High Performance Clustering for Web Person Name Disambiguation Using Topic Capturing

Zhengzhong Liu Qin Lu Jian Xu

Department of Computing
The Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong
+852-2766-7247

hector.liu@polyu.edu.hk csluqin@comp.polyu.edu.hk csjxu@comp.polyu.edu.hk

ABSTRACT

Searching for named entities is a common task on the web. Among different named entities, person names are among the most frequently searched terms. However, many people can share the same name and the current search engines are not designed to identify a specific entity, or a namesake. One possible solution is to identify a namesake through clustering webpages for different namesakes. In this paper, we propose a clustering algorithm which makes use of topic related information to improve clustering. The system is trained on the WePS2 dataset and tested on both the WePS1 and WePS2 dataset. Experimental results show that our system outperforms all the other systems in the WePS workshops using B-Cubed and Purity based measures. And the performance is also consistent and robust on different datasets. Most important of all, the algorithm is very efficient for web persons disambiguation because it only needs to use data already indexed in search engines. In other words, only local data is used as feature data to avoid query induced web search.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – Clustering; I.2.7 [**Artificial Intelligence**]: Natural Language Processing - Text analysis.

General Terms

Algorithms, Experimentation

Keywords

Web Person Name Disambiguation, Text Clustering

1. INTRODUCTION

When people search for a name, one may intend to find a specific entity, called the namesake associated with that name. Current search engines normally return a large set of documents containing the searched name string based on features such as hit ratio and hyperlinks associated with popular search interest as a whole in the internet rather than features to distinguish different namesakes. For example, when a user submits the query "`George Bush`", the returned documents by a search engine may include mentions such as (1)`George W. Bush (The 43rd President of U.S.` and (2)`George W.H. Bush (The 41st President of U.S.)` that are contained with different ranks. However, the mentions of `George Bush (Professor of Hebrew University, U.S)` would be hard to find as they are being returned as very low ranked files that would normally be missed.

Identifying a specific entity using the current search engine is time consuming because the scale of the returned documents must be large enough to ensure coverage of the different namesakes especially to contain all the documents with namesakes of much less popular persons.

The task of identifying different namesakes through web search is called web persons disambiguation. Recent works on web persons disambiguation have been reported in the WePS workshops [1,2]. The reported systems used different clustering techniques to organize the searched results according to either closed information or open information to disambiguate different namesakes. Some systems have achieved competitive performance by incorporating large amount of open information [7] or by conducting expensive training process [14]. In this paper, we present a novel algorithm based on Hierarchical Agglomerative Clustering(HAC) [16] which makes use of topic information using a so called hit list to make clustering more suitable and effective for web persons disambiguation.

The rest of the paper is organized as follows. Section 2 presents related works. Section 3 describes our algorithm design. Section 4 gives the performance evaluation and Section 5 is the conclusion.

2. RELATED WORKS

Researchers have adopted different assumptions about the data for web persons disambiguation. A common assumption is that each document is associated with only one person and thus a specific namesake [3,4,7]. Others may select some smaller units such as paragraph to represent a single person [18].

Different systems adopted different features for similarity measures. The performance of a system is highly related to the features used. Generally speaking, feature selection can be classified into either local methods or global methods. The local methods select and assign weights to features according to the given collection. Among the local methods, [3] extracts features from the sentences that contain a co-reference of the target name and [15] extracts biographical information from web pages. Many systems in the WePS workshops use simple word tokens from the webpages found in the given corpus [4,7,10]. Some systems use more advanced features like word bigrams [7], phrases [8] or named entities [18]. On the other hand, global methods

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
EOS, SIGIR 2011 workshop, July 28, 2011, Beijing, China.
Copyright is held by the author/owner(s).

incorporate information external to the given dataset such as external corpora or online information for similarity measures. For example, [19] searches for external snippets from Google to directly help in clustering. Other systems may use the external knowledge to tune the weightings of the different features. For example, [7] uses the Google 1T 5-gram data to learn frequencies of bigrams. [14] uses Wikipedia to find phrases in documents.

