Using Web Directories for Similarity Measurement in Personal Name Disambiguation

Quang Minh Vu
Graduate School of Information Science and Technology, The University of Tokyo
Tokyo, 113-8656 Japan
Email: vuminh@nii.ac.jp

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

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

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

Abstract

In this paper, we target on the problem of personal name disambiguation in search results returned by personal name queries. Usually, a personal name refers to several people. Therefore, when a search engine returns a set of documents containing that name, they are often relevant to several individuals with the same namesake. Automatic differentiation of people in the resulting documents may help users to search for the person of interest easier. We propose a method that uses web directories to improve the similarity measurement in personal name disambiguation. We carried out experiments on real web documents in which we compared our method with the vector space model method and the named entity recognition method. The results show that our method has advantages over these previous methods.

1 Introduction

The World Wide Web (WWW) has been becoming the largest text database. Nowadays, web users often use search engines to find their interested information. As pointed out in [3], an analysis of five million queries obtained from the search engine AllTheWeb shows that at least 5-10% of queries include names of people. Search engines can help users to search for a person of interest, but due to the namesake problem, the results often contain noisy documents that are related to people other than the person of interest. For example, the top 100 results for the query "Jim Clark" from the Google search engine contain at least eight different *Jim Clark*. Among them, the two people with the largest number of pages are *Jim Clark*, the Formula One world champion and *Jim Clark*, the founder of Netscape. In our research, we try to separate different people in such a set of results automatically and group documents related to the same person together.

Disambiguation of personal names has been researched by some other groups [1, 2, 3, 5, 7, 8]. Our research is different from theirs as follows. We focus on documents mentioning to ordinary people in the WWW, while the previous research focuses on other kinds of people in other kinds of database such as people mentioned in the newspaper, people belonging to the same community network, or famous people on the WWW. As our target people and document type have different characteristics from those of the previous research, we argue that the previous approaches do not work well with our target data. Therefore, we propose a new method that can work well with relevant documents of ordinary people in the WWW. The rest of this paper is organized as follows. In Section 2, we summarize the previous research. We describe the targeted data and approaches used in the previous research. Then in Section 3, we propose a new method to determine similarities among documents. We present our improvement to the conventional document similarity measurement method. In Section 4, we give details about our name disambiguation system. Experiment results and comparisons with other methods are given in Section 5. Finally, Section 6 concludes this paper.

2 Related research

In [5], Bagga and Baldwin targeted on newspaper articles. In newspaper articles, documents related to the same person often discuss the same topic and they share many temrs strongly related with the topic. Therefore, the vector space model used in this research worked well with this kind of documents. In [1], Pederson et al. targeted documents referring to famous people. Naturally, famous people have many relevant documents on the WWW. From this large set of documents, the authors used the *second order context vectors* method [1] to build context vectors of people. These vectors were used to differentiate documents of different people. In [3, 7, 8], the authors extracted profiles of namesake people contained in documents. They used databases like DBLP [13], Amazon [14], pattern matching technique and natural language processing technique to extract personal information. Information about social networks of people has been used to differentiate namesakes. In [2], Bekkerman and McCallum extracted a group of people simultaneously. The authors proposed two methods to extract these documents: one that used hyperlink information and another that used the Agglomerative Conglomerative Double Clustering (A/CDC) [2] algorithm.

The previous approaches mentioned above would have limitation if they were applied to the documents relevant to ordinary people on the WWW for the following reasons. First, people mentioned in web pages may be in regard to different specific topics (although their general topics are often close together). Therefore, the vector space model may not work well with documents of ordinary people. Second, as the people that we target are ordinary, their relevant documents would likely be few. Hence, methods of creating their contexts from keywords do not work. Third, documents on the WWW have many different formats and are noisy, hence personal information extraction method can only work with a small fraction of these documents. Finally, method of utilizing social network information is also limited, since, in general, we do not know the social network of the person of interest in advance.

3 Our approach

3.1 Review of vector space model

In the vector space model a term's weight is calculated as follows.

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

where $tf(t,d)$ is the count of term t appearing in document d, N is the number of documents in the collection, and $df(t)$ is the number of documents containing term t in the collection of documents.

