Automatic Discovery of Personal Name Aliases from the Web

Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka,

Abstract—An individual is typically referred by numerous name aliases on the web. Accurate identification of aliases of a given person name is useful in various web related tasks such as information retrieval, sentiment analysis, personal name disambiguation, and relation extraction. We propose a method to extract aliases of a given personal name from the web. Given a personal name, the proposed method first extracts a set of candidate aliases. Second, we rank the extracted candidates according to the likelihood of a candidate being a correct alias of the given name. We propose a novel, automatically extracted lexical pattern-based approach to efficiently extract a large set of candidate aliases from snippets retrieved from a web search engine. We define numerous ranking scores to evaluate candidate aliases using three approaches: lexical pattern frequency, word co-occurrences in an anchor text graph, and page-counts on the web. To construct a robust alias detection system, we integrate the different ranking scores into a single ranking function using ranking support vector machines. We evaluate the proposed method on three datasets: an English personal names dataset, an English place names dataset, and a Japanese personal names dataset. The proposed method outperforms numerous baselines and previously proposed name alias extraction methods, achieving a statistically significant mean reciprocal rank of 0.67. Experiments carried out using 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 different languages. Moreover, the aliases extracted using the proposed method are successfully utilized in an information retrieval task and improve recall by 20% in a relation-detection task.

Index Terms—Web Mining, Information Extraction, Web Text Analysis

1 Introduction

Searching for information about people in the Web is one of the most common activities of Internet users. Around 30% of search engine queries include person names [1], [2]. However, retrieving information about people from web search engines can become difficult when a person has nicknames or *name aliases*. For example, the famous Japanese major league baseball player *Hideki Matsui* is often called as *Godzilla* on the Web. A newspaper article on the baseball player might use the real name, *Hideki Matsui*, whereas a blogger would use the alias, *Godzilla*, in a blog entry. We will not be able to retrieve all the information about the baseball player if we only use his *real name*.

Identification of entities on the web is difficult for two fundamental reasons: first, different entities can share the same name (i.e. lexical ambiguity); secondly, a single entity can be designated by multiple names (i.e. referential ambiguity). For an example of lexical ambiguity consider the name *Jim Clark*. 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 [3], [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 of 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 in information retrieval [5]. In information retrieval, to improve recall of a web search on a person name, a search engine can automatically expand a query using aliases of the name [6]. In our previous example, a user who searches for *Hideki Matsui* might also be interested in retrieving documents in which Matsui is referred to as *Godzilla*. Consequently, we can expand a query on *Hideki Matsui* using his alias name *Godzilla*.

The Semantic Web is intended to solve the entity disambiguation problem by providing a mechanism to

• *The University of Tokyo, Japan.*
danushka@mi.ci.i.u-tokyo.ac.jp

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 [7] 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.

Identifying aliases of a name is important for extracting relations among entities. For example, Matsuo et al. [8] 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.

Along with the recent rapid growth of social media such as blogs, extracting and classifying sentiment on the web has received much attention [9]. 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.

We propose a fully automatic method to discover aliases of a given personal name from the web. Our contributions can be summarized as follows.

- We propose a lexical pattern-based approach to extract aliases of a given name using snippets returned by a web search engine. The lexical patterns are generated automatically using a set of real-world name-alias data. We evaluate the confidence of extracted lexical patterns and retain the patterns that can accurately discover aliases for various personal names. Our pattern extraction algorithm does not assume any language specific pre-processing such as part-of-speech tagging or dependency parsing etc., which can be both inaccurate and computationally costly in web-scale data processing.
- To select the best aliases among the extracted candidates, we propose numerous ranking scores based upon three approaches: lexical pattern frequency, word-cooccurrences in an anchor text graph, and page-counts on the web. Moreover, using real world name alias data, we train a ranking support vector machine to learn the optimal combination of individual ranking scores to construct a robust alias extraction method.
- We conduct a series of experiments to evaluate the various components of the proposed method. We compare the proposed method against numerous baselines and previously proposed name alias extraction methods on three datasets: an English personal names dataset, an English place names dataset, and a Japanese personal names dataset. Moreover, we evaluate the aliases extracted by the proposed method in an information retrieval task and a relation extraction task.

2 Related work

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 [10] 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*) [3], [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. The web people search task (WePS)[1] provided an evaluation dataset and compared 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*) [11]. Rules in the form of regular expressions and edit-distance-based methods have been used to compare names. Bilenko and Mooney [12] 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 [13] 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

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

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

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

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 generate such lexical patterns automatically using a training dataset of names and aliases.

3 METHOD

The proposed method is outlined in Figure 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[2] for the query *"Will Smith * The Fresh Prince"*.

Here, we use the wildcard operator * to perform a *NEAR* query and it matches with one or more words in a snippet. In Figure 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 Figure 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 [14] and meronyms [15].

2. www.google.com

Algorithm 3.1: 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 3.2: 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.

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. Our definition of lexical patterns include patterns that contain words as well as symbols such as punctuation markers. For example, from the snippet shown in Figure 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. In our experiments we limit the number of matched words with "*" to a maximum of five words. Because snippets returned by web search engines are very short in length compared to the corresponding source documents, increasing the matching window beyond five words did not produce any addional lexical patterns.

