Automatically Extracting Personal Name Aliases from the Web

Danushka Bollegala*, Taiki Honma, Yutaka Matsuo, and Mitsuru Ishizuka

The University of Tokyo, Hongo 7-3-1, Tokyo, 113-8656, Japan
{danushka,honma}@mi.ci.i.u-tokyo.ac.jp,
matsuo@biz-model.t.u-tokyo.ac.jp,
ishizuka@i.u-tokyo.ac.jp

Abstract. Extracting aliases of an entity is important for various tasks such as identification of relations among entities, web search and entity disambiguation. To extract relations among entities properly, one must first identify those entities. We propose a novel approach to find aliases of a given name using automatically extracted lexical patterns. We exploit a set of known names and their aliases as training data and extract lexical patterns that convey information related to aliases of names from text snippets returned by a web search engine. The patterns are then used to find candidate aliases of a given name. We use anchor texts to design a word co-occurrence model and use it to define various ranking scores to measure the association between a name and a candidate alias. The ranking scores are integrated with page-count-based association measures using support vector machines to leverage a robust alias detection method. The proposed method outperforms numerous baselines and previous work on alias extraction on a dataset of personal names, achieving a statistically significant mean reciprocal rank of 0.6718. Experiments carried out using a dataset of location names and Japanese personal names suggest the possibility of extending the proposed method to extract aliases for different types of named entities and for other languages. Moreover, the aliases extracted using the proposed method improve recall by 20% in a relation-detection task.

1 Introduction

Precisely identifying entities in web documents is necessary for various tasks such as relation extraction [16], search and integration of data [9] and entity disambiguation [14]. Nevertheless, identification of entities on the web is difficult for two fundamental reasons: first, different entities can share the same name (lexical ambiguity); secondly, a single entity can be designated by multiple names (referential ambiguity). As an example of lexical ambiguity the name *Jim Clark* is illustrative. Aside from the two most popular namesakes, the formula-one racing champion and the founder of Netscape, at least 10 different people are listed among the top 100 results returned by Google for the name. On the other

* Research Fellow of the Japan Society for the Promotion of Science (JSPS).

A. Ranta, B. Nordström (Eds.): GoTAL 2008, LNAI 5221, pp. 77–88, 2008.

hand, referential ambiguity occurs because people use different names to refer to the same entity on the web. For example, the American movie star *Will Smith* is often called the *the Fresh Prince* in web contents. Although lexical ambiguity, particularly ambiguity related to personal names, has been explored extensively in the previous studies of name disambiguation [14,4], the problem of referential ambiguity of entities on the web has received much less attention. In this paper, we specifically examine on the problem of automatically extracting the various references on the web to a particular entity.

For an entity e, we define the set A of its aliases to be the set of all words or multi-word expressions that are used to refer to e on the web. For example, *Godzilla* is a one-word alias for *Hideki Matsui*, whereas the alias *the Fresh Prince* contains three words and refers to *Will Smith*. Various types of terms are used as aliases on the web. For instance, in the case of an actor, the name of a role or the title of a drama (or a movie) can later become an alias for the person (e.g., *Fresh Prince, Knight Rider*). Titles or professions such as *president*, *doctor*, *professor*, etc. are also frequently used as aliases. Variants or abbreviations of names such as *Bill* for *William* and acronyms such as *J.F.K.* for *John Fitzgerald Kennedy* are also types of name aliases that are observed frequently on the web.

Identifying aliases of a name is important for extracting relations among entities. For example, Matsuo et al. [16] propose a social network extraction algorithm, in which they compute the strength of the relation between two individuals A and B by the web hits for the conjunctive query, *"A" AND "B"*. However, both persons A and B might also appear in their alias names in web contents. Consequently, by expanding the conjunctive query using aliases for the names, a social network extraction algorithm can accurately compute the strength of a relationship between two persons.

Searching for information about people on the web is an extremely common activity of Internet users. Around 30% of search engine queries include personal names [1]. However, retrieving information about a person merely using his or her real names is insufficient when that person has nicknames. Particularly with keyword-based search engines, we will only retrieve pages which use the real name to refer to the person about whom we are interested in finding information. In such cases, automatically extracted aliases of the name are useful to expand a query in a web search, thereby improving recall.