The vector space model measures document similarities well when documents discuss the same story. However, in the task of personal name disambiguation, documents may not have the same story for the following reason. Normally, a person attends various events, so his/her documents need not be about the same story (although they often have the same general topic). In such a case, there are very few common terms between documents, so the product of the feature vectors is not large. This makes the border between pairs of documents about the same topic and pairs of documents about different topics ambiguous.

3.2 Measure term feature

We use web directories to help the calculation of term's weigh. In a general sense, a web directories may be regarded as a kind of a knowledge base, where knowledge about several topics are collected in several set of documents on these topics. We choose to use web directories because many open web directories exist in the WWW, so the cost of preparing a collection is at minimum. In the discussion beyond, we will use "a knowledge base" and "web directories" alternatively to refer a collection of documents of several topics. We name our method "Similarity via Knowledge Base"(**SKB**). We use the knowledge base to find out keywords in documents and recalculate their weight. The details are as follows.

A knowledge base

The knowledge base consists of several sets of documents on several topics. We call a set of documents in the knowledge base a directory. In the traditional vector space model, all documents in the corpus are regarded as unrelated to each other. In contrast, in our knowledge base, documents belonging to the same set are related to each other, while documents belonging to different sets are unrelated to each other. We use this characteristic of the knowledge base to modify the term weight calculation.

Modification of tf

In [9], we proposed a modification method for tf to improve measurement of terms' weight. Here, we present a brief summarization of that method. Suppose that a document d is close in topic with a set of documents Dir in the knowledge base. Since relevant text related to a person in a web document tends to be short, terms strongly related to a document's topic may not appear frequently. On the other hand, a directory has abundant text, so terms strongly related to the directory's topic appear frequently in the directory. We can use the term frequency in the directory to modify the term frequency in the web document as $\sqrt{tf(t,d) \times tf(t,Dir)}$ to increase the term frequencies of terms strongly related to the document's topic.

As a knowledge base has several directories, we want to make this new feature measurement comparable among them to facilitate the calculation in the next step. Therefore, we normalize the term weight by dividing it by the size of each directory. The size of a directory Dir is the total number documents' word count in that directory and is denoted as $length(Dir)$. The following equation is used to compute the term weight.

$$tf_{SKB}(t,d,Dir) = \sqrt{\frac{tf(t,d) \times tf(t,Dir)}{length(Dir)}} \quad (2)$$

$$weight_{SKB1}(t,d,Dir) = tf_{SKB} \times log(\frac{N}{idf(t)}) \quad (3)$$

Modification of idf

We present our new idea to further improve measurement of term weight. In Equation 1 of the vector space model, the factor $log(\frac{N}{df(t)})$ (also called the idf value) measures the particularity of a term. The larger document frequency a term has, the lower idf value it has. With this treatment with the document frequency, all documents are assumed to be unrelated to each other. However, a term may appear frequently in some documents because either it is a common term or it is strongly related to the topic of the documents. The vector space model only considers the former case. We use the directory to consider the latter case.

If a term is strongly related to a topic and appears frequently in the directory corresponding to that topic, then among documents in the corpus that contain the term, a certain number of documents will be close to that topic. Therefore, these documents are not unrelated to each other and the idf value should be increased. We propose the following equation for idf that favors terms strongly related to the directory's topic and disfavors other terms.

$$df_{geo\_mean}(Dir) = \sqrt[K]{\prod_{i=1}^{K} df(t_i, Dir)} \quad (4)$$

$$idf_{SKB}(t,Dir) = log(\frac{N}{df(t)} \times \frac{df(t,Dir)}{df_{geo\_mean}(Dir)}) \quad (5)$$

New term feature measurement equation

Combining the above two modifications, we arrive at the following equation to measure term weight.

$$weight_{SKB2}(t,d,Dir) = tf_{SKB} \times idf_{SKB} \quad (6)$$

3.3 Measure document similarities

Find directories with topics close to the document topic

For each document, we choose k directories in the knowledge base whose similarites to the document are the top k largest values. The similarity between document d and directory Dir is measured as follows.

$$SIM(d,Dir) = \sum_{t \in d \cap Dir} weight_{SKB2}(t,d,Dir) \quad (7)$$

We call these top k directories the document's representative directories: $Dir_1, Dir_2, ...Dir_k$. The set of common terms between d and Dir_i is called the representative of d via directory Dir_i and is denoted by $Representative(d, Dir_i)$.