Once a set of lexical patterns is extracted, we use the

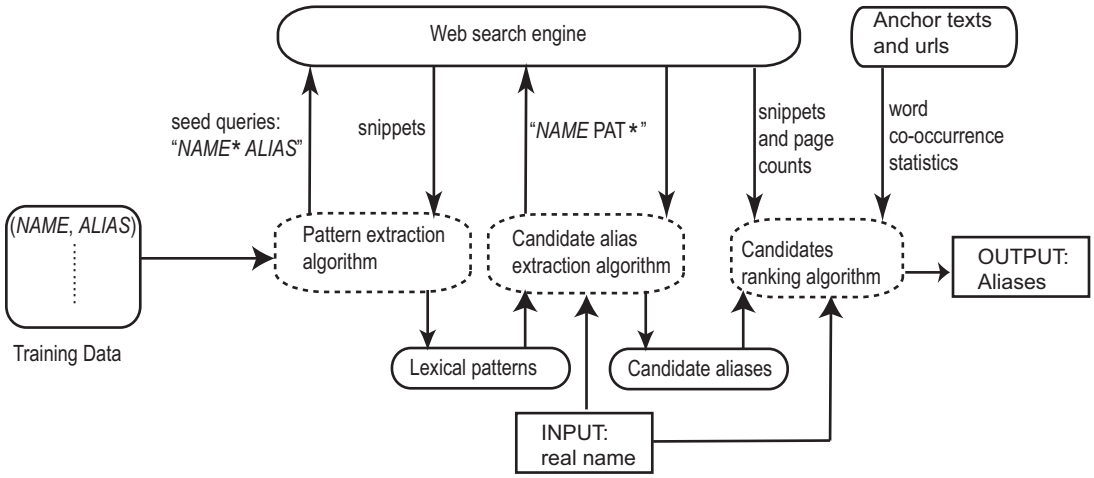

Fig. 1. Outline of the proposed method

.

patterns to extract candidate aliases for a given name as portrayed in Figure 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 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. For efficiency reasons, we limit the number of snippets downloaded by the function *GetSnippets* to a maximum of 100 in both Algorithm 3.1 and Algorithm 3.2. In Google it is possible to retrieve 100 snippets by issuing only a single query by setting the search parameter **num** to 100, thereby reducing the number queries required in practice.

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 three different approaches: lexical pattern frequency, word co-occurrences in an anchor text graph, and page-counts on the web. Next, we describe the those three approaches in detail.

3.3 Lexical Pattern Frequency

In Section 3.1 we presented an algorithm to extract numerous lexical patterns that are used to describe aliases of a personal name. As we will see later in Section 4, the proposed pattern extraction algorithm can extract a large number of lexical patterns. If the personal name under consideration and a candidate alias occur in many lexical patterns, then it can be considered as a good alias for the personal name. Consequently, we rank a set of candidate aliases in the descending order of the number of different lexical patterns in which they appear with a name. The lexical pattern frequency of an alias is analogous to the document frequency (DF) popularly used in information retrieval.

3.4 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 [16]. Anchor texts are particularly attractive because they not only contain concise texts, but also provide links that can be considered as expressing a citation. 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. Here, we use the term inbound anchor texts to refer the set of anchor texts pointing to the same url.

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. It is noteworthy that we do not consider co-occurrences of an alias and a name in the same anchor

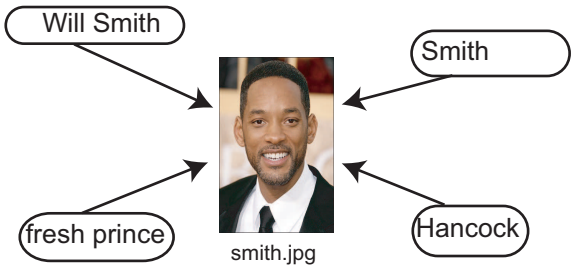

Fig. 5. A picture of Will Smith being linked by different anchor texts on the web.

	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 1
Contingency table for a candidate alias x

text. For example, consider the picture of Will Smith shown in Figure 5. Figure 5 shows a picture of Will Smith being linked to by four different anchor texts. According to our definition of co-occurrence, *Will Smith* and *fresh prince* are considered as co-occurring. Similarly, *Will Smith* and *Hancock* is also co-occurring in this example. It is noteworthy that we do not require the resource that is linked to by an acnhor text be a picture (or any resource related to the person under consideration for that matter). p and x are considered as co-occurring as long as they are linked to the same url, irrespective of the resource. Moreover, an anchor text can contain other texts beside a real name or an alias. We consider all the words in an anchor text and their bi-grams as potential candidate aliases if they co-occur with the real name according to our definition.

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 Figure 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: co-occurrence frequency (**CF**), tfidf measure (**tfidf**), chi-squared measure (**CS**), Log-likelihood ratio (**LLR**), hyper-geometric distributions (**HG**), cosine measure (**cosine**), overlap measure (**overlap**), and Dice coefficient (**Dice**). Next, we describe the computation of those association measures in detail.

3.4.1 Co-occurrence Frequency

This is the simplest of all association measures and was defined already in the previous section. The value k in Table 1 denotes the co-occurrence frequency of a candidate alias x and a name p. Intuitively, if there are many urls which are pointed to by anchor texts that contain a candidate alias x and a name p, then it is an indication that x is indeed a correct alias of the name p.

3.4.2 tfidf

The co-occurrence frequency is biased toward highly frequent words. A word that has a high frequency in anchor texts can also report a high co-occurrence with the name. For example, so-called *stop words* such as particles and articles appear in various anchor texts and have an overall high frequency. The *tfidf* measure [17], which is popularly used in information retrieval, is useful to normalize this bias. In fact, the tfidf measure reduces the weight that is assigned to words that appear across various anchor texts. The tfidf score of a candidate x as an alias of name p, $\mathrm{tfidf}(p,x)$, is computed from Table 1 as

$$\mathrm{tfidf}(p,x) = k \log\left(\frac{N}{K+1}\right). \quad (1)$$

3.4.3 Chi-square Measure (CS)

The χ^2 measure has been used as a test for dependence between two words in various natural language processing tasks including collocation detection, identification of translation pairs in aligned corpora, and measuring corpus similarity [18]. Given a contingency table resembling that shown in Table 1, the χ^2 measure compares the observed frequencies in the table with the frequencies expected for independence. Then it is likely that the candidate x is an alias of the name p if the difference between the observed and expected frequencies is large for some candidate x.

