Person Resolution in Person Search Results: WebHawk

Xiaojun Wan*
Institute of Computer Science and Technology,
Peking University
Beijing 100871, China
wanxiaojun@icst.pku.edu.cn

Jianfeng Gao, Mu Li
Natural Language Computing Group
Microsoft Research Asia
Beijing 100080, China
{jfgao, muli}@microsoft.com

Binggong Ding*
Department of Computer Science and Engineering
Dalian University of Technology
Dalian 116023, China
dbg_dlut@hotmail.com

ABSTRACT

Finding information about people on the Web using a search engine is difficult because there is a many-to-many mapping between person names and specific persons (i.e. referents). This paper describes a person resolution system, called **WebHawk**. Given a list of pages obtained by submitting a person query to a search engine, **WebHawk** facilitates person search in three steps: First of all, a *filter* removes those pages that contain no information about any person. Secondly, a *cluster* groups the remaining pages into different clusters, each for one specific person. To make the resulting clusters more meaningful, an *extractor* is used to induce query-oriented personal information from each page. Finally, a *namer* generates an informative description for each cluster so that users can find any specific person easily. The architecture of **WebHawk** is presented, and the four components are discussed in detail, with a separate evaluation of each component presented where appropriate. A user study shows that **WebHawk** complements most existing search engines and successfully improves users' experience of person search on the Web.

Categories and Subject Descriptors

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

General Terms

Design, Experimentation

Keywords

Person Resolution, Person Search, Clustering, Junk Filtering

1. INTRODUCTION

Person search is one of the most popular search types on the Web. For example, according to [6], 5-10% of the English queries from *AllTheWeb* include person names. However, most of the current search engines (e.g. *Google*, *Yahoo*, *MSN* and *AllTheWeb*, etc.) do not provide any specific function aiming at person search. They treat a person query the same as a general query, and return all web pages that contain one or more terms in the person query. It is difficult for users to find the expected person in such retrieval results due to following reasons.

First of all, some pages may not contain any person information, referred to as *junk pages* in this paper, because person names may refer to non-person entities, such as products, companies, or places. For example, "Ford" may refer to the Ford Company. Those junk pages should be removed from person search results.

Secondly, there are a lot of ambiguities in person search. As Taffet [17] pointed out, those ambiguities can be categorized into two kinds. One is called the *multi-referent* ambiguity due to the fact that many persons (i.e. referents) may have the same name. The other is called the *multi-morphic* ambiguity due to the fact that one name may have different written forms.

Multi-referent ambiguity is very common. For example, given a person query "David Lee", there are more than ten referents in the top-100 pages retrieved by *MSN*. Some examples are shown below:

1. *David (L.) Lee – lawyer*
2. *David (A.) Lee – surgeon*
3. *David (H.) Lee – professor*
4. *David Lee – coach*

To investigate how severe the multi-referent ambiguity is in reality, we selected the 200 most-frequent person queries from the log of *MSN*. We issued these queries to *MSN*, and manually counted for each query the number of different referents that occur in top-100 retrieved pages. The statistics are shown in Figure 1. We see that 68% of person queries retrieve two or more referents, and about 20% retrieve more than ten referents. The worst case is the query "Michael", which retrieves 48 referents.

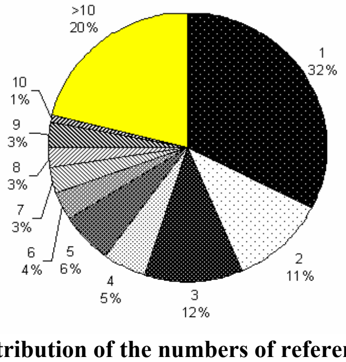


Figure 1: Distribution of the numbers of referents, estimated from the retrieved results of 200 person queries using *MSN*.

Multi-morphic ambiguity is also common because the same person usually goes by different names in different contexts such

* This work was done while the authors were visiting Microsoft Research Asia.

*CIKM'*05, October 31-November 4, 2005, Bremen, Germany.

as "Susan Dumais" and "Susan T. Dumais". This would also annoy users when they browse search results.

Unfortunately, the issues in person search mentioned above have not been dealt with successfully in most of the existing search engines. They treat person queries the same as other generic queries, and simply present the retrieval results as a mixed list of non-person (or junk) pages and person pages with different referents. Thus, users have to browse those pages one at a time to find the expected person, making person search a painful experience. Usually, veteran users can get refined results by adding appropriate contextual words (e.g. title, organization, etc.) into the query and elaborating the query expression. However, most users are not experienced search experts and the inappropriate contextual words and query expression will ever deteriorate the results e.g., lose relevant web pages and introduce irrelevant web pages.

Recently, some meta-search engines have been developed to handle ambiguities in search results. Some representative examples include *Vivisimo* (www.vivisimo.com), *Dogpile* (www.dogpile.com) and *iBoogie* (www.iBoogie.com). However, they focus on general information organization, and can only partially resolve the two kinds of ambiguities in person search.

In this paper we present a person resolution system, called **WebHawk**. Given a list of pages obtained by submitting a person query to a search engine (i.e. *MSN* in this study), **WebHawk** facilitates person search by re-organizing the search results in three steps: First of all, a *filter* removes junk pages that contain no person information. Secondly, a *cluster* groups the remaining pages into different clusters, each for one specific person (i.e. referent). To make the resulting clusters more meaningful in the context of person search, an *extractor* has been developed for inducing query-oriented personal information from each page. Finally, a *namer* generates an informative description (which consists of a name and a set of key words) for each cluster so that users can easily determine the specific person to which each cluster refers. Note that the system is currently focusing on English name resolution.

