Automatic Annotation of Ambiguous Personal Names on the Web

DANUSHKA BOLLEGALA[*]

The University of Tokyo

Hongo 7-3-1, Tokyo, 113-8656, Japan

danushka@iba.t.u-tokyo.ac.jp

YUTAKA MATSUO

The University of Tokyo

Hongo 7-3-1, Tokyo, 113-8656, Japan

matsuo@biz-model.t.u-tokyo.ac.jp

MITSURU ISHIZUKA

The University of Tokyo

Hongo 7-3-1, Tokyo, 113-8656, Japan

ishizuka@i.u-tokyo.ac.jp

[*]Tel:+81-3-5841-6347, Fax:+81-3-5841-8570

Abstract

Personal name disambiguation is an important task in social network extraction, evaluation and integration of ontologies, information retrieval, cross-document co-reference resolution and word sense disambiguation. We propose an unsupervised method to automatically annotate people with ambiguous names on the web using automatically extracted keywords. Given an ambiguous personal name, first, we download text snippets for the given name from a Web search engine. We then represent each instance of the ambiguous name by a *term-entity model* (TEM), a model that we propose to represent the web appearance of an individual. A TEM of a person captures named entities and attribute values that are useful to disambiguate that person from his or her namesakes (i.e. different people who share the same name). We then use group average agglomerative clustering to identify the instances of an ambiguous name that belong to the same person. Ideally, each cluster must represent a different namesake. However, in practice it is not possible to know the number of namesakes for a given ambiguous personal name in advance. To circumvent this problem, we propose a novel normalized cuts-based cluster stopping criterion to determine the different people on the web for a given ambiguous name. Finally, we annotate each person with an ambiguous name using keywords selected from the clusters. We evaluate the proposed method on a dataset of over 2500 documents covering 200 different people for 20 ambiguous names. Experimental results show that the proposed method outperforms numerous baselines and previously proposed name disambiguation methods. Moreover, the extracted keywords reduce ambiguity of a name in an information retrieval task, which underscores the usefulness of the proposed method in real-world scenarios.

Keywords

name disambiguation, automatic annotation, clustering, web mining

1 Introduction

World-Wide-Web is a rich source of information about people and their activities. Social Network Services (SNSs), personal home pages, research publications, online newspapers and magazines are among the major information sources about people on the Web. To retrieve information about people we search on the web using personal names as queries. In fact, 30% of all web queries have been reported to include personal names (Guha and Garg, 2004; Artiles et al., 2005). Despite the popular use of personal names as queries, personal names are one of the most ambiguous types of named entities on the Web. According to a report by the U.S. Census Bureau (Guha and Garg, 2004), only $90,000$ different names are shared by 100 million people. With the growing popularity of the Web, the problem of ambiguous personal names is expected to aggravate.

For example, consider searching for *Jim Clark* on Google[1]. Even among the top 100 search results returned by Google, we find eight different *Jim Clarks* including the two popular namesakes (i.e. different people with the same name); *Jim Clark* the Formula One racing champion (46 pages) and *Jim Clark* the founder of *Netscape* (26 pages). Both *Jim Clarks* are equally popular on the Web and using the name alone as a query is inefficient to retrieve information related to one of the Jim Clarks. A user who seeks for information regarding a particular namesake must read each search result separately and must decide whether it is relevant or

[1]http://google.com

not. This is a time-consuming and tedious task considering the vast number of search results returned by a search engine for popular names. Manual disambiguation might even become an impossible task for the users, if they do not possess a sufficient knowledge regarding the individual that they are searching for.

The problem of identifying people on the Web is further complicated by the existence of people with multiple web appearances. A web-based name disambiguation algorithm must consider these complications. For example, the renowned linguist, *Noam Chomsky* also appears as a critic of U.S. foreign policy on the Web. Moreover, many people prefer to have an official web page regarding their professional activities and a separate private blog, for example, where they express their personal opinions. Some people associate themselves with several different name aliases. For example, the popular movie star *Will Smith* often called as the *fresh prince* in web contexts. In anti-aliasing (Novak et al., 2004), the goal is to map the different alias names to an individual. On the other hand, in name disambiguation, we are faced with the challenge of identifying different people who share the same identical name.

In this paper, we propose an unsupervised algorithm to automatically annotate a given ambiguous personal name with automatically extracted keywords from the Web. A fundamental problem that needs to be solved in web-based name disambiguation is the accurate prediction of the number of different people who have an ambiguous name. We propose a novel method based on normalized cuts (Shi and Malik, 2000) cluster quality measure to predict this number.

This paper is organized as follows. First, we describe possible contributions of the proposed method to social network extraction and Semantic Web. Next, we present an overview of the related work in this field. In Section 2.1, we define the problem of annotating ambiguous personal names and detail the proposed solution. Central to the proposed method are *Term-Entity Models* (TEMs). We define TEMs in Section 2. In Section 4, we perform various experiments to evaluate the ability of the proposed method to disambiguate personal names. Finally, we discuss the experimental results and conclude the paper.

1.1 Personal Name Disambiguation in Social Networks

Social networks have grown both in size and popularity over the recent years. *mixi*[2], a popular social network system in Japan, reported over ten million registered users at the turn of the year 2007. As more and more people join these social networks, it is highly likely that more than one person with identical names exist in the network. In order to identify a particular person in a social network by his or her name, we must first resolve the ambiguity for that name. The proposed method can be used to annotate people using automatically extracted keywords, thereby reducing the ambiguity in the social network.

Disambiguating personal names is an essential first step in many social network extraction algorithms (Mika, 2004; Matsuo et al., 2006b). Given a pair of names, social network extraction algorithms attempt to capture the degree of association between the two people from the Web. Various association measures such as the Jaccard coefficient (Mika, 2004), and Overlap coefficient (Matsuo et al., 2006b) have been proposed to find the relationships between personal names on the Web. They use page-counts returned by a web search engine for the individual names and the conjunctive (AND) query to compute association measures. Page-count of a query is the number of different web pages in which the query words

[2]http://mixi.jp/

appear. Most major web search engines provide page-counts (or approximated page-counts) for user queries. However, if one or both of the names are ambiguous (i.e. if there are other people with the same names), then page-counts do not accurately reflect the association of the two persons that we are interested. One solution to this problem is to include a keyword that uniquely identifies the person under consideration from his or her namesakes. The keywords extracted by the proposed method have been successfully utilized to disambiguate real-world large-scale social networks (Matsuo et al., 2006a).

The friend of a friend (FOAF) project[3] is an initiative to create an annotated web of people (Mika, 2005). In FOAF, users can describe themselves using keywords, provide links to their home pages and introduce their friends. FOAF uses RDF (Resource Description Framework) to represent the information provided by the users. We can boost the manual annotation process in FOAF by automatically extracting keywords from the Web that describe individuals.

1.2 Personal Name Disambiguation in Ontologies

An ontology is a formal representation of knowledge that one has about a particular domain. An ontology expresses the different concepts that belong to the domain in interest and the relationships between those concepts. However, manually creating large-scale ontologies from the scratch can be time consuming and tedious. Several methods have been proposed to boost the process of ontology creation by aligning and merging existing ontologies to create larger ontologies (Noy and Musen, 1999; Kalfoglou and Schorlemmer, 2003; Hage et al., 2006), and extracting ontologies from the Web (Cimano et al., 2004). Moreover, proper alignment of ontologies is

[3]http://www.foaf-project.org/

important for evaluating ontologies (Euzenat, 2007), where an ontology is compared against a gold-standard using various evaluation metrics[4]. However, ambiguity among concepts in different ontologies lead to inaccurate alignments. For example, consider merging two ontologies representing employees in two companies. A particular personal name might appear multiple times in the two ontologies. A single person can be associated with different projects in an internal ontology of a company. Before merging the two ontologies one must first resolve the ambiguities for the concepts. Annotating concepts (in this paper we focus on personal names) with extra keywords is a useful way to disambiguate entries in an ontology.

2 Automatic annotation of ambiguous personal names

2.1 Problem definition

Definition 1. *Given an entity e which has the name n, we call it* ***ambiguous*** *if there is at least one other entity $e\prime$ which has the same name n.*

For example, in the case of people, if two or more people have the same personal name n, then they are collectively called as *namesakes* of the ambiguous name n.

Definition 2. *Given an ambiguous entity e, the problem of automatic annotation of e is defined as the task of finding a set of words (or multiword expressions) $W(e)$, that uniquely identify e from his or her namesakes.*

For example, in our example of *Jim Clark*, the set of words (or multiword expressions) racing driver, formula one, scotsman can identify the Formula One

[4]http://oaei.ontologymatching.org/2007/

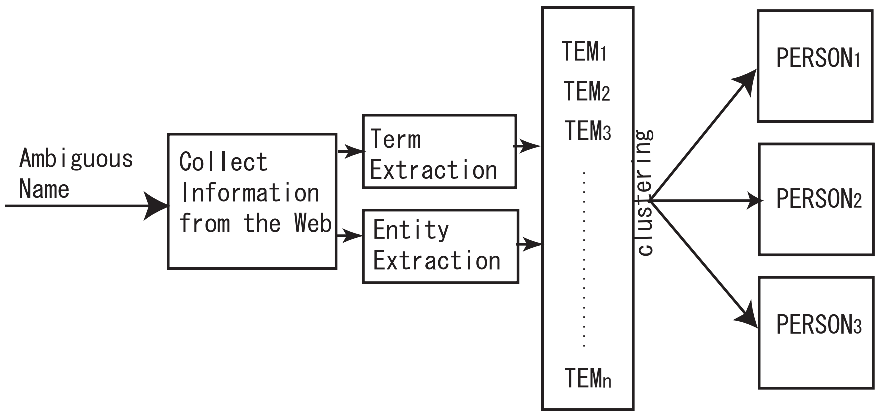

Figure 1: A TEM is created from each search result downloaded for the ambiguous name. Next, TEMs are clustered to find the different namesakes. Finally, discriminative keywords are selected from each cluster and used for annotating the namesakes.

racing champion Jim Clark from the other Jim Clarks in the Web. It is noteworthy that the name string (i.e. *jim clark*) itself is not unique and does not belong to the set of words. In practice, whether a particular word (or a multiword expression) can uniquely identify a namesake of a given ambiguous name, can be difficult to decide. In this paper, we take a pragmatic approach and decide a combination of words (or multiword expressions) that can uniquely identify a person, if that combination of words together with the person's name (e.g. in a conjunctive query) can return results only for that person in a web search engine.

