Computational Linguistics and Chinese Language Processing
Vol. 11, No. 3, September 2006, pp. 183-200

An Approach to Using the Web as a Live Corpus for Spoken Transliteration Name Access

Ming-Shun Lin[*], Chia-Ping Chen[+] and Hsin-Hsi Chen[*]

Abstract

Recognizing transliteration names is challenging due to their flexible formulation and lexical coverage. In our approach, we employ the Web as a giant corpus. The patterns extracted from the Web are used as a live dictionary to correct speech recognition errors. The plausible character strings recognized by an Automated Speech Recognition (ASR) system are regarded as query terms and submitted to Google. The top *N* snippets are entered into PAT trees. The terms of the highest scores are selected. Our experiments show that the ASR model with a recovery mechanism can achieve 21.54% performance improvement compared with the ASR only model on the character level. The recall rate is improved from 0.20 to 0.42, and the MRR from 0.07 to 0.31. For collecting transliteration names, we propose a named entity (NE) ontology generation engine, called the X_{NE}-*Tree* engine, which produces relational named entities by a given seed. The engine incrementally extracts high co-occurring named entities with the seed. A total of 7,642 named entities in the ontology were initiated by 100 seeds. When the bi-character language model is combined with the NE ontology, the ASR recall rate and MRR are improved to 0.48 and 0.38, respectively.

1. Introduction

Named entities [MUC 1998], which denote persons, locations, organizations, etc., are common foci of searchers. Thompson and Dozier [1997] showed that named entity recognition (NER) could improve the performance of information retrieval systems. Capturing named entities is challenging due to their flexible formulation and novelty. The issues behind speech recognition make named entity recognition more challenging on the

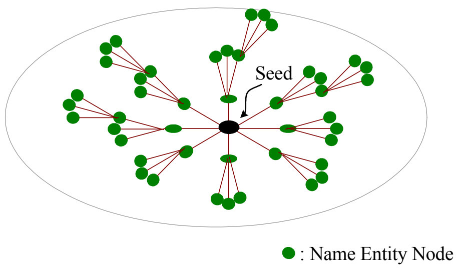
[*] Department of Computer Science and Information Engineering, National Taiwan University,
Taipei, 106 Taiwan
E-mail: {d91022 , hhchen}@csie.ntu.edu.tw
[+] Department of Computer Science and Engineering, National Sun Yat-Sen University,
70, Lien-Hai Road, Kaohsiung, 804 Taiwan
E-mail: cpchen@cse.nsysu.edu.tw

[Received February 16, 2006; Revised June 23, 2006; Accepted July 10, 2006]

spoken level than on the written level. This paper focuses on a special type of named entities, called transliteration names. They describe foreign people, places, etc. Spoken transliteration name recognition is useful for many applications. For example, cross language image retrieval via spoken queries aims to employ the latter in one language to retrieve images with captions in another language [Lin *et al*. 2004].

In the past, Appelt and Martin [1999] adapted the TextPro system to process transcripts generated by a speech recognizer. Miller *et al.* [2000] analyzed the effects of out-of-vocabulary errors and the loss of punctuation on name finding in automatic speech recognition. Huang and Waibel [2002] proposed an adaptive method of named entity extraction for the meeting understanding. Chen [2003] dealt with spoken cross-language access to image collections. The coverage of a lexicon is one of the major issues in spoken transliteration name access. Recently, researchers are interested in using the Web, which provides a huge collection of up-to-date data, as a corpus. Keller and Lapata [2003] employed the Web to obtain frequencies for bigrams that are unseen in a given corpus.

Named entities are important objects in web documents. Building named entity relationship chains from the web is an important task. Matsuo *et al*. [2004] found social networks of trust from related web pages. Google sets[1] extracts named entity from web pages by inputting a few named entities. For some emerging applications like personal name disambiguation [Fleischman and Hovy 2004] [Mann and Yarowsky 2003], social chain finding [Bekkerman and McCallum 2005] [Culotta *et al*. 2004] [Raghavan *et al*. 2004], etc., glossary-based representations of named entities are not enough. For collecting transliteration names and building a bi-character language model, we propose a named entity (NE) ontology generation engine, called the X_{NE}-*Tree* engine. This engine produces relational named entities by given a seed. The engine uses Google to incrementally extract high co-occurrence named entities from related web pages and those named entities have similar relational properties with the seed. In each iterative step, the seed will be replaced by its siblings or descendants, which form new seeds. In this way, the X_{NE}-*Tree* engine will build a tree structure as follows with the original seed as a root.



[1] http://labs.google.com/sets

In this paper, we discuss using the Web as a live dictionary for recognizing spoken transliteration names and employ the fuzzy search capability of Google to retrieve relevant web page summaries. In section 2, we sketch the steps in our method. In section 3, we discuss using PAT trees to learn patterns from the Web dynamically and to correct recognition errors. Section 4 shows the experiments, which are the ASR model with/without the recovery mechanism. Section 5 presents the X_{NE}-*Tree* named entity ontology engine and our experimental results. In section 6, we make concluding remarks.

2. Spoken Transliteration Name Recognition System

The spoken transliteration name recognition system shown in Figure 1 accepts a speech signal denoting a foreign named entity and converts it into a character string. It is composed of the following four major stages. Stages (1) and (2) consist of the fundamental tasks in speech recognition. In the Stages (3) and (4), speech-to-text errors are corrected by using the Web.