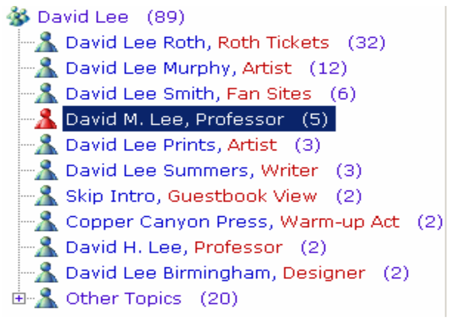

David Lee (89)
- David Lee Roth, Roth Tickets (32)
- David Lee Murphy, Artist (12)
- David Lee Smith, Fan Sites (6)
- David M. Lee, Professor (5)
- David Lee Prints, Artist (3)
- David Lee Summers, Writer (3)
- Skip Intro, Guestbook View (2)
- Copper Canyon Press, Warm-up Act (2)
- David H. Lee, Professor (2)
- David Lee Birmingham, Designer (2)
- Other Topics (20)

Figure 2: Referent list for the query "David Lee" in WebHawk.

Consider the example of "David Lee" aforementioned. The result of **WebHawk**, generated from the top-100 retrieved pages, is shown in Figure 2. We see that there are 11 junk pages being removed. For the remaining 89 pages, **WebHawk** groups them into a list of clusters, each for one referent. The clusters are ranked by the number of pages contained. The cluster named "Other topics" is a collection of single-page clusters. Each cluster is described by its name (i.e. person name like "David Lee Murphy") and a set of keywords that can best distinguish the referent from all of the others (e.g. David Lee Murphy is an "Artist"). Now, users can easily find the referent of each cluster (e.g. the professor named "David M. Lee" in Figure 2) without browsing pages.

In the rest of this paper, we first review related work in Section 2. Section 3 describes the corpora we used in our study. In Sections 4 to 7, we describe each of the four components (i.e., *filter*, *cluster*, *extractor* and *namer*) in detail, presenting a separate evaluation of each component where appropriate. In Section 8, we present a pilot user study, where **WebHawk** is compared with another state-of-the-art system. Results show that **WebHawk** complements most existing search engines and successfully improves users' experience of person search on the Web. Finally, we conclude the paper in section 9.

2. RELATED WORK

Mann and Yarowsky [11] employ a bottom-up centroid agglomerative clustering algorithm to generate person clusters based on extracted biographic features. However, the algorithm can only handle a small number of clusters, and has only been tested on some artificial test data. So it is questionable how well the method generalizes to real world problems.

Al-Kamha and Embley [1] also use an approach that clusters search results. But they use a different feature set, including attributes, links and page similarity. Then, a so-called relatedness confidence matrix is constructed for each page-pair using a probabilistic model. Clustering is performed by merging pairs whose matching confidence value is larger than a preset threshold.

Guha and Garg [6] follow a different scenario. Instead of the clustering algorithm, they use a re-ranking algorithm to disambiguate people. The algorithm requires a user to select one of the returned web pages as a start 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. In this scenario, a user needs to browse the documents in order to find one which matches the user's intended referent, which is inconvenient to the user.

Bekkerman and McCallum [3] present two unsupervised frameworks for finding those web pages referring to a particular person: one based on link structure and another using Agglomerative/Conglomerative Double Clustering (A/CDC). But their scenario focuses on simultaneously disambiguating an existing social network of people, or lists of people who are known to be somewhat connected, which is not the case for person search in reality.

On the one hand, **WebHawk** can be viewed as an extension of the above approaches. For example, **WebHawk** also uses a clustering algorithm to group pages based on extracted personal information, referred to as biographic features in [1, 6, 12]. On the other hand, **WebHawk** differs from the above work in that it has been designed as a *pragmatic* system where we take into account more pragmatic factors, among which three of the most important ones are as follows. First, we remove junk pages by a *filter*. This problem is not discussed thoroughly in previous works, but we think that it is very critical for any pragmatic person search system that is based on re-organizing search results of general search engines. Second, we perform an in-depth study of user-interface issues, such as how to provide an informative title/description for each cluster. Finally, **WebHawk** is tested on real data, and evaluated by real users, in comparison with other search systems.

In practice, some commercial systems, such as *Zoominfo* (www.zoominfo.com), have been developed to find people information. But these systems have high cost and low scalability because the person information in the systems is collected mainly by human labors.

In addition, our work is generally related to a number of other researches, e.g. object identification in [14, 16], citation matching in [12], and name matching in [7]. Search result clustering has been discussed in [4, 19, 20]. Person name resolution can also be formulated as a general co-reference resolution problem discussed in [8]. In particular, Wacholder et al. [18] describe how to resolve person names within one document. It has also been extended to the multi-document case in [2, 5, 13, 15].

3. DATA SET

We have selected the 200 most-frequent person queries from the log of *MSN*. For each page, we collected the top 100 web pages that are retrieved successfully by *MSN*.

Those pages have been manually annotated as follows. All junk pages (as defined in Section 4) have been labeled. We also assumed that one page refers to one referent, and removed all pages that refer to two or more referents. As a result, 2% of pages have been removed. All remaining person pages have been grouped into different clusters, each for one referent. For each person page, we have manually extracted a set of personal information features, including full person name, title, organization, email address, and phone number. The distribution of the numbers of referents for the 200 person queries is shown in Figure 1. The average number is 6.88.