2.2 Outline

The proposed method is outlined in Figure 1. Given an ambiguous personal name, the first step in our algorithm is to collect information regarding the different people with that name from the Web. Retrieving a relevant set of documents for a query, is a fundamental problem that has been studied extensively in information retrieval (Salton and McGill, 1986). In this work, we assume the availability of a web search engine and query for a given ambiguous personal name to collect information regarding the different people who are represented on the web by that name. Specifically, given an ambiguous personal name n, we download the top N ranked search results and only consider the namesakes with name n that appear in this set of N results.

How to represent information about people and their web activities is an important problem that any web-based identity disambiguation method must consider. For this purpose, we propose *Term-Entity Models* (TEMs). TEMs are sets of terms or named entities that are closely associated with the people with an ambiguous name on the web. We formally define TEMs in Section 2.3, and describe an unsupervised method to create TEMs for a given personal name in Section 2.4. To determine whether two documents disclose information about the same person, we must compare the TEMs created for the two documents. However, measuring the similarity between TEMs is not a trivial task. For example, consider the two phrases *Formula One* and *Racing Championship*. The two phrases are closely related because Formula One is a racing championship. However, there are no words in common (i.e. zero word overlap) between those two phrases. To infer that the two TEMs in this example correspond to two documents that are about the racing champion Jim Clark, we must accurately measure the similarity between phrases (terms or named-entities). We employ a contextual similarity measure (Section 2.5) for this purpose.

We make the assumption that all occurrences of a given ambiguous personal name within a document (e.g. a web page) refer to the same individual. Under this assumption, identifying the different namesakes for a given name can be modeled as a document clustering problem. Initially, a TEM is created from each downloaded document. Next, we cluster those TEMs to identify the different people (namesakes) for the ambiguous personal name. We use group-average agglomerative hierarchical clustering for this purpose. A document is assigned to only one cluster (i.e. hard clustering). Ideally, each final cluster formed by this process must represent a different namesake. Therefore, the number of clusters must be equal to the number of different namesakes of the given name. However, in reality it is not possible to know in advance the number of different namesake of a given name on the web. Therefore, we terminate the agglomerative clustering process when the overall cluster quality (defined in Section 2.6) drops below a pre-defined threshold value. The threshold is determined using a development dataset. Finally, we select unique keywords from each cluster, and annotate the different namesakes using the extracted keywords.

2.3 Term-Entity Models

When searching for an individual who has numerous namesakes on the Web, one quick solution that we frequently adopt is to append the query with one or two keywords that identify the person we are interested in from his or her namesakes. For example, if we want to search for *Jim Clark* the *racing driver* we could append the query with keywords such as *Racing Driver, Formula One* or *Scotsman*. These keywords enable us to filter-out the search results for other *Jim Clarks*. We extend this idea and propose a keyword-based model to represent individuals on the Web.

Definition 3. *A* ***Term-Entity Model*** *(TEM) of a person p is a set of terms or named-*

entities that uniquely describes that person from his or her namesakes. Terms and/or named entities that construct a TEM are called ***elements*** *of the TEM.*

We use the notation $T(p)$ to denote the TEM of a person p. Then with the conventional set notation we can write,

$$T(p) = \{e_1, e_2, \ldots, e_n\}.$$

Here, $e_1, e_2, \ldots, e_n$ are the elements of $T(p)$ and can be terms or named entities.

For example, TEM for $\mathrm{JimClark}_{\mathrm{driver}}$, the racing champion, could be,

$$T(\mathrm{JimClark}_{\mathrm{driver}}) = \{\mathrm{Formula\ One}, \mathrm{Racing\ Driver}, \mathrm{Champion}\}.$$

In this example, *Racing Driver* and *Champion* are terms whereas, *Formula One* is a named-entity. We use the subscript notation here to indicate a namesake of a name.

TEMs capture the essence of the keyword-based boolean web queries we are accustomed to. For simplicity, if we limit ourselves to conjunctive queries (*AND* queries), then the elements of an TEM act as the literals of the boolean query that identifies a person with the ambiguous name. Moreover, TEMs can be considered as a scaled down version of the bag-of-words (BOW) model (Manning and Schutze, 2002), which is commonly used in information retrieval. Bag-of-words model represents a document as a set of words. However, considering all the words as identifiers of a person is noisy and inefficient. The reasons for using both terms and named-entities in TEMs are two fold. Firstly, there are multi-word phrases such as the *secretary of state, chief executive officer, racing car driver* which are helpful when identifying people on the web but are not recognized as named-entities. Secondly, automatic term extraction (Frantzi and Ananiadou, 1999) can be carried out using statistical methods, and does not require extensive linguistics resources such as named entity dictionaries, which might not be readily available for some domains. It is noteworthy that we do not distinguish terms from named-entities once we have created a TEM for a person. All elements in a TEM are considered equally, irrespective of whether they are terms or named-entities in the subsequent processing.

2.4 Creating Term-Entity Models from the Web

Given an ambiguous personal name, we extract terms and named entities from the *contexts* retrieved from a web search engine for the name to create its TEM. If the ambiguous name n appears in a document D, then the context of n could be for example a paragraph containing n, a fixed window of words including n, or the entire text in document D. Other than for some exceptional cases, such as a search results page for an ambiguous name from a search engine or a disambiguation entry in Wikipedia, usually a single web page does not include more than one namesake of an ambiguous name. In fact, all previous work in web-based name disambiguation have modeled this problem as a web page clustering problem, where each cluster represents a different person of the given ambiguous name. Following these lines, we consider the entire document where an ambiguous name appears as its context.

For automatic multi-word term extraction, we use the *C-value* measure proposed by Frantzi et al. (1999). 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 a stop word list. First, the context from which we need to extract terms is tagged using a part of speech tagger. Next a set of pre-defined POS patterns are used to extract candidate terms. For example, the POS pattern $(NN+)$ extracts noun phrases, and $(AdJ)(NN+)$ extracts noun phrases modified by an adjective. Here, NN represents a single noun, Adj represents a single adjective, and $+$ matches one or more occurrences of the preceding term. A list of stop words can be used to prevent extracting common words that are not considered as terms in a particular domain. Having a stop words list improves the precision of term extraction. However, in our experiments we did not use a stop words list because it is not possible to determine in advance the domain which a namesake belongs to.

The sequences of words that remain after this initial filtering process (here onwards referred to as candidates) are evaluated for their *termhood* (likeliness of a candidate to be a term) using the *C-value* measure which is defined as,

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

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

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 (measured in the number of words). The higher the C-value of a candidate, more likely it is a term. For candidates that occur equal number of times in a corpus C-value method prefers the longer candidates to shorter candidates. In our experiments we select candidates with C-value greater than 2 as terms. (see (Frantzi and Ananiadou, 1999) for more details on C-value-based multi-word term extraction). However, there are cases where the terms extracted from the C-value method tend to be exceedingly longer and meaningless. For example, we get the 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 is not a genuine term. To avoid such terms we use two heuristics. First, we ignore any terms which are longer than four words. Second, for the remaining terms, we check their page-counts we obtain from a web search engine. Our assumption here is that, if a term is a meaningful, then it is likely to be used in many web pages. We ignore any terms with less than five page-counts. Those heuristics allow us to extract more expressive and genuine terms.

To extract entities for TEMs, the contexts are tagged using a named entity tagger developed by the Cognitive Computation Group at UIUC[5]. From the tagged contexts we select personal names, organization names and location names to include in TEMs. UIUC named-entity tagger has an F1-score of 90.80% on CoNLL03 evaluation dataset (Ratinov and Roth, 2009). Statistical methods for term extraction do not necessarily extract all the named entities in a document usually found by a named entity tagger. Term extraction algorithms (and the tools that implement those algorithms) first limit the candidate terms using part-of-speech-based filters that capture POS sequences common for technical terms. For example, in the C-

[5]http://l2r.cs.uiuc.edu/~cogcomp/

value method used in our work, (NN+) and (Adj)(NN+) POS filters are in place. On the other hand, a named entity recognizers (NER) use large lists of named entities and contextual rules (either learnt from training data or manually compiled) to detect potential candidate entities. For example, an NP that follows prefixes such as Mr., Prof., Dr., Ms. are likely to be person names. In addition to the difference in the candidate sets processed by term extraction tools and named entity recognizers, the candidates are weighted differently. In a statistical term extraction algorithm such as the C-value method, a candidate term to be selected it must either occur many times in the corpus or must be nested in many other candidates. Therefore, even if a named entity appears in the candidate set of a term extractor it might not necessarily receive a high score (termhood) in the subsequent weighting process. To avoid selecting incorrect extractions, we only select terms with termhood greater than two when we create Term-Entity models. Therefore, all named entities that are recognized by a named entity recognizer might not be selected from a term extraction tool. Therefore, we must use both a term extractor as well as a named entity recognizer in the system. From a theoretical point of view one could design a keyphrase extractor that extracts both terms and named entities and use its output to build the Term-Entity models. However, we could not find a keyphrase extraction tool that meets our particular requirements. A potential future research direction would be to design such a keyphrase extractor for the task of name disambiguation and compare the effect of the different term extraction algorithms on the overall performance of the proposed method.

2.5 Contextual Similarity

We must calculate the similarity between TEMs derived from different contexts, in order to decide whether they represent the same namesake or not. WordNet[6] based similarity metrics have been widely used to compute the semantic similarity between words in sense disambiguation tasks (Banerjee and Pedersen, 2002; McCarthy et al., 2004). However, most of the terms and entities are proper names or multi-word expressions which are not listed in the WordNet.