The χ^2 measure sums the differences between observed and expected values in the contingency table and is scaled by the magnitude of the expected values. Actually, the χ^2 measure is given as

$$\chi^2 = \sum_{i,j} \frac{(O_{ij}-E_{ij})^2}{E_{ij}}. \quad (2)$$

Here, O_{ij} and E_{ij} respectively denote the observed and expected frequencies. We apply Formula 2 to Table 1 and define χ^2 ranking score, $\mathrm{CS}(p,x)$, of a candidate x as an alias of a name p as follows:

$$\mathrm{CS}(p,x) = \frac{N\{k(N-K-n+k)-(n-k)(K-k)\}^2}{nK(N-K)(N-n)}. \quad (3)$$

The CS score is used to rank candidate aliases of a name.

3.4.4 Log Likelihood Ratio (LLR)

The log likelihood ratio (LLR) [19] is defined as the ratio between the likelihoods of two alternative hypotheses: that the name p and the candidate alias x are independent or the name p and that the candidate alias x are dependent. Likelihood ratios have been used often in collocation discovery [18]. Unlike the χ^2 measure, likelihood ratios are robust against sparse data and have a more intuitive definition. The LLR-based alias ranking score, $\mathrm{LLR}(p, x)$, is computed using values in Table 1 as

$$\begin{aligned}\mathrm{LLR}(p,x) &= k\log\frac{kN}{nK} + (n-k)\log\frac{(n-k)N}{n(N-K)} \\ &+ (K-k)\log\frac{N(K-k)}{K(N-n)} \\ &+ (N-K-n+k)\log\frac{N(N-K-n+k)}{(N-K)(N-n)}.\end{aligned} \tag{4}$$

3.4.5 Pointwise Mutual Information (PMI)

Pointwise mutual information [20] is a measure that is motivated by information theory; it is intended to reflect the dependence between two probabilistic events. For values of two random variables y and z, their pointwise mutual information is defined as

$$\mathrm{PMI}(y,z) = \log_2\frac{P(y,z)}{P(y)P(z)}. \tag{5}$$

Here, $P(y)$ and $P(z)$ respectively denote the probability of random variables y and z. Consequently, $P(y,z)$ is the joint probability of y and z. The probabilities in Formula 5 can be computed as marginal probabilities from Table 1 as

$$\mathrm{PMI}(p,x) = \log_2\frac{\frac{k}{N}}{\frac{K}{N}\frac{n}{N}} = \log_2\frac{kN}{Kn}. \tag{6}$$

3.4.6 Hypergeometric Distribution

Hypergeometric distribution [21] is a discrete probability distribution that describes the number of successes in a sequence of draws from a finite population without replacement. For example, the probability of the event that "k red balls are contained among n balls which are arbitrarily chosen from among N balls containing K red balls", is given by the hypergeometric distribution, $hg(N, K, n, k)$, as

$$hg(N,K,n,k) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}. \tag{7}$$

We apply the definition 7 of hypergeometric distribution to the values in Table 1 and compute the probability $HG(p, x)$ of observing more than k number of co-occurrences of the name p and candidate alias x. The value of $HG(p, x)$ is give by,

$$\begin{aligned}\mathrm{HG}(p,x) &= -\log_2(\sum_{l\geq k} hg(N,K,n,l)) \\ \max\{0, N+K-n\} &\geq l \geq \min\{n,K\}.\end{aligned} \tag{8}$$

The value $HG(p, x)$ indicates the significance of co-occurrences between p and x. We use $HG(p, x)$ to rank candidate aliases of a name.

3.4.7 Cosine Measure

The cosine measure is widely used to compute the association between words. For strength of association between elements in two sets, X and Y can be computed using the cosine measure:

$$\mathrm{cosine}(X,Y) = \frac{|X\cap Y|}{\sqrt{|X|}\sqrt{|Y|}}. \tag{9}$$

Here, $|X|$ denotes the number of elements in set X. Letting X be the occurrences of candidate alias x and Y the occurrences of name p, we define $\mathrm{cosine}(p, x)$ as a measure of association between a name and a candidate alias as

$$\mathrm{cosine}(p,x) = \frac{k}{\sqrt{n}\sqrt{K}}. \tag{10}$$

3.4.8 Overlap Measure

The overlap between two sets X and Y is defined as

$$\mathrm{overlap}(X,Y) = \frac{|X\cap Y|}{\min(|X|,|Y|)}. \tag{11}$$

Assuming that X and Y respectively represent occurrences of name p and candidate alias x, We define a ranking score based on the overlap measure to evaluate the appropriateness of a candidate alias.

$$\mathrm{overlap}(p,x) = \frac{k}{\min(n,K)} \tag{12}$$

3.4.9 Dice Coefficient

Smadja [22] proposes the use of the Dice coefficient to retrieve collocations from large textual corpora. The Dice coefficient is defined over two sets X and Y as

$$\mathrm{Dice}(X,Y) = \frac{2|X\cap Y|}{|X|+|Y|}. \tag{13}$$

As with cosine and overlap measures, using the co-occurrence values in Table 1, we define a ranking score based on the Dice coefficient as

$$\mathrm{Dice}(p,x) = \frac{2k}{n+K}.$$

The Jaccard coefficient, which is monotonic with the Dice coefficient, was not considered because it gives the exact ranking of nodes as that given by the Dice coefficient. In general, if the Dice coefficient is g, then the corresponding Jaccard coefficient is given by $g/(2-g)$.

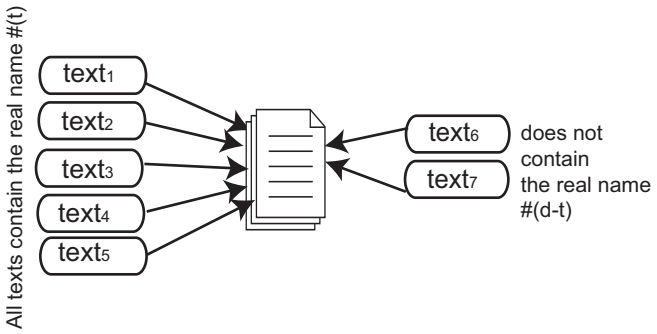

Fig. 6. Discounting the co-occurrences in hubs.