Measure similarities

Denote a pair of documents as (d_1, d_2). Denote the union set of representative directories of d_1, d_2 as $Dir_1, Dir_2, ...Dir_l$, $(k \leq l \leq 2k)$. We calculate the similarity between two representatives and consider it to be the similarity of d_1, d_2 via directory Dir_i as follows.

$$SIM(d_1,d_2,Dir_i) = \sum_t \begin{matrix} weight_{SKB2}(t,d_1,Dir_i) \times \\ weight_{SKB2}(t,d_2,Dir_i) \end{matrix} \quad (8)$$

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

After calculating the similarities of d_1, d_2 via all representative directories, we select the largest similarity value as the similarity of the document pair (d_1, d_2).

$$SIM(d_1,d_2) = \max_{i=1,2,..,l} SIM(d_1,d_2,Dir_i) \quad (9)$$

4 Name disambiguation system

4.1 System overview

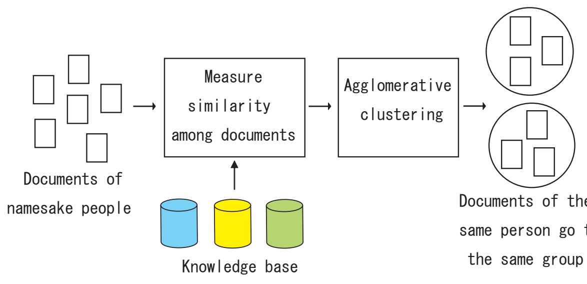

Figure 1. Name disambiguation system

Figure 1 shows the overview of our name disambiguation system. The document similarity calculation module calculates similarities among documents according to the SKB2 methods mentioned in Section 3. We chose documents from the Dmoz Open Directory Project (Dmoz ODP)[12] as the knowledge base for this module. We selected 56 directories from Dmoz ODP covering a broad range of topics in art, business, society, recreation, science, sports, etc. Each directory contains about 40 to 50 documents. The grouping module takes as input similarity results among documents and outputs several groups of documents.

4.2 Document processing

Preprocessing

We removed stop words and stem words to their root forms. The stemming algorithm was the Porter algorithm[15].

Next, we selected the 50 terms before and 50 terms after a personal name to create a bag of words representing that person. We ignored terms far away from the personal name because web documents are noisy sources and only text surrounding the personal name likely contains information about that person.

Document similarity calculation
We used 56 directories from the Dmoz ODP and equations 2, 4, 5 and 6 of the SKB2 method to calculate the document feature vectors via directories. On each document pair, we used equations 7, 8 and 9 to calculate the similarities via each representative directory and to select the largest value as the similarity of this document pair.

Document grouping
We used average link agglomerative clustering algorithm to cluster documents as follows