Sahami et al. (2005) proposed the use of snippets returned by a Web search engine to calculate the semantic similarity between words. A snippet is a brief text extracted from a document around the query term. Many search engines provide snippets alongside with a link to the original document. Snippets help a web search engine user to decide whether a search result is relevant without actually having to click the link. Because snippets capture the immediate surrounding of the query in a document, we can consider a snippet to be the context of the query term. Using snippets is also efficient because it obviates the need to download the source documents from the web, which can be time consuming depending on the size of the page. To calculate the contextual similarity between elements (terms and entities) in TEMs, we first collect snippets for each element by querying a web search engine for that element. We then represent each snippet S_i as a vector $\vec{v_i}$ of words that appear in S_i. Each word in a snippet is weighted using TF-IDF weighting method (Salton and McGill, 1986). Weight w_{ij} of a word T_j that appears

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

in a snippet S_i is defined as follows,

$$w_{ij} = tf_{ij} \log_2 \frac{N}{df_j}. \tag{2}$$

Here, tf_{ij} is the term-frequency of word T_j in the snippet S_i (i.e. the number of times that T_j occurs in S_i). N is the total number of snippets retrieved from the search engine for the query, and df_j is the document-frequency of word T_j (i.e. the number of snippets that contained the word T_j). We then compute the centroid vector, $\vec{C(a)}$, for a query a by averaging all the snippet-word vectors $\vec{v_i}$ as follows,

$$\vec{C(a)} = \frac{1}{N} \sum_{i=1}^{N} \vec{v_i}. \tag{3}$$

Moreover, we normalize centroid vectors such that their L_2 norm is 1. Normalizing snippet-word vectors to unit length enables us to compare snippets with different numbers of words. We define the contextual similarity, $\mathrm{ContSim}(a, b)$, between two elements a, b, in TEMs as the inner product between their centroid vectors $\vec{C(a)}, \vec{C(b)}$.

$$\mathrm{ContSim}(a, b) = \vec{C(a)} \cdot \vec{C(b)} \tag{4}$$

Let us illustrate the above mentioned contextual similarity measure by an example. Consider computing the association between the two phrases *"George Bush"* and the *"President of the United States"*. First, we issue the query *"George Bush"* to a web search engine and download snippets. In this example, we download the top 100 ranked snippets by Google for the query. We then use TF-IDF method (Equation 2) to weight the words in snippets. Each snippet is represented by a vector of words weighted by TF-IDF. Because we have 100 snippets in this example we obtain 100 vectors. Next, the centroid vector of those 100 vectors is computed using Equation 3. Similarly, a centroid vector is computed for the query *"President of the United States"*. Finally, the similarity between the two phrases is computed as the inner product between the corresponding centroid vectors using Equation 4.

Figure 2 shows the distribution of most frequent words in snippets for these two queries. We can observe a high overlap between the two distributions. Interestingly, the words *george* and *bush* appear with a high frequency among the snippets for the query *"President of the United States"* and word *president* appears with a high frequency for the query *"George Bush"*. The contextual similarity between the two queries is 0.2014.

On the other hand, if we compare snippets for the queries *"Tiger Woods"* and *"President of the United States"* (as shown in Figure 3) we get a relatively low similarity score of 0.0691. This indicates *"George Bush"* is more closely related to the phrase the *"President of the United States"* than *"Tiger Woods"* is.

Using the snippet-based contextual similarity measure, we define the similarity $\mathrm{sim}(T(A), T(B))$, between two TEMs $T(A) = \{a_1, \ldots, a_n\}$ and $T(B) = \{b_1, \ldots, b_m\}$ of contexts A and B as follows,

$$\mathrm{sim}(T(A), T(B)) = \frac{1}{nm} \sum_{i,j} \mathrm{ContSim}(a_i, b_j). \tag{5}$$

Therein; $\mathrm{ContSim}(a_i, b_j)$ is the contextual similarity between elements a_i and b_j, and it is given by Equation 4.

Contextual similarity measure that use Web snippets overcomes several limitations observed in other similarity measures. First, because contextual similarity

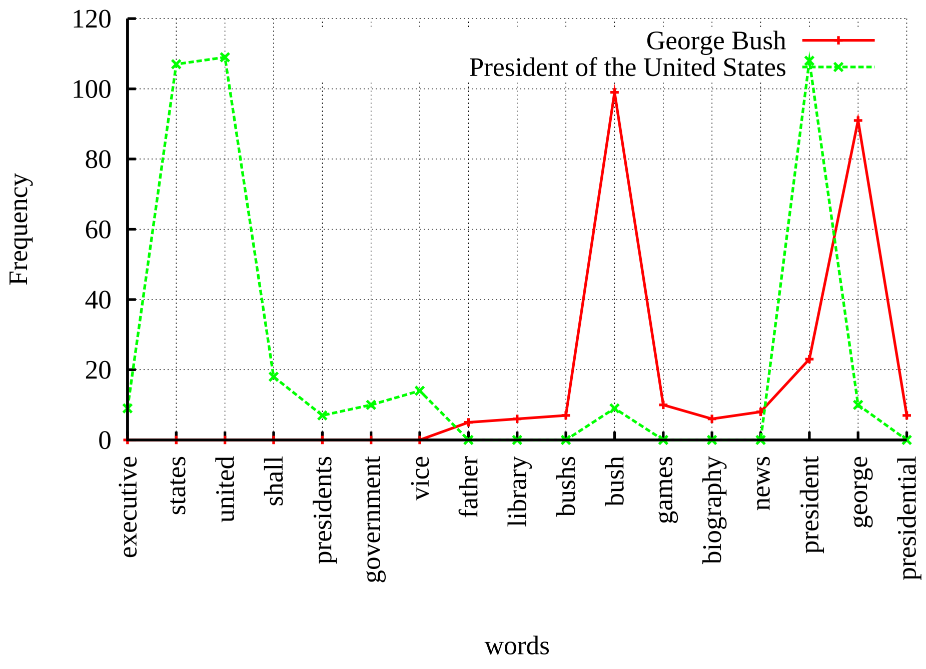

Figure 2: Distribution of words in snippets for “George Bush” and “President of the United States”

measure described above uses words that appear in Web snippets retrieved from a Web search engine, it is able to find more informative contexts for the queries (keyphrases or named entities) between which we must compute similarity, thereby overcoming the data sparseness problem. This is particularly important in our task because we must compute the similarity between person names and terms (multi-word expressions) that do not appear in manually created dictionaries such as WordNet or pre-compiled text corpora. Second, to compute the similarity between N terms or entities the number of queries that we must issue to the Web search engine is proportional to N. On the other hand, if we consider a similarity measure that requires the number of co-occurrences of two words (e.g., Jaccard co-

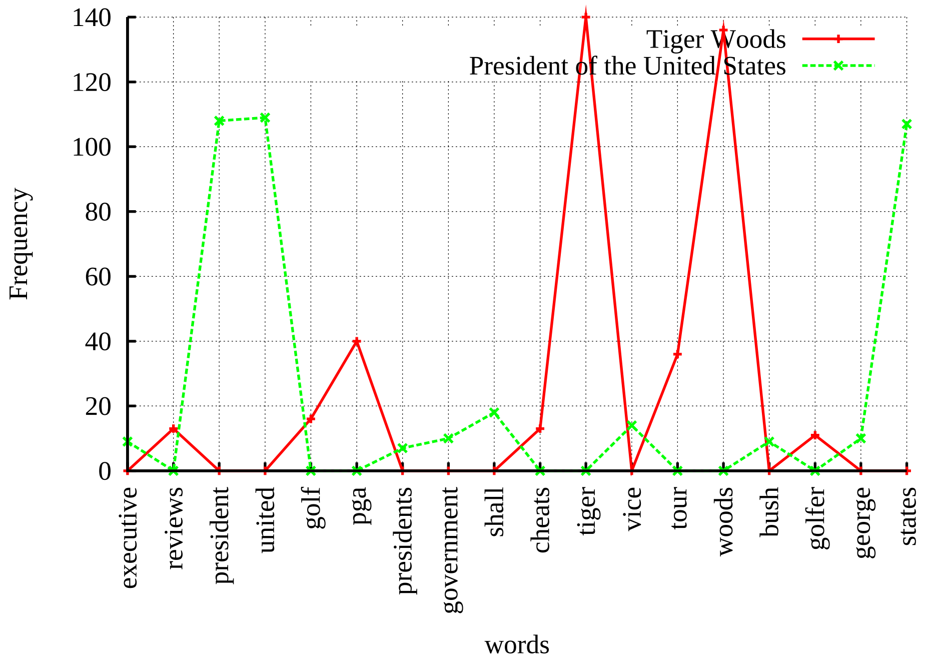

Figure 3: Distribution of words in snippets for “Tiger Woods” and “President of the United States”

efficient, pointwise mutual information), then we must issue a query for each pair of words between which we must compute similarity. For N words this requires Web queries proportional to $N(N-1)/2$. In fact, this property of the contextual similarity measure enables it to be kernalized (Sahami and Heilman, 2005). In Section 4.5, we empirically compare the contextual similarity measure against several other similarity measures.

2.6 Clustering

We use Group-Average Agglomerative Clustering (GAAC) (Cutting et al., 1992), a hybrid of single-link and complete-link clustering, to cluster the contexts that belong to a particular namesake. Initially, we assign a separate cluster for each of the contexts in the collection. Then, GAAC in each iteration executes the merger that gives rise to the cluster Γ with the largest average correlation $C(\Gamma)$ where,

$$C(\Gamma) = \frac{1}{2}\frac{1}{|\Gamma|(|\Gamma|-1)}\sum_{u\in\Gamma}\sum_{v\in\Gamma}\mathrm{sim}(T(u),T(v)). \qquad (6)$$

Here, $|\Gamma|$ denotes the number of contexts in the merged cluster Γ; u and v are two contexts in Γ, and $\mathrm{sim}(T(u),T(v))$ is given by Equation 5.

Ideally, the number of clusters formed by the GAAC process must be equal to the number of different namesakes for the ambiguous name. However, in reality it is impossible to exactly know the number of namesakes that appear on the Web for a particular name. Moreover, the distribution of pages among namesakes is not even. For example, among the top 100 results retrieved for the name "Jim Clark" from Google, 78 belong to the two famous namesakes; *Founder of Netscape* and *Formula One world champion*. The remaining 22 search results (web pages) are distributed among six other namesakes. If these outliers get attached to the otherwise pure clusters, both disambiguation accuracy and keywords selection deteriorate. Therefore, we monitor the *quality* of clustering and terminate further agglomeration when the cluster quality drops below a pre-set threshold value. Numerous metrics have been proposed for evaluating the quality of clustering (Kannan et al., 2000). In this paper, we use normalized cuts measure proposed by Shi and Malik (2000).

