A Person-Name Filter for Automatic Compilation of Bilingual Person-Name Lexicons

Satoshi Sato and Sayoko Kaide

Graduate School of Engineering, Nagoya University
Furo-cho, Chikusa-ku, Nagoya, 464-8603, JAPAN
ssato@nuee.nagoya-u.ac.jp, kaide@sslab.nuee.nagoya-u.ac.jp

Abstract

This paper proposes a simple and fast person-name filter, which plays an important role in automatic compilation of a large bilingual person-name lexicon. This filter is based on pn_score, which is the sum of two component scores, the score of the first name and that of the last name. Each score is calculated from two term sets: one is a dense set in which most of the members are person names; another is a baseline set that contains less person names. The pn_score takes one of five values, $\{+2, +1, 0, -1, -2\}$, which correspond to *strong positive*, *positive*, *undecidable*, *negative*, and *strong negative*, respectively. This pn_score can be easily extended to bilingual pn_score that takes one of nine values, by summing scores of two languages. Experimental results show that our method works well for monolingual person names in English and Japanese; the F-score of each language is 0.929 and 0.939, respectively. The performance of the bilingual person-name filter is better; the F-score is 0.955.

1. Introduction

Named entity recognition is important for many NLP applications such as information retrieval, information extraction, and question answering. Named entity translation is also important for bilingual NLP applications such as machine translation and cross-language information retrieval. Large monolingual and bilingual named entity lexicons are valuable resources for such applications.

Named entity lexicons should be updated continuously to cover new names. Automatic compilation of named entity lexicons is desired to respond this request. For person names, a major class of named entities, we have developed an automatic lexicon compiler (Sato, 2009a), which has already produced an English-Japanese person-name lexicon with 406K entries. In the compilation process, a person-name filter played an important role in producing the large and accurate lexicon. This paper describes the person-name filter, which is simple, fast, and accurate.

2. Method

2.1. Two Term Sets D and B

Intuition behind the method is that there are typical first names and last names, such as "John" and "Smith." In contrast, some words are rarely used as first and last names, such as "The" and "Have." Every word has typicality of first name and that of last name. Our method estimates such typicality from two term sets, D and B, which satisfy the following requirements.

1. The set D is a *dense* set of person names (full names); i.e., most of the members (e.g., more than 90%) in D are person names.

2. The set B is a *baseline* set. It includes person names but their proportion is not large, e.g., less than 30%.

3. The set D is a subset of B; i.e., $D \subset B$.

4. For every term $t \in B$, two functions, $\mathrm{first}(t)$ and $\mathrm{last}(t)$, are defined. The former extracts the word in the first-name position from the term t; the latter extracts the word in the last-name position. For example, in English, $\mathrm{first}(t)$ is the function that extracts the first word in the term t, and $\mathrm{last}(t)$ extracts the last word.

It is notable that the above requirements are much looser than the standard requirement of supervised learning; D may include negative instances and B may include positive instances. The sufficient condition is that the proportion of positive instances in D is much larger than that in B.

2.2. Person-Name Score

By using these two sets, D and B, we define pn_score of a term t as follows.

$$pn\_score(t, D, B) = score_{\mathrm{first}}(\mathrm{first}(t), D, B) + score_{\mathrm{last}}(\mathrm{last}(t), D, B) \quad (1)$$

The pn_score is the sum of two component scores, the score of the first name and that of the last name. Each component score $score_c$ (c = first or last) is determined as follows, where $|\cdot|$ means the size of a set.

