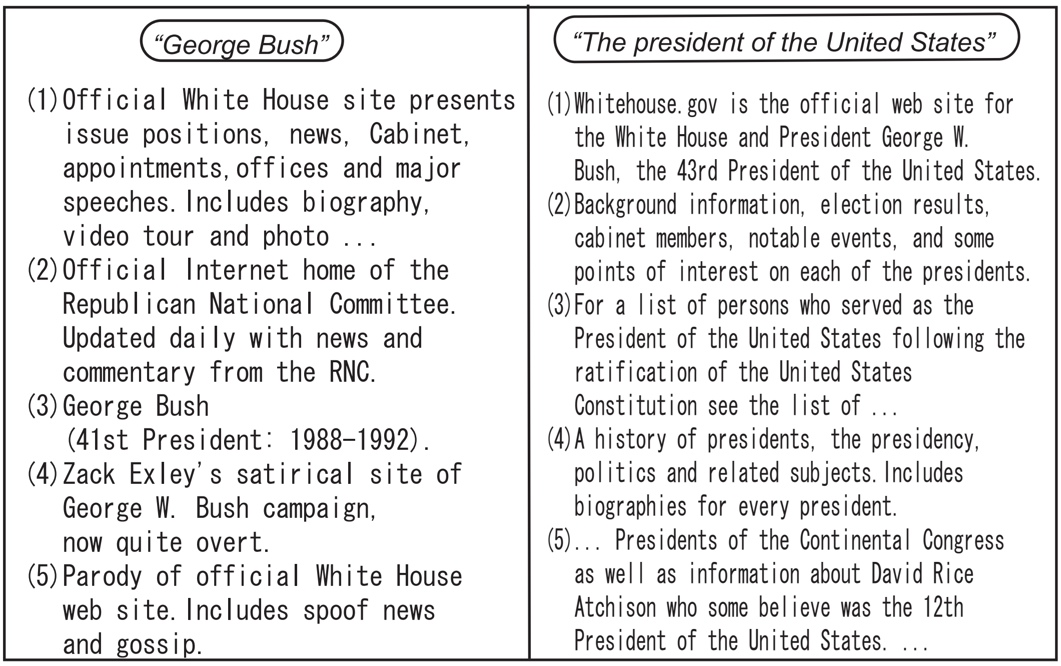
$$C - value(a) = \begin{cases} \log_2 |a| \cdot f(a) & a \text{ is not nested,} \\ \log_2 |a|(f(a) - \frac{1}{P(T_a)} \sum_{b \in T_a} f(b)) & \text{otherwise} \end{cases} . \quad (1)$$

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 these candidate terms.

We prefer the candidate terms with higher c-values to terms with lower c-values. However, there are cases where the terms extracted from Frantzi's [6] c-value method tend to be exceedingly longer and meaningless. For example, we get a term *Search Archives Contact Us Table Talk Ad* from a page about the netscape founder, Jim Clark. This term is a combination of words extracted from a navigation menu and not a genuine term. Using such terms to represent the context of a namesake cannot be acceptable. To avoid such terms we use two heuristics. First we ignore any term which is longer than four words. Then, for the remaining terms, we check the number of hits we get for the term in a web search engine. Our assumption here is if a term is a meaningful one it is likely to be used in many web pages. We ignore any terms with less than five hits. Using these heuristics does not only allow us to extract more expressive and genuine terms but also prevents data sparseness when calculating snippet based similarity between terms as explained in the next section.

3.2 Similarity Calculation

Exact matches of terms extracted from different documents 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 *The president*



"George Bush"

(1) Official White House site presents issue positions, news, Cabinet, appointments, offices and major speeches. Includes biography, video tour and photo ...
(2) Official Internet home of the Republican National Committee. Updated daily with news and commentary from the RNC.
(3) George Bush (41st President: 1988-1992).
(4) Zack Exley's satirical site of George W. Bush campaign, now quite overt.
(5) Parody of official White House web site. Includes spoof news and gossip.

"The president of the United States"

(1) Whitehouse.gov is the official web site for the White House and President George W. Bush, the 43rd President of the United States.
(2) Background information, election results, cabinet members, notable events, and some points of interest on each of the presidents.
(3) For a list of persons who served as the President of the United States following the ratification of the United States Constitution see the list of ...
(4) A history of presidents, the presidency, politics and related subjects. Includes biographies for every president.
(5) ... Presidents of the Continental Congress as well as information about David Rice Atchison who some believe was the 12th President of the United States. ...