In addition, because webpages can be noisy, some attempts are made to use only relevant content blocks in webpages. [13] identifies useful content by using predefined templates. [14] uses Gradient Boosting method to learn the content weighting on 1.3 million popular web pages and 400,000 manually annotated query-URL pairs. However, as the types of webpages can vary lot, and new pages are continuously being generated, it may be difficult to capture the most relevant information from the ever changing data types.

In the task summary of the WePS2 workshop, [2] pointed out the importance of selecting the optimal threshold for clustering. The BEST_TOKEN system which knows all the optimal thresholds beforehand, achieve a very good performance with only simple local features. Results show that finding the optimal termination criteria can greatly improve the clustering quality even with only local data. Most prior works on clustering focus on optimization of cluster numbers or threshold for a clustering task. [17] reviewed the set of criteria for cluster number optimization. However, these methods are normally application dependent. The PDDP method used by [6] measures the non-cohesiveness of clusters and split the data in the weakest cohesion point. The Chameleon system [12] adopts a similar method that computes both the inter-connectivity within the cluster and the proximity among clusters. These methods work well on regular clustering problems. However, most of them tend to avoid both singleton and "All-in-One" clusters, whereas in our problem, the sizes of the clusters can vary significantly. For web persons disambiguation, it is important to handle large variations of different cluster sizes.

3. ALGORITHM DESIGN

3.1 Preprocessing

For web persons disambiguation, the data are normally obtained from information already stored in a search engine indexed under the searched name including the webpages, the related snippets and metadata to form a so called local corpus as supplied by WePS workshops. Webpages as documents normally contain a large amount of noise including formatting tags, executable scripts and Meta information. The quality of preprocessing can largely influence the performance of the clustering system.

We use some simple rule-based methods for noise removal. The raw html pages are first parsed by the Beautiful Soup Parser[1] to remove html tags and scripts tags. Only plain text is reserved for clustering. Similar to the measures in [4], for block level tags (such as , <p>), only text blocks with more than 10 words are preserved. The preprocessing module also removes groups of text less than 4 words that are separated by bars or text grouped by <li> tags to eliminate the navigation links in a webpage. After the plaintext is produced, standard sentence segmentation, word tokenization and stemming are applied using the NLTK toolkit[2]. Stop words, punctuations, and numbers less than 4 digits are also removed.

3.2 Feature Selection

It is easy to understand that rich features and extra knowledge beyond the local data collection can improve the system performance. However, getting external information is rather expensive, both on storage and computation. Such information is also sometimes domain specific, which is difficult to obtain in some cases. In practice, we also know that as long as there is information for a specific namesake in the local data, it is often sufficient for human to identify the entity. In this work, we explore methods to make full use of local features, and try to explore the full potential of locally provided information.

Our algorithms use the standard Vector Space Model commonly used by many information retrieval systems. Each document is represented by a vector formed by 7 types of tokens extracted from local data as listed below:

1. **Webpage title**: the title of the webpage is split to single words and added to the feature space.
2. **URL of webpage**: they include both the host name of URL of a webpage and path on the server. We remove the common webpage extensions by a dictionary. Non-meaningful tokens such as pure numbers and punctuations are removed.
3. **Webpage metadata**: only two kinds of metadata in a webpage, "keywords" and "descriptions" are used if they exist. These metadata are highly summarized and informative. Sometimes they contain information that does not appear in the webpage.
4. **Snippet**: Query biased snippets returned by the Yahoo! search engine (included in WePS data) which normally has reference to the name. Snippets are highly summarized fragments of text. Query biased snippets summarize the query context and distant information relevant to the query term [21].
5. **Context Window**: Tokens in a context window within the query name. The window size is selected based on experiments.
6. **Context Sentence**: The whole sentence that contains the query name.
7. **Bag of Words**: Commonly used feature to represent the document. In our system, we simply index all the words in the document as tokens in the feature space.