Let V denote the set of contexts for a name. Consider, $A \subseteq V$ to be a cluster of contexts taken from V. For two contexts x,y in V, $\mathrm{sim}(x,y)$ represents the contextual similarity between the contexts (Equation 5). Then, the normalized cut

$N_{cut}(A)$ of cluster A is defined by,

$$N_{cut}(A) = \frac{\sum_{x \in A\, y \in (V-A)} \mathrm{sim}(x,y)}{\sum_{x \in A\, y \in V} \mathrm{sim}(x,y)}. \tag{7}$$

For a set, $\{A_1, \ldots, A_n\}$ of non-overlapping n clusters A_i, we define the *quality* of clustering, $\mathrm{Quality}(\{A_1, \ldots, A_n\})$, as follows,

$$\mathrm{Quality}(\{A_1, \ldots, A_n\}) = \frac{1}{n} \sum_{i=1}^{n} N_{cut}(A_i). \tag{8}$$

For the set of clusters formed at each iteration of the agglomerative clustering process, we compute the cluster quality using Equation 8. We terminate the clustering process, if the cluster quality drops below a fixed threshold θ. Finally, we assign the remaining contexts (singletons) to the already formed clusters based on the correlation (Equation 6) between a context and a cluster. We experimentally determine the cluster stopping threshold θ using a development dataset as described later in Section 4.4.

2.7 Automatic annotation of namesakes

GAAC process produces a set of clusters representing each of the different namesakes of the ambiguous name. To annotate the namesakes represented by the formed clusters, we select elements (terms and entities) from TEMs in each cluster. To select appropriate keywords to annotate a person represented by a cluster, we first compute the union of all TEMs in that cluster. We then remove any elements that appear in other clusters. This process yields a set of elements that uniquely represents each cluster. Finally, we rank each element in a cluster according to its similarity with the given ambiguous name. We use Equation 4 to compute the similarity between the given name and an element. The context of a name is approximated by the top ranking snippets. We used the top ranked 100 snippets in our experiments. For example, in Section 2.5, we computed the similarity between the name *George Bush* and the element (a term) *President of the United States* to be 0.2014. The motivation behind ranking elements is to identify the keywords which are closely related to the namesake. Each namesake is annotated using the top ranking elements in his or her cluster.

Alternatively, we can first rank all the elements in each cluster using the similarity between the name and the element using Equation 4, and subsequently remove any elements that are ranked below a certain rank. Optionally, we can remove elements that appear in more than one cluster to obtain a set of keywords that uniquely identify a cluster. This alternative approach is particularly useful when there are multiple namesakes who are popular in a particular field. However, this approach requires more web search queries compared to the previous approach, because we must first compare *all* elements in a cluster with the given name in order to rank them. On the other hand, first removing common elements in different clusters can significantly reduce the number of comparisons (thereby the web search queries). Furthermore, during our preliminary experiments with this second approach we did not notice any significant improvement in the quality of keywords obtained at the cost of additional web queries. Therefore, we adopted the first approach which require comparatively lesser number of web search queries, where we first remove elements that appear in multiple clusters and subsequently rank the remaining elements.

3 Evaluation Datasets

To evaluate the ability to disambiguate and annotate people with the same name, we create a dataset for ambiguous personal names; *Jim Clark* and *Michael Jackson*. For each of those names, we query Google and download the top ranking search results. We then manually annotate each search result by reading the content in each downloaded web page. We exclude pages that only contain non-textual data, such as images. A web page is assigned to only one namesake of the ambiguous personal name under consideration. Moreover, we evaluate on two datasets created in previous work on namesake disambiguation: Pedersen and Kulkarni (2007a; 2007b)'s dataset (5 ambiguous names: *Richard Alston, Sarah Connor, George Miller, Michael Collins* and *Ted Pedersen*), and Bekkerman and McCallum (2005)'s dataset (12 ambiguous names: *Adam Cheyer, William Cohen, Steve Hardt, David Israel, Leslie Pack Kaelbling, Bill Mark, Andrew McCallum, Tom Mitchell, David Mulford, Andrew Ng, Fernando Pereira* and *Lynn Voss*. By using the same datasets used in previous work, we can directly compare the proposed method with previous work on namesake disambiguation.

To create a gold standard for namesake disambiguation one must first manually annotate each search result retrieved for an ambiguous personal name. All datasets mentioned above take this approach. As an alternative approach that does not require manual annotation of search results, Pedersen et al. (2005) propose the use of *pseudo ambiguous names*. In this approach, first a set of unambiguous personal names are manually selected. For each of the names in this set there must be only one individual in the web. Next, a web search engine is queried with each of the unambiguous names separately and search results are downloaded. Finally, each occurrence of the queried name is replaced by an identifier (e.g. person-X) and the search results retrieved for all the unambiguous names are conflated to create a single dataset. This conflated dataset can be considered as containing namesakes for the pseudo-ambiguous name, person-X. Moreover, we know which search result belongs to which namesake without any manual annotation because we replace a name with an identifier by ourselves. Although this process obviates the need for manual annotation, thereby enabling us to easily create a large dataset, it is sometimes criticized because it does not reflect the natural distribution of namesakes for real-world ambiguous personal names. Following the previous work on this line, for automated pseudo-name evaluation purposes, we select the four names (*Bill Clinton*, *Bill Gates*, *Tom Cruise* and *Tiger Woods*) for conflation. We download the top 100 ranking search results from Google for each of these names and manually confirmed that the search results did not contain any namesakes of the selected names. We then replace the ambiguous personal name by the string "person-X" in the collection, thereby artificially introducing ambiguity. The complete dataset that we used for experiments is shown in Table 1. We have grouped the names in Table 1 according to the datasets that they belong to. Moreover, names within a particular dataset are sorted alphabetically.

4 Experiments and Results

4.1 Outline

In this section we present the numerous experiments we conduct to evaluate the proposed personal name disambiguation algorithm. In our experiments, we query

Table 1: Experimental Dataset

Name	number of namesakes	number of contexts
person-X	4	137
Jim Clark	8	100
Michael Jackson	2	82
Pedersen and Kulkarni's dataset (Pedersen and Kulkarni, 2007a,b)		
George Miller	3	286
Michael Collins	4	359
Richard Alston	2	247
Sarah Connor	2	150
Ted Pedersen	4	333
Bekkerman and McCallum's dataset (R.Bekkerman and A.McCallum, 2005)		
Adam Cheyer	2	97
Andrew McCallum	8	94
Andrew Ng	29	87
Bill Mark	8	94
David Israel	16	92
David Mulford	13	94
Fernando Pereira	19	88
Leslie Pack Kaelbling	2	89
Lynn Voss	26	89
Steve Hardt	6	81
Tom Mitchell	37	92
William Cohen	10	88
Total	205	2779

Google[7] for a given ambiguous personal name and download the top ranked 100 web pages. We eliminate pages that do not contain any text. We use Beautiful Soup[8], an HTML parser, to extract text from HTML pages. Next, we create a TEM from each resulting web page as described in Section 2.4. The set of web pages downloaded for the given ambiguous personal name is then clustered using the clustering algorithm described in Section 2.6. We use contextual similarity

[7]http://code.google.com/

[8]http://www.crummy.com/software/BeautifulSoup/

(Equation 4) to compute the similarity between elements (i.e. terms or named-entities that appear in a term-entity model) in TEMs created for web pages. In Equation 4, we use the top ranking 100 snippets returned by Google for an element as its context.

In Section 4.2, we describe *disambiguation accuracy*, the evaluation measure used in our experiments. In Section 4.3, we compare disambiguation accuracy with cluster quality introduced in Section 2.6. We determine the cluster stopping threshold θ using development data in Section 4.4. In Section 4.5, we compare the performance of the proposed method against baseline methods and previous work on namesake disambiguation. Moreover, in Section 5, we employ the keywords selected for different namesakes of an ambiguous personal name in an information retrieval task

4.2 Evaluation Measure

To evaluate the clusters produced by the proposed method we compare them with the gold-standard clusters for a name in a dataset. For each ambiguous personal name, the gold standard contains a set of contexts (web pages) downloaded and assigned to a namesake. In the gold standard a web page is assigned to only one of the namesakes of the given name. Therefore, we can consider the set of contexts in the gold standard for a particular name as a set of non-overlapping clusters. We compare the set of clusters in the gold standard with the set of clusters produced the proposed method. If the two sets of clusters are similar, then we can conclude that the proposed method can accurately disambiguate namesakes for a given personal name.

First, we assign each cluster to the namesake that has the most number of contexts (web pages) for him or her in that cluster. If there is more than one namesake in a cluster with the highest number of contexts, then we randomly select one of those namesakes and assign to the cluster. This process assigns a cluster to one of the namesakes for the given personal name. The purpose of this assignment is to align the set of clusters produced by the proposed method with the manually created gold standard. Assigning a cluster to a particular namesake of the given ambiguous name enables us to compute the accuracy of the clusters produced by the proposed method as described below. However, it should be noted that there exist numerous alignment methods other than the majority assignment approach adopted in this paper. For example, one could search for the alignment that maximizes some evaluation measure such as the micro-average F1 score (Manning et al., 2008). However, we employed the majority assignment method in this work for its simplicity.