Fig. 2. Top five snippets extracted for two terms

of the United States are closely related but do not have any words in common. Word Net [4] based similarity metrics have been widely used as semantic similarity measures between words in sense disambiguating tasks [12, 3]. However, the major problem with such approaches is the low coverage of words. For example, we would not find proper nouns such as *George Bush* in WordNet. However, such proper nouns (specially human names) are useful to disambiguate namesakes (see section 4). We use the World Wide Web as our knowledge source and define similarity between terms using web snippets. Mehra [14] proposes a method to calculate similarity between words (also can be used with 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 along with the links to the source pages. A user can read the snippet and decide whether the linked page is relevant to the query, thereby avoiding the time to download and read the complete page. The snippet gives the context in which the searched term appear in the page. We use Google as our web search engine and extract the top 100 snippets for each term we extract.

For example, consider the first five snippets we get for *George Bush* and *The president of the United States* shown in figure 2. Even among the first five snippets for these two terms, we find many common words such as *President, White*

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

House, Official, and, site, etc. However, some of these words (ex: and, of) have a purely grammatical functionality and do not carry any semantic information regarding the searched terms. We use a predefined list of stop words and remove such words from the snippets. We then merge the top hundred snippets together (here on, we will call this merged result as the snippet text) and compare the distribution of words to calculate the similarity between the terms. In order to calculate the word distribution we count the frequency of each word in the snippet text. We divide the frequency counts by the total number of words to convert the frequency distribution into a probability distribution. These normalized word distributions, calculated using snippets retrieved for different terms, are compared using Kullback-Liebler (KL) divergence. KL- divergence is a popular metric used in measuring the distance between two probability distributions. 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,

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

Where, X is the set of values that random variable x takes. Since we are concerned on word distributions, X is the vocabulary of words used in the snippets. However, due to the sparseness of data, some words may not appear in both distributions. KL-divergence becomes undefined when there are words with zero probabilities. Skew divergence is used to overcome this problem [8]. Skew divergence $S_\alpha(p, q)$ is defined as follows,

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

Therein: $\alpha \in [0, 1]$ is the degree of skewness between the two distributions p and q. It has been shown that([8]) skew divergence best expresses the divergence between two distributions when the value of α is closer to 1. In our experiments we set $\alpha = 0.99$.

However, both KL-divergence and skew divergence are not symmetric and does not satisfy the properties of distance metrics. We define a distance function $d(p, q)$ by considering the skew divergence on both ways. Thus, the distance $d(p, q)$ between two distributions p, q is defined as follows,

$$d(p, q) = \frac{1}{2}(s_\alpha(p, q) + s_\alpha(q, p)). \tag{4}$$

We further convert the distance values given by equation 4 to similarity values $\mathrm{sim}(p, q)$ by taking their negative exponential values as follows,

$$\mathrm{sim}(p, q) = \exp(-d(p, q)). \tag{5}$$

Equation 5 defines the similarity between two terms using the probability distributions calculated for each of the terms.

However, to cluster the documents, we need a pairwise similarity measure which is defined upon the documents. For this we extend the similarity function defined by equation 5 to two documents. We take the average of similarity for all the pairs of terms extracted by the two documents. The similarity, $\mathrm{DocSim}(A, B)$, between two documents A and B is defined as follows,

$$\mathrm{DocSim}(A, B) = \frac{1}{|A||B|} \sum_{(a,b) \in (A \times B)} \mathrm{sim}(a, b). \tag{6}$$

Therein: A and B are the sets of terms extracted from the corresponding documents. $|A|$ denotes the cardinality of the set A. $A \times B$ gives the set of pairs taken from the two sets. $a \in A$ and $b \in B$ are terms in the sets.

