# Identification of Personal Name Aliases on the Web

Danushka Bollegala *
The University of Tokyo
danushka@mi.ci.i.u-tokyo.ac.jp

Taiki Honma
The University of Tokyo
honma@mi.ci.i.u-tokyo.ac.jp

Yutaka Matsuo
The University of Tokyo
matsuo@biz-model.t.u-tokyo.ac.jp

Mitsuru Ishizuka
The University of Tokyo
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 [11, 26], social network extraction from the web [24, 25] search and integration of data [2, 17] and entity disambiguation [22, 14, 7, 1, 27]. 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 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 [22, 7, 14, 1, 27], 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. [24] 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.

The Semantic Web is intended to solve the entity disambiguation problem by providing a mechanism to add semantic metadata for entities. However, an issue that the Semantic Web currently faces is that insufficient semantically annotated web contents are available. Automatic extraction of metadata [13, 18, 29] can accelerate the process of semantic annotation. For named entities, automatically extracted aliases can serve as a useful source of metadata, thereby providing a means to disambiguate an entity.

Searching for information about people on the web is an

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


*WWW2008*, April 21–25, 2008, Beijing, China.

extremely common activity of Internet users. Around 30% of search engine queries include personal names [3]. However, retrieving information about a person merely using his or her real name 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.

Along with the recent rapid growth of social media such as blogs, extracting and classifying sentiment on the web has received much attention [28]. Typically, a sentiment analysis system classifies a text as positive or negative according to the sentiment expressed in it. However, when people express their views about a particular entity, they do so by referring to the entity not only using the real name but also using various aliases of the name. By aggregating texts that use various aliases to refer to an entity, a sentiment analysis system can produce an informed judgment related to the sentiment.

In this paper, we propose a fully automatic method to extract aliases of a given name. The proposed method includes two steps: given a name, extract all potential candidate aliases from the web; then rank the extracted candidates according to the likelihood that they are aliases of the given name. Our main contributions are the following:

- 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

Alias identification is closely related to the problem of cross-document coreference resolution [5, 6, 16], in which the objective is to determine whether two mentions of a name in different documents refer to the same entity. Bagga and Baldwin [5] 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*) [22, 7, 14, 1, 27]. 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. The web people search task

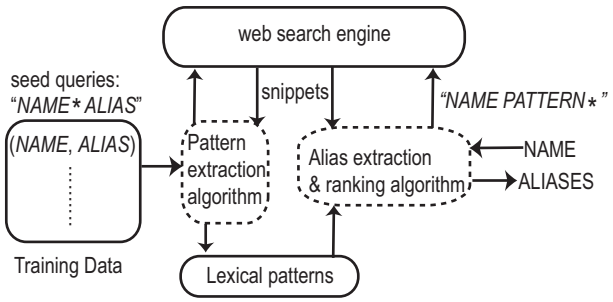


**Figure 1: Outline of the proposed method**

(WEPS) at SemEval 2007 [1] provided a dataset and evaluated various name disambiguation systems. 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.

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*) [15]. Rules in the form of regular expressions and edit-distance-based methods have been used to compare names. Bilenko and Mooney [9] proposed a method to learn a string similarity measure to detect duplicates in bibliography databases. 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 [20] 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 stricture on the web and page counts retrieved from a search

[1] http://nlp.uned.es/weps

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[2] for the query *"Will Smith * The Fresh Prince"*.

> ...Rock the House, the duo's debut album of 1987, demonstrated that **Will Smith**, aka **the Fresh Prince**, was an entertaining and amusing storyteller...

**Figure 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 [19] and meronyms (i.e. words in a part-whole relation) [8], measuring semantic similarity [10] and automatic metadata extraction [13].