We have separated the person queries into two data sets: 160 queries as the training set and the remaining 40 queries as the test set. The following evaluations are based on these two data sets, unless stated otherwise.

4. *FILTER*: REMOVING JUNK PAGES

4.1 Method

A junk page, in the context of person search, refers to a retrieved web page, with respect to a person query, where no occurrence of the person query in the page occurs as a (part of) person name.

Junk pages are retrieved because person names may refer to non-person entities such as products (e.g. Abercrombie, Bloomberg), companies (e.g. Ford, Disney), places (e.g. Edmunds), and nature/law systems (e.g. Claudette, White, Aarne). According to our study, junk pages amount to 35% in the top-100 retrieved pages, with respect to a person query. Therefore, junk page filtering is not only a critical initial step for subsequent processes such as clustering, but also prevents users from being detracted. Unfortunately, this problem has not been studied thoroughly in a systematic manner previously.

In **WebHawk**, a component called *filter* has been developed for this purpose, where junk page filtering is formulated as a binary classification problem. In our implementation, the *filter* employs a particular SVM classifier, SVM^{light} [9]. In the next subsection, we will discuss the features used in the classifier.

4.2 Evaluation

We evaluate the performance of the *filter* using the data set described in Section 3. In particular, we denote manually labeled junk pages as positive examples and other pages as negative examples. The evaluation metrics are standard precision (*p*), recall (*r*) and $F_1=2pr/(p+r)$. We explore the effectiveness of different feature sets in the *filter*. These feature sets are

1. **Simple lexical features**: These features include title words, meta words[1], and body text words. We use frequency as their real-valued weight.
2. **Stylistic lexical features**: These features include words in bold font, words in anchor text, and words in heading text. We use frequency as their real-valued weight.
3. **Query-relevant lexical features**: These features include words adjacent to any of the query terms (i.e. previous word and next word). We use frequency as their real-valued weight.
4. **Linguistic features**: These features include three real-valued numbers, which are the counts of person names, organizations, and locations extracted by an in-house named entity recognition (NER) tool.
5. **Query-relevant possessive feature**: This feature refers to the frequency of those occurrences of "'s" after query terms, e.g., "Bill's".
6. **Query-relevant abbreviation feature**: This feature refers to the number of those cases where a token of the form "X" or "X." appears before or next to any of the query terms. Here, X refers to any capital letter from "A" to "Z".

In our experiments, only the simple lexical features are used in the baseline system. Table 1 shows the results, where "w/o" means that we use all features except the specified features. We can observe from that both the lexical features (1-3) and the linguistic features (4) have a substantial positive impact on the performance while the query-relevant features (5 and 6) bring only marginal improvements. To further improve the performance of the *filter*, we will also employ new features such as link information and annotate more data for training.

Table 1: Results of junk page filtering

Feature Set	P	R	F_1
Baseline (w/ 1)	0.740	0.751	0.745
All features	0.803	0.774	**0.788**
w/o 1	0.712	0.738	0.725
w/o 2	0.751	0.802	0.776
w/o 3	0.778	0.773	0.776
w/o 4	0.746	0.775	0.760
w/o 5	0.783	0.777	0.780
w/o 6	0.785	0.775	0.780

5. *EXTRACTOR*: INDUCING PERSONAL INFORMATION

5.1 Method

Personal information includes person name, title, organization, email address and phone number. Intuitively, a combination of the above five biographical features can almost uniquely characterize a specific referent, and thus contribute most to the subsequent processes in **WebHawk,** such as clustering and naming.

[1] Meta words denote the words in the description metadata and keywords metadata in a web page, e.g. the italic words in the following html source: <meta name="description" content="*car company…*"> <meta name="keywords" content="*car…*">

Extracting personal information is similar to the problem of named entity recognition (NER), which is a long-standing research topic in natural language processing (NLP). However, it is challenging to extend traditional NER techniques to our case for two reasons. First, our task is query-oriented. That is, we only extract biographical attributes of a specific referent in a page, with respect to a person query. The second reason is that web data is very noisy, so traditional NLP techniques may not be robust enough. Our method uses a hybrid approach based on two techniques: *pattern matching* and *mutual reinforcement learning*.

Now, we discuss in detail how to extract each of the five biographical features.

We assume that terms in a person query form (partially) the person name and are key clues for person name extraction. Full English person names are usually composed of three fields: first name, middle name and last name, e.g. "Michael Jeffrey Jordan". Person names with two words or one word also appear frequently in documents, e.g. "Michael Jordan" and "Jordan". We prefer person names with more words because those names are more like full names and have a better distinguishing capability.

We can simply extract the full names from web pages given person queries with three words, but the difficult case lies in the person queries composed of only one or two words (e.g. "David" and ""David Lee"). We employ the technique of *mutual reinforcement learning*. Person names are extracted from a web page in the following three steps.

(1) To extract candidate person names from web text. Candidate names are those substrings that are composed of capitalized words and contain the query terms. These candidate person names may include incorrect names (e.g. "David Lee Homepage") and names of different persons (e.g. "David H. Lee", "David M. Lee" and "David Lee Arnold").