$$score_c(w, D, B) = \begin{cases} +1 & \text{if } diff_c(w, D, B) \geq 1 \\ 0 & \text{if } -1 < diff_c(w, D, B) < 1 \\ -1 & \text{if } diff_c(w, D, B) \leq -1 \end{cases} \quad (2)$$

$$diff_c(w, D, B) = freq_c(w, D) - \frac{freq_c(w, B)}{|B|}|D| \quad (3)$$

$$freq_c(w, X) = |\{x | x \in X, w = c(x)\}| \quad (4)$$

Let us explain the above definitions when c is first. In this case, only words that appear in the first-name position are considered. $freq_{\mathrm{first}}(w, X)$ means the frequency of w in the set X. Therefore, $diff_{\mathrm{first}}(w, D, B)$ means the difference between the frequency of w in D and the baseline frequency calculated from B. If $diff_{\mathrm{first}}(w, D, B) \geq 1$, w will be a first name because w appears more frequently in D than B. In the opposite side, if $\mathit{diff}_{\text{first}}(w, D, B) \leq -1$, w must not be a first name. If $-1 < \mathit{diff}_{\text{first}}(w, D, B) < 1$, the difference is not reliable because $\mathit{freq}_{\text{first}}(w, D)$ is an integer. When the baseline frequency $\mathit{freq}_{\text{first}}(w, B) \cdot |D|/|B|$ is 3.42, for example, both $\mathit{freq}_{\text{first}}(w, D) = 3$ and 4 do not conflict with the baseline frequency. As a result, $\mathit{score}_{\text{first}}(w, D, B)$ takes one of three values, $\{+1, 0, -1\}$, which correspond to *positive*, *undecidable*, and *negative*, respectively.

Because $\mathit{score}_{\text{last}}(w, D, B)$ also takes one of three values, $\mathit{pn\_score}(t, D, B)$ takes one of five values, $\{+2, +1, 0, -1, -2\}$, which correspond to *strong positive*, *positive*, *undecidable*, *negative*, and *strong negative*, respectively. This method is applicable to any language, by preparing two term sets, D and B.

The $\mathit{pn\_score}$ can be easily extended to the bilingual person-name score, by summing scores of two languages; the score takes one of nine values between $+4$ and -4.

In summary, this method provides a trinary classifier of first names and last names, a five-value classifier of person names (full names), and a nine-value classifier of bilingual person-name pairs.

2.3. Component Lexicon

In practice, when D and B are given, we calculate $\mathit{score}_{\text{first}}$ and $\mathit{score}_{\text{last}}$ for every word w in B in advance, and store them into a table, which we call *component lexicon*. Note that this lexicon stores not only positive instances (first and last names) but also negative instances, whereas traditional lexicons store only positive instances.

By using this component lexicon, $\mathit{pn\_score}$ can be computed very fast, according to Equation (1). Note that score_c is 0 (undecidable) if w does not exist in the lexicon (i.e., B), because both $\mathit{freq}_c(w, B)$ and $\mathit{freq}_c(w, D)$ are 0.

3. Bilingual Lexicon Compilation with the Person-Name Filter

3.1. Compilation Method

Our final goal is to compile bilingual (English-Japanese[1]) person-name lexicons automatically. The lexicon compilation process consists of three steps: collection of monolingual person names, transliteration, and final selection of bilingual person-name pairs. The person-name filter (pn-filter) is used in all steps.

The first step is collection of monolingual person names, in either English or Japanese. It is easy to collect terms that are possibly person names. From English texts, we simply extract all terms that consist of two or more words whose first letters are capital. In Japanese, transliterated proper names are written as Katakana[2] terms so we simply extract all Katakana terms from Japanese texts. As a result, a very huge and noisy list is produced. The pn-filter is used to reduce the size by selecting terms that are probably person names; high-speed execution is crucial here. In this step, the loose filter ($\mathit{pn\_score} \geq 0$) is used, because high recall is preferred.

The second step is transliteration, which is performed by a web-based transliterator (Sato, 2009b). It produces a list of bilingual person-name pairs from a monolingual list. In case we use a crawler version of the web-based transliterator (Sato, 2009a), new monolingual person-name candidates are also collected simultaneously with transliteration, where the pn-filter is used for selecting candidates.