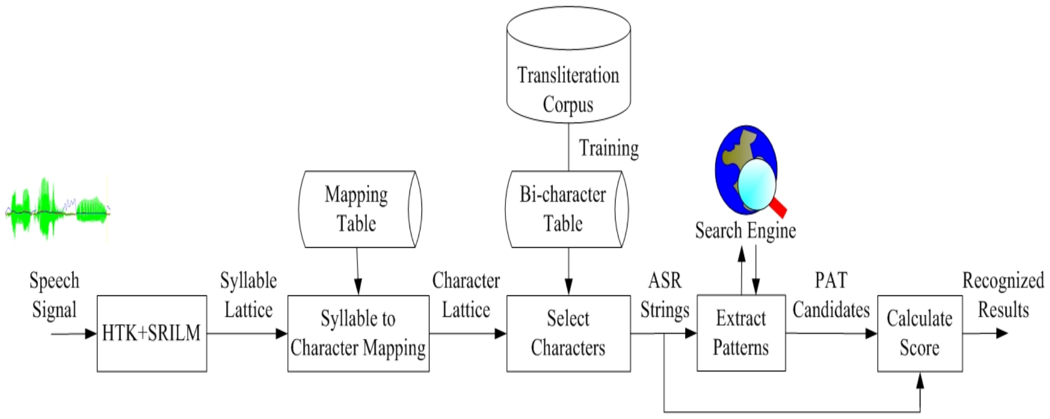


Figure 1. Stage in transliteration name recognition

(1) First, we employ the HTK[2] and SRILM[3] toolkits to build speech recognition models. For each speech signal, we use the model to get a syllable lattice.

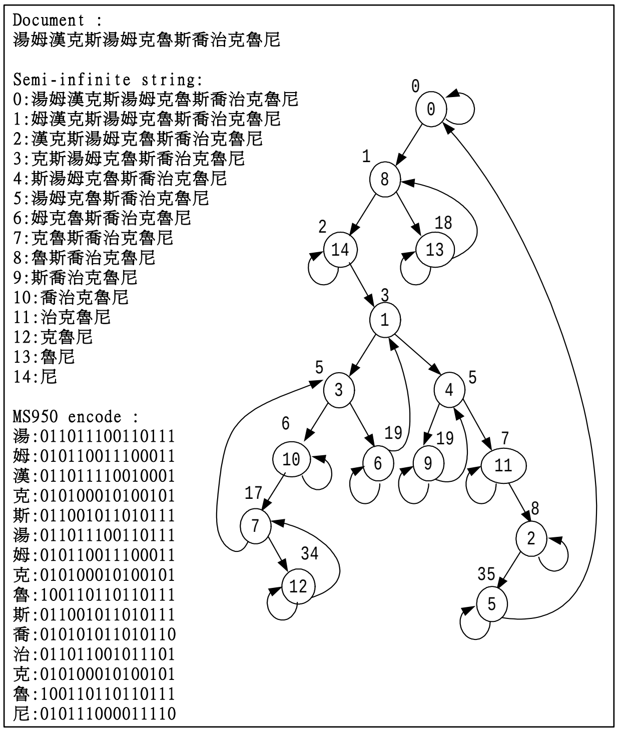
(2) Then, the syllable lattice is mapped into a character lattice by using a mapping table. The mapping table is a syllable-to-character mapping. Top-*N* character strings are selected from the character lattice by using Viterbe algorithm and a bi-character model which is trained from a transliteration name corpus. Such character strings will be called *ASR strings* in the following.

(3) Next, each ASR string is regarded as a query and is submitted to a web search engine like Google. From the top-*N* search result, we select higher frequency patterns from a PAT tree structure. The PAT tree [Chien 1997] [Gonnet *et al*. 1992], which was derived from the Patricia

2 http://htk.eng.cam.ac.uk/

3 http://www.speech.sri.com/projects/srilm/

tree, can be employed to extract word boundary and key phrases automatically. Because we employ the PAT tree to extract patterns, the patterns will be called *PAT candidates* in the following. A PAT tree example, “湯姆漢克斯湯姆克魯斯喬治克魯尼” in MS950 encoding, is shown in Figure 2. The circles represent semi-infinite string numbers. The number above each circle denotes the length, which indicates the first different bit of the character strings recorded in the sub-trees. In this example, the longest patterns are for “克魯” and “湯姆” on nodes (7, 12) and (0, 5), with lengths of 33 and 34 bits, respectively. The second longest patterns are for “克”, ”魯”, ”姆” and “斯” on nodes (3, 7, 12), (8, 13), (1, 6) and (4, 9), with lengths of 16, 17, 18 and 18 bits, respectively.



```
Document :
湯姆漢克斯湯姆克魯斯喬治克魯尼

Semi-infinite string:
0:湯姆漢克斯湯姆克魯斯喬治克魯尼
1:姆漢克斯湯姆克魯斯喬治克魯尼
2:漢克斯湯姆克魯斯喬治克魯尼
3:克斯湯姆克魯斯喬治克魯尼
4:斯湯姆克魯斯喬治克魯尼
5:湯姆克魯斯喬治克魯尼
6:姆克魯斯喬治克魯尼
7:克魯斯喬治克魯尼
8:魯斯喬治克魯尼
9:斯喬治克魯尼
10:喬治克魯尼
11:治克魯尼
12:克魯尼
13:魯尼
14:尼

MS950 encode :
湯:011011100110111
姆:010110011100011
漢:011011110010001
克:010100010100101
斯:011001011010111
湯:011011100110111
姆:010110011100011
克:010100010100101
魯:100110110110111
斯:011001011010111
喬:010101011010110
治:011011001011101
克:010100010100101
魯:100110110110111
尼:010111000011110
```

Figure 2. An example of extracting longest length pattern and its frequency

(4) Finally, the PAT candidates of all the ASR strings are merged together and ranked based on their number of occurrences and similarity scores. Candidates with the highest ranks are regarded as the recognition results for a spoken transliteration name.

Consider the example shown in Figure 3. The Chinese speech signal is a transliteration name, “湯姆克魯斯”, in Chinese, which denotes the name of the movie star “Tom Cruise.” The lattice shows different combinations of syllables. Each syllable corresponds to several Chinese characters. For example, “ke” is converted into “克”, “柯”, “科”, “可”, “喀”, “刻”, etc. The ASR strings “塔莫克魯斯”, “塔門克魯斯”, “塔莫柯魯斯”, etc. are selected from the character lattice. Through Google fuzzy search using the query “塔莫克魯斯”, some summaries of Chinese web pages are obtained and shown in Figure 4. Although the common transliteration of “Tom Cruise” in Chinese is “湯姆克魯斯”, which is different from the query “塔莫克魯斯”, fuzzy matching using Google can still identify relevant snippets containing the correct transliteration. We will call this operation “recognition error recovery using the Web” in the following.