(2) To assign each candidate person name a saliency score. The score is a linear combination of heuristic score (see Table 2) and normalized frequency, with equal weights. The higher the score, the better the candidate name is.

Table 2: Heuristic scores for candidate person names

Query	Cap Words	Score
A B C [David Murphy Lee]	X Y Z (XYZ=ABC) [David Murphy Lee]	0.4
A B [David Lee]	X Y Z (X=A, Z=B) [David Murphy Lee]	0.3
	X Y Z (XY=AB or YZ=AB) [David Lee Arnold, David Lee Homepage]	0.1
	X Y (XY=AB) [David Lee]	0.2
A [David]	X Y Z (Y = M.) [David M. Lee]	0.3
	X Y Z [David Murphy Lee, David Lee Arnold]	0.1
	X Y [David Lee, David Homepage]	0.1
	X [David]	0.0

3) To adjust the scores of candidate names and choose the one with the highest score. We employ *mutual reinforcement learning* to simultaneously compute the scores of the bi-names (person name with two words) and the tri-names (person name with three words). The scores of uni-names (person name with one word) are not changed in this step. The technique of *mutual reinforcement learning* is based on the following assumption.

A bi-name should have a high saliency score if it appears in many tri-names with high saliency scores and if its two terms are boundaries of a tri-name with high saliency scores. Conversely, a tri-name should have a high saliency score if it contains many bi-names with high saliency scores and if its boundary words are just the two terms of a bi-name with high saliency score.

For each web page, we generate two sets of candidate names: the set of bi-names $B=\{b_1,...,b_n\}$ and the set of tri-names $T=\{t_1,...,t_m\}$. We build a weighted bipartite graph from B and T as follows: we create an edge between b_i and t_j and specify nonnegative weights w_{ij} on the edge indicating the relation between them ($w_{ij}=0.1$ if b_i appears in t_j and $w_{ij}=0.5$ if terms in b_i are boundaries of t_j). We denote the weighted bipartite graph by $G(B,T,W)$ where $W=[w_{ij}]$ is the m-by-n weight matrix containing all the pairwise edge weights. For each bi-name b_i and each tri-name t_j we wish to compute their saliency scores $u(b_i)$ and $v(t_j)$, respectively. So the above principle is rendered as

$$u(b_i) \propto \sum_j w_{ij} v(t_j), \quad v(t_j) \propto \sum_i w_{ij} u(b_i)$$

We collect the saliency scores for bi-names and tri-names into two vectors u and v, respectively. The above equation can then be written in the following matrix format

$$u = \frac{1}{\sigma} W v, v = \frac{1}{\sigma} W^T u$$

where $\frac{1}{\sigma}$ is the proportionality constant.

We alternate computing and normalizing scores of bi-names and tri-names according to the above equation until convergence. At last we choose the person name with the highest saliency score.

Different from person name extraction, title, organization, email address and phone number are extracted using pattern matching. We manually author a set of extraction patterns for email address and phone number due to their simplicity. As for title and organization, we employ a simple learning method to collect extraction patterns. First, we select sentences in the annotated training corpus and tokenize them, smooth variations, and then subject them to simple generalization to constitute candidate patterns. For example, in this sentence "David Lee is a painter", "painter" is annotated as a title, so we replace "David Lee" by "<person>" and generate a candidate pattern like "<person> is a <title>". Then, all these patterns are applied to the same training corpus to evaluate their accuracies. Those patterns with high accuracy are adopted to extract title and organization. The numbers of patterns for title, organization, email address and phone number are 13, 11, 5 and 4, respectively.

Some patterns can handle most cases except a few peculiar instances, e.g. "webmaster@x..x" is not an appropriate email address though it conforms to the pattern of "x..x@x..x". Here, x represents any character. These negative examples are manually induced to form total 25 patterns, named *anti-patterns*, and we use these patterns to avoid similar extraction errors.

With these patterns, we might extract more than one title or organization for a web page in that various patterns could be applied appropriately. Then we use anti-patterns to filter the inappropriate instances. In order to improve the accuracy of title extraction, we have built a person title gazetteer beforehand, and those extracted titles not in this gazetteer will be excluded. Lastly we rank the remaining instances by frequency and select the best ones.

5.2 Evaluation

In our experiments, standard precision (p), recall (r) and $F_1=2pr/(p+r)$ are used as evaluation metrics. The baseline method of extracting personal information is to select the most frequent one from the corresponding set of candidate entities. The candidate sets for different types of personal information are produced as follows, respectively:

- Person name candidates are those names which are extracted by the in-house NER tool and contain query terms;
- Organization candidates are all organizations extracted by the in-house NER tool;
- Title candidates are those terms co-occur both in the web page and in the person title gazetteer;
- Email address candidates are those substrings containing character "@";
- Phone number candidates are those substrings containing only numbers and hyphen and having at least 6 characters.

Table 3 compares the performance of the baseline methods with our methods. In the first column, # denotes the number of names (or titles, etc.) in the test set. For name extraction, "w/o step 3" means that we only apply the first two steps to assign saliency scores to candidate names, and "w/ step3" means that mutual reinforcement learning is employed to adjust those scores. We can see that the learning step improves the performance of name extraction.