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

[2]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** { $D \leftarrow$ GetSnippets("$NAME * ALIAS$")
  **for each** snippet $d \in D$
  **do** $P \leftarrow P +$ CreatePattern($d$) }
**return** ($P$)

**Figure 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** { $D \leftarrow$ GetSnippets("$NAME\ p\ *$")
  **for each** snippet $d \in D$
  **do** $C \leftarrow C +$ GetNgrams($d, NAME, p$) }
**return** ($C$)

**Figure 4: Given a name and a set of lexical patterns, extract candidate aliases.**

*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 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) [21] 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 in-

**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** | $\frac{k}{\min(n,K)}$ |

formation retrieval, and ranking and classification of web pages [12], 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 [23] for a detailed discussion).

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

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

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.

**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) | $\frac{H(p \cap x)}{H(p)}$ |

## 3.4 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** [10] 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.

## 3.5 Training

Using a dataset of name-alias pairs, we train a ranking support vector machine [21] 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.

Given a set of personal names and their aliases, we model the training process as a preference learning task. For each name, we impose a binary preference constraint between the correct alias and each candidate. Then we consider one alias at a time and combine it with the candidates if more than one correct alias exists. For example, let us assume that for a name $p$ we selected the four candidates $a_1, a_2, a_3, a_4$. Without loss of generality, let us further assume that $a_1$ and $a_2$ are the correct aliases of $p$. Consequently, we form four partial preferences: $a_1 \succ a_3$, $a_1 \succ a_4$, $a_2 \succ a_3$ and $a_2 \succ a_4$. Here, $x \succ y$ denotes the fact that $x$ is preferred

to $y$. 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.

# 4. EXPERIMENTS

## 4.1 Datasets

To train and evaluate the proposed method, we create three name-alias datasets[3]: 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.

To compute the anchor text-based word co-occurrence measures, we crawled English and Japanese web sites and extracted anchor texts and urls linked by the anchor texts. A web site might use links for purely navigational purposes, which would convey no semantic clue. To remove navigational links in our dataset, we prepare a list of words that are commonly used in navigational menus, such as *top, last, next, previous, links*, etc., and remove anchor texts that contain those words. The resultant dataset contains $24,456,871$ anchor texts pointing to $8,023,364$ urls. All urls in the dataset contain at least two inbound anchor texts. The average number of inbound anchor texts per url is 3.05 and its standard deviation is 54.02.

## 4.2 Pattern Selection

We used the English personal name dataset to extract lexical patterns as described in algorithm 1. The proposed pattern extraction algorithm extracts over 8000 unique patterns that represent various ways in which names and aliases are introduced on the web. Of those patterns, 70% occur less than 5 times for name-alias pairs in the dataset. Given the relatively small number of training instances (i.e. 50 instances in the English personal names dataset), it is not possible to train with such numerous sparse patterns. From among these patterns, we must select the patterns that are most accurate. We use algorithm 1 to extract patterns and then evaluate those patterns based on the candidates they extract when used in algorithm 2. We perform 5-fold cross validation on English personal names dataset. Precision and recall of a pattern $s$ is defined as follows:

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

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

Consequently, the $F$-score, $F(s)$, can be computed as

$$F(s) = \frac{2 \times \text{Precision}(s) \times \text{Recall}(s)}{\text{Precision}(s) + \text{Recall}(s)}.$$

Table 4 shows the patterns with the highest precision scores. As shown in the table, unambiguous and highly descriptive patterns are extracted using the proposed method. Most of the patterns shown in Table 4 are asymmetric in the sense that the variable [NAME] and the wildcard * appear only in one combination among top ranked patterns. In contrast, pattern *aka* is symmetric and both combinations show high $F$-scores. Although not shown in Table 4 because of limited space, the proposed method also extracted some patterns written in other languages other than English. For example, *de son vrai nom* (French for *his real name*) and *vero nome* (Italian for *vero nome*) were also extracted as patterns using the proposed method, despite the fact that we searched only for English search results in Google.

