

A Three-Phase System for Chinese Named Entity Recognition

Conrad Chen

Hsi-Jian Lee

Department of Computer Science and
Information Engineering, National
Chiao Tung University, Hsinchu

drchen@csie.nctu.edu.tw

Department of Medical Informatics,
Tzu Chi University, Hualien

hjlee@mail.tcu.edu.tw

Abstract. The handling of out-of-vocabulary (OOV) words is one of the key points to a high performance lexical analysis in natural language processing. Among all OOV words, named entities (NE) are the most productive ones. They generally constitute the most meaningful parts of sentences (persons, affairs, time, places, and objects). In this paper, we propose a three-phase “generation, filtering, and recovery” system to address the NER problem. A set of stochastic models is first used to generate all possible NE candidates. Then we treat candidate filtering as an ambiguity resolution problem. To resolve ambiguities, we adopt a maximal-matching-rule-driven lexical analyzer. Last, a pattern matching method is applied to detect and recover abnormalities in the results of the previous two phases.

Pure lexical information is exploited in our system. We get a high recall of 96% with personal names (PER), satisfiable recall of 88%, 89%, and 80% with transliteration names (TRA), location names (LOC), and organization names (ORG), respectively. The overall precision and excluding rate is over 90% and 99%.

1. Introduction

Words are generally the basic unit to process natural languages. However, in Chinese, sentences are composed of string of characters without any delimiters to mark word boundaries. To process Chinese, sentences must be segmented into word sequences first. Most Chinese language processing systems rely on lexicons to recognize words in sentences. Because the number of Chinese words is tremendous, it is impossible to compile all words in a lexicon. Therefore, word segmentation processes often encounters the problem of out-of-vocabulary (OOV) words.

Among all OOV words, named entities are one of the most important sorts. It is impossible to list them exhaustively in a lexicon. They are the most productive type of words. Nearly no simple or unified generation rules for them exist. Besides, they are usually keywords in documents. Named entity recognition (NER) thus becomes a major task to many natural language applications, such as natural language understanding, question answering, and information retrieval.

Many researches have addressed the NE recognition problem in Chinese since 1990. Most of them focused on some specific types as *personal names* [5][13], *location names* [9], *organization names* [10], and *transliteration names* [11]. There are also type-independent approaches of NER. However, most of these approaches need type-dependent data such as role tags. Type-independent approaches can be roughly divided into two major sorts: over-generating & disambiguating [3][12] and over-segmenting & generating [4][8].

Generally speaking, there are two main approaches of the above studies, *rule-based* models and *machine learning* methods. Rule-based approaches could effectively exploit human knowledge and can be tuned conveniently. On the other hand, machine learning approaches, such as *maximum entropy* or *support vector machine*, is more independent from languages and simple to implement. Rule-based approaches is slightly outperform machine learning ones in MUC-7 tests [2].

In our consideration, rule-based approaches are more reasonable than machine learning ones. Boosting performances of rule-based approaches is easier than improving machine learning abilities. Therefore,

rule-based approaches is adopted in this paper, while machine learning methods still could be incorporate in our system under the present framework in future.

A three-phase “*generation, filtering, and recovery*” system is proposed to solve NER problem. In the generation phase, stochastic models are responsible for generating all possible candidates of different kinds of named entities in input documents. In the filtering phase, we treat the filtering of false candidates as an ambiguity resolution problem. A maximal-matching-rule-driven lexical analysis is performed to resolve ambiguities caused by false candidates. In the recovery phase, a rule-driven pattern matching method is applied to detect and recover abnormalities in the results of the previous two phases.

2. System Overview

In our system, we try to make use of both the tunability of stochastic models in candidate extraction and the power of lexical analyzers in disambiguation. To implement this idea, we propose a three-phase framework: candidate generation, filtering, and recovery, as shown in Figure 2.1:

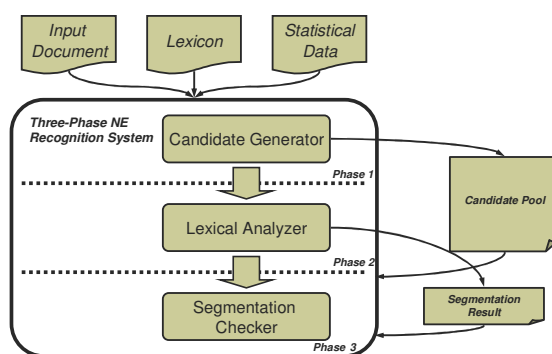


Fig. 2.1. An overview of our system

In the first phase, all possible candidates of various kinds of named entities in the input document are extracted. Notice that this process is inevitably both over-generating and under-generating. Because of the filtering process, the candidate extracting can be tuned to have a higher recall and to sacrifice precision a little for a moment.