Next, we evaluate experimental results based on the confusion matrix A, where $A[i.j]$ represents the number of contexts for "person i" predicted as "person j". $A[i,i]$ represents the number of correctly predicted contexts for "person i". We define *disambiguation accuracy* as the sum of diagonal elements divided by the sum of all elements in the matrix as follows,

$$\text{Disambiguation Accuracy} = \frac{\sum_i A(i,i)}{\sum_{i,j} A(i,j)}. \tag{9}$$

If all contexts are correctly assigned for their corresponding namesakes then the confusion matrix A becomes a diagonal matrix and the disambiguation accuracy becomes 1. In practice, the number of clusters produced by a namesake disambiguation system might not necessarily be equal to the number of namesakes for

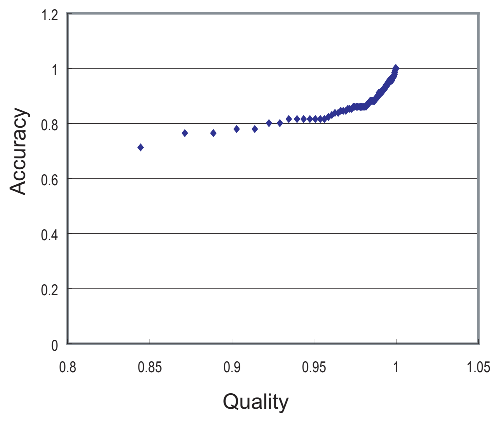

Figure 4: Accuracy vs Cluster Quality for person-X data set.

an ambiguous personal name. The above mentioned cluster assignment procedure can assign multiple clusters to a particular namesake, or not assign any cluster for some namesakes, depending on the set of clusters produced by a system. However, it is noteworthy that disambiguation accuracy can still be computed using the definition in Equation 9 even under such circumstances.

4.3 Correlation between cluster quality and disambiguation accuracy

In Section 2.6, we proposed the use of cluster quality (which can be computed in an unsupervised manner without using the gold standard clustering) to determine when to stop the agglomerative clustering process. However, it remains unknown whether the cluster quality can accurately approximate the actual accuracy of a clustering algorithm. In order to evaluate how well does normalized cuts-based cluster quality reflects the accuracy of clustering, we compare disambiguation accuracy (computed using the gold-standard) with cluster quality (computed using Equation 8) for person-X collection as shown in Figure 4. For the data points shown in Figure 4, we observe a high correlation between accuracy and quality (Pearson correlation coefficient between accuracy and quality is 0.865). This result enables us to guide the clustering process and determine the optimal number of clusters using cluster quality.

4.4 Determining the Cluster Stopping Threshold θ

In Section 2.6 we described a group-average agglomerative hierarchical clustering algorithm to cluster the contexts (i.e. web pages). We empirically determine the cluster stopping threshold θ using person-X collection as a development dataset. Figure 5 shows the accuracy of clustering against various threshold values. According to Figure 5 we set the threshold at 0.935 where accuracy maximizes for person-X collection. Threshold θ is fixed at this value (0.935) for the remainder of the experiments described in the paper.

The threshold that is used to determine the number of clusters is based on the cluster quality measure defined in Formula 8. Quality of the clustering depends only upon the clusters produced. Because it is a normalized score that does not depend on the number of clusters, even though the number of namesakes (there by the clusters) might vary with different ambiguous names, the threshold computed from the Person-X dataset can be used in other datasets. In our future work, we plan to investigate the effect of the popularity and ambiguity of different names on the determination of the threshold.

4.5 Clustering Accuracy

Table 2 summarizes experimental results for the accuracy of the clustering. For each ambiguous name in our dataset, the second column in Table 2 shows the

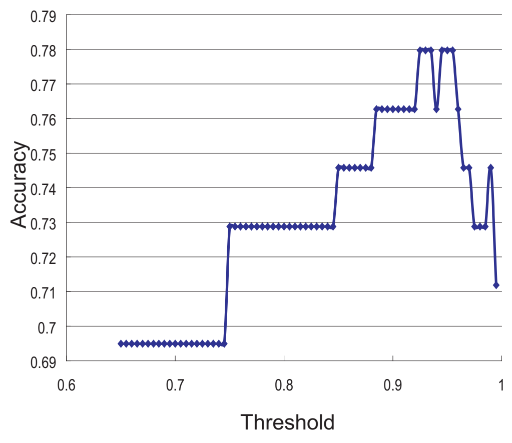

Figure 5: Accuracy vs Threshold value for person-X data set.

number of different people in the collection with that name. Moreover, we have visualized the experimental results in Figure 6 to show the overall trend. We compare the proposed method against the following three baselines.

Jaccard (Jaccard coefficient-based clustering) : This method computes the similarity between two TEMs using the Jaccard coefficient. Jaccard coefficient between two sets A and B is defined as follows,

$$\text{Jaccard} = \frac{|A \cap B|}{|A \cup B|}.$$

Here, $|A \cap B|$ denotes the number of elements in the intersection of sets A and B, $|A \cup B|$ is the number of elements in the union of sets A and B. If two TEMs share many elements then the Jaccard coefficient computed over the two TEMs will be high. Using the Jaccard coefficient as the similarity measure, we perform group average agglomerative clustering with cluster stopping enabled to discriminate the namesakes. This baseline shows the

Table 2: Comparing the proposed method against baselines.

Name	Namesakes	Jaccard	Overlap	Proposed	Majority
person-X	4	0.8382(4)	0.7941(4)	0.7941(4)	0.6985(1)
Jim Clark	8	0.8475(3)	0.8305(3)	0.8475(3)	0.6949(1)
Michael Jackson	2	0.9706(2)	0.9706(2)	1.0000(2)	0.6765(1)
Pedersen and Kulkarni's dataset (Pedersen and Kulkarni, 2007a,b)					
George Miller	3	0.9441(3)	0.9231(3)	0.9895(3)	0.7762(1)
Michael Collins	4	0.9777(4)	0.9861(4)	0.9889(4)	0.8357(1)
Richard Alston	2	0.9109(2)	0.9069(2)	0.9960(2)	0.7368(1)
Sarah Connor	2	0.9333(2)	0.9333(2)	0.9867(2)	0.7267(1)
Ted Pedersen	4	0.9820(4)	0.9670(4)	0.9850(4)	0.8378(1)
Bekkerman and McCallum's dataset (R.Bekkerman and A.McCallum, 2005)					
Adam Cheyer	2	0.9897(1)	0.9897(1)	0.9897(1)	0.9897(1)
Andrew McCallum	8	0.7447(3)	0.7340(3)	0.7766(4)	0.7660(1)
Andrew Ng	29	0.5172(7)	0.4943(6)	0.5747(5)	0.6437(1)
Bill Mark	8	0.6702(2)	0.6383(2)	0.8191(4)	0.6064(1)
David Israel	16	0.5217(3)	0.5217(4)	0.6739(4)	0.5435(1)
David Mulford	13	0.6702(3)	0.6809(2)	0.7553(4)	0.7128(1)
Fernando Pereira	19	0.4886(5)	0.4886(6)	0.6364(6)	0.5455(1)
Leslie Pack Kaelbling	2	0.9888(1)	0.9888(1)	0.9888(1)	0.9888(1)
Lynn Voss	26	0.4607(7)	0.4045(4)	0.6404(9)	0.5056(1)
Steve Hardt	6	0.8642(2)	0.8148(2)	0.8148(2)	0.8272(1)
Tom Mitchell	37	0.3478(9)	0.3696(8)	0.4891(13)	0.5870(1)
William Cohen	10	0.7955(2)	0.7841(2)	0.8295(3)	0.7955(1)
Overall Average		0.7732	0.7610	0.8288	0.7247

effect of the contextual similarity measure (Section 2.5) on the proposed method.

Overlap (Overlap coefficient-based clustering): This approach computes the similarity between two TEMs using the overlap (Simpson) coefficient between

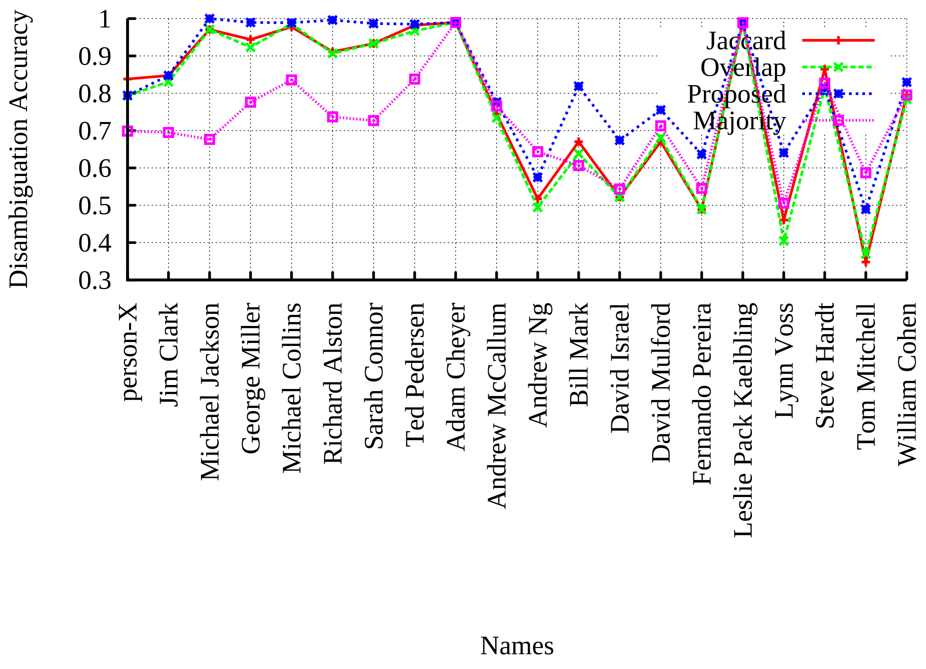

Figure 6: Comparing the proposed method against baselines.

them. Overlap coefficient between two sets A and B is defined as follows,

$$\text{Overlap} = \frac{|A \cap B|}{\min(|A|, |B|)}.$$

If one of the TEMs that we compare contains a lot of elements, then it is likely to share many elements in common with smaller TEMs. Overlap coefficient attempts to normalize the bias due to the difference in size (i.e., number of elements in a TEM) when computing similarity. Using the overlap coefficient as the similarity measure we perform group average agglomerative clustering with cluster stopping enabled to discriminate the namesakes. Overlap coefficient has been used in previous work on social network mining to measure the association between two names on the web (Matsuo et al., 2006b). Likewise the **Jaccard** baseline, **Overlap** baseline is expected to show the effect of using contextual similarity measure (Section 2.5) on the proposed method.

Proposed: This is the proposed namesake disambiguation algorithm. This approach uses the contextual similarity measure described in Section 2.5 to compute the similarity between TEMs. The clustering is performed using group average agglomerate clustering with cluster stopping enabled.