Our contributions in this paper are two fold:

- We propose a lexical pattern-based approach to extract aliases of a given name using snippets returned by a web search engine. We propose an algorithm to automatically generate lexical patterns using a set of real-world name-alias data.
- To select the best aliases among the extracted candidates, we propose numerous ranking scores based upon two approaches: a word co-occurrence model using anchor texts, and page-counts returned by a search engine. Moreover, using real world name alias data, we train a ranking support vector machine to learn the optimal combination of individual ranking scores to leverage a robust alias extraction method.

2 Related Work

The problem of extracting aliases of a given name can be considered as a special case of the more general problem of extracting the words Y that have a given relation R with a word X. For example, extracting hyponyms [10], synonyms [13], meronyms [5] are specific instances of this general problem of relation extraction. Manually created or automatically extracted lexico-syntactic patterns have been successfully used to identify various relations between words [17,18]. For example, patterns such as *X is a Y* and *X such as Y* are typically used to introduce hypernyms, whereas, *X of a Y* and *X's Y* are frequently used with meronyms. However, alias extraction poses several unique challenges that separates it from the more general relation extraction problem. Firstly, personal names and their aliases are not typically listed in manually created dictionaries. Therefore, an alias extraction algorithm must first extract a possible set of candidate aliases for a given name and then verify each extracted candidate. Secondly, names and aliases can be multi-word expressions. For example, in the case of *Will Smith*, who has a two-word alias *fresh prince*, it is inaccurate to extract *fresh* as an alias. Thirdly, unlike hypernyms or meronyms, it is not obvious as to which lexical patterns convey useful clues related to aliases of a given name. This makes it difficult to manually create a sufficiently large list of lexical patterns to cover various types of name aliases. In addition to above mentioned challenges, the lack of evaluation benchmark dataset for aliases makes it difficult to compare and evaluate different approaches. Although it is relatively easy to manually verify whether an extracted candidate is a correct alias of a given name, it is not always possible to obtain a list of all the aliases of a name, which makes it difficult to compute the recall or coverage of an alias extraction algorithm.

Alias identification is closely related to the problem of cross-document coreference resolution, in which the objective is to determine whether two mentions of a name in different documents refer to the same entity. Bagga and Baldwin [3] proposed a cross-document coreference resolution algorithm by first performing within-document coreference resolution for each individual document to extract coreference chains, and then clustering the coreference chains under a vector space model to identify all mentions of a name in the document set. However, the vastly numerous documents on the web render it impractical to perform within-document coreference resolution to each document separately and then cluster the documents to find aliases.

In personal name disambiguation the goal is to disambiguate various people that share the same name (*namesakes*) [14,4]. Given an ambiguous name, most name disambiguation algorithms have modeled the problem as one of document clustering, in which all documents that discuss a particular individual of the given ambiguous name are grouped into a single cluster. However, the name disambiguation problem differs fundamentally from that of alias extraction because, in name disambiguation the objective is to identify the different entities that are referred by the same ambiguous name; in alias extraction, we are interested in extracting all references to a single entity from the web.

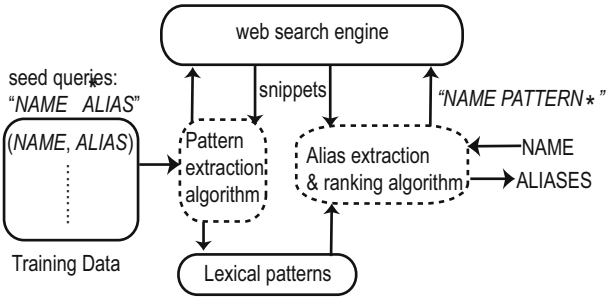

Fig. 1. Outline of the proposed method

Approximate string matching algorithms have been used for extracting variants or abbreviations of personal names (e.g. matching *Will Smith* with the first name initialized variant *W. Smith*) [8]. Rules in the form of regular expressions and edit-distance-based methods have been used to compare names. However, an inherent limitation of such string matching approaches is that they cannot identify aliases which share no words or letters with the real name. For example, approximate string matching methods would not identify *Fresh Prince* as an alias for *Will Smith*.

Hokama and Kitagawa [11] propose an alias extraction method that is specific to the Japanese language. For a given name p, they search for the query *"* koto p"* and extract the context that matches the asterisk. The Japanese word *koto*, roughly corresponds to *also known as* in English. However, *koto* is a highly ambiguous word in Japanese that can also mean *incident, thing, matter, experience* and *task*. As reported in their paper, many noisy and incorrect aliases are extracted using this pattern, which requires various post-processing heuristics that are specific to Japanese language to filter-out the incorrect aliases. Moreover, manually crafted patterns do not cover various ways that convey information about name aliases. In contrast, we propose a method to leverage such lexical patterns automatically using a training dataset of names and aliases.

