Using a Knowledge Base to Disambiguate Personal Name in Web Search Results

Quang Minh Vu
The University of Tokyo
Tokyo, 113-8656 Japan
vuminh@nii.ac.jp

Tomonari Masada
National Institute of Information
Tokyo, 101-8430 Japan
masada@nii.ac.jp

Atsuhiro Takasu
National Institute of Information
Tokyo, 101-8430 Japan
takasu@nii.ac.jp

Jun Adachi
National Institute of Information
Tokyo, 101-8430 Japan
adachi@nii.ac.jp

ABSTRACT

Results of queries by personal names often contain documents related to several people because of the namesake problem. In order to differentiate documents related to different people, an effective method is needed to measure document similarities and to find documents related to the same person. Some previous researchers have used the vector space model or have tried to extract common named entities for measuring similarities. We propose a new method that uses Web directories as a knowledge base to find shared contexts in document pairs and uses the measurement of shared contexts to determine similarities between document pairs. Experimental results show that our proposed method outperforms the vector space model method and the named entity recognition method.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*clustering, information filtering*

Keywords

Personal name searching, name disambiguation, document similarity

1. INTRODUCTION

The prevalence of the Internet in daily life has made the World Wide Web(WWW) a huge resource for information. Information in the WWW comes from many sources, including websites of companies, organizations, communities, and personal homepages, etc. In such a heterogeneous environment, information about one person tends to be scattered in various places. Suppose we want to search for information about a person. We may send a query containing his/her name to a search engine and get a set of documents containing his name. However, because of the name sake problem the set of documents may contain documents related to several people. For example, the top 100 pages from the Google search engine for the query *"Jim Clark"* contain at least eight different *Jim Clarks*. Among them, two people with the largest number of pages are *Jim Clark* the Formula one world champion (46 pages) and *Jim Clark* the founder of Netscape (26 pages). It would be more easily for end-users to find their interested person, if we can separate documents of different people.

Our reseach objective is to determine documents related to the same person and to group them together, so that end-users can get their desired information more easily. When determining documents related to the same person, correctly measuring the closeness between pairs of documents is very crucial because it directly affects determining performance.

In some previous research [1, 2, 3, 9, 10, 11, 12], several methods have been proposed to determine similarities between document pairs. We propose a new method to effectively measure document pair similarities. We use several sets of documents on several topics as intermediate documents to find out shared contexts in document pairs and measure the weight of these shared contexts. These sets of documents can be regarded as a knowledge base, an information source on various topics. We chose to use Web directories for the knowledge base because they are easy to get from the Web. We used the Dmoz Web directories [5] in our research.

The rest of this paper is organized as follows. In section 2, we summarize the related research. Then in section 3, we propose a new method to measure similarities among documents. We present our idea for measuring similarities and give details of the calculation process. Experiment results and comparisons with other methods are given in section 4. Finally, section 5 concludes our work.

2. PROBLEM STATEMENT AND RELATED WORKS

Suppose we send a query A to a search engine and get a set of documents $S = \{d_1, d_2, ..., d_n\}$ and documents in S are about k people $P_1, P_2, ..., P_k$. Our task is to group n documents in S into groups so that each group contains documents related to only one person. As we have k people, the result should contain k groups, each group corresponds to one person.

In [9], Bagga and Baldwin solved the problem of personal name coreference in news articles. They used the vector space model (VSM) [7] to measure similarities between articles. A person appearing in some news articles tends to be related to one event, so that person's relevant documents tend to discuss only one story. Therefore, the VSM model measures similarities very well. However, for people in the Web, they may appear with more than one event. Therefore, although their documents are about the same general topic, their specific topics may deffer. In such a case, where

SAC'07 March 11-15, 2007, Seoul, Korea

relationship between pages are weak, the VSM model may not measure similarities well.