For a token s, we use the TF.IDF weighting scheme with its weight calculated as:

$$W'_s = \log(tf_s + 1) \times \log(\frac{N_d}{df_s})$$

where tf_s is term frequency of s, N_d is the total number of documents, and df_s is the number of documents in which s is present. Furthermore, we give weight factor to each type of tokens denoted by *WF(s)* because some might be more important than others. Thus, the final normalized weight for each token using cosine co-efficiency is given as:

$$W = \frac{W'_s \times WF(s)}{\sqrt{\sum_{s \in V}(W'_s \times WF)^2}}$$

3.3 Clustering Algorithm

Our clustering algorithm is based on the common Hierarchical Agglomerative Clustering (HAC Algorithm) [16]. In HAC, all documents are treated as singleton clusters initially, and are

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

[2] http://www.nltk.org/

referred to as "leaf clusters". In each iteration of HAC, the two most similar clusters will be merged into a larger cluster. The centroid vectors of the two clusters will be merged to a new centroid vector. During the merging of clusters, there are generally two trends in the cluster's centroid:

1. A small set of keywords will be repeatedly matched, thus maintaining a higher weight in the cluster centroid vector.
2. As new tokens are constantly added to the cluster centroid, there are some rare words to be added too with relatively low weight. And these low weight tokens occupy a large portion in the vector.

The HAC algorithm generally works well at the beginning of the clustering. However, when the above two phenomena become more and more prominent, the performance begins to deteriorate. The system may mistakenly merge two large clusters together because of the large number of token hits as shown in **Figure 1**. Also, some newly added tokens may make the topic diverge to a wrong direction. In **Figure 2**, the set of clusters about a journalist may be mistakenly merged together with the cluster about sergeants in a war because of the match on some non-topic words.

Figure 1 Merge of Clusters with low weight terms

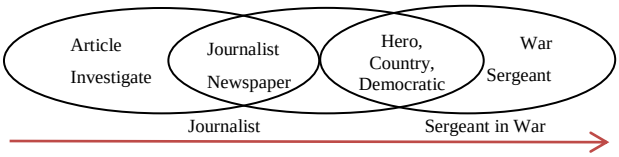

Figure 2 Topic Divergence

3.3.1 Topic Capturing

To solve the problem created by noise data not related to certain topic, we assume that most of the webpages describing the same namesake are likely to have similar topics. For example, documents about a journalist should mainly contain topics about news, journals and so on. Documents about a sergeant, however, should primarily contain topics about war and the like. Based on this assumption, we try to merge document clusters only if they share consistent and useful topics. In other words, our system favors merging clusters that are likely to mention the same topics.

In order to obtain the core topic words, we created a vector called "Hit List" during the clustering process. This vector is maintained for each non-singleton cluster in addition to the centroid vector used in the conventional HAC.

The Hit List of a cluster *C* is actually the vector containing the shared tokens of the two source clusters that are merged to form cluster *C*. Each token in the Hit List is associated with a weight, which is the product of its origin normalized weights in the two source clusters. The construction of a Hit List vector can be illustrated using the example in **Table 1**. It is worth noting that the sum of the weights in the Hit List is the dot product of the centroid vector in cluster C_1 and cluster C_2. Because the vectors are already normalized, the dot product represents the cosine similarity of C_1 and C_2. So the Hit List actually records the contribution of each token to the formation of this new cluster *C*.

Table 1 Example of Hit List construction by merging two clusters of the same size (number of documents in the cluster)

C_1	umass	virginia	research	student
Centroid Vector	0.5	0.1	0.4	0

C_2	umass	virginia	research	student
Centroid Vector	0.5	0.2	0.2	0.1

Merged *C*	umass	virginia	research	student
Centroid Vector	0.5	0.15	0.3	0.05
Hit List	**0.25**	**0.02**	**0.08**	**0**

* *Hit List for cluster 1 and cluster 2 are omitted.*
** *For demonstration purpose, the centroid vector for the merged cluster is not normalized.*

Table 1 shows that if a token gets a high weight in the Hit List, it should have high weights in its source clusters as well. This means that tokens with high weights in the Hit List are normally important words to documents in the cluster. Accordingly, they can represent the topic of the cluster. As the clustering process continues, the Hit List gradually grows larger. But only the topic words, which are likely to be repeated in many documents, will gain higher weights than general words. So we only keep words with higher weights as topic words. To maintain topic words with significance, we select a fixed threshold that is proportional to the cosine similarity threshold value used in HAC which will be explained in the performance evaluation section.