The final step selects only reliable pairs from the obtained bilingual list by using several heuristics including the bilingual pn-filter. In this step, the strict filter ($\mathit{pn\_score} \geq 1$) is used, because high precision is preferred.

Note that this compilation process can be viewed as generation process of a larger D for the pn-filter. It means that our pn-filter can be enhanced in a bootstrap manner.

3.2. Actual Compilation

By the above procedure, two English-Japanese person-name lexicons have been complied automatically. The small one has 230K entries with 90% accuracy, which was complied in the first cycle. The large one has 406K entries with 93% accuracy, which was complied in the second cycle.

At the beginning in the first cycle, we prepared an initial set of Japanese person names, D_{J0}. In Japanese newspaper articles, person names often appear with *San*, which corresponds to "Mr." and "Ms." in English. In case a Katakana term appears with *San*, it must be a transliterated person name. By using this heuristic, we obtained the initial set with 64,438 entries from a 15-year volume of newspaper articles; the proportion of positive instances is more than 95%. We use this set in the first cycle of lexicon compilation. Note that we did not use the bilingual pn-filter in this cycle because of a lack of English D.

At the end of the first cycle, we obtained the smaller bilingual lexicon. From this lexicon, we made an enhanced Japanese set D_{J1} and an English set D_{E1} as follows.

$$D_{J1} = D_{J0} \cup \text{"Japanese person names in the obtained lexicon"} \quad (5)$$

$$D_{E1} = D_{E0}(= \phi) \cup \text{"English person names in the obtained lexicon"} \quad (6)$$

In the second cycle, we used D_{J1} and D_{E1} for the monolingual and bilingual pn-filters. At the end of the second cycle, we obtained the larger bilingual lexicon. From this lexicon, we made D_{J2} and D_{E2} in the same way, for the next cycle.

In summary, by the above compilation process, we obtained three Japanese dense sets, D_{J0}, D_{J1}, and D_{J2}; and two English dense sets D_{E1} and D_{E2}.

4. Performance

We have conducted an experiment that measures performance of our method. In this experiment, we used a test set that consists of manually labeled English-Japanese term pairs (1,253 positive and 373 negative instances),

[1]In this paper, we use the term *English person names* as full names appeared in English texts, and the term *Japanese person names* as Japanese transliterations of English person names.

[2]Katakana is one component of the Japanese writing system along with Hiragana and Kanji. It mainly used for writing foreign-language origin words.

				component lexicon					
				$score_{\text{first}}$			$score_{\text{last}}$		
ID	D	$\|D\|$	$\|B\|$	$+1$	0	-1	$+1$	0	-1
J0	D_{J0}	64,438	2,720,278	14,629	376,640	6,004	36,002	552,514	5,123
				0.037	0.948	0.015	0.061	0.931	0.009
J1	D_{J1}	268,299	2,818,880	22,119	357,710	19,197	90,597	504,606	15,740
		(+203,861)	(+98,602)	0.055	0.896	0.048	0.148	0.826	0.026
J2	D_{J2}	473,560	2,980,307	36,817	341,141	28,027	145,497	475,270	23,516
		(+205,261)	(+161,427)	0.091	0.840	0.069	0.226	0.738	0.036
E1	D_{E1}	186,201	1,792,635	15,298	176,255	10,569	67,088	270,338	10,220
				0.076	0.872	0.052	0.193	0.778	0.029
E2	D_{E2}	356,810	1,929,068	30,713	163,417	16,763	114,316	246,212	16,329
		(+170,609)	(+136,433)	0.146	0.775	0.079	0.303	0.653	0.043

upper line: number, lower line: proportion

Table 1: Five monolingual pn-filters

which were randomly selected from the HeiNER dictionary (Wentland et al., 2008). Note that the HeiNER dictionary contains all types of named entities, not limited to person names.