In [1], Pederson et al. calculated words' context vectors using word co-occurrence information. Each document was represented by a context vector using the method *second order context vectors* [17] (they calculated the average vector of all context vectors in a document). They used the documents' context vectors to cluster the documents into groups. However, this approach is suitable only for people whose names appear in a large number of documents because calculation of words's context vectors requires word co-occurrence information from a large number of documents.

In [2], Bekkerman and McCallumn proposed a method to extract a group of people simultaneously. People in this group are related to one another so their relevant web pages may share a same topic and be connected. The researches proposed two methods to extract a group of people: one that uses link information in web pages and another that uses the Agglomerative Conglomerative Double Clustering (A/CDC) [2] clustering algorithm to group together web pages with the same topic. The use of this method is limited because when we search for a person on the Internet, we may not know about his social network in advance.

Extraction of personal profiles has been used in some other researches [11, 3, 12]. In [11], Mann et al. used the pattern matching method to extract personal profiles (birthday, birth place, occupation, etc). In [3], Guha et al. used databases like DBLP [14], Amazon [15] to extract books' author names and research keywords. In [12], Wan et al. used natural language processing techniques to extract named entities in documents.

Our method can be seen as an improvement on the vector space model method: we give more weight to terms strongly related with the topic of document. To do this, we look for other documents that are close in topic to the current document and count the keywords' frequencies in the other documents.

3. SIMILARITY VIA KNOWLEDGE BASE (SKB)

3.1 Measuring document similarities

The vector space model (VSM) method is a traditional and basic method used to measure similarities between documents. It measures the weight of terms based on the number of times a term occurs in a document (term frequency) and the number of documents that contain the term (document frequency). The VSM method works well when related documents discuss the same specific topic. When documents are about the same specific topic, they share many common terms, so the document similarities calculated by VSM model become high. However, a person in the web may appear in different circumstances. Therefore, although these documents may be about the same general topic, their specific topics may be different. For example, a computer scientist may have his publications on several different specific topics under the same general computer science topic. In such a case, common terms among documents are very few. Moreover, information about a people may appear in only few lines in a web page so the text relates to him is short. The shortness of relevant texts also makes common terms being few. When the number of common terms is few, similarities calculated by VSM are not so effective for differentiating documents relevant to different people.

We propose a new method to boost the weight of important terms in order to measure document similarities when the number of common terms is small. Suppose that we have a set of documents that are about topics close to those of a pair of documents. In the pair of documents, because of the small number of documents and the shortness of documents' length, keywords related with the topic may not appear more frequently than other words. However, in the set of documents of the same topic, keywords appear more frequently than other words. It is reasonable to assume that keywords in the document pair appear as frequently as they do in the set of documents if the relevant texts are longer. Therefore, we use the frequencies of keywords in the set of documents to modify the frequencies of keywords in the pair of documents.

This approach requires external sets of documents, so we prepared some sets of documents on some topics. We call these sets the knowledge base. Then we used this knowledge base to find document sets that are close in topic to a pair of documents and modify their keywords' term frequencies. We call our method "Similarity via Knowledge Base" (**SKB**).

Knowledge base used in our SKB method can be seen as a kind of training data. In a research of name disambiguation in citation data[18], the authors used a supervised learning method. They prepared training documents for every person in advance and used these documents to train the classifier. However, this approach is infeasible for people in the web because preparing training documents for every person in the web is an impossible task. On the other hand, our approach of using knowledge base is feasible because we only prepare documents on several topics. Also, this preparation is easier because we may use already existing document categories (e.g web directories) as knowledge base.

3.2 Calculation algorithm

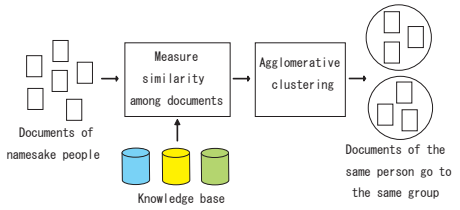

Figure 1: Name disambiguation system

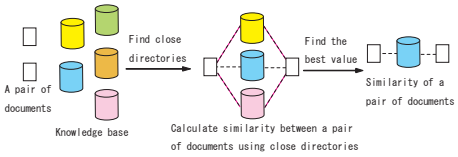

Figure 2: Measure similarity using a knowledge base

Figure 1 and Figure 2 show the overview of the name disambiguation system using the knowledge base. It has three steps as follows.