Procedure ClusterDocument()

```
1: Initially, each document forms a singleton cluster
2: Calculate similarities 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 similarities 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.

5 Experiment

5.1 Baseline methods

We compared our method with two previous methods: the vector space model (VSM) method[5] and the named entity recognition (NER) method[8].

Vector Space Model method
In the VSM method, we removed stop words and stemmed words into their root form by using the Porter stemming algorithm [15]. Then, we chose the 50 terms before and 50 terms after a personal query and used $tf\_idf(t,d)$ weight of these terms to build the feature vectors of documents (equation 1). We took the inner products of document feature vectors as the similarities between document pairs.

Named Entity Recognition method
In the NER method, we used the LingPipe software[11] to extract the names of entities in a document. Then we used these names to construct feature vectors of the document. The constitutents of vectors were binary values (1 if a name appears in the document, 0 otherwise). We took the inner products of the document feature vectors as the similarities between document pairs.

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

5.2 Test data

We selected 24 researchers in four fields: computer science, physics, medicine and history, as shown in Table 1. We sent these names as queries to the Google search engine and selected the top 100 document results containing the personal name. 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 our selected person. Hereafter, we call a namesake with 20 to 60 relevant documents in the data set a major person and other namesakes with a few number of relevant documents in the data set minor people.

We tried to create test data as close to those of the real application as possible. In a real information retrieval system, we would not know the number of namesakes that results documents refer to in advance. Also, in the results set, some people would have many relevant documents while other people would have only a few relevant documents. Therefore, we created pseudo-namesake data by mixing two document collections corresponding to the search results for two names of two people in different research fields. Each pseudo-namesake data created in this way had two major namesakes and several other minor namesakes.

We divided the 24 collections into two sets: a training set and a test set. The training set had 16 collections (4 collection/field $\times$ 4fields), and the test set had 8 collections (2 collection/field $\times$ 4fields). We created pseudo-namesake data as mentioned above from collections of the training set and the test set. 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.

5.3 Evaluation of top clusters

Labeling documents
Each test data has two major people referred in the dataset. We removed clusters whose size was less than or equal to

three. From remaining clusters, we select two clusters, each of the two contains the most number of documents relevant to each major person. We mark documents in each cluster with the label of that major person's name.

Evaluation metrics

Denote $N_{i\_labeled}$ as the number of documents being labeled with ith person's name label ($i = 1, 2$). Denote $N_{i\_correct}$ as the number of documents correctly being labeled with ith person's name label ($i = 1, 2$). Denote N_{i_total} as the total number of documents relevant to ith person ($i = 1, 2$). We calculate the top averaged precision (P_{top_aver}), top averaged recall (R_{top_aver}) of the labeling result. We also calculated the harmonic mean ($F_{measure\_top\_aver}$) of P_{top_aver} and R_{top_aver}.

$$P_i = \frac{N_{i\_correct}}{N_{i\_labeled}} \quad (10)$$

$$R_i = \frac{N_{i\_correct}}{N_{i\_total}} \quad (11)$$

$$P_{top\_aver} = \frac{P_1 + P_2}{2} \quad (12)$$

$$R_{top\_aver} = \frac{R_1 + R_2}{2} \quad (13)$$

$$F_{measure\_top\_aver} = \frac{2P_{top\_aver} \times R_{top\_aver}}{P_{top\_aver} + R_{top\_aver}} \quad (14)$$

Performances

We investigated the performances of different methods by varying the stopping condition of the clustering algorithm (i.e. the number of clusters) and measured P_{top_aver}, R_{top_aver} and $F_{measure\_top\_aver}$ metrics of top clusters. Figures 2, 3, 4 and 5 show the results for VSM, NER, SKB1, and SKB2, respectively. The comparison of the four methods in terms of $F_{measure\_top\_aver}$ value are shown in Figure 6.

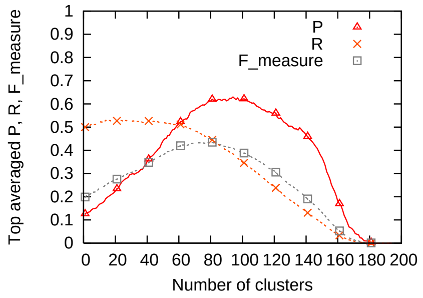

Figure 2. Performance of VSM method for top clusters

Tuning the number of clusters