3.3.2 Similarity Modification Based on Topics

With the help of the Hit List, the similarity measure used in the HAC is slightly modified by a simple scheme: If two clusters that are not likely to be talking about the same topic, their similarity value will be reduced by a penalty value. We consider two cases for the penalty. The first one is when the Hit List of the merged cluster has only few words, which means that the two clusters are generally matched based on the large amount of low weight tokens. The second one is when the Hit List of the merged cluster has only few overlaps with the respective Hit Lists of the source clusters which indicate topic divergence.

For the first case, we use the Vital Match Ratio to handle the problem. Given two clusters, C_1 and C_2 with their feature vector F_1 and F_2, and their corresponding Hit List, H_1 and H_2, and the merged cluster Hit List H_c, The Vital Match Ratio of C, denoted by VMR_c is calculated using the formula:

$$VMR_c = \frac{card(H_c)}{card(C_1) + card(C_2)}$$

where $card(H_c)$, $card(C_1)$, and $card(C_2)$ denote the respective cardinalities of the respective vectors. If VMR_c is less than a threshold, we will give penalty to the similarity by subtracting a penalty score P_c to be determined experimentally.

For the second case, we use the overlapping similarities between H_1 and H_c and between H_2 and H_c to detect the divergence of the clusters to consider penalty. For two vectors *V* and *V'*, let us use $v(u)$ and $v'(u)$ to denote the weights of a term *u* in the corresponding vectors. The overlap similarity is then given as:

$$Overlap\_sim(V, V') = \frac{\sum_{u \in V, u \in V'} (v(u) + v'(u))}{\sum_{w \in V} v(w) + \sum_{w' \in V'} v'(w')}$$

If one of the two to be merged clusters is a singleton cluster, which would not have had a Hit List associated with it, we let the

overlap similarity equals to the overlap threshold. Then, we define the divergence value (*DV*) as the harmonic mean of the two values, which is given as:

$$DV = \frac{2}{\frac{1}{Overlap\_sim(H1, Hc)} + \frac{1}{Overlap\_sim(H2, Hc)}}$$

In particular, *DV* value will be 0 if either one of the overlap similarity is 0. If *DV* value is lower than a threshold, we also give a penalty by subtracting a penalty score P_o to be determined experimentally.

4. PERFORMANCE EVALUATION

Our system is developed based on the WePS2 dataset, which contains 30 ambiguous names. Each name is associated with around 150 documents. The total number of documents is 3,444. The names are extracted from different domains, which can help to test the system in real application without any bias. We use both the B-Cubed scores described in [3] and Purity-based scores described in [1] to evaluate our system. The official ranking score for WePS2 is the F-measure (Harmonic Mean) of B-cubed precision and recall, and for WePS1 is the purity based F-measure. The workshops provide two F-measures, one gives equal weighting to precision and recall ($\alpha = 0.5$), the other give higher weighting to recall ($\alpha = 0.2$).

As the training data are provided, all the algorithm parameters are determined experimentally based on WePS2 test data. Due to the limit of the paper, we will simply give out the parameters used without giving details of the experiments.

Optimal clustering threshold is difficult to find when the size of the clusters can vary a lot from person to person [2]. In WePS2's local data collection, the minimum number of clusters is 1 and the maximum number of clusters is 56. Also, the number of documents can be from just one document in a cluster to 99 documents in another. The high dimensionality of the document vectors also makes it difficult to model clustering behaviors. In our system, the similarity measurement is Cosine Similarity of two vectors. The algorithm stops if the maximum similarity between clusters is less than the cosine similarity threshold. The threshold is **0.1,** determined experimentally, and also consistent with other systems [9,20]. The weighting factors for different tokens are tuned based on their importance to the clustering, their values are given in **Table 2**. All these parameters are set according to experiments on the WPS2 test data. Experimental data show that metadata and context sentences play more important roles. Snippets and context window are less important perhaps because their information is less coherent.