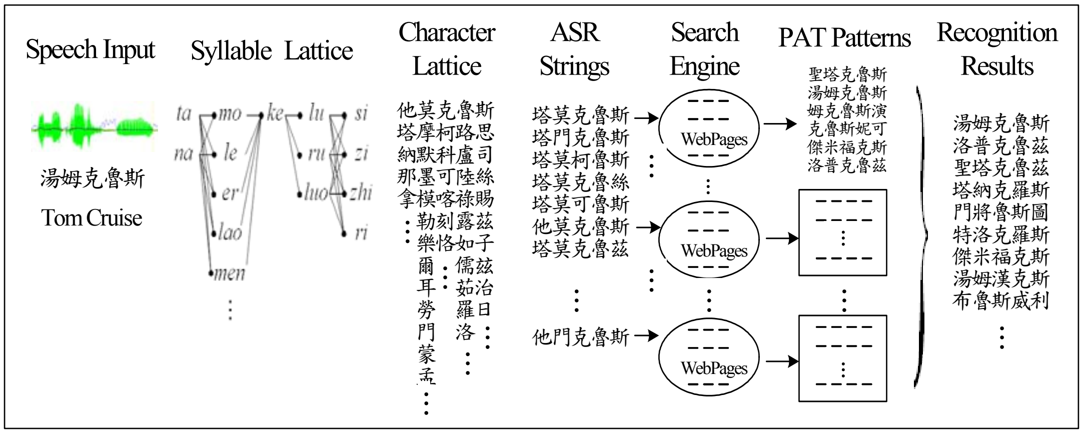


Figure 3. An example of recognizing a transliteration name: “湯姆克魯斯” (“Tom Cruise”)

(1) ... 至於贏家部份，則還是湯姆漢克斯、湯姆克魯斯、喬治克魯尼這些老面孔，...

(2) ... 第 76 分鐘，克魯斯換下梅開二度的維埃裏。

(3) ... 國際米蘭 (4-4-2)：豐塔納/科爾多瓦，布爾迪索，馬特拉齊，法瓦利/斯坦科維奇，貝隆，扎內蒂，埃姆雷/克魯斯，馬丁斯。

(4) ... 提起妮可即發火湯姆克魯斯“想殺死記者”.

(5) ... 電影節最具看點的明星當然非妮可基德曼與湯姆克魯斯有望在水城的戲劇性重逢莫屬。

Figure 4. Summaries obtained through fuzzy search for the query “塔莫克魯斯”

In the above examples, each partial matching part is enclosed in a rectangle symbol and the correct transliteration name is underlined. Summaries (1), (4) and (5) mention the movie star “湯姆克魯斯” (Tom Cruise) and summaries (2) and (3) mention a football star, “克魯斯” (Cruz). Figure 3 shows that PAT patterns like “聖塔克魯斯”, “湯姆克魯斯”, “姆克魯斯演”, etc. are proposed. After merging and ranking are performed, the possible recognition results are “湯姆克魯斯”, “洛普克魯茲”, “聖塔克魯茲”, etc.

3. Recognition Error Recovery Using the Web

The error recovery module tries to select higher frequency patterns from the Web search results and substitute the speech recognition results of Stages 1 and 2 (shown in Section 2) with the pattern. In this approach, Web search results obtained with an ASR string are placed in a PAT tree, and PAT candidates are selected from the tree. Two points are worth noting. A PAT candidate should occur many times in the PAT tree and should be similar to the ASR string.

The frequency, *Freq*, of a *PAT* candidate can be computed easily based on the PAT tree structure. The similarity between a *PAT* candidate and an *ASR* string is modeled by edit distance, which is the minimum number of insertions, deletions and substitutions needed to transform one character string (*ASR*) into another string (*PAT*). The smaller number is, the more similar they are. The similarity score, between an *ASR* string and a *PAT* string, is the frequency of the *PAT* string minus their number of edit operations. Finally, the score of a *PAT* string relative to an *ASR* string is defined as follows:

$$Score(ASR, PAT) = \alpha \times Freq(PAT) - \beta \times Dis\tan ce(ASR, PAT)\,. \tag{1}$$

It is computed through weighted merging of the frequency of the *PAT* string and by using the similarity between the *ASR* string and *PAT* string. This value determines if the *ASR* string will be replaced by the *PAT* string. In the above example, *Freq*(湯姆克魯斯)=43 and *Distance* (塔莫克魯斯, 湯姆克魯斯)=2.

4. Experimental Results

The speech input to the transliteration name recognition system is a Chinese utterance. We employed 51,111 transliteration names [Chen *et al*. 2003] to train the bi-character language model discussed in Section 2. In the experiments, the test data include 50 American state names, 29 names of movie stars from the 31st Annual People's Choice Awards (http://www.pcavote.com), and 21 names of NBA stars from the 2005 NBA All Star Team (http://www.nba.com/allstar2005/). The 50 American state names are not very active on the Web. In contrast, the 50 names of stars are very active. The test set is different from the training data set, so it is an open test. Because there may be more than one transliteration for a foreign named entity, the answer keys are manually prepared. For example, "Arizona" has four possible transliterations in Chinese: "亞利桑納", "亞歷桑納", "亞利桑那", and "亞歷桑那". On average, there are 1.9 Chinese transliterations for a foreign name in our test set. Appendix A lists the name test set and its answer keys. As explained in Section 2, the transliteration name recognition system is composed of four major stages. Stages 1 and 2 include the fundamental speech recognition tasks, and Stages 3 and 4 comprise the error recovery task. To examine the effects of these two parts, we will evaluate them separately in the following two subsections.