3.5 Hub discounting

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. Consider the situation shown in Figure 6 where a certain web page is linked to by two sets of anchor texts. One set of anchor texts contains the real name for which we must find aliases, whereas the other set of anchor texts contains various candidate aliases. If the majority of anchor texts linked to a particular web site use the real name to do so, then the confidence of that page as a source of information regarding the person whom we are interested in extracting aliases increases. We use this intution to compute a simple discouting measure for co-occurrences in hubs as follows.

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

$$\alpha(h, p) = \frac{t}{d}. \tag{14}$$

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, Formula 14 boosts hubs that are likely to contain information related to p, while penalizing those that contain various other topics.

3.6 Page-count-based Association Measures

In section 3.4 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 AND x"*, for a name p and a candidate alias x can be regarded as an approximation of their co-occurrences in the web. We compute popular word association measures using page counts returned by a search engine.

3.6.1 WebDice

We compute the Dice coefficient, $\text{WebDice}(p, x)$ (Section 3.4.9) between a name p and a candidate alias x using page counts as

$$\text{WebDice}(p, x) = \frac{2 \times \text{hits}(\text{“}p\ AND\ x\text{”})}{\text{hits}(p) + \text{hits}(x)}. \tag{15}$$

Here, $\text{hits}(q)$ is the page counts for the query q.

3.6.2 WebPMI

We compute the pointwise mutual information, $\text{WebPMI}(p, x)$ using page counts as follows:

$$\text{WebPMI}(p, x) = \log_2 \frac{L \times \text{hits}(\text{“}p\ AND\ x\text{”})}{\text{hits}(p) \times \text{hits}(x)}. \tag{16}$$

Here, 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. It should be noted however that the actual value of L is not required for ranking purposes because it is a constant and can be taken out from the definition of WebPMI (Equation 16) as an additive term. Both WebDice and WebPMI measures are described in detail in [23].

3.6.3 Conditional Probability

Using page counts, we compute the probability of an alias, given a name, as

$$Prob(x|p) = \frac{\text{hits}(\text{“}p\ AND\ x\text{”})}{\text{hits}(p)}.$$

Similarly, the probability of a name, given an alias, is

$$Prob(p|x) = \frac{\text{hits}(\text{“}p\ AND\ x\text{”})}{\text{hits}(x)}.$$

Unlike pointwise mutual information and the Dice coefficient, conditional probability is an asymmetric measure.

3.7 Training

Using a dataset of name-alias pairs, we train a ranking support vector machine [24] to rank candidate aliases according to their strength of association with a name. For a name-alias pair, we define three types of features: anchor text-based co-occurrence measures, web page-count-based association measures, and frequencies of observed lexical patterns. The nine co-occurrence measures described in Section 3.4.1 (CF, tfidf, CS, LLR, PMI, HG, cosine, overlap, Dice) 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 Section 3.6 and the frequency of lexical patterns extracted by algorithm 3.1 are used as features

in training the ranking SVM. We normalize each measure to range $[0, 1]$ to produce feature vectors for training. 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. Specifically, for each extracted candidate alias and the name under consideration, we compute the above mentioned feature and represent by a feature vector. The trained SVM model can then be used to assign a ranking score to each candidate alias. Finally, the highest-ranking candidate is selected as the correct alias of the name.

3.8 Dataset

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. Aliases were manually collected after referring various information sources such as Wikipedia and official home pages.

We crawled Japanese web sites and extracted anchor texts and the urls linked by the anchor texts. We find anchor texts pointing to any url, not limiting to pictures or any other resource type. A website might use links for purely navigational purposes which do not convey any semantic clues. In order 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 ignore anchor texts that contain these words. Moreover, we remove any links that point to pages within the same site. After removing navigational links our 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 anchor texts per url is 3.05 and the standard deviation is 54.02. Japanese texts do not use spaces to separate words. Therefore, tokenizing text is an important pre-processing step. We tokenize anchor texts using the Japanese morphological analyzer MeCab [25].

4 Experiments

4.1 Pattern Extraction

Algorithm 3.1 extracts over 8000 patterns for the 50 English personal names in our dataset. However, not all patterns are equally informative about aliases of a real name. Consequently, 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. First, for a pattern s we compute its precision and recall as follows:

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

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

Then, its F-score can be computed as:

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

Table 2 shows the patterns with the highest F scores extracted using the English personal names dataset. As shown in Table 2, unambiguous and highly descriptive patterns are extracted by the proposed method. Likewise, Table 3 shows the patterns with the highest F scores extracted using the English location names dataset. The same pattern extraction algorithm (Algorithm 3.1) can be used to extract patterns indicating aliases of personal names as well as aliases of location names, provided that we have a set of name, alias pairs for each entity type. This is particularly attractive because we do not need to modify the pattern extraction algorithm to handle different types of named entities. Figure 7 shows precision, recall and F-score for the top ranking patterns extracted for English personal names. From Fig. 7 we see that when we use more lower ranked patterns the recall improves at the expense of precision. Upto a certain point (ca. 200 patterns) F-score improves as a result of the improvement in recall. However, beyond this point the use of low precision patterns decreases the F-score. Consequently, we select the top 200 ranked patterns for the remainder of the experiments in this paper. Figure 7 implies that alias extraction methods that use a small number of manually created patterns [13] have low recall because they cannot cover all the different ways in which an alias can be expressed. Interestingly, among the automatically 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*). Although not shown in Table 2 and 3, patterns that contain punctuation symbols also appear among the top 200 patterns used for alias extraction.