Table 2 Token Weighting Factors (WF)

Token type	Title	URL	Metadata	Snippets	Context Window	Context Sentence	BOW
WF_t	1	1	2	0.8	0.8	2	1

Threshold values for VMR, the Divergence Value and the corresponding penalty values should be scaled accordingly to the specific application. If these values are higher, then they can have a better control power on topic. The upper bound and lower bound for these values are 1 and 0 respectively. These values should also be proportional to the cosine similarity threshold. Normally, setting the value of penalty scores similar to the cosine similarity threshold will achieve a good performance. The parameters related to the penalties in our system are listed in **Table 3**.

Table 3 Threshold Settings in the Evaluation

VMR Threshold	VMR Penalty (P_c)	DV Threshold	DV Penalty (P_o)
0.02	0.08	0.01	0.1

Table 4 gives the performance evaluation of our system, labeled as HAC_Topic, compared to 2 algorithms within known upper bound and the top 3 performers in the WePS2 evaluation. The BEST-HAC-TOKENS and the BEST-HAC-BIGRAMS systems are the upper bound systems provided by the WePS workshop committee. These two systems have the optimal threshold on each namesake beforehand. The Top1 performer is the PolyUHK system [7] which uses both global and local features. PolyUHK used the Google 1T corpus to learn the weighting of unigrams and bigrams and query Google to find extra information about a person. The Top 2 system UVA_1 [5] uses simple tokens from the html cleaned documents. The Top 3 system ITC-UT-1 [11] uses rich features including named entities, compound nouns and URL links within the local page. **Table 4** shows that our system outperforms all the top 3 teams in both F-0.5 and F-0.2 scores beating systems using both local features and global features. Compared to the systems using local features, ours is at least **5.7%** improvement. This indicates that with a better designed clustering algorithm, the system can effective even if only local features are used. In other words, the clustering algorithm can make full use of local features so the performance can be improved without the loss of run time efficiency through the use of global features.

Table 4 Performance of WePS2 Data on B-Cubed Measures

	F-measures		B-Cubed	
SYSTEMS	$\alpha = 0.5$	$\alpha = 0.2$	Pre.	Rec.
BEST-HAC-TOKENS	*0.85*	*0.84*	*0.89*	*0.83*
BEST-HAC-BIGRAMS	*0.85*	*0.83*	*0.91*	*0.81*
Top1:PolyUHK	0.82	0.80	0.87	0.79
Top2:UVA_1	0.81	0.80	0.85	0.80
Top3:ITC-UT_1	0.81	0.76	**0.93**	0.73
HAC_Topic	**0.85**	**0.83**	0.92	**0.82**

It is also worth noting that our HAC_Topic system has shown a similar performance to the BEST-HAC-TOKENS, which is the upper limit of basic token based method. This implies that our method can help find relatively good stopping termination criteria.

Table 5 Performance of WePS2 data on Purity Measures

	F-measures			
SYSTEMS	$\alpha = 0.5$	$\alpha = 0.2$	Pur.	Inv_Pur.
BEST-HAC-TOKENS	*0.90*	*0.89*	*0.93*	*0.88*
BEST-HAC-BIGRAMS	*0.90*	*0.87*	*0.94*	*0.86*
Top1:PolyUHK	0.88	0.87	0.91	0.86
Top2:UVA_1	0.87	0.87	0.89	0.87
Top3:ITC-UT_1	0.87	0.83	0.95	0.81
HAC_Topic	**0.90**	**0.89**	**0.94**	**0.88**

Table 5 shows the performance evaluation of the different systems using purity based scores. Again, our system achieves the best result in almost all the performance measures. It even outperforms the best upper bound system BEST-HAC-TOKENS. The consistent high performance in both scoring schemes proves that our algorithm is rather robust which is very important in real applications.