4.1 Performance in the Error Recovery Task

We assume that correct syllables have been identified in the speech recognition task. We simulate this assumption by transforming all the characters in the answer keys into syllables. Then, in Stage 2, we map the syllable lattice to obtain a character lattice. A total of 50 ASR strings are extracted from the character lattice in Stage 2 and submitted to Google. Finally, the best 10 PAT candidates are selected. We use the MRR (Mean Reciprocal Rank) [Voorhees 1999] and recall rate to evaluate the performance. The MRR represents the average rank of the correctly identified transliteration names among in the proposed candidates and it is defined as follow:

$$MRR = \frac{1}{M}\sum_{i=1}^{M} r_i \quad , \tag{2}$$

where M is the total number of test cases ; r_i equals $1/rank_i$ if $rank_i > 0$ and r_i is 0 if no answer is found. The $rank_i$ is the rank of the first correct answer for the i^{th} test case. That is, if the first correct answer is ranked 1, then the score is 1/1; if it is ranked 2, the score is 1/2, and so on. The MRR value is between 0 and 1. The inverse of the MRR denotes the average position of the correct answer in the proposed candidate list. The higher the MRR value is, the better the performance is. The recall rate is the number of correct references divided by M. It indicates how many transliteration names are correctly recognized.

Table 1. Performance of models wo/with error recovery

Models	Recall	MRR
ASR only	0.79	0.50
ASR + Web	0.90	0.88
ASR/Pre-removed + Web	0.59	0.48

Table 1 summarizes the experimental results obtained with models without/with the error recovery procedure. With the “ASR only” model, the top 10 ASR strings produced in Stage 2 are regarded as answers. This model does not employ the error recovery procedure. The recall rate is 0.79 and the MRR is 0.50. That is, 79 of 100 transliteration names are recognized correctly, and they appear in the first 2 (=1/0.50) position. In contrast, the “ASR + Web” model utilizes the error recovery procedure. PAT candidates extracted from the Web are selected in Stage 4. The recall rate is 0.90 and the MRR is 0.88. A total of 90 transliteration names are recognized correctly, and they appear in the first 1.13 (=1/0.88) position on average. In other words, when they are recognized correctly, they are always the top 1. Compared with the first model, the recall rate is increased 13.92%. As for the third model, i.e., the "ASR/Pre-removed + Web" model, we try to evaluate the error recovery ability. The correct transliteration names appearing in the set of ASR strings are removed. That is, all of the ASR strings contain at least one incorrect character. In such cases, the recall rate is 0.59 and the MRR is 0.48. This means that 59 transliteration names are recovered, and they appeared in the first 2.08 (=1/0.48) position on average. We further examine the number of errors produced by the "ASR/Pre-removed + Web" model to study the error tolerance when using the Web. Table 2 shows the lengths of the transliteration names (in the rows), and the number of matching characters (in the columns). For a transliteration name of length *l*, the number of matching characters is 0 to *l*. Each cell denotes how many strings belong to the specific category. For example, before error recovery, there are 6, 25, 90, 184, and 0 strings of length 4, which have 0, 1, 2, 3, and 4 characters matching the corresponding answer keys, respectively. After error recovery, there are 19, 52, 66, 62, and 106 strings of length 4, which have 0, 1, 2, 3, and 4 characters matching the answer keys, respectively. In other words, the recovery procedure corrects some wrong characters. The number of 1-character (2-character) errors decreased from 184 (90) to 62 (66), and total number of correct strings are increased from 0 to 106.

Table 2. Distribution before/after error recovery

	Before Error Recovery							After Error Recovery						
Length of NEs	Number of Matching Characters							Number of Matching Characters						
	0	1	2	3	4	5	6	0	1	2	3	4	5	6
2	11	23	0	-	-	-	-	13	21	0	-	-	-	-
3	6	29	76	0	-	-	-	6	39	64	2	-	-	-
4	6	25	90	184	0	-	-	19	52	66	62	106	-	-
5	9	10	12	77	193	0	-	11	23	36	41	53	137	-
6	0	0	1	8	20	39	0	0	3	19	12	7	5	22

Table 3. Effects of error positions and string lengths

Error Positions	Length=2	Length=3	Length=4	Length=5	Length=6	Total
Position 1	0	0	37	42	7	86
Position 2	0	2	35	42	4	83
Position 3	-	0	20	19	9	48
Position 4	-	-	17	24	3	44
Position 5	-	-	-	14	3	17
Position 6	-	-	-	-	1	1
Total	0	2	109	141	27	279

Table 3 shows the effects of the error position (in the rows) and the string length (in the columns). A total of 0, 2, 106, 137, and 22 utterances recover 1 character with length 2, 3, 4, 5, and 6, respectively. A total of 0, 0, 3, 4, and 5 utterances recover 2 characters with length 2, 3, 4, 5, and 6, respectively. No utterances can recover over 3 characters. The cell denotes how many strings can be recovered under the specific position and length. For example, a total of 37, 35, 20, and 17 errors for strings of length 4 appearing at positions 1, 2, 3, and 4, respectively, can be recovered by using the Web. In the experiments, 0% (=0/34), 1.80% (=2/111), 35.74% (=109/305), 46.84% (=141/301), and 39.71% (=27/68) of the strings of length 2, 3, 4, 5, and 6 can be recovered, respectively. The 34 is the number of the PAT candidates with length 2. Similarly, the 111, 305, 301, and 68 are the number of the PAT candidates with length 3, 4, 5, and 6. As for length, the longer strings facilitate better recovery than the shorter strings. Another results show that 30.82% (=86/279), 29.75% (=83/279), 17.20% (=48/279), 15.77% (=44/279), 6.09% (=17/279), and 0.36% (=1/279) of the strings with incorrect character appearing at positions 1, 2, 3, 4, 5 and 6 can be recovered, respectively. The 279 is the number of characters on which the 100 test data. Because the bi-character language model proceeds from the left side to the right side, the errors occurring at the beginning are easier to recover than those at the end.