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

**Table 4: Lexical patterns with the highest $F$-scores as extracted using the proposed method**

| pattern | | | $F$-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 |
| [NAME] | nickname | * | 0.224 |
| * | whose real name is | [NAME] | 0.205 |
| [NAME] | aka the | * | 0.187 |
| * | was born | [NAME] | 0.153 |

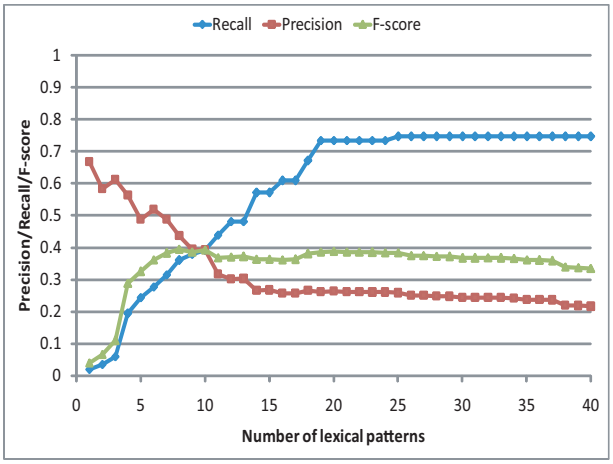


**Figure 5: Selecting patterns for training**

To find the optimum number of patterns that should be used in training, we sort the patterns by their precision and measure the overall recall when more patterns are used to extract candidate aliases. Here, the overall recall of using a set of patterns is computed as the ratio of the number of aliases extracted using all the patterns in the set to the total number of correct aliases in the dataset. Experimental results are shown in Fig.5. It is apparent in Fig.5 that overall recall is rapidly enhanced by a greater number of patterns. However, low-precision patterns do not increase recall to a great degree. Consequently, recall settles to a maximum value of 0.75 at 25 patterns.

## 4.3 Accuracy of Alias Extraction

In Table 5, we compare the proposed SVM-based method

**Table 5: Proposed method vs. baselines and previous studies of alias extraction**

| 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 et al. [20] | 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 |

**Table 6: Overall performance**

| Dataset | MRR | Average Precision |
|---|---|---|
| English Personal Names | 0.6150 | 0.6865 |
| English Place Names | 0.8159 | 0.7819 |
| Japanese Personal Names | 0.6718 | 0.6646 |

against various individual ranking scores (baselines) and previous studies of alias extraction (Hokama and Kitagawa. [20]) on Japanese personal names dataset. We used linear, polynomial (quadratic), and radial basis functions (RBF) kernels for ranking SVM. We use mean reciprocal rank (MRR) [4] 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 over-fitting. Hokama and Kitagawa's [20] 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 used as features for training, 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 hub-weighting method given by Eq.1. 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). Between the two conditional probabilities, conditioning on the real name (i.e. Prob($x|p$)) gives slightly better performance. This result implies that we have a better chance in identifying an entity given its real name than an alias.

We evaluate the proposed method using three types of alias data: personal names in English, place (location) names in English and personal names in Japanese using the mean reciprocal rank and average precision [4]. Different from the mean reciprocal rank, which focuses only on rank, average precision incorporates consideration of both precision

**Table 8: Effect of aliases on relation detection**

| Method | real name only | | | real name and top alias | | |
|---|---|---|---|---|---|---|
| | P | R | F | P | R | F |
| Jaccard | .4902 | .5229 | .4527 | .4999 | .7748 | .5302 |
| PMI | .4812 | .7185 | .4792 | .4833 | .9083 | .5918 |

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 or a question-answering (QA) system. 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 aliases extracted for some entities included in our datasets. The *gold standard* is the aliases assigned by humans for the named entities in the datasets. Overall, in Table 7 the proposed method extracts most aliases assigned in the gold standard. It is interesting to note that, for actors the extracted aliases include their roles in movies or television dramas (e.g. *Michael Knight* for *David Hasselhoff* and *Susan Mayer* for *Teri Hatcher*). We extract $n$-grams from snippets as candidate aliases. Therefore, some of the extracted aliases do overlap (e.g. aliases for *Texas*). This might be prevented by using a post-processing heuristic such as ignoring aliases that are nested within an alias that has a higher rank. However, to keep the proposed method as simple as possible, we use no such post-processing heuristics.