Majority: Majority sense clustering assigns all the contexts in a collection to the person that has the most number of contexts in the collection (dominant sense). Majority sense acts as a baseline for sense disambiguation. In personal name disambiguation on web, although there are lots of people with the same name, only a few are very popular. Assigning all the documents to this popular namesake can still report high clustering accuracies. For this reason, majority sense has been used as a baseline in previous work on name disambiguation (Fleischman and Hovy, 2004; Pedersen and Kulkarni, 2007b,a).

Disambiguation accuracies and the number of correctly identified namesakes (shown within brackets) for the different approaches are reported in Table 2. From Table 2, we see that the proposed method (**Proposed**) reports the highest disambiguation accuracy of 0.8288. Moreover, all three methods; **Jaccard**, **Overlap** and **Proposed** report significant improvements (pair-wise t-tests with $\alpha = 0.05$) over the **Majority** sense baseline. The proposed method, which uses contextual similarity, outperforms both Jaccard and Overlap baselines because those similarity measures are computed using exact matches between elements. They do not utilize the snippet-based contextual similarity described in section 2.5. Therefore, both Jaccard and Overlap baselines suffer from data sparseness (i.e. only few elements appear in common for two TEMs). It is interesting to note that the majority sense baseline has similar or better performance to the proposed method for the names *Adam Cheyer, Leslie Pack Kaelbling, Andrew McCallum, Steve Hardt, Tom Mitchell, Andrew Ng*. All those names appear in the dataset proposed by Bekkerman and McCallum (2005). The datasets for those names are highly skewed and the majority of the documents collected for a name belong to one namesake. For example, in the case of *Adam Cheyer*, 96 out of the total 97 documents are about the founder of Siri Inc. However, the majority sense baseline can only find one namesake and performs poorly when there are more than one popular namesake for an ambiguous personal name.

For collections *person-X, Michael Jackson, Richard Alston, Sarah Connor, George Miller, Michael Collins,Ted Pedersen* all three methods: Jaccard, Overlap, and Proposed, correctly identify all the different namesakes. Correct identification of the number of namesakes is essential because the selection of keywords depends on it. However, Jaccard and Overlap do not perform well with ambiguous names with lots of different namesakes, such as *Tom Mitchell* (37 namesakes), *Andrew Ng* (29 namesakes), *Lynn Voss* (26 namesakes) and *Fernando Pereira* (19 namesakes). In particular, *Tom Mitchell* is a very ambiguous name in the dataset containing 37 namesakes and only 15 out of 92 contexts is for the dominant sense (CMU professor). The quality and the length of text in a document affects the performance of term extraction and named-entity extraction. In particular, the statistical computations in the C-value method depends on the length (i.e. number of words) in a document. Moreover, the named-entity tagger, which is trained using news-

Table 3: The number of namesakes detected by the proposed method

Name	no. of namesakes	no. of clusters	detected	undetected
person-X	4	4	4	0
Jim Clark	8	8	3	5
Michael Jackson	2	2	2	0
Pedersen and Kulkarni’s dataset (Pedersen and Kulkarni, 2007a,b)				
George Miller	3	3	3	0
Michael Collins	4	4	4	0
Richard Alston	2	2	2	0
Sarah Connor	2	2	2	0
Ted Pedersen	4	4	4	0
Bekkerman and McCallum’s dataset (R.Bekkerman and A.McCallum, 2005)				
Adam Cheyer	2	2	1	1
Andrew McCallum	8	8	4	4
Andrew Ng	29	8	5	24
Bill Mark	8	4	4	4
David Israel	16	7	4	12
David Mulford	13	10	4	9
Fernando Pereira	19	13	6	13
Leslie Pack Kaelbling	2	2	1	1
Lynn Voss	26	12	9	17
Steve Hardt	6	5	2	4
Tom Mitchell	37	17	13	24
William Cohen	10	5	3	7

paper articles, produces invalid entities when tested on web documents, which are noisy. Better term and entity extraction methods can produce more accurate TEMs, thereby improving overall performance of the proposed method.

Table 3 compares the number of clusters produced by the proposed cluster stopping approach against the number of namesakes for a name in the gold standard. For each name in our dataset, Table 3 shows the number of namesakes in the gold standard dataset, the number of clusters produced by the proposed method, the number of namesakes correctly detected by the proposed method, and the number of undetected namesakes (i.e. total namesakes in the dataset minus no. of correctly detected namesakes). From Table 3 we see that for 11 out of the 20 names in our dataset the cluster stopping approach produces exactly the same number of clusters as the number of namesakes. Moreover, for 6 of those names, all namesakes are accurately detected. In particular, for Pedersen and Kulkarni's dataset, we have perfectly detected all namesakes.

There are eight different people with the name Jim Clark in our dataset and the proposed clustering algorithm created eight clusters. However, there were multiple clusters for the same person. Specifically, the Netscape CEO has four clusters, film editor has three clusters and the Formula One racing champion has a single cluster. In the case of Netscape CEO the four clusters correspond to different information related to the person: a book about the person (one cluster), Netscape company (two clusters), and Silicon graphics company in which he worked before he started Netscape (one cluster). We observed that when a person has multiple personalities on the Web, the proposed method creates a cluster for each of the personalities. Most of the documents that describes a particular personality of a person do not describe the other personalities. Considering the inter-site link structure can be useful to detect such pages that refer to the same individual. In our future work we plan to explore these possibilities.

Table 4 compares the proposed method against the best F-scores reported by Pedersen and Kulkarni (2007b) for the names in their dataset. To be able to directly compare with their results, we compute F-scores instead of accuracies in Table 4. From Table 4 we can see that the proposed method performs better than the method proposed by Pedersen and Kulkarni (2007b).

Table 4: Comparison with results reported by Pedersen and Kulkarni (2007b) using F-scores.

Name	Pedersen and Kulkarni	Proposed
George Miller	0.7587	0.9835
Michael Collins	0.9304	0.9866
Richard Alston	0.9960	0.9935
Sara Connor	0.9000	0.9779
Ted Pedersen	0.7658	0.9770

Table 5: Comparison with results reported by Bekkerman and McCallum (2005) using disambiguation accuracy

Name	Bekkerman and McCallum (2005)	Proposed
Adam Cheyer	0.6495	0.9897
Andrew McCallum	0.9787	0.7766
Andrew Ng	0.9080	0.5747
Bill Mark	0.8511	0.8191
David Israel	0.9456	0.6739
David Mulford	1.0000	0.7553
Fernando Pereira	0.7159	0.6364
Leslie Pack Kaelbling	0.9438	0.98888
Lynn Voss	0.9888	0.6404
Steve Hardt	0.3827	0.8148
Tom Mitchell	0.9348	0.4891
William Cohen	0.9545	0.8295

In Table 5, we compare the proposed method against the previous work on namesake disambiguation by Bekkerman and McCallum (2005). Bekkerman and McCallum consider the namesake disambiguation problem as a one of separating a set of given documents collected for an ambiguous personal name into two clusters: a cluster with all documents relevant to a particular namesake of the given name, and a cluster with all other documents. We compute disambiguation accuracy for those two clusters using Equation 9 for each name as shown in Table 5. However,

Table 6: Clusters for Michael Jackson

CLUSTER 1	CLUSTER 2
fan club	beer hunter
trial	ultimate beer FAQ
world network	christmas beer
superstar	great beer
new charity song	pilsner beer
neverland ranch	bavaria

Table 7: Clusters for Jim Clark

CLUSTER 1	CLUSTER 2
racing driver	entrepreneur
rally	story
scotsman	silicon valley
driving genius	CEO
scottish automobile racer	silicon graphics
british rally news	SGI/Netscape

it must be emphasized that their method can find *only one namesake* of the given ambiguous personal name. Moreover, they assume the availability of information regarding the social network of the person that they attempt to disambiguate. In contrast, the proposed method attempts to disambiguate *all namesakes* of a given personal name and does not require any information regarding the social network of a person. Despite the fact that the proposed method does not require external information regarding a namesake, such as his or her social network, and attempts to identify all namesakes, it has comparative performance with Bekkerman and McCallum's method.

Tables 6 and 7 shows the top ranking keywords extracted for Michael Jackson and Jim Clark. First cluster for Michael Jackson represents the singer while the second cluster stands for the expert on beer. The two Michael Jacksons are annotated with very different TEMs. Jim Clark the Formula one champion is represented by the first cluster in Table 7, whereas the second cluster stands for the founder of Netscape.

On an Intel Core2Duo 2.8GHz, 2GB RAM desktop, the proposed method requires approximately one minute to disambiguate a given name. The major portion of time is spent on querying a web search engine to compute contextual similarity. We cache the search results to reduce the amount of web accesses. The proposed method is used to disambiguate people in a social network system with more than $200,000$ people (Matsuo et al., 2006a).

5 Information Retrieval Task

We conduct an information retrieval task to evaluate the ability of the extracted keywords to uniquely identify an individual. We use a keyword k, selected for a namesake p, of an ambiguous name n to retrieve documents that contain k from a collection of documents D downloaded from the web using n as the query. If a document $d \in D$ contains the keyword k then we retrieve the document. We use the top 5 ranked keywords selected by the proposed method for all the namesakes it identifies for the 19 names in our gold standard datasets shown in Table 1 (person-X is not considered in this evaluation because it is not an actual name). If a keyword can accurately retrieve documents regarding a particular namesake, then such keywords are useful when searching for that person. We measure the ability of a keyword to retrieve documents related to a particular namesake using precision, recall and F1-score. The precision of a keyword k, precision(k), is defined as