4.2 Performance in the Speech Recognition Task

The set of 100 transliteration names discussed in Section 4.1 are spoken by 2 males and 1 female, so 300 transliteration names are recorded. We employ HTK and SRILM to get the best 100 syllable lattices (N-Best, N=100). The TCC-300 dataset for Mandarin is used to train the acoustic models. There are 417 HMM models, and each one has 39 feature vectors. The syllable accuracy is computed as follows: (*M-I-D-S*)/*M* * 100%, where *M* is the number of correct syllables; *I*, *D*, and *S* denote the number of insertion, deletion, and substitution errors, respectively. The syllable accuracy is 76.57%. To estimate the character recovery ability, we consider the correct character number. The accuracy of the ASR only and ASR+Web models on the character level are computed as follows, respectively:

$$\sum_{i=1}^{M} \max_{j=1toK} \left(\frac{Word_{Length}(TestName_i) - Distance(AnsKey_{ij}, ASR_i)}{Word_{Length}(TestName_i)}\right) \tag{3}$$

and

$$\sum_{i=1}^{M} \max_{j=1toK} \left(\frac{Word_{Length}(TestName_i) - Distance(AnsKey_{ij}, PAT_i)}{Word_{Length}(TestName_i)}\right), \tag{4}$$

where *M* is the total test number and *K* is the answer key number for test name *i*. A total of 50

ASR strings are extracted from the character lattice, and the best 50 PAT candidates are selected. Table 4 shows the character level results. The "ASR+Web" model achieves 21.54% better performance than the "ASR Only" model on average. Table 5 shows the word level results. The "ASR+Web" model using error recovery procedure improves the recall rate and the MRR of the "ASR Only" model from 0.20 and 0.07 to 0.42 and 0.31, respectively. In other words, the average ranks of the correct transliteration names move from the 14th position (=1/0.07) to the 3rd position (=1/0.31) after error recovery.

Table 4. Performance on the character level

ASR Only (Character Level Accuracy)					ASR + Web (Character Level Accuracy)				
Top 1	Top 2	Top 3	Top 4	Top 5	Top 1	Top 2	Top 3	Top 4	Top 5
38.01%	43.34%	47.30%	49.07%	50.93%	48.18%	54.01%	55.93%	58.03%	59.48%

Table 5. Performance on the word level

ASR Only (Word Level)		ASR + Web (Word Level)	
Recall	MRR	Recall	MRR
0.20	0.07	0.42	0.31

Web fuzzy search produces useful patterns for error recovery. Our fault tolerance experiments show that longer transliteration names have stronger tolerance than shorter transliteration names and that the incorrect characters appearing at the beginning of a transliteration name are relatively easier to correct than those appearing at the end. Thus, the improvement in the character level accuracy is helpful for the recovery mechanism, and vice versa.

5. Re-training the Bi-Character Language Model

For collecting transliteration names to build a bi-character language model, we propose using a named entity (NE) ontology generation mechanism, called the X_{NE}-*Tree* engine. Given a seed, the engine incrementally extracts relational named entities with the seed from related web pages and the output is a tree structure. Each node in the structure is a named entity (NE).

5.1 A Named Entity Ontology Generation Engine

Recognizing a named entity and calculating the relational property score with a seed are two crucial tasks. Firstly, we submit the given seed to a search engine and select the top *N* returned snippets. Then, we use the suffix tree to extract possible patterns automatically. The patterns, which are extracted based on the global statistic, may be impacted by the frequency variance of patterns with the same substrings [Yang and Li 2002]. Because our aim is to generate named entities, most of the max-duplicated strings can be filtered out by using a named entity recognition (NER) system [Chen *et al.* 1998]. The NER system will re-segment a candidate pattern to obtain some substrings and give each substring a part of speech (POS) and a possible name tag. If any substring is tagged as a location, an organization, or a person by using an NER-POS server [Chen *et al.* 1998], the candidate pattern is considered to be a named entity. Because prepositions frequently occur before/after a named entity, the suffix tree approach may introduce an incorrect boundary. Thus, we filter out substrings that have a preposition tag.

Secondly, we calculate a relational property score, called the *Co-Occurrence Double-Check score (CODC*, for each extracted name entity (denoted Y_i) with a seed (denoted X). We postulate that X and Y_i have a strong relationship if we can find Y_i from X (a forward process) and find X from Y_i (a backward process). The forward and backward processes form a double check operation. $CODC(X, Y)$ is defined as follows:

$$CODC(X,Y) = e^{\log\left(\frac{f(Y@X)}{f(X)}\times\frac{f(X@Y)}{f(Y)}\right)^{\alpha}}, \qquad (5)$$

where $f(X@Y_i)$ is the total number of occurrences of X in the top N snippets when query Y_i is submitted to the search engine. Similarly, $f(Y_i@X)$ is the total number of occurrences of Y_i in the top N snippets for query X; $f(X)$ is the total number of occurrences of X in the top N snippets for query X, and $f(Y)$ is the total number of occurrences of Y in the top N snippets of query Y. In each iterative step, Y_i is added to a queue when the $CODC(X,Y_i)$ value is larger than a threshold θ. Then, we get a new seed X from the queue. The $CODC$ measure is best when α=0.15. The overall process is shown in Figure 5.

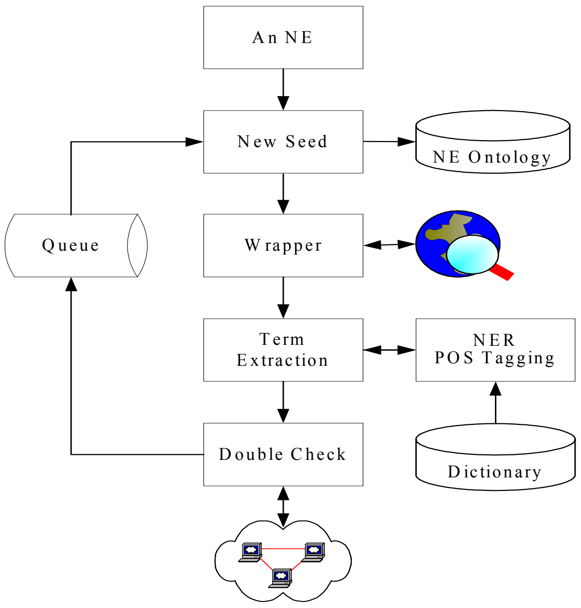


Figure 5. Named entity ontology generation process

5.2 Constructing a Named Entity Ontology

When building a bi-character language model, we choose 100 seeds, which are the same 100 utterances described in Section 4. Here, we set a condition to control generation of the ontology. Each initial seed can derive at most four layers, and no more than 15 children are allowed in the first layer. The maximum number of children of a named entity in layer i is bounded by the number in layer (i-1) multiplying a decreasing rate. In the experiments, we set the decreasing rate to be 0.7, so that at most 15, 15×0.7, 15×0.7^2, and 15×0.7^3 children can be expanded by a named entity in layers 0-3, respectively. We set the threshold θ at 0.1. Those named entities with *CODC* scores larger than the predefined threshold are sorted, and a sufficient number of named entities are selected in a sequence for expansion. In this way, a total of 7,642 nodes are generated by the 100 seeds. We employ Touch-Graph (http://www.touchgraph.com) to represent the named entity ontology. Figure 6 shows an example by using “湯姆克魯斯” as a seed, which is a Mandarin transliteration name of an actor “Tom Cruise”, to build an ontology. To evaluate the performance, we consider the following four types.