In order to fully validate the effectiveness of our approach, we also tried to apply the algorithm to different datasets. In principle, there are two more datasets to use: WePS1 and WePS3. Even though the WePS3 dataset is relatively large and comprehensive,

the answer set is problematic. It was produced using online crowd sourcing method with little manual verification. So, the set contains incorrect data and also missing data. Thus, comparison to others is not meaningful. Thus, we only used the manually prepared WePS1 dataset for further evaluation and comparison.

The WePS1 dataset contains a test set with 30 names and a training set with 49 names. Our system ran the test set to compare to the other systems in WePS1. The top 3 systems [5,8,18] are all using rich local features such as tokens, URLs, Named Entities and time expressions. **Table 6** shows the performance evaluation based on Purity measures as the official ranking in WePS1 workshop only provided purity-based performance measures, where purity is a measure for precision level and inverse purity is for recall level. As shown in **Table 6,** our system outperforms all the other systems in terms of purity scores. This behavior is expected because the measures are designed to prevent the merging of two clusters referring to different namesakes. Our purity score has a **15.2%** improvement to the best system CU_COMSEM. In terms of the inverse purity score, our system outperforms CU_COMSEM by **3.4%**. The overall improvement in F-score is a significant **10.2%**.

Table 6 Performance of WePS1 data on Purity Measures

	F-measures			
SYSTEMS	$\alpha = 0.5$	$\alpha = 0.2$	Pur.	Inv_Pur
Top1:CU_COMSEM	0.78	0.83	0.72	0.88
Top2: IRST-BP	0.75	0.77	0.75	0.80
Top3: PSNUS	0.75	0.78	0.73	0.82
HAC_Topic	**0.86**	**0.89**	**0.83**	**0.91**

We further evaluate the effectiveness of our clustering method compared to the regular HAC method without the use of topic information (labeled as HAC_NoTopic) using WePS2 as training data. **Table 7a** shows the experiment results for evaluation based on the test dataset of WePS1 only and **Table 7b** shows the evaluation using both the test dataset and training dataset of WePS1.

Table 7a Performance of HAC Using WePS1 Test Data

	B-Cubed		Purity		F-Measure	
SYSTEMS	BEP	BER	P	IP	B-Cubed	Purity
HAC_Topic	**0.79**	0.85	**0.83**	0.91	**0.81**	**0.86**
HAC_NoTopic	0.75	0.85	0.67	0.91	0.78	0.76

Table 7b Performance on HAC Using WePS1 Complete Data

	B-Cubed		Purity		F-Measure	
SYSTEMS	BEP	BER	P	IP	B-Cubed	Purity
HAC_Topic	**0.88**	0.82	**0.70**	0.89	**0.84**	**0.76**
HAC_NoTopic	0.84	**0.84**	0.67	**0.90**	0.82	0.75

Table 7a shows that the B-Cubed F-measure is improved by **3.8%**. Furthermore, the Purity based F-measure is improved by **13.1%.** This further shows that the improvement by our system is mainly contributed by the improvement in precision so the resulting data is more reliable. For the full dataset as shown in the **Table 7b**, the corpus is composed of 79 name queries with much more variations in terms of numbers of documents in each collection. It is obvious that when the dataset gets larger, our algorithm has further improvement in all the precision related measures. However, HAC_NoTopic is better in terms of recall related measures. It is certainly understandable that our system has gains in precision at the cost of recall, at least statistically. The important issue is, our algorithm is better for both datasets in terms of F-measure which further confirms that our algorithm can give overall performance improvement compared to the regular HAC. This means that the introduction of the Hit List vector is very effective.

Further investigation shows, however, that our algorithm sometimes can improve both the precision and recall. **Table 8** shows the micro level performance in B-Cubed measures on all query names in the WePS-1 test set. Take the query "Neil_Clark", as an example, the precision is improved by **29%** and the recall is improved by **6%** as well. The overall performance shows our algorithm has significant improvement in precision while keeping the recall at a similar level. This is because we have enough features to distinguish different namesakes without losing any useful information. In other words, the algorithm has the ability to successfully guide clustering to the correct direction.