1. Preprocess documents.
2. Find directories from the knowledge base that are close in topic to a document and measure weight of terms using these directories.

3. Measure similarity between a pair of documents using knowledge base.

3.2.1 Preprocessing

In this step we remove stop words and use the Porter algorithm[16] to stem words to their root forms. As web pages are noisy information source, only information around personal name should be considered as information related to people. Therefore, we extract only 50 terms in before and 50 terms after a personal name to create a bag of words representing that person.

3.2.2 Finding close directories and measuring term weights

The traditional VSM uses the following formulas to calculate term weights.

$$tf\_idf(t,d) \quad = \quad tf(t,d) \times log(\frac{N}{df}) \tag{1}$$

Here $tf(t,d)$ is the frequency of term t in the document d. We use the TREC-Web collection[13] to calculate the inverse document frequency $log(\frac{N}{df})$. N is the number of documents in the TREC-Web collection.

Suppose that a directory Dir is close in topic to the document d. Then the distribution of a topic's keyword term t in d and Dir should be similar if d is long enough. Therefore, the larger a term t's weight $tf\_idf(t, Dir)$ is, the larger that term t's importance in the document $weight(t,d)$ should be.

We may use $tf\_idf(t, Dir)$ in place of $tf\_idf(t,d)$, but we still want to keep the importance of $tf\_idf(t,d)$, so we choose to use the geometric mean: $weight(t,d,Dir) \propto \sqrt{tf\_idf(t,d) \times tf\_idf(t,Dir)}$

We have many directories and we want to make this importance comparable among them, so we normalize this importance by dividing it by the size of each directory. Finally, we use the following formula to compute terms' weight.

$$weight(t,d,Dir) = \sqrt{\frac{tf\_idf(t,d) \times tf\_idf(t,Dir)}{length(Dir)}} \tag{2}$$

Formula 2 is used to find directories that are close in topic to document d. We use the following formula to calculate similarity between a document d and a directory Dir.

$$SIM(d,Dir) = \sum_{t \in d \cap Dir} weight(t,d,Dir) \tag{3}$$

Then we select the top k directories with the highest $SIM(d,Dir)$ values and call these k directories $Dir_1, Dir_2, ...$ Dir_k as representative directories. The common terms between d and Dir_i are called representative of d via directory Dir_i and denoted by $Representative(d, Dir_i)$.

3.2.3 Measuring document pair similarities

Let a pair of documents to be measured be (d_1, d_2). For each directory Dir in the knowledge base, we represent d_1, d_2 via directory Dir using common terms between documents and directory: $Representative(d_1, Dir)$, $Representative(d_2, Dir)$. The weight of representative terms is calculated using formula 2. Then, we use the weight of common representative terms to calculate the similarity between the document pair (d_1, d_2) as follows.

$$SIM(d_1,d_2,Dir) = \sum_t weight(t,d_1,Dir) \times weight(t,d_2,Dir) \tag{4}$$

where $t \in Representative(d_1, Dir) \cap Representative(d_2, Dir)$.

$$SIM(d_1,d_2) = \max_i SIM(d_1,d_2,Dir_i) \tag{5}$$

3.3 Grouping documents

Suppose we have two document sets, and each set has only documents related to the same person. If these two document sets are similar enough to each other, both of them may be about the same person, so we merge them together. The similarity between two sets of documents is calculated as follows.

$$SIM(C_1,C_2) = \frac{\sum_{d_i \in C_1 \ , d_j \in C_2} SIM(d_i,d_j)}{|C_1| \times |C_2|} \tag{6}$$

At the initial step each document itself forms a singleton cluster. Then we consecutively merge the closest cluster pair until the number of clusters is small enough. The details of clustering algorithm are as follows.