(1) Named Entity (NE) type: In this case, the proposed candidate should be a named entity and should not have incorrect boundary. A personal name with a title or a first name with more than 4 characters is regarded as being correct. In contrast, patterns with a last name only are considered incorrect.

(2) Relational property of NE (RNE) type: For those acceptable strings in (1), which have the same relational property with the initial seed or its parents are considered to be correct. The remaining nodes are incorrect.

(3) Partial Named Entity (PNE) type: We relax the restriction on boundary errors specified in (1). Patterns consisting of partial named entities are regarded as being correct. The remaining nodes are incorrect.

(4) Relational property of PNE (RPNE) type: For those acceptable strings in (3), which have the same relational property with the initial seed or its parents are considered to be correct. The remaining nodes are incorrect.

Table 6 shows the performance in ontology generation. Of those 7,642 nodes, the error rates for the NE type, the RNE type, the PNE type, and the PRNE type are 19.60%, 34.20%, 12.62%, and 29.82%, respectively.

Table 6. Performance in ontology generation

Size of Seed	Size of Ontology	NE	RNE	PNE	RPNE
100	7,642	19.60%	34.20%	12.62%	29.82%

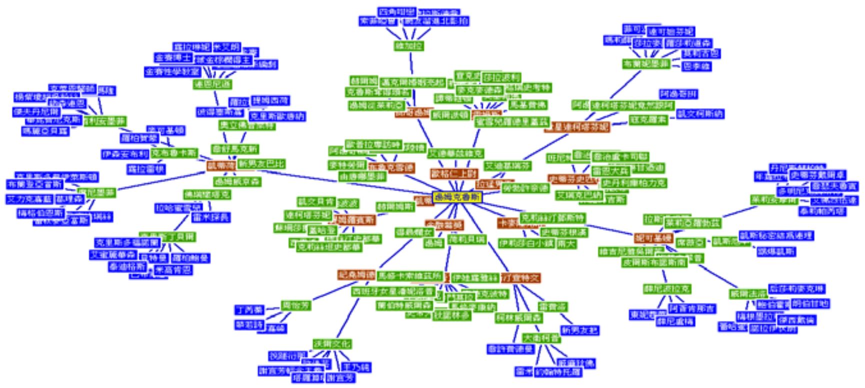


Figure 6. A snapshot of the named entity ontology of "湯姆克魯斯" ("Tom Cruise")

5.3 Combining the Bi-Character Language Model with the NE Ontology

In the previous experiments, we employed 51,111 transliteration names (BaselineTN) to build the bi-character language model. However, these transliteration names might not be active on the Web. We submitted these transliteration names to a search engine (i.e., Google). For a transliteration name, if the search engine does not return any web pages, we filter it out. Finally, we filter out 14,933 named entities and get 36,178 transliteration names (FilterTN) with this method. Refer to Table 6. Of the 7,642 named entities (Total-Ontology) reported by X_{ne}-engine, 6,146 named entities (NE-Ontology) are of the correct NE type, and 5,023 named entities (RNE-Ontology) are of the correct RNE type.

In the experiments, we consider word level accuracy only. Two basic transliteration name corpora, i.e., BaselineTN and FilterTN, are employed to build bi-character language models. In ideal case, correct syllables have been identified in the ASR (ASR_Perfect). Table 7 shows that FilterTN is a little better than BaselineTN. We further combine FilterTN with the NE ontology derived by the X_{NE}-*Tree* engine to perform evaluation. In this way, we employ the FilterTN+RNE-Ontology, FilterTN+NE-Ontology, and FilterTN+Total-Ontology to build bi-character language models. Table 7 summarizes the experimental results obtained with the language model with the NE ontology. The three models with the NE ontology outperform the baseline model. In particular, the NE ontology improve the recall rate and the MRR from 0.79 and 0.50 (BaselineTN) to 0.84 and 0.55 (FilterTN+RNE-Ontology), respectively. Table 8 lists the results obtained using both the NE ontology and error recovery procedure. The NE ontology is still helpful, in particular for the recall rate. In the best case, it improves the recall rate from 0.90 (BaselineTN) to 0.94 (FilterTN+RNE-Ontology). In summary, the model using NE ontology resources, the recall rate is improved 13.92%. On comparing the "FilterTN+RNE-Ontology" model with the ASR model without the error recovery procedure and NE ontology resources, the recall rate is improved 18.98%.

Table 7. Bi-character language models with the NE ontology but without error recovery.