4.2 Alias Extraction

In Table 4, we compare the proposed SVM-based method against various individual ranking scores (baselines) and previous studies of alias extraction (Hokama and Kitagawa (HK) [13]) on Japanese personal names dataset. We used linear, polynomial (quadratic), and radial basis functions (RBF) kernels for ranking SVM. Mean reciprocal rank (MRR) and Average Precision (AP) [26] is used to evaluate the different approaches. MRR is defined as follows,

$$\text{MRR} = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{R_i}. \tag{17}$$

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

TABLE 2
Lexical patterns with the highest F-scores for English personal name aliases.

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

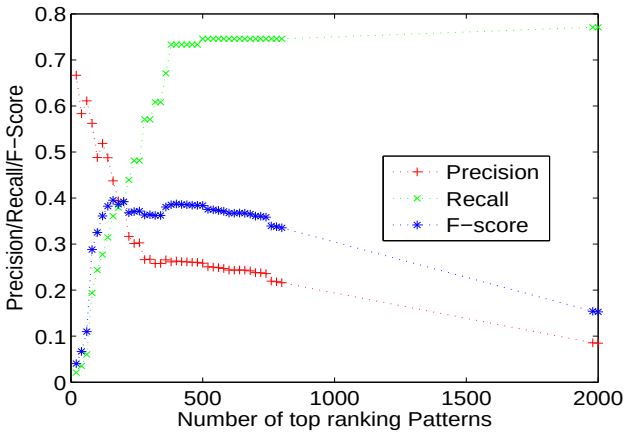

Fig. 7. Quality of the top ranked patterns for English personal name aliases.

Therein: R_i is the rank assigned to a correct alias and n is the total number of aliases. AP is defined as follows,

$$\text{AveragePrecision} = \frac{\sum_{r=1}^{k} \text{Pre}(r) \times \text{Rel(r)}}{\text{No of correct aliases}}. \quad (18)$$

Here, $\text{Rel}(r)$ is a binary valued function that returns 1 if the candidate at rank r is a correct alias for the name. Otherwise, it returns zero. Furthermore, $\text{Pre}(r)$ is the precision at rank r, which is given by,

$$\text{Pre}(r) = \frac{\text{no. of correct aliases in top r candidates}}{r}. \quad (19)$$

Different from the mean reciprocal rank, which focuses only on rank, AP 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. If a method ranks the correct aliases of a name on top, then it receives a higher MRR and AP values.

As shown in Table 4, 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 in performance occurs with more complex kernels, which is attributable to over-fitting.

TABLE 3
Lexical patterns with the highest F-scores for English location name aliases.

Patterns	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 4
Comparison with baselines and previous work.

Method	MRR	AP	Method	MRR	AP
SVM (Linear)	0.67	0.66	Prob$(p\|x)$	0.14	0.14
SVM (Quad)	0.65	0.64	CS(h)	0.12	0.06
SVM (RBF)	0.61	0.60	CF	0.08	0.03
HK	0.63	0.62	cosine	0.08	0.03
tfidf(h)	0.39	0.15	tfidf	0.07	0.03
WebDice	0.39	0.38	Dice	0.07	0.03
LLR(h)	0.39	0.15	overlap(h)	0.07	0.03
cosine(h)	0.37	0.14	PMI(h)	0.06	0.02
CF(h)	0.37	0.15	LLR	0.06	0.02
HG(h)	0.33	0.12	HG	0.04	0.01
Dice(h)	0.29	0.12	CS	0.01	0
Prob$(x\|p)$	0.21	0.21	PMI	0.01	0
WebPMI	0.14	0.13	overlap	0.01	0

Hokama and Kitagawa's method (HK) which uses manually created patterns, can only extract Japanese name aliases. Their method reports an MRR value of 0.63 on our Japanese personal names dataset. In Table 4, 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.

In Table 5, we evaluate the overall performance of the proposed method on each dataset. With each dataset we performed a 5-fold cross validation. As shown in Table 5, the proposed method reports high scores for both MRR and AP on all three datasets. Best results are achieved for the place name alias extraction task.

Table 6 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). It is noteworthy that most aliases do not share any words with the name nor acronyms, thus would not be correctly extracted from approximate string matching methods. It is interesting to see that, for actors the extracted aliases include their

TABLE 5
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 6
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**

roles in movies or television dramas (e.g. *Michael Knight* for *David Hasselhoff*).

Table 7 shows the top three ranking aliases extracted for *Hideki Matsui* by the proposed SVM (Linear) measure and the various baseline ranking scores. For each candidate we give its English translation within brackets alongside with the score assigned by the measure. The correct alias, *Godzilla*, is ranked first by SVM (RBF). Moreover, the correct alias is followed by the his last name *Matsui* and the team which he plays for, *New York Yankees*. tfidf(h), LLR(h), CF(h) all have the exact ranking for the top three candidates. *hide*, which is an abbreviated form of *Hideki* is ranked second by these measures. However, none of them contain the alias *Godzilla* among the top three candidates. The non-hub weighted measures have a tendency to include general terms such as *Tokyo*, *Yomiuri* (a popular Japanese newspaper), *Nikkei* (a Japanese business newspaper) and *Tokyo stock exchange*. A close analysis revealed that such general terms frequently co-occur with a name in hubs. Without adjusting the co-occurrences coming from hubs, such terms receive high ranking scores as shown in Table 7. It is noteworthy that the last name *Matsui* of the baseball player is ranked by most of the baseline measures as the top candidate.

4.3 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 [8], [27], we measured the association between two people using the pointwise mutual information (Equation 16) between their names on the web. We then use group average agglomerative clustering (GAAC) [18] 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, $\mathrm{Corr}(\Gamma)$, between two clusters X and Y is defined as

$$\mathrm{Corr}(\Gamma) = \frac{1}{2}\frac{1}{|\Gamma|(|\Gamma|-1)}\sum_{(u,v)\in\Gamma}\mathrm{sim}(u,v). \qquad (20)$$

Here, Γ is the merger of the two clusters X and Y. $|\Gamma|$ denotes the number of elements (persons) in Γ and $\mathrm{sim}(u,v)$ is the association between two persons u and v in Γ.