3 Method

The proposed method is outlined in Fig.1 and comprises two main components: pattern extraction, and alias extraction and ranking. Using a seed list of name-alias pairs, we first extract lexical patterns that are frequently used to convey information related to aliases on the web. The extracted patterns are then used to find candidate aliases for a given name. We define various ranking scores using the hyperlink structure on the web and page counts retrieved from a search engine to identify the correct aliases among the extracted candidates.

3.1 Extracting Lexical Patterns from Snippets

Many modern search engines provide a brief text snippet for each search result by selecting the text that appears in the web page in the proximity of the query.

Such snippets provide valuable information related to the local context of the query. For names and aliases, snippets convey useful semantic clues that can be used to extract lexical patterns that are frequently used to express aliases of a name. For example, consider the snippet returned by Google[1] for the query *"Will Smith * The Fresh Prince"*.

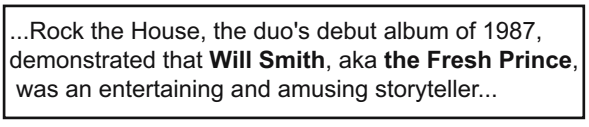

Fig. 2. A snippet returned for the query *"Will Smith * The Fresh Prince"* by Google

Here, we use the wildcard operator * to perform a *NEAR* query and it matches with one or more words in a snippet. In Fig.2 the snippet contains *aka* (i.e. *also known as*), which indicates the fact that *fresh prince* is an alias for *Will Smith*. In addition to *a.k.a.*, numerous clues exist such as *nicknamed, alias, real name is, nee*, which are used on the web to represent aliases of a name. Consequently, we propose the shallow pattern extraction method illustrated in Fig.3 to capture the various ways in which information about aliases of names is expressed on the web. Lexico-syntactic patterns have been used in numerous related tasks such as extracting hypernyms [10] and meronyms.

Given a set S of (NAME, ALIAS) pairs, the function *ExtractPatterns* returns a list of lexical patterns that frequently connect names and their aliases in web-snippets. For each (NAME, ALIAS) pair in S, the *GetSnippets* function downloads snippets from a web search engine for the query *"NAME * ALIAS"*. Then, from each snippet, the *CreatePattern* function extracts the sequence of words that appear between the name and the alias. Results of our preliminary experiments demonstrated that consideration of words that fall outside the name and the alias in snippets did not improve performance. Finally, the real name and the alias in the snippet are respectively replaced by two variables [**NAME**] and [**ALIAS**] to create patterns. For example, from the snippet shown in Fig.2, we extract the pattern [**NAME**] *aka* [**ALIAS**]. We repeat the process described above for the reversed query, *"ALIAS * NAME"* to extract patterns in which the alias precedes the name.

Once a set of lexical patterns is extracted, we use the patterns to extract candidate aliases for a given name as portrayed in Fig.4. Given a name, $NAME$ and a set, P of lexical patterns, the function *ExtractCandidates* returns a list of candidate aliases for the name. We associate the given name with each pattern, p in the set of patterns, P and produce queries of the form: *"NAME p *"*. Then the *GetSnippets* function downloads a set of snippets for the query. Finally, the *GetNgrams* function extracts continuous sequences of words (n-grams) from the beginning of the part that matches the wildcard operator *. Experimentally, we selected up to 5-grams as candidate aliases. Moreover, we removed candidates

[1] www.google.com

Algorithm 1: ExtractPatterns(S)

comment: S is a set of (NAME, ALIAS) pairs

$P \leftarrow null$
for each $(NAME, ALIAS) \in S$
  do $\begin{cases} D \leftarrow \text{GetSnippets}(\text{“}NAME \ * \ ALIAS\text{”}) \\ \textbf{for each} \text{ snippet } d \in D \\ \quad \textbf{do } P \leftarrow P + \text{CreatePattern}(d) \end{cases}$
return (P)