Table 3: Results of Personal Information Extraction

category		P	R	F_1
Name (# =133)	baseline	85.8%	58.8%	0.698
	w/o step 3	91.0%	90.0%	0.905
	w/ step 3	**93.0%**	**92.3%**	**0.926**
Title (# =127)	baseline	30.7%	30.7%	0.307
	WebHawk	**62.5%**	**55.1%**	**0.586**
Organization (# = 88)	baseline	5.6%	9.5%	0.070
	WebHawk	**22.1%**	**33.0%**	**0.265**
Email Address (# =27)	baseline	50.4%	66.7%	0.574
	WebHawk	**92.3%**	**88.9%**	**0.906**
Phone Number (# =48)	baseline	12.0%	25.0%	0.162
	WebHawk	**84.3%**	**89.6%**	**0.869**

As shown in Table 3, our approaches significantly outperform the baseline methods. In our pilot study, we have found that though the in-house NER tool achieves a good performance on the WSJ corpus, it performs much worse on web pages due to the complexity and irregularity of web pages. For example, a web page has a person name "Ashton Kutcher" in heading text, but there are no contextual words surrounding it and the terms in this name do not exist in the name list of the NER tool. So, the NER tool cannot recognize it as a person name. While in our approach, the query "Ashton Kutcher" is a good contextual clue for distinguishing this person name. Overall, the performance of the pattern matching approach to extraction of title, email address and phone number benefits from the use of query terms as contextual clues and the use of anti-patterns for filtering noisy answers. However, organization extraction is still a difficult task because it is hard to determine the boundary of organization.

6. *CLUSTER*: RESOLVING REFERENTS

6.1 Method

Cluster is used to group person pages into different clusters, each for one specific person. We use the agglomerative clustering algorithm to produce clusters in a bottom-up way as follows:

Initially, each web page is an individual cluster; then we iteratively merge two clusters with the largest similarity value to form a new cluster until this similarity value is below a pre-set merging threshold. The merging threshold can be determined through cross-validation. We employ the widely used average-link method to compute the similarity between two clusters as follows:

$$sim(c_1, c_2) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} sim(p_i, p_j)}{m \times n}$$

where p_i, p_j are web pages in cluster c_1 and cluster c_2, and m is the number of web pages in cluster c_1 and n is the number of web pages in cluster c_2.

The principal problem of using the clustering algorithm described above is how to measure the similarity between two web pages p_i and p_j. Different types of features extracted from web pages are explored in the experiments, including lexical features, linguistic features and personal information (i.e. PersonInfo) extracted as described in Section 5. The lexical features include title words, meta words and text words (We consider both unigrams and bigrams). The linguistic features include baseNP (base Noun Phrase) and NE (Named Entity, including person, organization and location), which are produced by the in-house NER and baseNP recognizer. Here, the NE features refer to all named entities extracted from the web page by the tool, no matter whether they are correct or who they are related to. The baseNP features refer to all baseNPs extracted from the web page.

For each type of features, we generate a feature vector for a web page and the weight of a feature unit is its frequency. Take NE features as an example. The vector is composed of all named entities in a web page. Having generated those feature vectors for two web pages, we use the cosine measure to calculate similarity value between each pair of the same typed feature vectors. We then linearly combine such similarity values to get the final similarity value. The weights for different types of features are hard to be estimated empirically, so we use the perceptron algorithm with uneven margins (PAUM) [10] to estimate them. The PAUM is an extension of the perceptron algorithm specially designed to cope with two class problems where positive examples are rare compared to negative ones, which is suitable for the clustering context.

6.2 Evaluation

6.2.1 Metrics

In the test set, each query corresponds to several non-overlapping clusters annotated by hand. For simplicity, the manually annotated clusters are called classes and the automatically generated clusters are called clusters. Our evaluation involves two steps: First, we evaluate the performance for each query. Second, we average the results over the 40 queries in test set.

For each cluster of one query, we calculate the recall and precision of that cluster for each given class. More specifically, for cluster j and class i

$$Recall(i,j) = n_{ij}/n_i, \quad Precision(i,j) = n_{ij}/n_j$$

where n_{ij} is the number of common members in class i and cluster j, n_j is the number of members of cluster j and n_i is the number of members of class i.

The F measure of cluster j and class i is then calculated by

$F(i,j) = (2*Precision(i,j)*Recall(i,j))/(Precision(i,j)+Recall(i,j))$

For an entire clustering for the query, the F measure of any class is the maximum value it attains at any cluster and an overall value for the F measure is computed by taking the weighted average of all values for the F measure as follows.

$$F = \sum_i \frac{n_i}{n} \max \{F(i,j)\}$$

where the *max* is taken over all clusters and n is the number of all web pages for the query.

After we get the performance values of all 40 queries, we average the values to produce the overall performance value.

6.2.2 Results

In the experiments, we compare the performance of the *clusters* using different types of features and explore the influence of the *filter* on the performance. The comparison results are shown in Figure 3. In the figure, "all data" refers to all retrieved web pages including junk pages, "clean data" refers only to person pages and "auto-cleaned data" refers to the remaining web pages after applying filtering, which may include both person pages and junk pages.

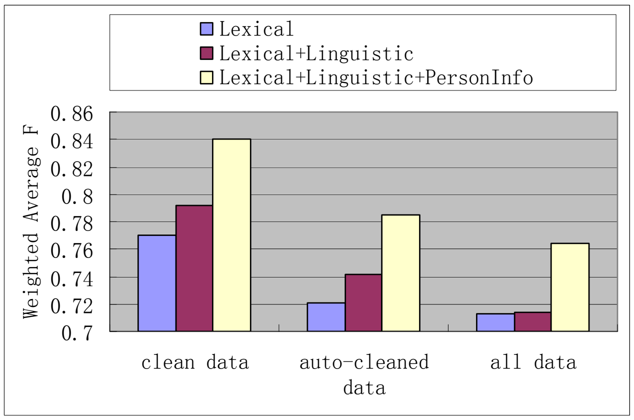


Figure 3: Performance comparison for different feature sets over different data sets.