Table 8 Micro Performance on WePS-1 Test Set (B-Cubed)

	Normal HAC			Using Topic Capturing		
topic	BEP	BER	F-0.5	BEP	BER	F-0.5
Alvin_Cooper	0.85	0.85	0.85	0.85	0.85	0.85
Arthur_Morgan	0.66	0.82	0.73	0.76	0.79	0.77
Chris_Brockett	0.93	0.86	0.89	0.94	0.82	0.88
Dekang_Lin	1.00	0.86	0.93	1.00	0.90	0.95
Frank_Keller	0.87	0.79	0.83	0.86	0.81	0.84
George_Foster	0.72	0.75	0.74	0.71	0.83	0.77
Harry_Hughes	0.86	0.91	0.88	0.88	0.85	0.87
James_Curran	0.71	0.81	0.76	0.66	0.83	0.73
James_Davidson	0.86	0.91	0.88	0.84	0.92	0.88
James_Hamilton	0.63	0.72	0.68	0.75	0.74	0.75
James_Morehead	0.59	0.88	0.71	0.60	0.86	0.71
Jerry_Hobbs	0.92	0.77	0.84	0.92	0.77	0.84
John_Nelson	0.76	0.89	0.82	0.80	0.88	0.84
Jonathan_Brooks	0.87	0.91	0.89	0.87	0.91	0.89
Jude_Brown	0.68	0.85	0.75	0.78	0.83	0.80
Karen_Peterson	0.72	0.98	0.83	0.76	0.98	0.86
Leon_Barrett	0.91	0.78	0.84	0.93	0.78	0.85
Marcy_Jackson	0.73	0.79	0.76	0.71	0.79	0.75
Mark_Johnson	0.58	0.92	0.71	0.77	0.91	0.83
Martha_Edwards	0.42	0.93	0.58	0.51	0.93	0.66
Neil_Clark	**0.68**	**0.81**	**0.74**	**0.97**	**0.87**	**0.92**
Patrick_Killen	0.70	0.81	0.75	0.87	0.76	0.81
Robert_Moore	0.70	0.70	0.70	0.87	0.67	0.76
Sharon_Goldwater	0.98	0.82	0.89	0.99	0.80	0.88
Stephan_Johnson	0.88	0.85	0.87	0.92	0.83	0.88
Stephen_Clark	0.87	0.88	0.88	0.83	0.87	0.85
Thomas_Fraser	0.42	0.88	0.57	0.46	0.85	0.60
Thomas_Kirk	0.57	0.88	0.70	0.71	0.88	0.79
Violet_Howard	0.55	0.96	0.70	0.61	0.96	0.75
William_Dickson	0.59	0.89	0.71	0.55	0.90	0.68
Average	0.74	0.85	0.78	0.79	0.85	0.81

5. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed an effective HAC algorithm using additional topic information for web persons disambiguation. The experimental results show that our proposed clustering algorithm can achieve very good performance over the conventional methods. The key to the high performance of this algorithm is that it can effectively reduce the over merging of namesakes in clustering especially apparent when the cluster sizes can vary a lot. The disambiguation power of the method is thus improved significantly. As a result, features required in the algorithm are less demanding than other algorithms used for web persons disambiguation. In fact, our algorithm only uses simple local features from the training data which is readily available in most of the current search engines. This means that the processing time and the storage requirement in our system is much less demanding. This is rather important in practice where timely feedback to user queries is essential. As the features selected in our algorithm are already common indexed terms by modern search engines, the method can be developed easily on any existing search engine.

However, there is still room for improvement. Firstly, the parameter settings are done based on WePS2 data. In principle, the parameters are sensitive to the data as they are threshold based algorithms. This is particularly true if the application is used in different domains. Adjusting parameters for a given domain is important for the success for the system. The algorithm will be more robust if the number of parameters can be reduced. Possible reduction can be investigated over the parameters used for the topic related penalties. We can also further investigate methods to reduce the dimension of the feature vectors used in the algorithm. We can also study the possibility of using relatively cheap global data sources for the training phase. Some global data can be achieved offline without knowing user queries, such as pre-compiled corpus for learning term frequencies. The use of such offline data will not greatly affect the query processing time. Other possible directions for feature enrichment include using biographical information from the webpages and utilizing more semantic information such as synonyms information.