Fig. 3. Given a set of (NAME, ALIAS) instances, extract lexical patterns

Algorithm 2: ExtractCandidates($NAME, P$)

comment: P is the set of patterns

$C \leftarrow null$
for each pattern $p \in P$
  do $\begin{cases} D \leftarrow \text{GetSnippets}(\text{“}NAME \ p \ *\text{”}) \\ \textbf{for each} \text{ snippet } d \in D \\ \quad \textbf{do } C \leftarrow C + \text{GetNgrams}(d, NAME, p) \end{cases}$
return (C)

Fig. 4. Given a name and a set of lexical patterns, extract candidate aliases

that contain only stop words such as *a, an*, and *the*. For example, assuming that we retrieved the snippet in Fig.3 for the query *“Will Smith aka *”*, the procedure described above extracts *the fresh* and *the fresh prince* as candidate aliases.

3.2 Ranking of Candidates

Considering the noise in web-snippets, candidates extracted by the shallow lexical patterns might include some invalid aliases. From among these candidates, we must identify those which are most likely to be correct aliases of a given name. We model this problem of alias recognition as one of ranking candidates with respect to a given name such that the candidates which are most likely to be correct aliases are assigned a higher rank. First, we define various ranking scores to measure the association between a name and a candidate alias using two approaches: co-occurrences in inbound anchor texts of a url and page-counts retrieved from a search engine. Next, we integrate those ranking scores using ranking support vector machines (SVMs) [12] to leverage a robust ranking function.

3.3 Co-occurrences in Anchor Texts

Anchor texts have been studied extensively in information retrieval and have been used in various tasks such as synonym extraction, query translation in

cross-language information retrieval, and ranking and classification of web pages [7], However, anchor texts have not been exploited fully in Semantic Web applications. We revisit anchor texts to measure the association between a name and its aliases on the web. Anchor texts pointing to a url provide useful semantic clues related to the resource represented by the url. For example, if the majority of inbound anchor texts of a url contain a personal name, it is likely that the remainder of the inbound anchor texts contain information about aliases of the name.

We define a name p and a candidate alias x as *co-occurring*, if p and x appear in two different inbound anchor texts of a url u. Moreover, we define *co-occurrence frequency* (**CF**) as the number of different urls in which they co-occur. We can use this definition to create a contingency table like that shown in Table 1. Therein, C is the set of candidates extracted by the algorithm described in Fig.4, V is the set of all words that appear in anchor texts, $C - \{x\}$ and $V - \{p\}$ respectively denote all candidates except x and all words except the given name p, k is the co-occurrence frequency between x and p. Moreover, K is the sum of co-occurrence frequencies between x and all words in V, whereas n is the same between p and all candidates in C. N is the total co-occurrences between all word pairs taken from C and V. To measure the strength of association between a name and a candidate alias, using Table 1 we define nine popular co-occurrence statistics: chi-squared measure (**CS**), Log-likelihood ratio (**LLR**), hyper-geometric distributions (**HG**) and the six measures shown in Table 2. Because of the limited availability of space, we omit the definitions of these measures (see Manning and Schutze [15] for a detailed discussion).

Table 1. Contingency table for a candidate alias x

	x	$C - \{x\}$	C
p	k	$n - k$	n
$V - \{p\}$	$K - k$	$N - n - K + k$	$N - n$
V	K	$N - K$	N

Table 2. Anchor text-based co-occurrence measures

Measure	Definition	Measure	Definition
CF	k	**tfidf**	$k \log \frac{N}{K+1}$
PMI	$\log_2 \frac{kN}{Kn}$	**cosine**	$\frac{k}{\sqrt{n}+\sqrt{K}}$
Dice	$\frac{2k}{n+K}$	**Overlap**	$\frac{k}{\min(n,K)}$

A frequently observed phenomenon related to the web is that many pages with diverse topics link to so-called *hubs* such as Google, Yahoo, or MSN. Two anchor texts might link to a hub for entirely different reasons. Therefore, co-occurrences coming from hubs are prone to noise. To overcome the adverse effects of a hub h when computing co-occurrence measures, we multiply the number of co-occurrences of words linked to h by a factor $\alpha(h, p)$, where

$$\alpha(h, p) = \frac{t}{d}. \tag{1}$$

Here, t is the number of inbound anchor texts of h that contain the real name p, and d is the total number of inbound anchor texts of h. If many anchor texts

that link to h contain p (i.e. larger t value), then the reliability of h as a source of information about p increases. On the other hand, if h has many inbound links (i.e. larger d value), then it is likely to be a noisy hub and gets discounted when multiplied by $\alpha(<< 1)$. Intuitively, Eq.1 boosts hubs that are likely to contain information related to p, while penalizing those that contain various other topics.