From Figure 3 we can see that linguistic features slightly improve the performance, while automatically extracted personal information contributes substantially to the performance over all kinds of data. With linguistic features and personal information, the performance on clean data is 7% higher than that with only lexical features. The results validate the intuition that personal information can almost uniquely characterize a specific referent.

We can observe that given the same feature set, the gap between the performance on clean data and the performance on all data is large. It demonstrates that junk pages deteriorate significantly the performance and junk page filtering is necessary for person resolution. The *filter* that we have developed, though by no means perfect, can already improve the visible performance across all types of features.

7. *NAMER*: GENERATING DESCRIPTION

Namer is used to generate for each cluster an informative description so that users can find any specific person easily.

Here we propose a method to name the cluster concisely and informatively. We define a name template which consists of two parts: full person name and informative term. The informative term is content-focused and contains information unique to a particular referent. Title is used as a preferential informative term. The filling of this template consists of two steps: candidate generation and ranking.

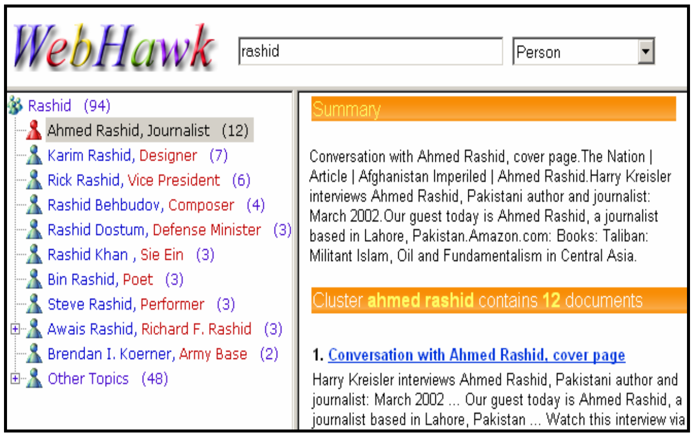
1) In the candidate generation process, we collect those names and titles, which have been extracted from web pages, as the candidates.

2) In the ranking process, we rank the candidate names and titles by their frequencies in the cluster and then fill the template with the most frequent ones. Note that the *extractor* may not extract any title from the web pages in some clusters. In such a case, as backoff, we use the most salient word or phrase in the cluster as the informative term. For example, for "David Lee Murphy", we use the title "Artist" as the informative term, but for "David Lee Smith", we use the phrase "Fan Sites" as the backoff informative term. The most salient term is the highest weighted term in the set of uni-grams and bi-grams extracted from those web pages, after excluding stop words and all bi-grams containing stop words. We assign the weight of a uni-gram as its frequency and the weight of a bi-gram as the double of its frequency.

The name for a cluster is very concise. Users may want to take a look in more detail at the cluster. So we provide a short summary as a supplement. The summary is produced based on a simple sentence extraction method. The details are skipped due to the page limit. The summary is located in the top of the right frame of the user interface of **WebHawk** as shown in Figure 4.

8. USER STUDY

In order to assess how easily and effectively to use **WebHawk** for person search on the web, we perform a pilot user study. We compare **WebHawk** with traditional *MSN* and the award-winning *Vivisimo*. *MSN* returns a ranked list of search results while both *Vivisimo* and **WebHawk** show clusters in the left pane and a ranked list of documents in the right pane (as shown in Figures 4-5). Because *Vivisimo* can combine various results returned by a variety of search engines, in order to evaluate the three systems on the same data source, we select *MSN* as the underlying search engine for *Vivisimo* in the user study. Note that previous work in [1, 3, 6, 11] does not provide available pragmatic systems for person search, the comparison between **WebHawk** and those works through user study is impossible. The **WebHawk**'s system implementation details, which are important for real-time person search, are omitted due to the space limit.



Rashid (94)
- Ahmed Rashid, Journalist (12)
- Karim Rashid, Designer (7)
- Rick Rashid, Vice President (6)
- Rashid Behbudov, Composer (4)
- Rashid Dostum, Defense Minister (3)
- Rashid Khan , Sie Ein (3)
- Bin Rashid, Poet (3)
- Steve Rashid, Performer (3)
- Awais Rashid, Richard F. Rashid (3)
- Brendan I. Koerner, Army Base (2)
- Other Topics (48)

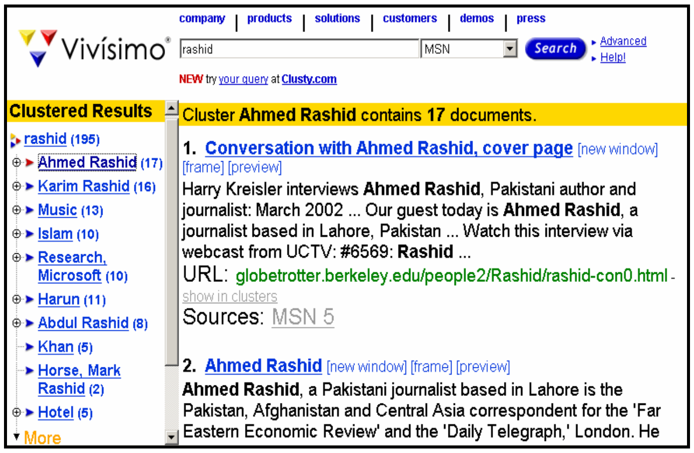
Conversation with Ahmed Rashid, cover page.The Nation | Article | Afghanistan Imperiled | Ahmed Rashid.Harry Kreisler interviews Ahmed Rashid, Pakistani author and journalist: March 2002.Our guest today is Ahmed Rashid, a journalist based in Lahore, Pakistan.Amazon.com: Books: Taliban: Militant Islam, Oil and Fundamentalism in Central Asia.