Using the training set we found the number of clusters for the stopping condition of the clustering algorithms to best achieve the $F_{measure\_top\_aver}$ value. We divided the original data set to a training set and a test set randomly 50 times. This yields 50 (training data, test data) pairs of data sets. We repeated tuning $N_{threshold}$ using training data and verifying performances using test data 50 times and took the averaged

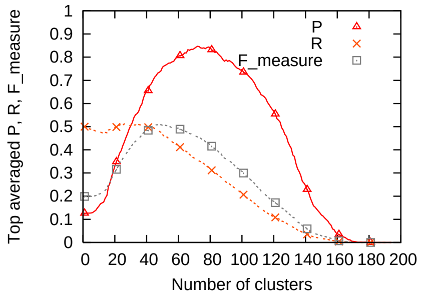

Figure 3. Performance of NER method for top clusters

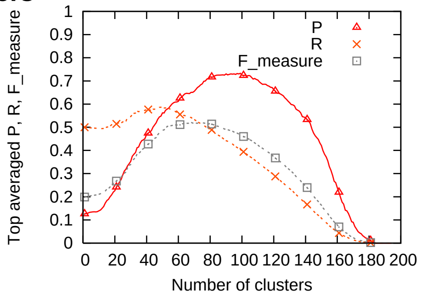

Figure 4. Performance of SKB1 method for top clusters

results. The averaged $N_{threshold}$ values are 74.5, 49.1, 66.5, and 73.6, for VSM, NER, SKB1, and SKB2, respectively (Table 2). At these averaged $N_{threshold}$ values, VSM, NER, SKB1, and SKB2 achieved the averaged $F_{measure\_top\_aver}$ values of 44.3%, 51.3%, 52.7%, and 52.0% with the training set. These numbers of clusters were applied for the stopping condition of the clustering algorithm in the test experiment. The performances of the training and test sets in terms of the averaged $F_{measure\_top\_aver}$ value are shown in Table 2.

Comparison

As shown in Table 2, the best $F_{measure\_top\_aver}$ values of SKB1 and SKB2 are 52.7% and 52.0%, respectively. These values are a litte better than that of NER (51.3%) and are much better than that of VSM (44.3%). In the test set, the SKB1 and SKB2 also outperformed VSM and NER in terms of the averaged $F_{measure\_top\_aver}$. The performance of SKB1 and SKB2 are 50.3% and 51.4%, respectively,

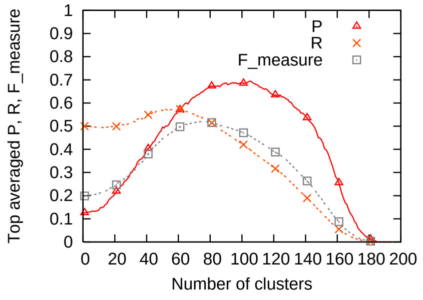

Figure 5. Performance of SKB2 method for top clusters

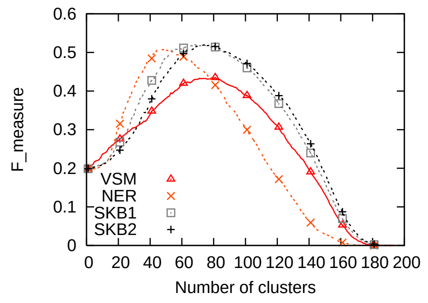

Figure 6. Comparison of F_measure_top_aver for top clusters among methods

Table 2. Tuning parameters and testing results for top clusters

Method	$N_{threshold}$	Training set	Test set
VSM	74.5	44.3%	41.5%
NER	49.1	51.3%	49.0%
SKB1	66.5	**52.7%**	50.3%
SKB2	73.6	52.0%	**51.4%**

compared with 49.0% for NER and 41.5% for VSM (Table 2).

5.4 Evaluation of large clusters

Normally, people may appear in the WWW with many appearances, so their documents' clustering result may have several groups, each group corresponds with an appearance. Even though we cannot group different appearances together, by showing different appearances in different groups we can still help users to navigate the search results and make it easier to find their person of interest. Therefore, we try to evaluate all clusters whose sizes are larger than three. The details on calculation of evaluation metrics can be found in [9]. The evaluation results of four methods VSM, NER, SKB1 and SKB2 and comparison among them are shown in Table 3.

Table 3. Tuning parameters and testing results for large clusters

Method	$N_{threshold}$	Training set	Test set
VSM	66.0	50.2%	47.2%
NER	57.3	51.8%	49.6%
SKB1	70.9	56.7%	55.0%
SKB2	77.7	**57.9%**	**57.6%**

6 Conclusion

We have proposed a new method to measure document similarities in personal name disambiguation. Our method uses a knowledge base in the similarity calculation. Basically, a knowledge base is a collection of documents on several topics. We use this collection of documents to calculate the documents' feature vectors and use the feature vectors' inner product for the document similarities. We then input these document similarity results into an agglomerative clustering algorithm to group related documents together. We carried out experiments on real web data and verified the advantage of our method over the vector space model method and the named entity recognition method.

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 Appearances of People in a Social Network. WWW2005.

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

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

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

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

[7] G. S. Mann, D. Yarowsky. Unsupervised Personal Name Disambiguation. Computational Natural Language Learning 2003.

[8] X. Wan, J. Gao, M. Li, B. Ding. Person Resolution in Person Search Results: WebHawk. CIKM 2005.

[9] Q. M. Vu, T. Masada, A. Takasu, J. Adachi. Using a Knowledge Base to Disambiguate Personal Name in Web Search Results. SAC-IAR 2007 (to appear).

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

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

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

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

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

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