Ideally, people who work in the same field should be clustered into the same group. We used the B-CUBED metric [10] to evaluate the clustering results. The B-CUBED evaluation metric was originally proposed for evaluating cross-document co-reference 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

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

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

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

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

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

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

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

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

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. only using the "real name" as the query to obtain web hits). Next, we repeat the experiment by expanding the query with the top ranking alias extracted by the

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

TABLE 7
Top ranking candidate aliases for Hideki Matsui

Method	First		Second		Third	
	candidate	score	candidate	score	candidate	score
SVM (Linear)	Godzilla	8.50	Matsui	8.43	Yankees	8.11
tfidf(h)	Matsui	415.24	*hide*	147.78	Yankees	131.42
LLR(h)	Matsui	218.74	*hide*	61.39	Yankees	59.74
cosine(h)	Matsui	0.67	Yankees	0.28	*hide*	0.25
CF(h)	Matsui	43.99	*hide*	15.40	Yankees	12.27
HG(h)	Matsui	173.01	Yankees	49.21	*hide*	47.78
Dice(h)	Matsui	0.04	Yankees	0.02	*hide*	0.01
CS(h)	Matsui	5912.66	Major league	2262.52	player	2144.81
CF	Tokyo	171	Yomiuri	111	Nikkei	95
cosine	Yomiuri	0.79	Tokyo stock exchange	0.68	Matsui	0.60
tfidf	Yomiuri	1040.75	Tokyo	995.09	Tokyo stock exchange	884.64
Dice	Yomiuri	0.019	Tokyo stock exchange	0.017	Matsui	0.015
overlap(h)	play	1.25	Godzilla	1.20	Steinbrenner	0.80
PMI(h)	play	11.15	Godzilla	11.10	Steinbrenner	10.70
LLR	Yomiuri	243.58	Tokyo stock exchange	219.15	jiji.com	172.32
HG	Yomiuri	592.36	Tokyo stock exchange	500.84	Matsui	427.32
CS	jiji.com	104943.29	Tokyo stock exchange	53222.85	Yomiuri	51420.51
PMI	Komdatzien	11.62	picture	10.92	contents	10.90
overlap	Komdatzien	1.00	picture	1.00	contents	1.00

proposed algorithm (i.e."real name" OR "alias"). Table 8 shows 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. 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.

4.4 Web Search Task

To retrieve information about a particular person from a web search engine, it is common to include the name of the person in the query. In fact it has been reported that approximately one third of all web queries contain a person name [1]. A name can be ambiguous in the sense that there might exist more than one individual for a given name. As such, searching only by the name is insufficient to locate information regarding the person we are interested in. By including an alias that uniquely identify a person from his or her namesakes, it might be possible to filter out irrelevant search results. We set up an experiment to evaluate the effect of aliases in a web search task.

The experiment is conducted as follows. For a given individual, we search Google with the name as the query and collect top 50 search results. We then manually go through the search results one by one and decide whether they are relevant for the person we searched for. We count the total number of relevant results. We then append the name query with an alias of the person and repeat the above mentioned process. Table 9 summarizes the experimental results for 10 Japanese names in our dataset: **Hideki Matsui** (a major league baseball player playing for New York Yankees), **Minako Nakano** (an announcer for the Fuji TV corporation), **Maki Goto** (a J-pop singer), **Hidetoshi Nakata** (former member of the Japanese national football team), **Takuya Kimura** (an actor and a singer), **Yuichi Kimura** (a comedian), **Takafumi Horie** (former CEO of Japanese internet portal *Livedoor*), **Hikaru Utada** (a J-pop singer and song writer), **Ryuichi Sakamoto** (a musician), **Miki Ando** (a figure skater).

In Table 9, we use **F**, **L**, and **A** to denote the first name, last name and alias respectively. The notation *X*+*Y* stands for the conjunctive query, "*X*" AND "*Y*" of two terms X and Y. For example, for Hideki Matsui, F+A represents the query *"Hideki"* AND *"Godzilla"*. The numbers in Table 9 are the relevant results out of 50 top ranking results returned by Google for each query. Because we searched for Japanese names, we searched in google.co.jp for pages written in Japanese language.

In Table 9, the number of relevant results have improved for both first name (F) and last name (L) only queries when the aliases was added to the query (F vs F+A and L vs L+A). In particular, last names alone produce very poor results. This is because most Japanese last names are highly ambiguous. For example, the last name Nakano (person no. 2) is a place name as well as a person name and does not provide any results for Minako Nakano, the television announcer. Compared to last names, first names seem to have better precision for the people tested. It is also evident from Table 9 that aliases alone are insufficient to retrieve relevant results. Despite the popular alias *godzilla* for Hideki Matsui, searching alone by the alias produces just 3 relevant results. The highly ambiguous alias, *professor*, for Ryuichi Sakamoto (person no. 8) returns only the Wikipedia disambiguation entry for the musician. However, the combination of last name and alias significantly improves relevant results for all individuals.

We see that searching by the full name (F+L) returns perfect results for 7 out of the 10 people. However, it is noteworthy that including the aliases still improve

TABLE 9
Effect of aliases in personal name search

No	First Name	Last Name	Alias	F	L	A	F+A	L+A	F+L	F+L+A
1	Hideki	Matsui	godzilla	48	1	3	50	50	50	50
2	Minako	Nakano	nakami	8	0	4	50	38	50	50
3	Maki	Goto	gomaki	16	6	24	48	44	48	50
4	Hidetoshi	Nakata	*hide*	44	6	12	50	50	50	50
5	Takuya	Kimura	kimutaku	34	2	30	50	50	50	50
6	Yuichi	Kimura	brother kimu	29	0	45	49	45	50	50
7	Takafumi	Horie	horiemon	29	5	49	48	50	50	50
8	Hikaru	Utada	hikki	30	48	5	50	48	50	50
9	Ryuichi	Sakamoto	professor	22	3	1	36	17	49	50
10	Miki	Ando	mikiti	41	8	4	47	35	48	50

relevancy even in the remaining 3 cases. If we only consider the last names, *Takuya Kimura* (person no. 5) and *Yuichi Kimura* (person no 6) corresponds to namesakes for the last name *kimura*. Searching only by the last name retrieves very few (almost zero) relevant results. However, they have unique name aliases. Even searching only by their aliases improves relevancy by a high margin (30 and 45 results respectively).