Procedure ClusterDocument()
1: At initial status, each document forms a singleton cluster
2: Calculate similarity between all clusters
3: **while** (number of clusters $> N_{threshold}$) **do**
4: Find the pair of clusters (C_1, C_2) with the maximum similarity
5: Merge C_1, C_2 to form a new cluster C_{new}
6: Update similarity between C_{new} and other clusters C_i
7: **end while**
8: **return** a set of clusters

Here $N_{threshold}$ is tuned using a training data set.

4. EXPERIMENT

4.1 Baseline methods

We chose two methods to compare with our method as baseline methods: the vector space model (VSM) method and the named entity recognition (NER) method.

4.1.1 Vector space model method

In the VSM method, we do preprocessing same as preprocessing in our SKB method: we remove stop words and select 50 words before and 50 words after each personal query name. Using this bag of words we construct a document vector whose constituents are $tf\_idf(t,d)$ values of all words in the bag calculated using equation 1. We use the inner vector product of document vectors as the similarity measurement of document pairs.

4.1.2 Named entity recognition method

In[12], the authors use named entities recognition (NER) method for the measurement of document similarities. We use the LingPipe software[4] (a named entity extraction tool) to extract named entities inside a document and build a document vector using these named entities. Constituents of vector are binary value (1 if a named entity appear in the document, 0 otherwise). The inner vector product between document vectors is used for similarity measurement.

Table 1: Data sets

Field	Name
Computer science	Adachi Jun, Sakai Shuichi Tanaka Katsumi, John D. Lafferty Tom M. Mitchell, Andrew McCallum
Physics	Paul G. Hewitt, Edwin F. Taylor Frank Bridge, Kenneth W. Ford Paul W. Zitzewitz, Michael A. Dubson
Medicine	Scott Hammer, Thomas F. Patterson Henry F. Chambers, David C. Hooper Michele L. Pearson, Lindsay E. Nicolle
History	John M. Roberts, David Reynolds Thomas A. Brady, William L. Cleveland Thomas E. Woods, Peter Haugen

4.2 Data sets

4.2.1 *Knowledge base directories*

We chose directories in dmoz.org [5] for knowledge base directories. We chose 56 specific directories from various general topics including art, business, computer, games, history, home, news, recreation, science, shopping, society and sports. Each directory contained about 40 to 50 documents.

4.2.2 *Test sets*

We got from the Google search engine [6] documents of researchers in four fields: computer science, physics, medicine and history. In each research field we chose six people as shown in table 1.

For each person we selected top 100 documents from the searching results. After removing the non-html documents, each collection had about 75 to 90 documents, among them about 20 to 60 documents were documents related to the same person.

We divided 24 collections into two sets: a training set and a test set. The training set has 16 collections (4 collections per each field $\times$ 4 fields) and the test set has eight collections (2 collections per each field $\times$ 4 fields). In each set, we created pseudo namesake data by mixing together two collections: two each from two people in different research fields. This yielded $4 \times 4 \times \binom{4}{2} = 96$ pseudo namesake data for the training set and $2 \times 2 \times \binom{4}{2} = 24$ pseudo namesake data for the test set. The training set is for tuning the number of clusters and the test set is for verifying and comparing the performance of each method.

4.3 Evaluation of clustering performance

We evaluate the performance of the clustering results as follows. From the clustering results, we first remove clusters whose size is less than or equal to three. For each remaining cluster, we choose the person who has the largest number of documents in the cluster and label all documents in the cluster with the label of that person. Then we calculate the precision (P), recall (R) of labeled documents using the following equations. We also calculate the harmonic mean ($F_{measure}$) of P and R.

$$P = \frac{\text{Number of documents correctly labeled}}{\text{Number of documents labeled}} \quad (7)$$

$$R = \frac{\text{Number of documents correctly labeled}}{\text{Number of documents should be labeled}} \quad (8)$$

$$F_{measure} = \frac{2PR}{P+R} \quad (9)$$

4.4 Experimental results

4.4.1 *Performances of methods*

We varied the stopping condition of the clustering algorithm (i.e. the number of clusters) and measure the values $P, R,$ and $F_{measure}$. Figure 3,4, and 5 show the results for three methods: VSM, NER, and SKB, respectively. Figure 6 shows a comparison of the three methods in terms of F_measure value. We also counted the number of remaining large clusters (clusters with size larger than 3) for all sets and took the average (Figure 7).