3.4 Page-Count-Based Association Measures

In previous section we defined various ranking scores using anchor texts. However, not all names and aliases are equally well represented in anchor texts. Consequently, in this section, we define word association measures that consider co-occurrences not only in anchor texts but in the web overall. Page counts retrieved from a web search engine for the conjunctive query, $p \cap x$, for a name p and a candidate alias x can be regarded as an approximation of their co-occurrences in the web. We define the four measures shown in Table 3 using page-counts retrieved from a search engine. Therein, the function $H(q)$ denotes the page-counts for a query q. **WebDice** and **WebPMI** [6] respectively are based on the Dice coefficient and pointwise mutual information. In WebPMI, L is the number of pages indexed by the web search engine, which we approximated as $L = 10^{10}$ according to the number of pages indexed by Google. **Prob**$(x|p)$ and **Prob**$(p|x)$ respectively denote the conditional probabilities of a candidate (x) given a name (p) and a name given a candidate.

Table 3. Page-count-based association measures

Measure	Definition	Measure	Definition
WebPMI	$\log_2 \frac{L \times H(p \cap x)}{H(p) \times H(x)}$	Prob(p|x)	$\frac{H(p \cap x)}{H(x)}$
WebDice	$\frac{2 \times H(p \cap x)}{H(p) + H(x)}$	Prob(x|p)	$\frac{H(p \cap x)}{H(p)}$

3.5 Training

Using a dataset of name-alias pairs, we train a ranking support vector machine [12] to rank candidate aliases according to their strength of association with a name. For a name-alias pair we define three feature types: anchor text-based co-occurrence measures, web page-count-based association measures, and frequencies of observed lexical patterns. The nine co-occurrence measures: **CF, tfidf, CS, LLR, PMI, HG, cosine, overlap, Dice** (Table 2) are computed with and without weighting for hubs to produce $18(2 \times 9)$ features. Moreover, the four page-count-based association measures defined in Table 3 and the frequency of lexical patterns extracted by algorithm 1 are used as features in training the ranking SVM. If numerous patterns connects a name and a candidate alias in snippets, then the confidence of the candidate alias as a correct alias of the name increases. During training, ranking SVMs attempt to minimize the number of discordant pairs in the training data, thereby improving the average precision. The trained SVM model is used to rank the set of candidates that were extracted for a name. Finally, the highest-ranking candidate is selected as the alias of the name.

Table 4. Lexical patterns with the highest F-scores

patterns for personal names	F-score
* aka [NAME]	0.335
[NAME] aka *	0.322
[NAME] better known as *	0.310
[NAME] alias *	0.286
[NAME] also known as *	0.281
* nee [NAME]	0.225
patterns for location names	F-score
[NAME] nickname the *	0.739
[NAME] is nicknamed the *	0.723
[NAME] employment nickname *	0.627
[NAME] state flag or *	0.589
[NAME] nicknamed the *	0.5567
[NAME] is called the *	0.3199

Table 5. Comparison with baselines and previous work

Method	MRR	Method	MRR
SVM (Linear)	0.6718	Prob($p\|x$)	0.1414
SVM (Quad)	0.6495	CS(h)	0.1186
SVM (RBF)	0.6089	CF	0.0839
Hokama & Kitagawa	0.6314	cosine	0.0761
tfidf(h)	0.3957	tfidf	0.0757
WebDice	0.3896	Dice	0.0751
LLR(h)	0.3879	overlap(h)	0.0750
cosine(h)	0.3701	PMI(h)	0.0624
CF(h)	0.3677	LLR	0.0604
HG(h)	0.3297	HG	0.0399
Dice(h)	0.2905	CS	0.0079
Prob($x\|p$)	0.2142	PMI	0.0072
WebPMI	0.1416	overlap	0.0056

4 Experiments

To train and evaluate the proposed method, we create three name-alias datasets[2]: the English personal names dataset (50 names), the English place names dataset (50 names), and the Japanese personal names (100 names) dataset. Both our English and Japanese personal name datasets include people from various fields of cinema, sports, politics, science, and mass media. The place name dataset contains aliases for the 50 U.S. states. Aliases were manually collected after referring various information sources such as Wikipedia and official home pages. The anchor texts collection we used to compute the measures in Table 2 contains $24,456,871$ anchor texts pointing to $8,023,364$ unique urls.