The pn-filter requires two sets, D and B, that satisfy $D \subset B$. As mentioned above, we have three different Japanese D and two different English D. In order to satisfy the requirement $D \subset B$ for each D, we first prepared B^- and then made B as follows.

$$B = B^- \cup D \tag{7}$$

The Japanese B^- consists of 2.72M Katakana terms, which were extracted from a web corpus. The English B^- consists of 1.67M English terms that consist of two or more words whose first letters are capital; these terms were extracted from English Wikipedia's abstracts.

Table 1 shows the data of five monolingual pn-filters: the size of D, the size of B, and the size of the component lexicon. From this table, we can see that a larger D increases both the number of positive words and that of negative words and decreases the number of undecidable words. From this fact, we expect that a larger D brings better performance.

Another interesting observation is that the proportion of positive first names is much smaller than that of positive last names; in contrast, the proportion of negative first names is larger than that of negative last names. It means that words used as first names are much limited than words used as last names.

Table 2 shows the experimental results of the five monolingual pn-filters. From this table, we can see that our simple method works very well. For each language, the best performance is obtained by using the largest D, as we expected. The performance of the J2 strict filter (i.e., $pn\_score \geq 1$) is recall = 0.917, precision = 0.962, and F-score = 0.939. That of the E2 strict filter is recall = 0.886, precision = 0.977, and F-score = 0.929, which is slightly lower than that of the Japanese one. This performance difference probably comes from the fact that we did not use any seed set for English.

The parenthesized values show the performance of noise reduction. For example, in case of J2, when we remove terms

		pn_score				
ID		+2	+1	0	-1	-2
J0	p	488	606	147	12	0
	n	3	20	115	149	86
	R	0.389	0.873	0.990	(0.630)	(0.231)
	P	0.994	0.979	0.900	(0.951)	(1.000)
	F	0.560	0.923	0.943	(0.758)	(0.375)
J1	p	730	390	117	15	1
	n	8	29	88	123	125
	R	0.583	0.894	0.987	(0.665)	(0.335)
	P	0.989	0.968	0.908	(0.939)	(0.992)
	F	0.733	0.929	0.946	(0.779)	(0.501)
J2	p	843	306	87	17	0
	n	16	29	88	99	141
	R	0.673	**0.917**	0.986	(**0.643**)	(0.378)
	P	0.981	**0.962**	0.903	(**0.934**)	(1.000)
	F	0.798	**0.939**	0.943	(0.762)	(0.549)
E1	p	607	430	157	53	6
	n	1	16	90	118	148
	R	0.484	0.828	0.953	(0.713)	(0.397)
	P	0.998	0.984	0.918	(0.818)	(0.961)
	F	0.652	0.899	0.935	(0.762)	(0.562)
E2	p	753	357	108	29	6
	n	2	24	77	107	163
	R	0.601	**0.886**	0.972	(0.724)	(0.437)
	P	0.997	**0.977**	0.922	(0.885)	(0.964)
	F	0.750	**0.929**	0.946	(0.796)	(0.601)

p: # of positive instances, n: # of negative instances,
R: recall, P: precision, F: F-score,
parenthesized values: performance of noise reduction

Table 2: Result: monolingual pn-filters

whose scores are less than zero, 64.3% of noise (negative instances) are removed with 93.4% precision.

Table 3 shows the performance of two bilingual pn-filters. It is better than the monolingual performance because of the two-side check. The performance of the EJ2 strict filter ($pn\_score \geq 1$) is recall = 0.940, precision = 0.970, and F-score = 0.955.