3.3 Clustering

Having defined a similarity metric in section 3.2, our next task is to cluster the documents using this similarity metric. In this paper, we use group-average agglomerative clustering (GAAC) as our clustering algorithm, a hybrid of single-link and complete-link clustering. We begin by assigning a separate cluster for each document in the collection. GAAC in each iteration executes the merger that gives rise to the cluster $\mathit{\Gamma}$ with the largest average correlation $C(\mathit{\Gamma})$ where,

$$C(\mathit{\Gamma}) = \begin{cases} 1 & |\mathit{\Gamma}| = 1, \\ \frac{1}{2}\frac{1}{|\mathit{\Gamma}|(|\mathit{\Gamma}|-1)} \sum_{u \in \mathit{\Gamma}} \sum_{v \in \mathit{\Gamma}} \mathrm{DocSim}(u, v) & \text{otherwise.} \end{cases} \tag{7}$$

Therein: $|\mathit{\Gamma}|$ denotes the number of documents in the merged cluster $\mathit{\Gamma}$; u and v are two documents in $\mathit{\Gamma}$ and $\mathrm{DocSim}(u, v)$ is given by equation 6. Ideally, the clustering process should terminate when there is exactly one cluster representing each of the namesakes in the collection. However, in practice, the exact number of different namesakes that exist in the collection is not known. Therefore, we define a measure which we will call *Cluster Quality* in section 3.4 and terminate the above mentioned GAAC process when the cluster quality falls below a predefined threshold. In section 4 we show empirical evidence to the fact that the cluster quality measure approximates well the disambiguation accuracy and stops the clustering when there are sufficient clusters to represent the different namesakes.

3.4 Cluster Quality

Clustering in general can be considered as an optimizing problem. In clustering we try to;

1. maximize the similarity of items (documents) within a cluster,
2. minimize the similarity of items (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. 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,

$$\mathrm{IntCor}(\Lambda) = \frac{1}{n} \sum_{\Gamma \in \Lambda} C(\Gamma). \tag{8}$$

Where, $C(\Gamma)$ is the average correlation defined in equation 7.

We define *external correlation* as a measure of how well the second condition is satisfied (i.e. the degree of dis-similarity between clusters). 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,

$$\mathrm{ExtCor}(\Lambda) = 1 - \frac{1}{|\Gamma_a||\Gamma_b|} \sum_{u \in \Gamma_a} \sum_{v \in \Gamma_2} \mathrm{DocSim}(u, v). \tag{9}$$

Where,

$$(\Gamma_a, \Gamma_b) = \arg_{\Gamma_i, \Gamma_j \in \Lambda} \min C(\Gamma_i \oplus \Gamma_j) \tag{10}$$

and the operator $\oplus$ denotes the merging operation between two clusters. Using equations 8 and 9 we define *Cluster Quality*, $Q(\Lambda)$, as follows,

$$Q(\Lambda) = \frac{1}{2}(\mathrm{IntCor}(\Lambda) + \mathrm{ExtCor}(\Lambda)). \tag{11}$$

We terminate GAAC when cluster quality drops below a predefined threshold and assign the remaining documents to the already formed clusters.

3.5 Term Selection and Ranking

GAAC (section 3.3) creates clusters for different namesakes. The next step is to select a set of terms from these clusters that describes each namesake. We select all the terms that appear in a cluster for a certain namesake but does not appear in other clusters. We then rank these terms by the relevancy of the term to the ambiguous name. We use snippets based similarity measure described in section 3.2 to calculate relevancy.

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 50 documents from the web for three different people for conflation. Our collection contains documents for *Maria Sharapova* the tennis player, *Bill Gates* chairman and chief software architect of Microsoft corporation and *Bill Clinton* former president of the United Sates. We then replace every occurrence of these names in the documents with *person-x*.

To evaluate our algorithm on naturally ambiguous names we selected names that appear in previous work([10, 4]) in this field, such as *Jim Clark, William Cohen, Tom Mitchell, Michael Jackson.* To evaluate our algorithm on people with different web appearances we tested for *Noam Chomsky.*