Algorithm 1 extracts over 8000 patterns for the 50 English personal names in our dataset. We rank the patterns according to their F scores to identify the patterns that accurately convey information about aliases. F score of a pattern s is computed as the harmonic mean between the precision and recall of the pattern:

$$\text{Precision}(s) = \frac{\text{No. of correct aliases retrieved by } s}{\text{No. of total aliases retrieved by } s},$$

$$\text{Recall}(s) = \frac{\text{No. of correct aliases retrieved by } s}{\text{No. of total aliases in the dataset}}.$$

Table 4 shows the patterns with the highest F scores extracted using English personal names. As shown in the table, unambiguous and highly descriptive patterns are extracted by the proposed method. Experimentally, we selected the top ranked 200 patterns as features for training. Interestingly, among the extracted pattens we found patterns written in languages other than English, such as *de son vrai nom* (French for *his real name*) and *vero nome* (Italian for *real name*).

[2] www.miv.t.u-tokyo.ac.jp/danushka/aliasdata.zip

In Table 5, we compare the proposed SVM-based method against various individual ranking scores (baselines) and previous studies of alias extraction Hokama and Kitagawa [11]) on Japanese personal names dataset. We used linear, polynomial (quadratic), and radial basis functions (RBF) kernels for ranking SVM. Mean reciprocal rank (MRR) [2] is used to evaluate the various approaches. If a method ranks the correct aliases of a name on top, then it receives a higher MRR value. As shown in Table 5, the best results are obtained by the proposed method with linear kernels (SVM(Linear)). Both ANOVA and Tukey HSD tests confirm that the improvement of SVM(Linear) is statistically significant ($p<0.05$). A drop of MRR occurs with more complex kernels, which is attributable to overfitting. Hokama and Kitagawa's method which uses manually created patterns, can only extract Japanese name aliases. Their method reports an MRR value of 0.6314 on our Japanese personal names dataset. In Table 5 we denote the hub-weighted versions of anchor text-based co-occurrence measures by (h). Among the numerous individual ranking scores, the best results are reported by the hub-weighted tfidf score (tfidf(h)). It is noteworthy that, for anchor text-based ranking scores, the hub-weighted version always outperforms the non-hub-weighted counterpart, which justifies the proposed hub-weighting method. Among the four page-count-based ranking scores, WebDice reports the highest MRR. It is comparable to the best anchor text-based ranking score, tfidf(h). The fact that Prob($x|p$) gives slightly better performance over Prob($p|x$) implies that we have a better chance in identifying an entity given its real name than an alias.

Table 6. Overall performance

Dataset	MRR	AP
English Personal Names	0.6150	0.6865
English Place Names	0.8159	0.7819
Japanese Personal Names	0.6718	0.6646

Table 7. Aliases extracted by the proposed method

Real Name	Extracted Aliases
David Hasselhoff	**hoff**, **michael knight**, **michael**
Courteney Cox	**dirt lucy**, **lucy**, **monica**
Al Pacino	**michael corleone**
Teri Hatcher	**susan mayer**, **susan**, mayer
Texas	**lone star state**, lone star, lone
Vermont	**green mountain state**, green,
Wyoming	**equality state**, **cowboy state**
Hideki Matsui	**Godzilla**, **nishikori**, **matsui**

In Table 6 we evaluate the overall performance of the proposed method on each dataset using MRR and average precision (AP) [2]. Different from the mean reciprocal rank, which focuses only on rank, average precision incorporates consideration of both precision at each rank and the total number of correct aliases in the dataset. Both MRR and average precision have been used in rank evaluation tasks such as evaluating the results returned by a search engine. With each dataset we performed a 5-fold cross validation. As shown in Table 6, the proposed method reports high scores for both MRR and average precision on all three datasets. Best results are achieved for the place name alias extraction task. Table 7 presents the aliases extracted for some entities included in our datasets. Overall, the proposed method extracts most aliases in the manually created gold standard (shown in bold).