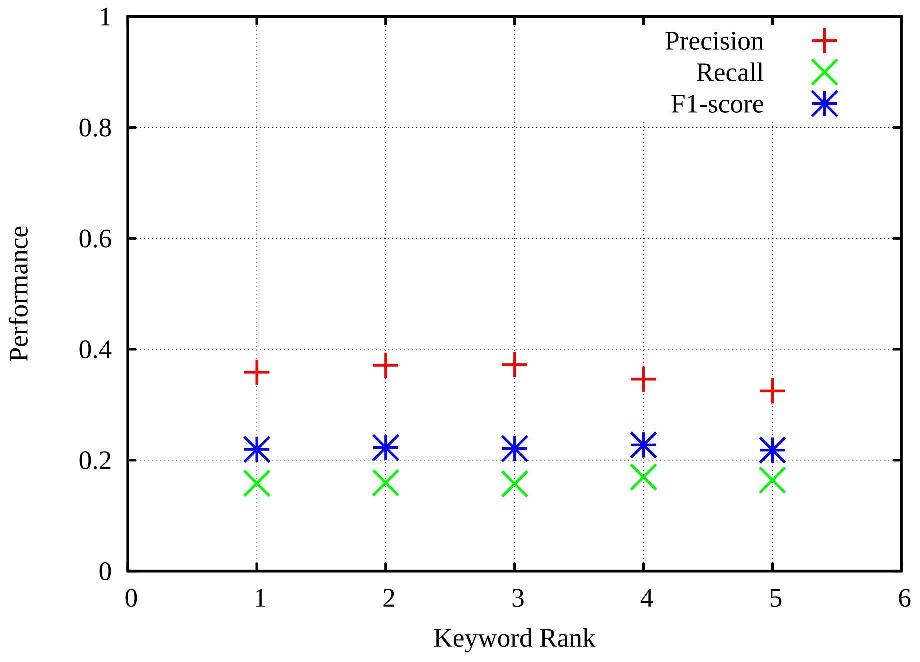

Figure 7: Performance vs. the keyword rank

follows,

$$\text{precision}(k) = \frac{\text{no. of documents that contain } k\text{, and belongs to } p}{\text{no. of documents that contain } k}. \tag{10}$$

Likewise, recall of a keyword k, recall(k), is defined as follows,

$$\text{recall}(k) = \frac{\text{no. of documents that contain } k\text{, and belongs to } p}{\text{no. of documents that belong to } p}. \tag{11}$$

The F1-score of a keyword k, F1-score(k), can then be computed as follows,

$$\text{F1-score}(k) = \frac{2 \times \text{precision}(k) \times \text{recall}(k)}{\text{precision}(k) + \text{recall}(k)}. \tag{12}$$

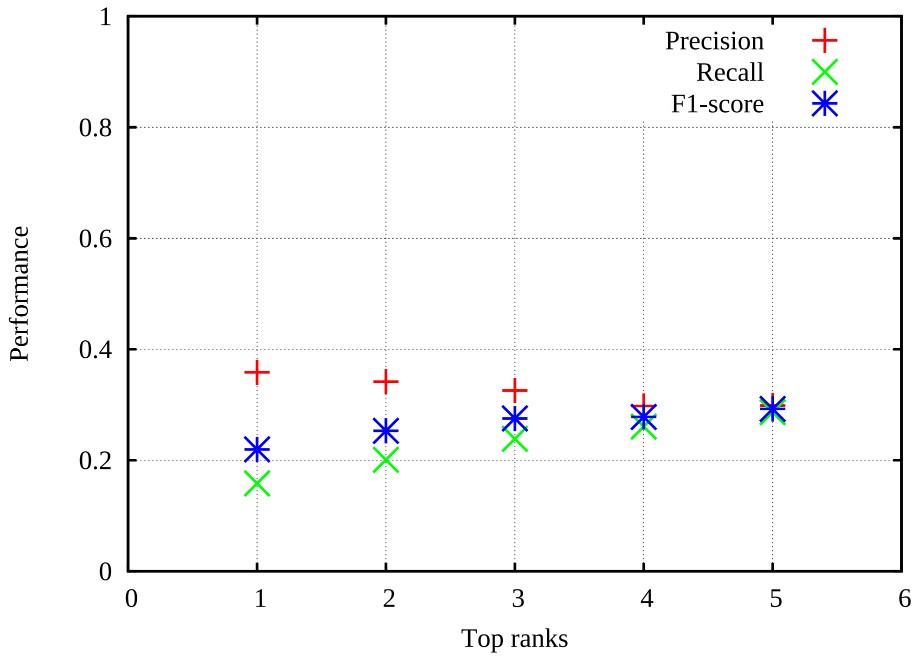

Figure 8: Performance vs. combinations of top ranking keywords

For the top 5 ranked keywords extracted for each detected namesake by the proposed method, we compute their precision, recall and F1-score using the above mentioned equations and take the average over the 19 names selected from the gold standard. Experimental results are shown in Figure 7. From Figure 7, we see that using any one of the top ranking keywords, on average, we obtain a precision of around 0.38. Moreover, a slight decrease in precision can be observed with the rank of the keywords used. However, the low recall (therefore the F1-score) indicates that using a single keyword alone is not sufficient to retrieve all the documents related to a namesake. A combination of top ranking keywords can be useful to improve recall. To evaluate the effect of using multiple keywords on retrieval performance, we combined the top r ranked keywords in a disjunctive (OR) query. Specifically, we retrieve a document for a namesake, if any one of the top k ranked keywords selected for that namesake appears in that document. We experiment with top 1-5 ranks as shown in Figure 8. From Figure 8 we see that combining the top ranked keywords indeed improve the recall. Although a slight drop in precision can be seen when we combine lower ranked keywords, overall, the F1-score improves as a result of the gain in recall. Ideally, one would like to minimize the drop in precision while maximizing the recall in an information retrieval task. In the case of namesake disambiguation we can model this as a problem of searching for the optimum combination of keyphrases (queries) over the space spanned by all terms and entities extracted and clustered for a particular namesake. Because the number of keyphrase combinations grows exponentially with the size a keyphrase cluster, we must resort to heuristic approaches that cut-down the search space. For example, one could first rank keyphrases in a cluster in the descending order of their page counts in a web search engine and present to a user. We plan to explore the possibility of using such ranked lists of keyphrase suggestions for the task of namesake disambiguation in our future work.

As a specific example of how the keywords extracted by the proposed method can be used in a real-world web search scenario, we search Google for the namesakes of the ambiguous personal name *Jim Clark* using the extracted keywords. We first search Google only using the name *Jim Clark*. We then modify the query by including a keyword selected for a particular namesake. We manually check the top 100 ranked search results and determine how many results are relevant for the namesake that we are searching for. Experimental results are summarized in Table 8.

Table 8: Effectiveness of the extracted keywords to identify an individual on the web.

Keyword	person-1	person-2	others	Hits
NONE	41	26	33	1,080,000
racing driver	81	1	18	22,500
rally	42	0	58	82,200
scotsman	67	0	33	16,500
entrepreneur	1	74	25	28,000
story	17	53	30	186,000
silicon valley	0	81	19	46,800

In Table 8 we classify Google search results into three categories. "person-1" is the formula one racing world champion, "person-2" is the founder of Netscape, and "other" category contains remainder of the pages that we could not classify to previous two groups (some of these pages were on other namesakes, and some were not sufficiently detailed to properly classify). We first searched Google without adding any keywords to the ambiguous name. Including the keywords *rally* and *scotsman*, which are selected from the cluster for *Jim Clark* the formula one champion, return no results for the other popular namesake. Likewise, the keywords *entrepreneur* and *silicon valley* yield results largely for the founder of Netscape. However, the keyword *story* returns results for both namesakes. A close investigation revealed that, the keyword *story* is extracted from the title of the book "The New New Thing: A Silicon Valley Story", a book on the founder of Netscape. This is an example where the term extraction has failed to detect the title of the book. The word *story* is a polysemous which has various senses. As can be seen from Table 8, it is inadequate to discriminate the two namesakes under consideration. A named entity tagger that covers numerous entity types such as products can be potentially useful to overcome this problem. In future, we plan to explore these possibilities.

6 Related Work

Personal name disambiguation is closely related to the Word Sense Disambiguation (WSD) (Schutze, 1998; McCarthy et al., 2004; Snyder and Palmer, 2004; Agirre and Soroa, 2007) problem which has been studied extensively in Natural Language Processing. In WSD, the objective is to predict the correct sense of an ambiguous word that appears in a text. The different senses of the ambiguous word are presented to the disambiguation algorithm. Alternatively, in word sense discrimination problem we must determine all the different senses a word can have. Pantel and Lin (2002) proposed the clustering by committee (CBC) algorithm to automatically discover the word senses from text. Their algorithm first discovers a set of tight clusters called committees that are well scattered in the similarity space. A cluster is represented by a feature vector that is computed as the centroid of the members of a committee. Next, each word is assigned to their most similar clusters, where similarity is computed using the cosine coefficient. They remove the overlapping features in a cluster to prevent discovering duplicate senses. Similarly, in the person name disambiguation problem, we must find the different people who have the same given ambiguous name. Here, each person with the ambiguous name can be viewed as a *sense* for the ambiguous name.

In large citation databases, author names can easily become ambiguous. In order to efficiently search for a particular publication, one must first disambiguate the author names. Besides the author names, a citation usually contains information such as the title of the publication, conference or journal name, year of publication and the number of pages. Such information have been utilized in previous work on citation disambiguation to design both supervised and unsupervised algorithms (Han et al., 2005; Kanani et al., 2007; Song et al., 2007; Huang et al., 2006b,a). Pallika et al. (2007) formulates the citation disambiguation problem as one of graph partitioning with discriminatively-trained edge weights, and incorporate web information either as additional features or as additional nodes in the graph. However, unlike in a citation which has a short and semi-structured format, in personal name disambiguation one must first extract salient information related to the ambiguous name from the Web. The term-entity models introduced in Section 2.3 attempt to represent the salient information for a particular person using terms and named entities. This extra step involved when disambiguating names on the Web makes it a more challenging task.

Research on multi-document personal name resolution (Bagga and Baldwin, 1998; Mann and D.Yarowsky, 2003; Fleischman and Hovy, 2004; Ravin and Kaiz, 1999) focuses on the related problem of determining if two instances with the same name and from different documents refer to the same individual. Bagga and Baldwin (1998) first perform within-document co-reference resolution to form co-reference chains for each entity in each document. They then use the text surrounding each reference chain to create summaries about each entity in each document. These summaries are then converted to a bag-of-words feature vector and are clustered using the standard vector space model, often employed in information retrieval. The use of simplistic bag-of-words clustering is an inherently limiting aspect of their methodology. On the other hand, Mann and Yarowsky (2003) propose a richer document representation involving automatically extracted features. However, their clustering technique can be basically used only for separating two people with the same name. Fleischman and Hovy (2004) constructs a maximum entropy classifier to learn distances between documents that are subsequently clustered. However, their method requires a large training set.