It is important to point out that clustering is only the first step in web persons disambiguation. To give a full picture for the users, the system should also be able to label the resulting clusters with the corresponding attributes of the namesakes.

6. ACKNOWLEDGMENTS

The project is partially supported by China Soong Ching Ling Foundation and Poly Project(COMP): RPVW.

7. REFERENCES

[1] Artiles, J., Gonzalo, J., & Sekine, S. 2007. The semeval-2007 weps evaluation: Establishing a benchmark for the web people search task. *Proceedings of Semeval*, (June), 64-69.

[2] Artiles, J., Gonzalo, J., & Sekine, S. 2009. Weps 2 evaluation campaign: overview of the web people search clustering task. *2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference*.

[3] Bagga, A., & Baldwin, B. 1998. Entity-based cross-document coreferencing using the vector space model. *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, 79–85. Association for Computational Linguistics.

[4] Balog, K., Azzopardi, L., & Rijke, M. de. 2005. Resolving person names in web people search. *Weaving services and people on the World Wide Web*, 301–323.

[5] Balog, K., He, J., Hofmann, K., Jijkoun, V., Monz, C., Tsagkias, M., et al. 2009. The University of Amsterdam at WePS2. *2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference*.

[6] Boley, D. 1998. Principal direction divisive partitioning. *Data mining and knowledge discovery*, *2*(4), 325–344. Springer.

[7] Chen, Y., Lee, S. Y. M., & Huang, C. R.2009. Polyuhk: A robust information extraction system for web personal names. *2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference*.

[8] Chen, Y., & Martin, J. 2007. Cu-comsem: Exploring rich features for unsupervised web personal name disambiguation. *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, 125-128.

[9] Elmacioglu, E., Tan, Y. F., Yan, S., Kan, M. Y., & Lee, D. 2007. PSNUS: Web people name disambiguation by simple clustering with rich features. *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, 268–271.

[10] Han, X., & Zhao, J. 2009. CASIANED: Web Personal Name Disambiguation Based on Professional Categorization. *2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference*, 2-5.

[11] Ikeda, M., Ono, S., Sato, I., Yoshida, M., & Nakagawa, H.2009. Person Name Disambiguation on the Web by Two-Stage Clustering. *2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference*.

[12] Karypis, G., & Kumar, V.1999. Chameleon: hierarchical clustering using dynamic modeling. *Computer*, *32*(8), 68-75. doi: 10.1109/2.781637.

[13] Lin, S.-hua, & Ho, J. M. 2002. Discovering informative content blocks from Web documents. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (p. 588–593). ACM.

[14] Long, C., & Shi, L. 2010. Web person name disambiguation by relevance weighting of extended feature sets. *Third Web People Search Evaluation Forum (WePS-3), CLEF* (Vol. 2010, pp. 1-13).

[15] Mann, G. S., & Yarowsky, David. 2003. Unsupervised personal name disambiguation. *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 -*, 33-40. Morristown, NJ, USA: Association for Computational Linguistics. doi: 10.3115/1119176.1119181.

[16] Manning, D. C., Raghavan, P., & Schutze, H. 2008. Hierarchical Clustering. *Introduction to Information Retrieval. Cambridge University Press, New York, 2008*, 377 - 401.

[17] Milligan, G. W., & Cooper, M. C.1985. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, *50*(2), 159–179. Springer.

[18] Popescu, O., & Magnini, B.2007. Irst-bp: Web people search using name entities. *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, (June), 195-198.

[19] Rao, D., Garera, N., & Yarowsky, D.2007. JHU1: an unsupervised approach to person name disambiguation using web snippets. *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, 2-5.

[20] Smirnova, K. A. E., & Trousse, B. 2010. Using web graph structure for person name disambiguation. *Third Web People Search Evaluation Forum (WePS-3), CLEF* (Vol. 2010).

[21] Tombros, A. and Sanderson, M. Advantages of query biased summaries in information retrieval. *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, ACM (1998), 2–10.