Cluster **ahmed rashid** contains **12** documents

1. Conversation with Ahmed Rashid, cover page

Harry Kreisler interviews Ahmed Rashid, Pakistani author and journalist: March 2002 ... Our guest today is Ahmed Rashid, a journalist based in Lahore, Pakistan ... Watch this interview via

Figure 4: The user interface of WebHawk.



Cluster **Ahmed Rashid** contains **17** documents.

1. Conversation with Ahmed Rashid, cover page [new window] [frame] [preview]
Harry Kreisler interviews **Ahmed Rashid**, Pakistani author and journalist: March 2002 ... Our guest today is **Ahmed Rashid**, a journalist based in Lahore, Pakistan ... Watch this interview via webcast from UCTV: #6569: **Rashid** ...
URL: globetrotter.berkeley.edu/people2/Rashid/rashid-con0.html - show in clusters
Sources: MSN 5

2. Ahmed Rashid [new window] [frame] [preview]
Ahmed Rashid, a Pakistani journalist based in Lahore is the Pakistan, Afghanistan and Central Asia correspondent for the 'Far Eastern Economic Review' and the 'Daily Telegraph,' London. He

Figure 5: The user interface of *Vivisimo*.

Prior to the user study, a list of 12 tasks was developed. Each of them was to find a specific piece of information about a specific person. Each task had only one correct answer. The person queries were selected from *MSN's* logs and the specific information need for each person query was designed. It was guaranteed that the correct answer could be found from the top 100 web pages returned by *MSN*. For example, one of the tasks was to *find the name of the college where a person named "Michael Williams", the vice president of Viral Products, got his B.S. degree,* and the correct answer was *Lynchburg College*.

The usability study metrics were as follows:

1) Effectiveness metric - accuracy: This was the percentage of tasks successfully completed by a system. A task could be completed successfully, completed unsuccessfully or aborted. All these occurrences were recorded. A successfully completed task was the one where a user completed the task and obtained the correct answer. An unsuccessfully completed task was the one where a user completed the task but obtained the incorrect answer. An aborted task was the one that a user quit while performing the task.

2) Efficiency metric - completion time: This was the time taken for a user to complete a task using the system.

3) Subjective acceptance: This relates a user's subjective satisfactory level by asking the user to fill out pre-designed questionnaires.

36 students from different departments were employed as subjects for performing person searches and filling out questionnaires. A Graeco-Latin Square was used to establish task order for each participant and to confound task order effects. According to the task order, each subject carried out all 12 searches, four on *MSN*, four on *Vivisimo* and four on **WebHawk**. Note that besides the provided person name query, subjects were allowed to design and use any other queries they wished on *MSN* or *Vivisimo* during a search task. Each search task had to be finished in 10 minutes[2]; otherwise the task was considered to be aborted. If the user already knew the answer, she or he still had to perform the search and find it. The start time and the end time of each task were recorded to measure the completion time if the task was not aborted. The obtained answer for each task was saved and then compared with the correct answer. The averaged completion time and the accuracy were obtained for each participant and then the values were averaged across all the participants, as shown in Table 4.

[2] The time limit of 10 minutes is defined in the TREC 2002 Interactive Track Guidelines (http://trec.nist.gov/data/t11_interactive/guidelines.html).

As a complement to the objective measures, an exit-system questionnaire was designed to gauge a user's overall acceptance of a system after the user performed four tasks on it. The questionnaire asked subjects to assess each system in the following four aspects:

- Ease of use: *How easy the system was to use?*
- Informativeness: *Were the cluster descriptions produced by the system (i.e. Vivisimo or **WebHawk**) informative enough to help the search?*
- Satisfaction: *Were you satisfied with your search experience with the system?*
- Preference: *Did you prefer to use the system when you searched specific information about a person?*

Subjects were required to express an opinion over a 5-point scale for each of the above questions, where 1 stood for "*not at all*", 3 for "*somewhat*" and 5 for "*extremely*". We collected the responses of subjects and averaged them, as shown in Table 5.

Table 4 shows that completion time for **WebHawk** was the lowest and the accuracy for **WebHawk** was a little higher than other systems. As can be seen in Table 5, both *Vivisimo* and **WebHawk** were a little more difficult to use than traditional *MSN* in that the two-pane-based user interface was more complex than the simple ranked list. **WebHawk** produced an informative person description for each cluster and these descriptions helped the person search. Lastly, users were more satisfied with **WebHawk** and preferred to use **WebHawk** for person search.

This user study showed that **WebHawk** was efficient and effective and could improve users' search experience for person search, which could be attributed to its good performance for person resolution and its ability to provide an informative interface.