Table 4 shows examples of classification errors by JE2. From this table, we can see that most terms in classification errors are not English-origin. For example, six Ital-

			bilingual pn_score								
ID	D		+4	+3	+2	+1	0	-1	-2	-3	-4
EJ1	D_{E1} & D_{J1}	p	548	180	312	96	84	26	7	0	0
		n	1	2	13	8	71	33	94	57	94
		R	0.437	0.581	0.830	0.907	0.974	(0.745)	(0.657)	(0.405)	(0.252)
		P	0.998	0.996	0.985	0.979	0.928	(0.894)	(0.972)	(1.000)	(1.000)
		F	0.608	0.734	0.901	0.942	0.950	(0.813)	(0.784)	(0.576)	(0.403)
EJ2	D_{E2} & D_{J2}	p	662	202	256	58	56	12	6	1	0
		n	2	8	14	13	61	30	76	60	109
		R	0.528	0.690	0.894	**0.940**	0.985	(0.737)	(0.657)	(0.453)	(0.292)
		P	0.997	0.989	0.979	**0.970**	0.926	(0.935)	(0.972)	(0.994)	(1.000)
		F	0.691	0.812	0.935	**0.955**	0.955	(0.825)	(0.784)	(0.622)	(0.452)

p: # of positive instances, n: # of negative instances, R: recall, P: precision, F: F-score
parenthesized values: performance of noise reduction

Table 3: Result: bilingual pn-filters

ian municipality names are incorrectly classified as person name. In general, words originated in other languages less frequently appear in English texts, and less frequent words tend to be inaccurate in the classification, which is inevitable when we use any statistical method.

5. Conclusion

This paper proposed a simple and fast person-name filter, which plays an important role in automatic compilation of a large bilingual person-name lexicon. This filter is based on pn_score, which is calculated from two sets: one is a dense set in which most of the members are person names; another is a baseline set that contains less person names. Experimental results show that our method works well not only for monolingual person names but also for bilingual person names.

6. Acknowledgments

This work was supported by JSPS KAKENHI 21300094 and the Kayamori Foundation of Information Science Advancement.

7. References

Satoshi Sato. 2009a. Crawling English-Japanese person-name transliterations from the Web. In *Proc. of 18th International World Wide Web Conference (WWW 2009)*, pages 1151–1152.

Satoshi Sato. 2009b. Web-based transliteration of person names. In *WI-IAT '09: Proc. of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, pages 273–278.

Wolodja Wentland, Johannes Knopp, Carina Silberer, and Matthias Hartung. 2008. Building a multilingual lexical resource for named entity disambiguation, translation and transliteration. In *Proc. of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*.

p/n	b	term	m
n	4	Annabel Lee (*poem*)	2
		アナベル・リー	2
n	4	Meana Sardo (*municipality*)	2
		メアーナ・サルド	2
n	3	Banja Luka (*city*)	1
		バニャ・ルカ	2
n	3	Dizzy Miss Lizzy (*song*)	1
		ディジー・ミス・リジー	2
n	3	McDonnell Douglas (*company*)	1
		マクドネル・ダグラス	2
n	3	Orio Canavese (*municipality*)	1
		オーリオ・カナヴェーゼ	2
n	3	Pieve San Giacomo (*municipality*)	1
		ピエーヴェ・サン・ジャコモ	2
n	3	Romano Canavese (*municipality*)	1
		ロマーノ・カナヴェーゼ	2
n	3	Solbiate Olona (*municipality*)	1
		ソルビアーテ・オローナ	2
n	3	Vico Canavese (*municipality*)	1
		ヴィーコ・カナヴェーゼ	2
p	-2	Augusta of Saxe-Weimar	-1
		アウグスタ・フォン・ザクセン＝ヴァイマル＝アイゼナハ	-1
p	-2	Gottlieb Daimler	-1
		ゴットリープ・ダイムラー	-1
p	-2	Haji Mohammad Chamkani	-1
		ハジ・モハンマド・チャムカニ	-1
p	-2	Princess Louise-Élisabeth of France	-2
		ルイーズ・エリザベート・ド・フランス	0
p	-2	Sher Ali Khan	-2
		シール・アリー・ハーン	0
p	-2	Thor Hushovd	-1
		トル・フースホフト	-1
p	-3	Ebenezer Hazard	-2
		エベニーザー・ハザード	-1

p: positive instance, n: negative instance,
b: bilingual pn_score, m: monolingual pn_score

Table 4: Classification errors