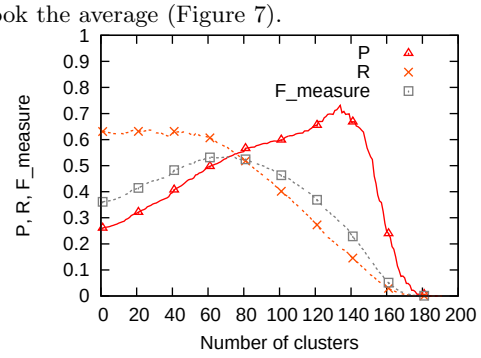

Figure 3: Performance of VSM method

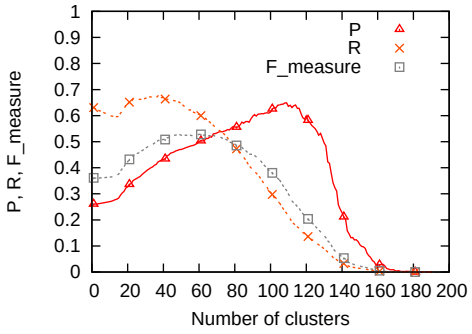

Figure 4: Performance of NER method

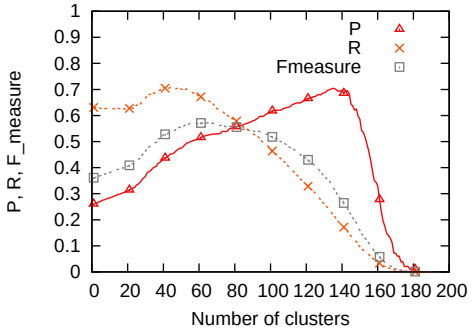

Figure 5: Performance of SKB method

4.4.2 *Tuning the number of clusters*

We used the training set to find the best number of clusters for the stopping condition of the clustering algorithm. The results are 71, 60, and 63, for VSM, NER, and SKB, respectively. At these thresholds, VSM, NER, and SKB achieved the F_measure values of 53.3%, 52.9%, and 57.2%. We applied these number of clusters for the stopping condition of the clustering algorithm in the testing experiment.

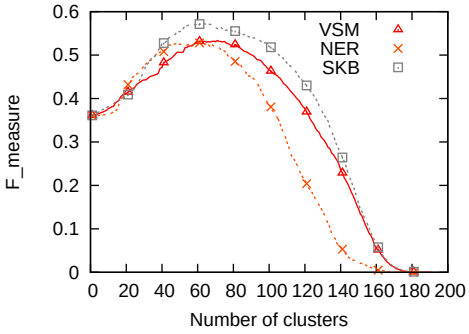

Figure 6: Comparison of F_measure among methods

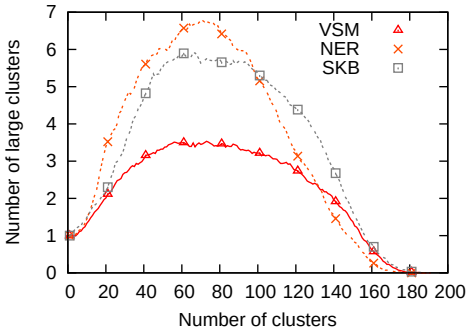

Figure 7: Number of large clusters

Table 2 shows the performance of the training and test sets in terms of $F_{measure}$ value.