## 4.4 Relation Detection

We evaluate the effect of aliases on a real-world relation detection task as follows. First, we manually classified 50 people in the English personal names dataset, depending on their field of expertise, into four categories: *music, politics, movies,* and *sports*. Following earlier research on web-based social network extraction [24, 25], we measured the association between two people using the Jaccard coefficient and pointwise mutual information. We then use group average agglomerative clustering (GAAC) [23] to group the people into four clusters. Initially, each person is assigned to a separate cluster. In subsequent iterations, group average agglomerative clustering process, merges the two clusters with the highest correlation. Correlation, Corr($\Gamma$), between two clusters $X$ and $Y$ is defined as

$$\text{Corr}(\Gamma) = \frac{1}{2}\frac{1}{|\Gamma|(|\Gamma|-1)}\sum_{(u,v)\in\Gamma}\text{sim}(u,v). \quad (1)$$

Here, $\Gamma$ is the merger of the two clusters $X$ and $Y$. $|\Gamma|$ denotes the number of elements (persons) in $\Gamma$ and $\text{sim}(u,v)$ is the association between two persons $u$ and $v$ in $\Gamma$. We used the Jaccard coefficient, which is calculated using page counts as

$$\text{Jaccard}(u,v) = \frac{hits(\text{“}u\text{”}AND\text{“}v\text{”})}{hits(\text{“}u\text{”}OR\text{“}v\text{”})},$$

and pointwise mutual information (Table 3) to measure the association between two persons $u$ and $v$. We terminate the GAAC process when exactly four clusters are formed.

**Table 7: Aliases extracted using the proposed method**

| Real Name | gold standard | First | Second | Third |
|---|---|---|---|---|
| David Hasselhoff | hoff, michael knight, michael | hoff | michael knight | michael |
| Courteney Cox | cece, lucy, dirt lucy, monica geller, monica | dirt lucy | lucy | monica |
| Al Pacino | sonny, alfredo james pacino, michael corleone | michael corleone | alfredo james pacino | alphonse pacino |
| Teri Hatcher | hatch, susan mayer, susan, lois lane, lois | susan mayer | susan | mayer |
| Texas | lone star state | lone star state | lone star | lone |
| Vermont | green mountain state | green mountain state | green | green mountain |
| Wyoming | equality state, cowboy state | equality | equality state | cowboy state |
| Hideki Matsui | Godzilla, nishikori, matsu hide | Godzilla | nishikori | matsui |

Ideally, people who work in the same field should be clustered into the same group. We used the *B-CUBED* metric [5] to evaluate the clustering results. The B-CUBED evaluation metric was originally proposed for evaluating cross-document coreference chains. We 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. For each person $p$ in our dataset, let us denote the cluster that $p$ belongs to as $C(p)$. Moreover, we use $A(p)$ to represent the affiliation of person $p$, e.g. $A$("Bill Clinton") ="politics". Then we calculate the precision and recall for person $p$ as

$$\text{Precision}(p) = \frac{\text{No. of people in C(p) with affiliation A(p)}}{\text{No. of people in C(p)}},$$

$$\text{Recall}(p) = \frac{\text{No. of people in C(p) with affiliation A(p)}}{\text{Total No. of people with affiliation A(p)}}.$$

Then, the $F$-score of person $p$ is defined as

$$\text{F}(p) = \frac{2 \times \text{Precision}(p) \times \text{Recall}(p)}{\text{Precision}(p) + \text{Recall}(p)}.$$

The overall precision (**P**), recall (**R**) and $F$-score (**F**) are computed by taking the averaged sum over all the names in the dataset.

$$\text{Precision} = \frac{1}{N} \sum_{p \in DataSet} \text{Precision}(p)$$

$$\text{Recall} = \frac{1}{N} \sum_{p \in DataSet} \text{Recall}(p)$$

$$F\text{−Score} = \frac{1}{N} \sum_{p \in DataSet} \text{F}(p)$$

Here, *DataSet* is the set of 50 names selected from the English personal names dataset. Therefore, $N = 50$ in our evaluations.