Language Model	Size of TN	ASR_Perfect Only (Word Level)	
		Recall	MRR
BaselineTN	51,111	0.79	0.50
FilterTN	36,178	0.80	0.50
FilterTN + RNE-Ontology	41,201	0.84	0.55
FilterTN + NE-Ontology	42,324	0.83	0.57
FilterTN + Total-Ontology	43,820	0.82	0.57

Table 8. Bi-character language models with both the NE ontology and error recovery procedure

Language Model	ASR_Perfect + Web (Word Level)	
	Recall	MRR
BaselineTN	0.90	0.88
FilterTN	0.90	0.87
FilterTN + RNE-Ontology	0.94	0.88
FilterTN + NE-Ontology	0.93	0.88
FilterTN + Total-Ontology	0.93	0.90

Table 9. Combining the bi-character language model with the NE ontology without/with the error recovery procedure in ASR systems

Language Model	ASR Only (Word Level)		ASR+Web (Word Level)	
	Recall	MRR	Recall	MRR
BaselineTN	0.20	0.07	0.42	0.31
FilterTN	0.20	0.06	0.41	0.32
FilterTN + RNE-Ontology	0.23	0.11	0.48	0.38
FilterTN + NE-Ontology	0.24	0.11	0.48	0.37
FilterTN + Total-Ontology	0.24	0.12	0.47	0.39

Table 9 summarizes the experimental results obtained with language models that use the NE ontology without/with error recovery procedure in the complete transliteration name ASR system. The system without the error recovery procedure (ASR Only), the NE ontology still improves the performance. Comparing the "FilterTN+RNE-Ontology" with BaselineTN, the recall rate is increased 15%. When the ASR system incorporates the error recovery procedure (ASR+Web), the recall rate is increased 14.28% (FilterTN+RNE-Ontology vs. BaselineTN).

6. Conclusions

In this study, we employ the Web as a giant corpus to correct transliteration name recognition errors. Web fuzzy search produces useful patterns for error recovery. In the ideal case, we input the correct syllable sequences, convert them into text strings, and test the recovery capability by using the Web corpus. On comparing with the model without the web recovery procedure, the recall rate is improved 13.92%. For collecting transliteration names beforehand, we propose using a named entity (NE) ontology generation engine, called the X_{NE}-*Tree* engine. The engine automatically creates named entity ontology for a given seed. In the experiments, a total of 7,642 named entities in the ontology were initiated by 100 seeds. After the language model for speech recognition combined the named entity ontology, the recall rate is improved 18.98%. With a complete transliteration name ASR system, the error recovery experiments show that the recall rate is increased from 0.20 to 0.42 and the MRR from 0.07 to 0.31. When the RNE-Ontology is incorporated, the recall rate and the MRR is increased 0.48 and 0.38, respectively. Thus, we conclude that the error recovery procedure and NE ontology can helpful to the ASR model.

Acknowledgements

This research was partially supported by the National Science Council, Taiwan, under contracts NSC94-2752-E-001-001-PAE and NSC95-2752-E-001-001-PAE.

References

Appelt, D. E., and D. Martin, "Named Entity Extraction from Speech: Approach and Results Using the TextPro System," In *Proceedings of DARPA Broadcast News Workshop*, 1999, pp. 51-54.

Bekkerman, R., and A. McCallum, "Disambiguating Web Appearances of People in a Social Network," In *Proceedings of WWW*, 2005, pp. 463-470.

Chen, H. H., "Spoken Cross-Language Access to Image Collection via Captions," In *Proceedings of 8th Eurospeech*, 2003, pp. 2749-2752.

Chen, H. H., C. H. Yang, and Y. Lin, "Learning Formulation and Transformation Rules for Multilingual Named Entities," In *Proceedings of the Association for Computational Linguistics on Multilingual and Mixed-language Named Entity Recognition*, 2003, pp. 1-8.

Chen, H. H., Y. W. Ding, and S. C. Tsai, "Named Entity Extraction for Information Retrieval," *Computer Processing of Oriental Languages*, *Special Issue on Information Retrieval on Oriental Languages*, 1998, pp. 75-85

Chien, L. F., "PAT-Tree-Based Keyword Extraction for Chinese Information Retrieval," In *Proceedings of 20th ACM SIGIR Conference*, 1997, pp. 50-58.

Culotta, A., R. Bekkerman, and A. McCallum, "Extracting Social Networks and Contact Information from email and the Web," In *Proceedings of the First Conference on Email and Anti-Spam*, 2004.

Fleischman, M. B., and E. Hovy, "Multi-document Person Name Resolution," In *Proceedings of the Association for Computational Linguistics (ACL), Reference Resolution Workshop*, 2004. (presentation order: second)

Gonnet, G. H., R. A. Baeza-Yates, and T. Snider, "New Indices for Text: PAT Trees and PAT Arrays," *In Information Retrieval Data Structures Algorithms*, 1992, pp. 66-82.

Huang, F., and A. Waibel, "An Adaptive Approach of Name Entity Extraction for Meeting Application," In *Proceedings Of Human Language Technology Conference*, 2002.

Keller, F., and M. Lapata, "Using the Web to Obtain Frequencies for Unseen Bigrams," *Computational Linguistics*, 2003, pp. 459-484.

Lin, W. C., M. S. Lin, and H. H. Chen, "Cross-Language Image Retrieval via Spoken Query," In *Proceedings of RIAO*, 2004, pp. 524-536.

Mann, G. S., and D. Yarowsky, "Unsupervised Personal Name Disambiguation," In *Proceedings of Conference on Computational Natural Language Learning*, 2003.

Matsuo, Y., H. Tomobe, K. Hasida, and M. Ishizuka, "Finding Social Network for Trust Calculation," In *Proceedings of 16th European Conference on Artificial Intelligence*, 2004, pp. 510-514.

Miller, D., S. Boisen, R. Schwartz, R. Stone, and R. Weischedel, "Named Entity Extraction from Noisy Input: Speech and OCR," In *Proceedings of 6th Applied Natural Language Processing Conference*, 2000, pp. 316-324.

MUC Message Understanding Competition, http://www.itl.nist.gov/iaui/894.02/related_projects/muc/index.html, 1998.

Raghavan, H., J. Allan, and A. McCallum, "An Exploration of Entity Models, Collective Classification and Relation Description," In *Proceedings of LinkKDD*, 2004.

Thompson, P., and C. Dozier, "Name Searching and Information Retrieval," In *Proceedings of Second Conference on Empirical Methods in Natural Language Processing*, 1997, pp. 134--140.

Yang, W., and X. Li, "Chinese Keyword Extraction Based on Max-Duplicated Strings of the Documents," In *Proceedings of 25th ACM SIGIR Conference*, 2002, pp. 439-440.

Voorhees, E., "The TREC-8 Question Answering Track Evaluation," In *Proceedings of the 8th TREC*, 1999, pp. 23-37.