Table 4: Averaged completion time and accuracy

	MSN	*Vivisimo*	WebHawk
Accuracy	78.2%	79.5%	82.0%
Complete time (Sec)	203.5	185.6	144.4

Table 5: Subject's responses per system to exit-system questionnaires averaged across query and subject on a scale of 1 (worst) to 5 (best)

	MSN	*Vivisimo*	WebHawk
Ease of use	4.3	3.4	3.3
Informativeness	–	2.9	3.8
Satisfaction	3.2	3.6	4.1
Preference	3.5	3.7	4.1

9. CONCLUSIONS AND FUTURE WORK

We have presented an effective system for person resolution in person search results, called **WebHawk**. It is composed of four components: *filter*, *extractor*, *cluster* and *namer*. The experiments and results have shown the performance of each component and demonstrated that personal information extracted by the *extractor* improves the performance substantially and the *filter* does benefit the system by removing noisy data. It is verified through a user study that users' person search experience is indeed improved by **WebHawk**. Either by simply providing a search option on the search interface indicting whether the search is a person search or a general search, or by providing an automatic person query identification mechanism, general search engines can integrate with **WebHawk** easily and issue person queries to the person search engine-**WebHawk**.

We have attempted a new method where we categorized person pages into more elaborate sub-categories (e.g. newswire, citation list, short bios, etc.), clustered web pages within each category, and combined these clusters. The intuition underlying this method is that each kind of web pages has their own characteristics and has better be clustered using their own distinguishing features. This method however did not improve the performance as we expected in our pilot experiments.

We will explore new clustering scenarios in our future work. New features, e.g. link information, will also be explored for different components in **WebHawk**. At present, **WebHawk** focuses only on English person names and we will adapt the system to other languages in the near future.

10. ACKNOWLEDGMENTS

We thank Tom Huang, Chuan Lin and John Chen for their earlier work and are grateful to Jian-Yun Nie and the anonymous reviewers for their helpful suggestions. We also thank Mary D. Taffet for her kindly help to provide her dissertation proposal and other valuable references. Lastly, we thank those part-time students for data annotation and user study.

11. REFERENCES

[1] R. Al-Kamha and D. W. Embley. Grouping search-engine returned citations for person-name queries. In *Proceedings of WIDM'04, Washington*, DC, USA, pp. 96-103, 2004.

[2] A. Bagga and B. Baldwin. Entity-based cross-document co-referencing using the vector space model. In *Proceedings of COLING-ACL'98*, pp. 79–85, 1998.

[3] R. Bekkerman and A. McCallum. Disambiguating web appearances of people in a social network. In *Proceedings of WWW'05*, pp. 463-470, 2005.

[4] D. Cutting, D. Karger, J. Pedersen and J. W. Tukey. Scatter/Gather: a cluster-based approach to browsing large document collections. In *Proceedings of SIGIR'92*, Copenhagen, 1992.

[5] M. B. Fleischman and E. Hovy. Multi-document person name resolution. In *Proceedings of the Workshop on Reference Resolution and its Applications: ACL'04*, Barcelona, 2004.

[6] R. Guha and A. Garg. Disambiguating people in search. Stanford University, 2004.

[7] H. Han, L. Giles, H. Y. Zha, et al. Two supervised learning approaches for name disambiguation in author citations. In *Proceedings of JCDL' 04*, Tucson, Arizona, USA, pp. 296-305, 2004.

[8] J. R. Hobbs. Resolving pronoun references. *Readings in Natural Language Processing*, Los Altos, CA: Morgan Kaufmann Publishers, pp. 339-352, 1978.

[9] T. Joachims. Making large-scale svm learning practical. In *Advances in Kernel Methods – Support Vector Learning*. MIT-Press, 1999.

[10] Y. Y. Li, H. Zaragoza, R. Herbrich, et al. The perceptron algorithm with uneven margins. In *Proceedings of ICML'02*, 2002.

[11] G. S. Mann and D. Yarowsky. Unsupervised personal name disambiguation, In *Proc. of CoNLL'03*, Edmonton, Canada, 2003.

[12] A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Knowledge Discovery and Data Mining*, pp. 169–178, 2000.

[13] C. Niu, W. Li and R. K. Srihari. Weakly supervised learning for cross-document person name disambiguation supported by information extraction. In *Proceedings of ACL'2004*.

[14] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *Advances in Neural Information Processing Systems 15*, MIT Press, 2003.

[15] Y. Ravin and Z. Kazi. Is Hillary Rodham Clinton the President? Disambiguating names across documents. *Proceedings of the ACL '99 Workshop on Coreference and its Applications*, pp. 9-16, 1999.

[16] S. Russell. Identity uncertainty. In *Proceedings of IFSA'01*, Vancouver, 2001.

[17] M. D. Taffet. Person resolution: resolving multireferent and multimorphic person names within and across full text documents. *Dissertation Proposal*, Syracuse University, 2004.

[18] N. Wacholder, Y. Ravin, and M. Choi. Disambiguation of proper names in text. In *Proceedings of ANLP'1997*.

[19] O. Zamir and O. Etzioni. Web document clustering: a feasibility demonstration, In *Proceedings of SIGIR'1998*.

[20] H. J. Zeng, Q. C. He, Z. Chen, W. Y. Ma and J. W. Ma. Learning to cluster web search result. In *Proceedings of SIGIR'04*, Sheffield, UK, 2004.