We first conduct the experiment only using real names (i.e. $u, v$ ="real name") Next, we repeated the experiment by expanding the query with the top ranking alias extracted by the proposed algorithm (i.e. $u, v$ ="real name" OR "alias"). Experimental results are summarized in Table 8. From Table 8, we can see that F-scores have increased as a result of including aliases with real names in relation identification. Moreover, the improvement is largely attributable to the improvement in recall. In both Jaccard and PMI, the inclusion of aliases has boosted recall by more than 20%. By considering not only real names but also their aliases, it is possible to discover relations that are unidentifiable solely using real names.

# 5. DISCUSSION

The concepts of entities and relations are central to numerous web search and mining tasks. However, uniquely identifying entities on the web is made complicated by lexical and referential ambiguities in entities. This study specifically examined referential ambiguity of names. However, lexical and referential ambiguities are closely connected. For example, in the case of extracting aliases for a personal name, the given name itself might be ambiguous. If more than one entity is represented by the name, then merely stating the real name does not enable us to identify the entity uniquely. In such situations, we must first disambiguate the real name (i.e. resolve the lexical ambiguity) before we attempt to extract aliases (i.e. resolve referential ambiguity). On the other hand, two web pages about the same individual might use different aliases of the person's real name. A namesake disambiguation system that attempts to cluster these two pages together might require the knowledge about aliases. Moreover, aliases themselves can sometimes be ambiguous. For example, *Godzilla*, an alias for *Hideki Matsui* is also a movie and an imaginary monster. A single alias might be insufficient to identify an entity on the web uniquely. In fact, during an error analysis, we discovered that the phrase *beer hunter* was incorrectly extracted as an alias for *Michael Jackson*. However, *Michael Jackson* has several namesakes on the web; one of whom was, in fact, an expert on beer and introduces himself as the *beer hunter*. In our future work in alias extraction, we intend to explore methods that can identify aliases for different namesakes of a given name.

Consider the problem of detecting whether a particular relation $R$ holds between two entities $A$ and $B$. One approach to solve this problem is to find contexts in which $A$ and $B$ co-occur and decide whether the relation $R$ pertains between the entities. For example, if $A$ and $B$ are two researchers, then we can expect a high co-occurrence on the web if they publish their mutual works together or work on the same project. In fact, previous studies of social network extraction [24, 25] have considered co-occurrences on the web as a measure of the social association among people. However, if $A$ and $B$ have name aliases, then it is not possible to collect all the contexts in which they co-occur merely by searching using the real names. To illustrate this point, let us assume the aliases of $A$ and $B$ to be $a$, $b$. Then there exists four possible co-occurrences: ($A$,$B$), ($A$,$b$), ($a$, $B$) and ($a$,$b$). The query which contains only real names, $A$ **AND** $B$, covers only one of the four outcomes. Moreover, the number of possible combinations grows exponentially along with the number of aliases for each entity. As seen from the relation detection experiment in section 4.4, knowledge related to

aliases can improve a relation detection system by providing more accurate information related to the co-occurrences of entities.

## 6. CONCLUSION

In this paper, we specifically addressed the problem of extracting aliases of a given name from the web. We proposed a lexical-pattern-based approach to represent the various ways in which names and aliases are introduced on the web. Using a set of name-alias pairs, we proposed a method to extract such lexical patterns automatically from snippets returned by a web search engine. We then used the extracted patterns to determine candidate aliases of a given name. We proposed a word co-occurrence measures using anchor texts and page counts to evaluate the confidence of an candidate alias for a name. Moreover, the various ranking scores proposed in the paper were integrated using ranking support vector machines to leverage a robust ranking function. We evaluated the proposed method using both personal and place names. The proposed method outperformed numerous baselines introduced in the paper and previous work on alias extraction. Moreover, independent evaluations on English and Japanese datasets suggest the possibility of extending the proposed method to other languages.

## 7. REFERENCES


[1] B. Aleman-Meza, M. Nagarajan, and I. Arpinar. Ontology-driven automatic entity disambiguation in unstructured text. In *Proc. of ISWC'06*, 2006.