Statistical approaches are adopted in the candidate generation phase. The reason is that names are given by people. Therefore, there is no exact answer if a string is a name or not. The only thing can be judged is how likely the string is to be a name. As for computers, to estimate the likelihood of names is basically a fuzzy problem. If a character is more likely to appear in a name, it has a better fuzzy value. The detail of how fuzzy logic and statistic estimation are applied will be discussed later.

The second phase of the system is *false candidate filtering*. How do we verify which candidates are true named entities and which ones are false? False candidates are either a common word or composed of fragments of common words and named entities. The first case has less impact on subsequent applications. The second case usually results ambiguous segmentations. Verification of these candidates could be viewed as an ambiguity resolution problem. If we can judge which segmentation is correct or more proper, we could also verify which candidates are true named entities.

Because of the regularity of lexical choices in modern Chinese, many simple approaches of segmentation ambiguity resolution have good performances. No matter what simple methods it takes, heuristic rules or stochastic estimations, if there are no OOV words, most lexical analysis methods show great precision in ambiguity resolution. That is to say, if we got a high recall in the extraction of NE candidates, most of the segmentation ambiguities caused by false candidates are supposed to be resolved by conventional word segmentation methods. We choose a heuristic approach, which is mainly driven by maximal matching rules, to resolve segmentation ambiguities.

The third phase of the system is *recovery*. The recovery mechanism is used to revive some obviously incorrect results of the first two phases. There are two major target types to be recovered: over-segmentations caused by under-generation and under-segmentations caused by over-generation.

Through the detection of these anomalies, e.g. a succession of single-character words indicating over-segmentations, part of un-extracted named entities could be revived.

3. Candidate Generation

The candidate generator is used to extract all possible named entity candidates in input documents. There are four layers in the candidate generator to handle four sorts of NEs: close-ended NEs, genuine names, whole named entities, and abbreviations.

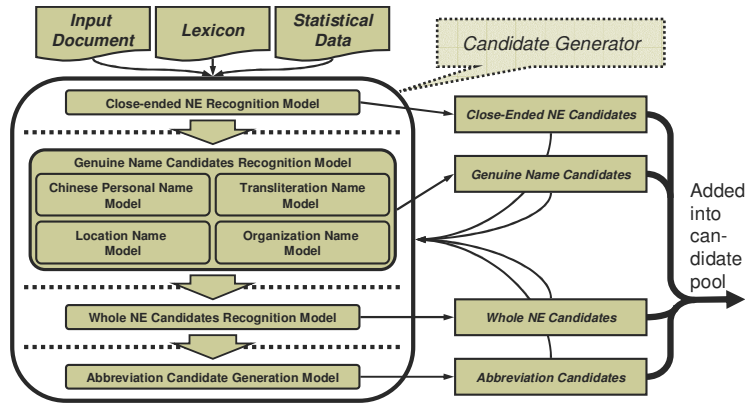


Fig. 3.1. The overview of the candidate generator

Close-ended named entities comprise time and quantity expressions. Since the extraction of close-ended NEs is not the focus of this paper, and previous researches [6] have solved this problem well, a single simplified rule is applied to recognize most of them in our system. The rule is as follows:

$$[“第”] + (Numerals)^+ + [Qualifier] + [Unit]$$

This simple rule cannot cover all close-ended NEs, of course. The purpose of this rule is just to prevent unrecognized close-ended NEs affect the performance of the recognition of open-ended ones.

In general, the structure of whole open-ended NEs except for abbreviations can be represented as:

$$[prefixes] + \text{genuine name} + [suffixes]$$

For example, “台北” is a *genuine name* and “台北市” is a *whole named entity* with *suffix* “市” indicating that “台北” is a city. The handling of prefixes is much similar to that of suffixes, and on the other hand prefixes are much more rarely seen than suffixes. Therefore, for simple implementation, whole NEs with prefixes would not be recognized in our system.