Table 8. Effect of aliases on relation detection

Real name only			Real name and top alias		
Precision	Recall	F	Precision	Recall	F
.4812	.7185	.4792	.4833	.9083	.5918

We evaluate the effect of the extracted aliases on a real-world relation detection task. First, we manually classify 50 people in the English personal names dataset, depending on their field of expertise, into four categories: *music, politics, movies,* and *sports*. Then, we measure the association between two people using the pointwise mutual information (WebPMI) as defined in Table 3. We then use group average agglomerative clustering (GAAC) to group the people into four clusters. Initially, each person is assigned to a separate cluster. In subsequent iterations, GAAC merges the two clusters with the highest correlation. We terminate the GAAC process when exactly four clusters are formed. Ideally, people who work in the same field should be clustered into the same group. We use the *B-CUBED* method [3] and compute the precision, recall and F-score for each name in the dataset and average the results over the number of people in the dataset. Table 8 shows performance of clustering when only the real name is used and the real name disjunctively coupled with the top alias extracted by the proposed method for the name. The use of aliases significantly improves recall (ca. 20%) and consequently the F score. This significant improvement in recall can be attributed to the discovery of relations between entities that use not only their real names but also numerous aliases. In such cases, using only the real name would extract only a fraction of the relations between the entities under consideration. By considering not only real names but also aliases, it is possible to discover relations that are unidentifiable solely using real names.

5 Conclusion

We proposed a lexical-pattern-based approach to extract aliases of a given name. The extracted candidates were ranked using various ranking scores computed using the hyperlink structure on the web and page-counts retrieved from a search engine. The proposed method reported high MRR scores on three different datasets and outperformed numerous baselines and a previously proposed method. Moreover, the extracted aliases significantly improved recall in a relation detection task.

References

1. Artiles, J., Gonzalo, J., Verdejo, F.: A testbed for people searching strategies in the www. In: Proc. of SIGIR 2005, pp. 569–570 (2005)
2. Baeza-Yates, R.A., Ribeiro-Neto, B.A.: Modern Information Retrieval. ACM Press, New York (1999)

3. Bagga, A., Baldwin, B.: Entity-based cross-document coreferencing using the vector space model. In: Proc. of COLING 1998, pp. 79–85 (1998)
4. Bekkerman, R., McCallum, A.: Disambiguating web appearances of people in a social network. In: Proc. of WWW 2005, pp. 463–470 (2005)
5. Berland, M., Charniak, E.: Finding parts in very large corpora. In: Proc. of ACL 1999, pp. 57–64 (1999)
6. Bollegala, D., Matsuo, Y., Ishizuka, M.: Measuring semantic similarity between words using web search engines. In: Proc. of WWW 2007, pp. 757–766 (2007)
7. Chakrabarti, S.: Mining the Web: Discovering Knowledge from Hypertext Data. Morgan Kaufmann, San Francisco (2003)
8. Galvez, C., Moya-Anegon, F.: Approximate personal name-matching through finite-state graphs. Journal of the American Society for Information Science and Technology 58, 1–17 (2007)
9. Guha, R.V., McCool, R., Miller, E.: Semantic search. In: Proc. of WWW 2003, pp. 700–709 (2003)
10. Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora. In: Proc. of COLING 1992, pp. 539–545 (1992)
11. Hokama, T., Kitagawa, H.: Extracting mnemonic names of people from the web. In: Sugimoto, S., Hunter, J., Rauber, A., Morishima, A. (eds.) ICADL 2006. LNCS, vol. 4312, pp. 121–130. Springer, Heidelberg (2006)
12. Joachims, T.: Optimizing search engines using clickthrough data. In: Proc. of KDD 2002 (2002)
13. Lin, D.: Automatic retrieval and clustering of similar words. In: Proc. of COLING 1998, pp. 768–774. Association for Computational Linguistics, Morristown (1998)
14. Mann, G.S., Yarowsky, D.: Unsupervised personal name disambiguation. In: Proc. of CoNLL 2003, pp. 33–40 (2003)
15. Manning, C., Schutze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge (1999)
16. Matsuo, Y., Mori, J., Hamasaki, M., Ishida, K., Nishimura, T., Takeda, H., Hasida, K., Ishizuka, M.: Polyphonet: An advanced social network extraction system. In: Proc. of WWW 2006 (2006)
17. Ravichandran, D., Hovy, E.: Learning surface text patterns for a question answering system. In: Proc. of ACL 2002, pp. 41–47 (2001)
18. Snow, R., Jurafsky, D., Ng, Y.: Learning syntactic patterns for automatic hypernym discovery. In: Proc. of NIPS 2005 (2005)