The time taken to process 100 names in the Japanese personal names is as follows: pattern extraction ca. 3m (processing only top 100 snippets), candidate alias extraction ca. 5.5h (using 200 patterns), feature generation 2.4h, training ca. 5m, and testing 30s (50 names). Overall it takes 8.1h to run the system end to end. However, it is noteworthy that once the system is trained, detecting an alias of a given name requires only ca. 4.3m (candidate alias extraction ca. 3.1m, feature generation 1m, ranking 10s). To avoid overloading, we have set a graceful 1s time interval between queries sent to the Web search engine. Moreover, several steps in training such as pattern extraction and candidate alias extraction could be parallelized because we can process each name in the training dataset individually. Currently, we have not performed any optimization or parallelization in our implementation.

5 Implementation Considerations

In order to create a contingency table like the one shown in Table 1, we require anchor texts pointing to an url. In this section, we describe two independent approaches to efficiently extract all anchor texts pointing to a partcular url.

5.1 Extracting inbound anchor texts from a web crawl

In the case where we have a large web crawl at our disposal, then extracting all the inbound anchor texts of an url can be accomplished using a hash table, where the hash key is the url and the corresponding value contains a list of anchor texts that point to the url specified by the key. Algorithm 5.1 illustrates the pseudo code to achieve this task. Given a web crawl, we extract the set L of links from the crawled data, where L contains tuples (a_i, u_i) of anchor texts a_i and the urls u_i pointed by the anchor texts. The function **ANCHORS** in algorithm 5.1 processes the links in L and returns a hash H where keys are urls u_i and values, $H[u_i]$ are lists of anchor texts pointing to u_i. H is initialized to an empty hash and for each tuple (a_i, u_i) in L, if H already contains the key u_i then the function $\mathrm{Append}(H[u_i], a_i)$ appends the anchor text a_i to the list $H[u_i]$. If u_i does not exist in H then a new list containing a_i is created and assigned to key u_i. If the number of links to be processed is large, then H can be implemented as a distributed hash table.

Algorithm 5.1: ANCHORS(L)

comment: Extract inbound anchor texts from L.

$H = \{\}$
for each $(a_i, u_i) \in L$
do **if** $u_i \in H$
then $Append(H[u_i], a_i)$
else $H[u_i] \leftarrow [\,]$
return (H)

Fig. 8. Extract inbound anchor texts from a web crawl.

5.2 Retrieving inbound anchor texts from a web search engine

Crawling and indexing a large set of anchor texts from the web requires both computation power and time. However, there already exist large web indexes created by major web search providers such as Google, Yahoo and MSN, Most web search engines provide query APIs and search operators to search in anchor texts. For example, Google provides the search operator *inanchor* to retrieve urls pointed by anchor texts that contain a particular word. Moreover, the search operator *link* returns urls which are linked to a given url. In Figure 9 we describe an algorithm to extract a set of anchor texts that points to urls which are also pointed by a given name using the basic search operators provided by a web search engine.

Given a name p, the function $\mathrm{ANCHORS}(p)$ described in algorithm 5.2 returns a hash H where keys of H are urls and the corresponding values contain lists of anchor

Algorithm 5.2: ANCHORS(p)

comment: Extract inbound anchor texts for p

$H = \{\}$
$S \leftarrow InAnchor(p)$
for each url $u_i \in S$
do
  $H[u_i] \leftarrow [\,]$
  $T \leftarrow Link(u_i)$
  for each page $t \in T$
  do
    $L \leftarrow ExtractLinks(t)$
    for each $(a_k, q_k) \in L$
    do
      if $q_k = u_i$
      then $Append(H[u_i], a_k)$
return (H)

Fig. 9. Get inbound anchor texts from a web search engine for a name p.

texts pointing to the url specified by the key. Moreover, for each url in H, at least one of the retrieved anchor texts contains the name p. Therefore, all the words that appear in anchor texts extracted by the algorithm contains candidate aliases of the name p. The algorithm first initializes the hash H to empty hash. Then the function $\mathrm{InAnchor}(p)$ retrieves urls that are pointed by anchor texts that contain p. For example, in Google, the search operators *inanchor* or *allinanchor* can be used for this purpose. For each url u_i retrieved by the function InAnchor we initialize the list of anchor texts to empty list and retrieve the set of pages T that link to u_i using the function $\mathrm{Link}(u_i)$. For example, in Google, the search operator *link* provides the desired functionality. Then for each page t in T the function $\mathrm{ExtractLinks}$ extracts anchor texts and links from t. Finally we accumulate anchor texts that point to u_i and store them as a list in H.

5.3 Determining the number of aliases

The number of aliases of a given real name depends on numerous factors such as the popularity of the person, and the field of work. For examples, celebrities in the movie industry tend to have multiple name aliases. When applying the proposed method in practice, it is important to know how many candidate aliases must be displayed for a given real name. For this purpose, we compute a threshold on the ranking scores. Specifically, for each real name in our training dataset, we use the trained SVM model to assign a ranking score to the extracted candidate aliases. We then normalize the ranking scores to range $[0, 1]$ and sort the candidate aliases (for each real name) in the descending order of their ranking scores. Finally, we compute the cutoff threshold as the average ranking score of the candidate at the correct number of aliases for each name in the dataset. For example, let us assume that we have two aliases for a name A and three aliases for another name B. Moreover, let the normalized ranking score of the candidate at rank two for the name A is $A@2$ and the same for the candidate at rank three for the name B is $B@3$. Then the cutoff threshold value is $(A@2 + B@3)/2$. To evaluate this method of finding the number of aliases for a given name, we employ 5-fold cross validation method on the Japanese personal names dataset. The average cutoff threshold over the five folds for the Japanese personal names dataset is 0.9716. If we select only those candidates with ranking score greater than this threshold, then we correctly predict the number of aliases 84% of the times.