[2] K. Anyanwu, A. Maduko, and A. Sheth. Semrank: ranking complex relationship search results on the semantic web. In *Proc. of WWW'05*, pages 117–127, 2005.

[3] J. Artiles, J. Gonzalo, and F. Verdejo. A testbed for people searching stategies in the www. In *Proc. of SIGIR'05*, pages 569–570, 2005.

[4] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[5] A. Bagga and B. Baldwin. Entity-based cross-document coreferencing using the vector space model. In *Proc. of COLING'98*, pages 79–85, 1998.

[6] A. Bagga and A. Biermann. A methodology for cross-document coreference. In *Proc. 5th Joint Conf. on information sciences*, pages 207–210, 2000.

[7] R. Bekkerman and A. McCallum. Disambiguating web appearances of people in a social network. In *Proc. of WWW'05*, pages 463–470, 2005.

[8] M. Berland and E. Charniak. Finding parts in very large corpora. In *Proc. of ACL'99*, pages 57–64, 1999.

[9] M. Bilenko and R. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proc. of SIGKDD'03*, 2003.

[10] D. Bollegala, Y. Matsuo, and M. Ishizuka. Measuring semantic similarity between words using web search engines. In *Proc. of WWW'07*, pages 757–766, 2007.

[11] R. C. Bunescu and R. J. Mooney. Learning to extract relations from the web using minimal supervision. In *Proc. of ACL'07*, pages 576–583, 2007.

[12] S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann, 2003.

[13] P. Cimano, S. Handschuh, and S. Staab. Towards the self-annotating web. In *Proc. of WWW'04*, 2004.

[14] M. Fleischman and E. Hovy. Multi-document person name resolution. In *Proc. of 42nd ACL, Reference Resolution Workshop*, 2004.

[15] C. Galvez and F. Moya-Anegon. Approximate personal name-matching through finite-state graphs. *Journal of the American Society for Information Science and Technology*, 58:1–17, 2007.

[16] C. H. Gooi and J. Allan. Cross-document coreference on a large scale corpus. In *Proc. of HLT/NAACL '04*, pages 89–98, 2004.

[17] R. V. Guha, R. McCool, and E. Miller. Semantic search. In *Proc. of WWW'03*, pages 700–709, 2003.

[18] S. Handschuh and S. Stabb. Cream creating metatada for the semantic web. *Computer Networks*, 42:579–598, 2003.

[19] M. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proc. of COLING'92*, pages 539–545, 1992.

[20] T. Hokama and H. Kitagawa. Extracting mnemonic names of people from the web. In *Proc. of 9th Intl. Conf. on Asian Digital Libraries (ICADL'06)*, pages 121–130, 2006.

[21] T. Joachims. Optimizing search engines using clickthrough data. In *Proc. of KDD'02*, 2002.

[22] G. Mann and D.Yarowsky. Unsupervised personal name disambiguation. In *Proc. of CoNLL'03*, pages 33–40, 2003.

[23] C. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[24] Y. Matsuo, J. Mori, M. Hamasaki, K. Ishida, T. Nishimura, H. Takeda, K. Hasida, and M. Ishizuka. Polyphonet: An advanced social network extraction system. In *Proc. of WWW'06*, 2006.

[25] P. Mika. Ontologies are us: A unified model of social networks and semantics. In *Proc. of ISWC2005*, 2005.

[26] J. Mori, Y. Matsuo, and M. Ishizuka. Extracting keyphrases to represent relations in social networks from the web. In *Proc. of IJCAI'07*, pages 2820–2852, 2007.

[27] T. Pedersen, A. Kulkarni, R. Angheluta, Z. Kozareva, and T. Solorio. An unsupervised language independent method of name discrimination using second order co-occurance features. In *Proc. of CICLing'06*, 2006.

[28] P. Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proc. of ACL'02*, pages 417–424, 2002.

[29] M. Vargas-Vera, E. Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna. Minm: Ontology driven semi-automatic and automatic support for semantic markup. In *Proc. of 13th Intl. Conf. on Knowledge Engineering and Management*, 2002.