4.2 Disambiguation Accuracy

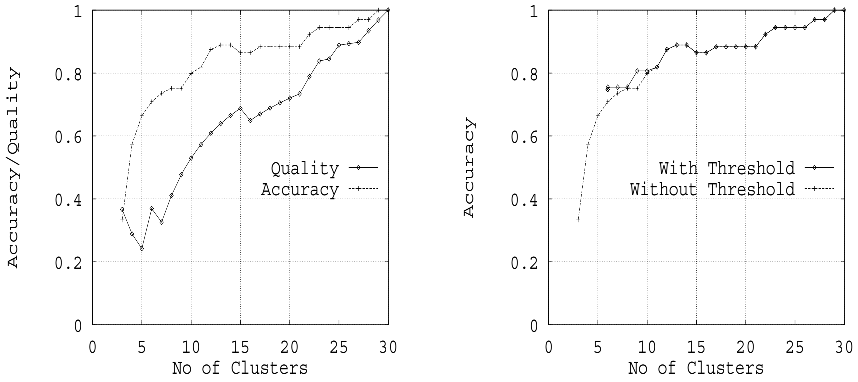
To evaluate the clusters produced by the proposed algorithm, we first assign each cluster to the namesake that appears most (we will call this namesake the *holder* of the cluster) in that cluster. We then count the number of documents in the collection for each of the different namesakes. Precision, $P(C)$, of cluster C is calculated as follows;

$$P(C) = \frac{\text{Number of documents in C, representing the holder of C}}{\text{Total number of documents in C}}. \tag{12}$$

However, the distribution of documents for each of the different namesakes in the collection is not even. Some namesakes have lots of pages representing them, where as for some namesakes they are mentioned only in one or two documents. To reflect this fact in our evaluation metric we define *Disambiguation Accuracy*, as the weighted sum of each cluster's precision. Accuracy (disambiguation accuracy) is defined as follows,

$$\text{Accuracy} = \sum_{C \in \Lambda} P(C) \frac{\text{Number of documents in the collection for the holder of C}}{\text{Total number of documents in the collection}}. \tag{13}$$

Where, Λ is the set of clusters and $P(C)$ is given by equation 12.



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

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

Fig. 3. Effect of the quality threshold

Figure 3(a) depicts the accuracy/quality vs the number of clusters in the experiment with pseudo names. From figure 3(a) it can be seen that when we do not impose a threshold on quality, there exists a steep drop in accuracy when ten clusters are formed. This is due to the outliers that get attached to the otherwise pure (representing the dominant namesakes) clusters. The value of quality when this happens is 0.6. Although the number of clusters when this happens is different for different names, our experiments show that the value of quality is around 0.6 in all the experiments. Therefore, we set the threshold of quality to 0.6. When the threshfold is imposed, accuracy does not drop as it can be seen from figure 3(b).

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 Chomskey	2	94.19	82.79

Table 1 shows results of our experiments. We implemented a TF-IDF based similarity metric and compared our algorithm against it. In the TF-IDF based method, we consider all the words in each document (except stop words) and represent documents with TF-IDF weighted vectors. Then we take the cosine similarity between the vectors as the similarity measure in the group average agglomerative clustering in section 3.3. However, note that the TF-IDF based method does not produce any key phrases. Table 1 reports higher accuracy values for the proposed method compared to this TF-IDF based method.

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 Gates. Although, Noam Chomskey is not an ambiguous name, we tested on it to evaluate the algorithm on individuals with different web appearances. Interestingly, the algorithm ranks 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. In future, we intend to explore the possibilities to extend the proposed method to disambiguate other types of entities such as location names and organization names.

References

1. P. Andritsos, R. 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*, pages 79–85, 1998.
3. S. Banerjee and T. 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*, pages 136–145, 2002.
4. R. Bekkerman and A. McCallum. Disambiguating web appearances of people in a social network. In *Proceedings of the 14th international conference on World Wide Web*, pages 463–470, 2005.
5. K. Frantzi and S. Ananiadou. Extracting nested collocations. In *16th Conference on Computational Lingustics*, pages 41–46, 1996.
6. K. 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.
7. M. Hernandez and S. Stolfo. The merge/purge problem for large databases. In *SIGMOD Conference*, pages 127–138, 1995.
8. L. Lee. On the effectiveness of the skew divergence for statistical language analysis. *Artificial Intelligence and Statistics*, pages 65–5, 2001.
9. X. Li, P. Morie, and D. Roth. Semantic integration in text, from ambiguous names to identifiable entities. *AI Magazine, American Association for Artificial Intelligence*, Spring:45–58, 2005.
10. G. S. Mann and D. Yarowsky. Unsupervised personal name disambiguation. In *Proceedings of CoNLL-2003*, pages 33–40, 2003.
11. 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.
12. 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)*, pages 279–286, 2004.
13. T. Pedersen, A. Purandare, and A. Kulkarni. Name discrimination by clustering similar contexts. In *Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics*, 2005.
14. M. Sahami and T. 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.
15. H. Schutze. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123, 1998.