6 Discussion

We utilized both lexical patterns extracted from snippets retrieved from a web search engine as well as anchor texts and links in a web crawl. Lexical patterns can only be matched within the same document. In contrast, anchor texts can be used to identify aliases of names across documents. The use of lexical patterns and anchor texts respectively, can be considered as an approximation of within document and cross-document alias references. Experimentally, in Section 4.2 we showed that by combining both lexical patterns-based features and anchor text-based features we can achieve better performance in alias extraction. The contingency table introduced in Section 3.4 can only incorporate first order co-occurrences. An interesting future research direction would be to create a word co-occurrence graph using the definition of anchor texts-based co-occurrences given in the paper and run graph mining algorithms to identify second or higher order associations between a name and candidate aliases.

An alias might not always uniquely identify a person. For example, the alias Bill is used to refer many individuals who has the first name *William*. The namesake disambiguation problem focuses on identifying the different individuals who have the same name. The existing namesake disambiguation algorithms assume the real name of a person to be given, and does not attempt to disambiguate people who are referred only by aliases. In Section 4.4, we showed experimentally that the knowledge of aliases is helpful to identify a particular person from his or her namesakes on the web. Aliases are one of the many attributes of a person that can be useful to identify that person on the web. Extracting common attributes such as date of birth, affiliation, occupation, and nationality have been shown to be useful for namesake disambiguation on the web [28].

7 Conclusion

We proposed a lexical-pattern-based approach to extract aliases of a given name. We use a set of names and their aliases as training data to extract lexical patterns that describe numerous ways in which information related to aliases of a name is presented on the web.

Next, we substitute the real name of the person that we are interested in finding aliases in the extracted lexical patterns, and download snippets from a web search engine. We extract a set of candidate aliases from the snippets. The candidates are ranked using various ranking scores computed using three approaches: lexical pattern frequency, co-occurrences in anchor texts, and page counts-based association measures. Moreover, we integrate the different ranking scores to construct a single ranking function using ranking support vector machines. We evaluate the proposed method using three datasets: an English personal names dataset, an English location names dataset, and a Japanese personal names dataset. The proposed method reported high MRR and AP scores on all three datasets and outperformed numerous baselines and a previously proposed alias extraction algorithm. Discounting co-occurrences from hubs is important to filter the noise in co-occurrences in anchor texts. For this purpose we proposed a simple and effective hub discounting measure. Moreover, the extracted aliases significantly improved recall in a relation detection task and render useful in a web search task.

REFERENCES

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

[2] J. Artiles, J. Gonzalo, and F. Verdejo, "A testbed for people searching strategies in the www," in *Proc. of SIGIR'05*, 2005, pp. 569–570.

[3] G. Mann and D.Yarowsky, "Unsupervised personal name disambiguation," in *Proc. of CoNLL'03*, 2003, pp. 33–40.

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

[5] G. Salton and M. McGill, *Introduction to Modern Information Retreival*. New York, NY: McGraw-Hill Inc., 1986.

[6] M. Mitra, A. Singhal, and C. Buckley, "Improving automatic query expansion," in *Proc. of SIGIR'98*, 1998, pp. 206–214.

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

[8] 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.

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

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

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

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

[13] 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)*, 2006, pp. 121–130.

[14] M. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *Proc. of COLING'92*, 1992, pp. 539–545.

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

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

[17] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing and Management*, vol. 24, pp. 513–523, 1988.

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

[19] T. Dunning, "Accurate methods for the statistics of surprise and coincidence," *Computational Linguistics*, vol. 19, pp. 61–74, 1993.

[20] K. Church and P. Hanks, "Word association norms, mutual information and lexicography," *Computational Linguistics*, vol. 16, pp. 22–29, 1991.

[21] T. Hisamitsu and Y. Niwa, "Topic-word selection based on combinatorial probability," in *Proc. of NLPRS'01*, 2001, pp. 289–296.

[22] F. Smadja, "Retrieving collocations from text: Xtract," *Computational Linguistics*, vol. 19, no. 1, pp. 143–177, 1993.

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

[24] T. Joachims, "Optimizing search engines using clickthrough data," in *Proc. of KDD'02*, 2002.

[25] T. Kudo, K. Yamamoto, and Y. Matsumoto, "Applying conditional random fields to japanese morphological analysis," in *Proc. of EMNLP'04*, 2004.

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

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

[28] S. Sekine and J. Artiles, "Weps 2 evaluation campaign: overview of the web people search attribute extraction task," in *proc. of the 2nd Web People Search Evaluation Workshop (WePS 2009) at 18th International World Wide Web Conference*, 2009.

Danushka Bollegala received his BS, MS and PhD degrees from the University of Tokyo, Japan in 2005, 2007, and 2009. He is currently a research fellow of the Japanese society for the promotion of science (JSPS). His research interests are natural language processing, Web mining and artificial intelligence.

Yutaka Matsuo is an associate professor at Institute of Engineering Innovation, the University of Tokyo, Japan. He received his BS, MS, and PhD degrees from the University of Tokyo in 1997, 1999, and 2002. He joined National Institute of Advanced Industrial Science and Technology (AIST) from 2002 to 2007. He is interested in social network mining, text processing, and semantic web in the context of artificial intelligence research.

Mitsuru Ishizuka is a professor at Graduate School of Information Science and Technology, the University of Tokyo, Japan. He received his BS and PhD degrees in electronic engineering from the University of Tokyo in 1971 and 1976. His research interests include artificial intelligence, Web intelligence, and lifelike agents.