Suffixes generally indicate the type of named entities. There are many types of named entities with different suffixes. Many sorts of them rarely appear in the document. It is not worth to build models for each type of these names. However, suffixes are strong features. It is easier to recognize them, and chances of error recognition are comparatively low. Therefore, a compromised method is adopted that only models for four kinds of genuine names are implemented at present in our system. They are *personal names*, *transliteration names*, *location names*, and *organization names*. These four kinds of genuine name candidates would be used to form various types of NEs with corresponding suffixes. For instance, if a personal name candidate is followed by a publication suffix, they will be recognized as a whole publication name, like:

$$“余光中”(personal\ name) + “詩選”(publication\ suffix) \rightarrow “余光中詩選”(publication\ name)$$

For the same reason above, all NE suffixes are roughly classified into three categories: ones with similar corresponding genuine name types to location suffixes, ones with similar corresponding genuine name types to organization suffixes, and others. The first category covers all location names, racial names, etc. The second one comprises all organization names except for racial names, facility names, publication names, etc. The third one includes feat names, culture names, and so on. Among these three categories, only the first two are addressed by our system. These two categories are called “*location-like NE*” and “*organization-like NE*”. Names belonging to the same category will be

addressed by the same corresponding model. There are two main advantages following this way. First, times spent on designing models and collecting data are saved. Second, confidences brought by suffixes could alleviate the deviation on statistics brought by a compromised approach. The extraction of genuine names and whole named entities will be detailed later.

Open-ended named entities extracted above are used to find possible abbreviations and some rule-recognizable aliases in the abbreviation generation model. Four simple rules are adopted to complete this job:

Rule 1: Take the first characters of genuine name and all suffixes other than typing suffix, and the last character of typing suffix from NE candidates (e.g. “中央研究院” → “中研院”)

Rule 2: Surnames of personal name candidates (e.g. “呂秀蓮” → “呂”)

Rule 3: Given names of personal names (e.g. “陳信安” → “信安”)

Rule 4: *Modifier + Surname* or *any character of Given names* (e.g. “陳水扁” → “小陳”, “阿水”, “阿扁”, etc.)

Notice that only abbreviations and aliases with original names appearing in the document could be addressed by our system.

3.1. Statistic Estimation

The recognition of *genuine names* is basically a fuzzy decision problem to computers. There is no exact right or wrong answer for a string to be a name. The only problem is how likely it is. Fuzzy values represent strings' likelihood or properness to be a name. Since Chinese is a character-based language, methods of estimating fuzzy values are generally also character-based. Names are composed of several characters. There are several ways to transform the member characters' fuzzy value to the string's fuzzy value.

Stochastic language models are usually adopted to estimate the likelihood of a candidate to be a named entity. The fundamental principle is that the string with a higher probability or frequency to be a name has a higher fuzzy value or likelihood. There are several ways to estimate the fuzzy value of a string from the statistic data based on characters. These models include Markov models, bi-gram models, unigram models, etc.

Each model has its advantages and disadvantages. Generally speaking, more complex the model is, more precisely it estimate, and more training data it needs. Besides that, the data-sparseness problem is more likely to happen. Since the amounts of features of different types of named entities are varied, each type has its own best-fit model. In this paper, to simplify data collecting and training, unigram models are adopted. Additionally, some supplementary information such as positional feature is exploited to support statistical models.

Generally there are two major ways to estimate fuzzy values of a single character:

Frequency: $freq(typ|c)=counts(typ, c)$

Probability: $prob(typ|c)=counts(typ, c)/counts(c)=freq(typ|c)/counts(c)$

Frequencies stand for differences among naming-characters. They represent popularities of characters to be used in names of some type. If some character is used in more names, it has a higher frequency. If frequencies are used as fuzzy values, a higher recall will be obtained with common names.

Probabilities stand for differences among all characters. They represent possibilities of characters to be used in a name of some type. If some character appears more frequently in names than in common words, it has a higher probability. If probabilities are used as fuzzy values, a higher precision and a higher recall will be obtained with rare names. However, it has a lower recall with common names comparing with using frequencies.

A hybrid statistics is adopted in our system to take advantages of both frequencies and probabilities. With common naming-characters, frequencies are adopted to get a higher recall with common names. With rare naming-characters, probabilities are adopted to complement frequencies' insufficiency with rare names. The resulting model looks like:

$$\mathcal{L}(typ|c) = \text{Max}\{freq(typ|c), prob(typ|c)\}$$

Data sparseness and reappearances of names make it hard to estimate probabilities. To overcome these difficulties, we propose to use inverse common frequencies to approximate probabilities:

$$icf(c)=1/(freq(common\ word|c)+1)=1/(counts(common\ word, c)+1)$$

Since probabilities are mainly used to estimate the probability of rarely seen events, usually:

$$\text{counts}(\text{common words}, c) \approx \text{counts}(\sim \text{typ}, c), \text{ where } \text{counts}(\text{typ}, c) \leq 2$$

In this case, $\text{icf}(c)$ is approximate to $\text{prob}(c)$:

$$\begin{aligned} \text{prob}(c) &= \text{counts}(\text{typ}, c) / (\text{counts}(\text{typ}, c) + \text{counts}(\sim \text{typ}, c)) \text{ where } \text{counts}(\text{typ}, c) \leq 2 \\ &\approx 1 / (\text{counts}(\sim \text{typ}, c) + 1) \approx \text{icf}(c) \end{aligned}$$