4.5 Discussion

4.5.1 Comparison among methods

As we can see from Figures 3,4,5, and 6, the best $F_{measure}$ value of SKB is 57.2%, which is better than the $F_{measure}$ values of VSM (53.3%) and NER (52.9%). In the test set, the SKB also outperformed VSM and NER in term of $F_{measure}$: SKB achieved a value of 52.8%, compared with 45.1% of VSM and 49.6% of NER. We also investigated the number of large clusters in the result. As shown in the Figure 7, the number of large clusters for each method varies in the range from 3 to 7 when the numbers of all clusters are around $N_{threshold}$, which is suitable for practical use.

4.5.2 Computation complexity

Let n and M be the number of documents and the number of directories, respectively. In the VSM method, we have to calculate similarity between every document pair. Therefore, the computation complexity of the VSM method is $O(n^2)$. In our SKB method, we have to calculate similarity between every document pair using $2k$ top directories and have to calculate similarity between every document and every directory. Therefore, the computation complexity of the SKB method is $O(2k \times n^2 + M \times n)$. As k and M are constants, the computation complexity is $O(n^2)$. This complexity is of the same order as the VSM method, but it is more expensive than the VSM method by a constant factor.

Table 2: Tuning parameters and testing results

Method	$N_{threshold}$	Training set	Test set
VSM	71	53.3%	45.1%
NER	60	52.9%	49.6%
SKB	63	**57.2%**	**52.8%**

5. CONCLUSION

In this research we focused on the problem of personal name disambiguation in web search results. To solve this problem, we have proposed a new method to measure the similarities between documents: similarity via knowledge base (**SKB**). Our method uses a knowledge base to find out topic words, which are important keywords in documents, in order to find out shared contexts of documents and to more easily calculate the weight of the shared contexts. Then, we use these similarity results for the agglomerative clustering to group related documents together. Our SKB method performed better than two traditional methods: the vector space model (VSM) method and the named entity recogniton (NER) method. In the future, we will use our SKB method in cooperation with other clustering techniques to improve the grouping performance. We will also try to reduce the calculation complexity induced by using a knowledge base.

6. REFERENCES

[1] T. Pedersen, A. Kulkarni, R. Angheluta, Z. Kozareva, T. Solorio. Name Discrimination by Clustering Similar Contexts. CICLing2005.

[2] R. Bekkerman, A. McCallum. Disambiguating Web Apprearances of People in a Social Network. WWW2005.

[3] R. Guha, A. Garg. Disambiguating People in Search. WWW2004.

[4] http://www.alias-i.com/lingpipe/

[5] http://www.dmoz.org/

[6] http://www.google.com/

[7] R. Baeza-Yates, B. Ribeiro-Neto. Modern Information Retrieval. Addison Wesley Longman Publishing 1999.

[8] C. D. Manning, H. Schutze. Foundations of Statistical Natural Language Processing. The MIT Press 2003.

[9] A. Bagga, B. Baldwin. Entity-Based Cross-Document Coreferencing Using the Vector Space Model. ACL 1998.

[10] B. Malin. Unsupervised Name Disambiguation via Social Network Similarity. SIAM ICDM 2005.

[11] Gideon S. Mann, David Yarowsky. Unsupervised Personal Name Disambiguation. Computational Natural Language Learning 2003.

[12] Xiaojun Wan, Jianfeng Gao, Mu Li, Binggong Ding. Person Resolution in Person Search Results: WebHawk. CIKM 05.

[13] http://trec.nist.gov/

[14] http://dblp.uni-trier.de

[15] http://www.amazon.com

[16] http://www.tartarus.org/martin/PorterStemmer/

[17] H. Schutze. Automatic Word Sense Discrimination. Computational Linguistics, 24(1):97-123, 1998

[18] H. Han, L. Giles, H. Y. Zha, C. Li, K. Tsioutsioliklis. Two Supervised Learning Approaches for Name Disambiguation in Author Citations. JCDL 2004.