Pedersen et al. (2005; 2006) propose an unsupervised approach to resolve name ambiguity by clustering the instances of a given name into groups, each of which is associated with a unique entity. They use statistically significant bigrams that occur in the same context as ambiguous name to create feature vectors that represent the context of an ambiguous name. Next, they create a co-occurrence matrix where the rows and columns represent the first and second words in bigrams, and the cells contain their log-likelihood scores. Then they represent each of the contexts in which an ambiguous name appears with a second order context vector. Second order context vectors are created by taking the average of the vectors from the co-occurrence matrix associated with the words that make up each context. Next, singular value decomposition (SVD) is performed on this matrix to reduce the dimensionality of the matrix. Finally, the different individuals with the given ambiguous name are grouped into clusters using a repeated bisections algorithm. Performance of their algorithm is evaluated using pseudo-names by following the method we described in Section 3.

Li et al. (2005) propose two approaches to disambiguate entities in a set of documents: a supervised pairwise classifier and an unsupervised generative model. In the supervised approach they train a pairwise local classifier to determine whether two mentions of a name represent the same real world entity. Next, using the trained pairwise classifier as the similarity measure, they perform a global clustering on a set of documents to identify the different people with the same name. In the unsupervised approach they define a global generative model of documents over names of entities. The proposed generative model has three components: a joint distribution over entities, an author model, and an appearance model. The joint distribution over entities attempts to capture the associations between entities. For example, a document that mentions “President Kennedy” is more likely to mention “Oswald” or “White House” than “Roger Clemens”. The contextual similarity measure described in Section 2.5 in this paper computes such associations between entities using information retrieved from a web search engine. The author model assumes that at least one mention of an entity in a document is easily identifiable and generates other mentions of the entity in a document using the appearance model. However, they do not assign keywords to the different people with an ambiguous name.

Bekkerman and McCallum (2005) present two unsupervised methods for finding web pages referring to a particular person: one based on link structure and another using Agglomerative/Conglomerative Double Clustering (A/CDC). Their scenario focuses on simultaneously disambiguating an existing social network of people, who are closely related. Therefore, their method cannot be applied to disambiguate an individual whose social network (for example, friends, colleagues) is not known.

Guha and Grag (2004) present a re-ranking algorithm to disambiguate people. The algorithm requires a user to select one of the returned pages as a starting point. Then, through comparing the person descriptions, the algorithm re-ranks the entire search results in such a way that pages referring to the same person described in the user-selected page are ranked higher. A user needs to browse the documents in order to find which one matches the user’s intended referent, which puts an extra burden on the user.

7 Conclusion

We proposed an unsupervised method to automatically extract keywords from the web, and annotate people with ambiguous personal names. Terms and named-entities are extracted from name-contexts to create term-entity models. Then group average agglomerative clustering is used to cluster the term-entity models. The proposed method is evaluated on a dataset covering 20 ambiguous names, including names from previous work on web-based personal name disambiguation. We proposed a method to determine the number of clusters using cluster quality. Experimental results showed improved performances over the baselines and previous work. We selected unique keywords from the clusters and annotated the different people with the ambiguous name. Extracted keywords are useful to retrieve information regarding a particular namesake.

There are many potential future research directions of this work. Ambiguity is not limited for personal names but exists in other types of named entities such as, locations and products. It would be interesting to apply the proposed method in these areas to find useful keywords to uniquely identify an entity. Term entity models can be further improved by experimenting with other term and entity extraction tools. In particular, the usability of general purpose keyword extraction algorithms must also be explored. There is a wide array of clustering algorithms proposed for word or document clustering. Experimenting with different clustering algorithms will provide valuable insights as to which clustering algorithms are useful for the current task. Determining the number of different namesakes for a given particular name is important when disambiguating person names on the Web. The ability to accurately predict the number of namesakes for a particular name is also useful when determining the number of clusters. Detecting and mapping the multiple Web appearances of a person is an important task to further improve the quality of clustering. Using link structure between Web sites and information about the social network of a person can be useful to detect such multiple appearances. Selecting keywords to expand name queries from clusters is particularly important to improve recall in information retrieval. Our experimental results suggest that a combination of multiple keywords is more useful to determine a particular namesake. We hope that our work will provide a foundation upon which others can explore the above-mentioned research directions.

Acknowledgements

We would like to extend our sincere thanks to the anonymous reviewers who provided invaluable comments that have improved the quality of the paper.

References

Eneko Agirre and Aitor Soroa. Semeval-2007 task 02: Evaluating word sense induction and discrimination systems. In 7-12, editor, *International Workshop on Semantic Evaluations (SemEval-2007)*, 2007.

J. Artiles, J. Gonzalo, and F. Verdejo. A testbed for people searching strategies in the WWW. In *Proc. of SIGIR*, pages 569–570, 2005.

A. Bagga and B. Baldwin. Entity-based cross-document coreferencing using the vector space model. In *Proc. of COLING'98*, pages 79–85, 1998.

S. Banerjee and T. Pedersen. An adapted lesk algorithm for word sense disambiguation using word net. In *Proc. of the third international conference on computational linguistics and intelligent text processing*, pages 136–145, 2002.

P. Cimano, S. Handschuh, and S. Staab. Towards the self-annotating web. In *Proc. of 13th International World Wide Web Conference (WWW)*, 2004.

D.R. Cutting, J.O. Pedersen, D. Karger, and J.W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proc. SIGIR '92*, pages 318–329, 1992.

J. Euzenat. Semantic precision and recall for ontology alignment evaluation. In *Proc. of International Joint Conferences on Artificial Intelligence (IJCAI'07)*, pages 348–353, 2007.

M.B. Fleischman and E. Hovy. Multi-document person name resolution. In *Proc. of 42nd Annual Meeting of the Association for Computational Linguistics (ACL), Reference Resolution Workshop*, 2004.

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.

R. Guha and A. Garg. Disambiguating people in search. In *Stanford University*, 2004.

W.V. Hage, H. Kolib, and G. Schreiber. A method for learning part-whole relations. In *Proc. of 5th International Semantic Web Conferences*, 2006.

H. Han, H. Zha, and C.L. Giles. Name disambiguation in author citations using a k-wayspectral clustering method. In *Proc. of the International Conference on Digital Libraries*, 2005.

J. Huang, S. Ertekin, and C.L. Giles. Fast author name disambiguation in citeseer. *ISI Technical Report*, 2006a.

J. Huang, S. Ertekin, and C.L. Giles. Efficient name disambiguation for large scale databases. In *Proc. of 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 536–544, 2006b.

Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *The Knowladge Engineering Review*, 18:1–31, 2003.

P. Kanani, A. McCallum, and C. Pal. Improving author coreference by resource-bound information gathering from the web. In *Proc. of International Joint Conferences on Artificial Intelligence (IJCAI'07)*, pages 429–434, 2007.

R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad, and spectral. In *Proc. of the 41st Annual Symposium on the Foundation of Computer Science*, pages 367–380, 2000.

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.

G.S. Mann and D.Yarowsky. Unsupervised personal name disambiguation. In *Proceedings of CoNLL-2003*, pages 33–40, 2003.

Christopher D. Manning and Hinrich Schutze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts London, England, 2 edition, 2002.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

Y. Matsuo, M. Hamasaki, Y. Nakamura, T. Nishimura, K. Hashidaand H. Takeda, J. Mori, D. Bollegala, and M. Ishizuka. Spinning multiple social networks for semantic web. In *Proc. of the American Association for Artificial Intelligence*, 2006a.

Y. Matsuo, J. Mori, and M. Hamasaki. POLYPHONET: An advanced social network extraction system. In *Proc. of the World Wide Web Conference (WWW)*, 2006b.

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

P. Mika. Bootstrapping the foaf-web: and experiment in social networking network mining. In *Proc. of 1st Workshop on Friend of a Friend, Social Networking and the Semantic Web*, 2004.

P. Mika. Flink: Semantic web technology for the extraction and analysis of social networks. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3:211–223, 2005.

J. Novak, P. Raghavan, and A. Tomkins. Anti-aliasing on the web. In *Proc. of 13th International World Wide Web Conference (WWW)*, pages 30–39, 2004.

N.F. Noy and M.A. Musen. An algorithm for merging and aligning ontologies: Automation and tool support. In *Proc. of the AAAI workshop on ontology management*, 1999.

Patrick Pantel and Dekang Lin. Discovering word senses from text. In *KDD 2002*, pages 613–619, 2002.

T. Pedersen and A. Kulkarni. Unsupervised discrimination of person names in web contexts. In *Proc. of Eighth International Conference on Intelligent Text Processing and Computational Linguistics*, pages 18–24, 2007a.

T. Pedersen and A. Kulkarni. Discovering identities in web contexts with unsupervised clustering. In *Proc. of the IJCAI'07 Workshop for Noisy Unstructured Text Data*, 2007b.

T. Pedersen, A. Purandare, and A. Kulkarni. Name discrimination by clustering similar contexts. In *Proc. of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics*, 2005.

T. Pedersen, A. Kulkarni, R. Angheluta, Z. Kozareva, and T. Solorio. An unsupervised language independent method of name discrimination using second order co-occurance features. In *Proc. of the Seventh International Conference on Intelligent Text Processing andComputational Linguistics (CICLing)*, 2006.

L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL'09)*, Jun 2009.

Y. Ravin and Z. Kaiz. Is hilary rodham clinton the president? disambiguating names across documents. In *Proc. of the ACL '99 Workshop on Coreference and its Applications*, 1999.

R.Bekkerman and A.McCallum. Disambiguating web appearances of people in a social network. In *Proc. of the World Wide Web Conference (WWW)*, pages 463–470, 2005.

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'05)*, 2005.

G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill Inc., New York, NY, 1986.

H. Schutze. Automatic word sense discrimination. *Computational Linguistics*, 24 (1):97–123, 1998.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

Benjamin Snyder and Martha Palmer. The english all-words task. In *International Workshop on Semantic Evaluations (SemEval-2004)*, 2004.

Y. Song, J. Huang, I.G. Councill, J. Li, and C.L. Giles. Generative models for name disambiguation. In *Proc. of International World Wide Web Conference (WWW)*, pages 1163–1164, 2007.