Appendix A

Transliteration name	Answer keys list	Transliteration name	Answer keys list
科羅拉多	克羅拉多 柯羅拉多 科羅拉多	喬治克隆尼	喬治克隆尼 喬治克龍尼 喬治柯隆尼
加利福尼亞	加里福尼亞 加利福尼亞 加利弗尼亞	丹佐華盛頓	丹佐華聖頓 丹佐華盛頓
喬治亞	喬治亞 喬治雅	湯姆克魯斯	湯姆克魯斯
密西根	密希根 密西根	強尼戴普	強尼戴普
阿拉斯加	阿拉斯加	湯姆漢克斯	湯姆漢克斯
北卡羅萊納	北卡羅萊納 北卡羅萊那 北卡羅來納 北卡羅來那	*芮妮齊薇格	芮妮齊薇格 芮尼齊維格
康乃狄克	康乃迪克 康乃狄克 康迺迪克	莎莉賽隆	莎莉賽隆 莎莉塞隆 沙莉賽隆
德拉瓦	德拉瓦	妮可基嫚	妮可基嫚 尼可基曼 妮可基曼
佛羅里達	佛羅里達 佛羅理達	茱莉安摩爾	朱利安摩爾 茱莉安摩爾 朱莉安摩爾
南卡羅萊納	南卡羅萊納 南卡羅萊那 南卡羅來納 南卡羅來那	茱莉亞羅勃茲	茱莉亞羅勃茲 朱利亞羅伯茲 朱利亞羅勃茲 朱莉亞羅伯茲 茱莉亞羅伯茲
*夏威夷	夏威夷 夏威宜	威爾史密斯	威爾史密斯
愛荷華	艾荷華 愛何華 愛荷華	維果莫天森	維果莫天森 維果摩天森 維果墨天森
愛達荷	艾達荷 愛達荷	麥特戴蒙	麥特戴蒙
伊利諾	伊利諾 伊立諾 依利諾	休傑克曼	休傑克曼 休杰克曼
*印地安那	印地安那 印地安納 印弟安納	托貝馬奎爾	托貝馬奎爾
堪薩斯	坎薩斯 堪薩斯	鄔瑪舒嫚	烏瑪舒曼 鄔瑪舒曼
肯塔基	肯塔基	琪拉奈特莉	琪拉奈特莉 奇拉奈特莉
路易斯安那	路易斯安納	凱特貝琴薩	凱特貝琴薩
麻薩諸塞	麻薩諸塞 馬薩諸塞 麻塞諸塞	荷莉貝瑞	荷莉貝瑞 荷利貝瑞
緬因	緬因	安潔莉娜裘莉	安潔莉娜裘莉
馬里蘭	馬里蘭 馬利蘭	柴克巴爾夫	柴克巴爾夫
亞利桑那	亞利桑納 亞歷桑納 亞利桑那 亞歷桑那	布萊德彼特	布萊德比特 布萊德彼特 布萊得比特 布萊得彼特
明尼蘇達	明尼蘇達 明尼蘇答	金凱瑞	金凱瑞
密蘇里	米蘇里 密蘇里	柯林法洛	柯林法洛 科林法洛
密西西比	密西西比	裘德洛	裘德羅 裘德洛
蒙大拿	蒙大納 蒙大那 蒙大拿	娜塔莉波曼	娜塔利波曼 娜塔莉波曼
內布拉斯加	內布拉斯加	凱特溫絲蕾	凱特溫斯雷 凱特溫斯蕾 凱特溫絲雷 凱特溫絲蕾
阿拉巴馬	阿拉巴馬	*珍妮佛嘉納	珍妮佛嘉納
北達科他	北達科他 北達科塔	*茱兒芭莉摩	茱兒芭莉摩
新罕布夏	新漢布夏 新罕布夏	姚明	姚明
紐澤西	紐澤西	俠客歐尼爾	俠克歐尼爾 俠客歐尼爾
*新墨西哥	新墨西哥	凱文賈奈特	凱文加奈特 凱文賈奈特
內華達	內華達	*崔西麥格瑞迪	崔西麥格瑞迪 崔西麥葛瑞迪
紐約	紐約	柯比布萊恩	柯比布萊恩 科比布萊恩
俄亥俄	俄亥俄	文斯卡特	文斯卡特 文思卡特
奧克拉荷馬	奧克拉荷馬 奧克拉荷瑪 奧克拉河馬	提姆鄧肯	提姆鄧肯
奧勒	奧勒岡	葛蘭特希爾	格蘭特希爾 葛蘭特希爾
賓汐法尼亞	賓希法尼亞 賓西法尼亞 賓夕凡	勒布朗詹姆斯	勒布朗詹姆斯 勒布郎詹姆斯

	尼亞		
*羅德島	羅德島	艾倫艾弗森	艾倫艾弗森 艾倫埃弗森 艾倫艾佛森
阿肯色	阿肯色 阿肯瑟 阿肯塞	*小歐尼爾	小歐尼爾
南達科他	南達科塔 南達科他 南達柯塔	拉希德華萊士	拉希德華萊士 拉西德華萊士
田納西	田納西 田那西	普林斯	普林斯
德克薩斯	德克薩斯 得克薩斯	賈米森	賈米森
*猶他	猶他	比盧普斯	比魯普斯 比盧普斯
佛蒙特	佛蒙特	斯托賈科維奇	斯托賈克維奇 斯托賈科維奇 斯托賈可維奇
維吉尼亞	維基尼亞 維吉尼亞 維吉尼雅	德克諾維茨基	德克諾維茨基 德克諾威茨基 德克諾維斯基
華盛頓	華聖頓 華盛頓 華勝頓	班華萊士	班華萊士 班華勒斯
西維吉尼亞	西維基尼亞 西維吉尼亞 西維吉尼雅	卡梅隆安東尼	卡梅隆安東尼 卡麥隆安東尼
威斯康辛	威斯康辛 威斯康新	斯塔德邁爾	斯塔德麥爾 斯塔德邁爾 斯塔達邁爾
懷俄明	懷俄明	基里連科	基里連科

* “印、島、兒、猶、小、芮” characters are not in training set and “尼、妮”, ” 辛、新” and “奇、琪” differentia of frequency is too high.