Further, we assume that $\text{counts}(\text{common word}, c)$ is in direct proportion to the number of lexicon entries in which the character c appears. Under these assumptions, we use inverse lexicon counts to approximate probabilities:

$$\text{ilc}(c) = 1 / (\text{Num_of_Lex_Entries}(c) + 1) \approx \text{icf}(c) \approx \text{prob}(\text{typ}|c)$$

Because $\text{ilc}(c)$ is ranged from 0 to 1, $\text{freq}(\text{typ}|c)$ also needs to be normalized to 0 to 1. The distribution of raw data of $\text{freq}(\text{typ}|c)$ is conformed to Zipf's Law, that:

$$P_n \approx 1/n^a, \text{ where } P_n \text{ is the frequency of occurrence of the } n^{\text{th}} \text{ ranked item and } a \text{ is close to } 1.$$

Values with often seen characters are too high and the distinctions among low frequency characters are not wide enough. Therefore, a logarithm function is taken on the raw data to smooth the distribution curve, and then the result is normalized to 0.1 to 1.

$$\text{freq}^*(\text{typ}|c) = \text{Norm}_{0.1,1}(\log(\text{freq}(\text{typ}|c))) \text{ while } \text{freq}(\text{typ}|c) \geq 1$$

Notice that the lower bound of $\text{freq}^*(\text{typ}|c)$ is set to 0.1, not 0. This is because the meaning of events that appear once is greatly different from the meaning of unseen events.

The final character likelihood model looks like:

$$\mathcal{L}(\text{typ}|c) = \text{Max}\{\text{freq}^*(\text{typ}|c), \text{ilc}(c)\}$$

Notice that there are two exceptions to this model. With surnames and transliterating characters, likelihoods of unseen events in training data are assigned to zero. This is because generally surnames and transliterating characters are not arbitrarily given. Probabilities of most characters to be surnames or transliterating characters are actually zero. The original model might cause unnecessary over-generation. To prevent this problem, only surnames and transliterating characters appearing in our training data are adopted as possible ones.

3.2. Open-ended Named Entity Extraction

Open-ended named entity extraction models would estimate likelihoods of strings to be some type of named entity from character likelihoods. Unigram models are adopted as the basis of our models. They could be represented as follows:

$$\begin{aligned} \mathcal{L}^*(\text{typ}|g \cdot s) &= \mathcal{L}'(\text{typ}|g) \times \text{ConRe}(g) \times \text{ConSuff}(\text{typ}, s) \\ &\text{where } g \text{ denotes the genuine name and } s \text{ denotes the suffix part} \end{aligned}$$

If $\mathcal{L}^*(\text{typ}|g \cdot s)$ is over some pre-defined threshold, which is decided by maximizing the f -measure of the recall of training data and the excluding rate of lexicon entries, $g \cdot s$ would be recognized as a possible candidate and added into the candidate pool. Each member of the formula is detailed below:

- $\text{ConRe}(g)$ estimates the confidence could be brought by reoccurrences of the genuine name, which is defined as:

$$\text{ConRe}(g) = k^{\text{Reoccurrence}(g)}$$

$$k = \begin{cases} 2 & \text{when the length of the input document is} \\ & \text{less than 400 characters} \\ 1 + 400 / \text{LEN}(\text{Document}) & \text{Elsewhere} \end{cases}$$

- $\text{ConSuff}(\text{typ}|s)$ estimates the confidence could be brought by the suffix part. Different types of suffixes could bring different quantities of confidence. One suffix part might comprise many different suffixes. The summation of each member's confidence is computed:

$$\text{ConSuff}(\text{typ}, s) = \text{Conf}(\text{typ}_1, s_1) + \text{Conf}(\text{typ}_2, s_2) + \dots + \text{Conf}(\text{typ}_n, s_n) + 1 \text{ where } s = s_1 s_2 \dots s_n$$

Notice that if s is empty, i.e., there are no suffix parts, $\text{ConSuff}(\text{typ}, s)$ would be 1.

- The definition of $\mathcal{L}'(\text{typ}|g)$ is varied from different types of *genuine names*. As we mentioned before, there are four types of genuine names that would be dealt by our system: *personal names*,

location names, organization names, and transliteration names. \mathcal{L} (typ|g) of different types of names is defined as follows:

- ◆ $\mathcal{L}(PER|s) = \text{ArgMax} \{ \mathcal{L}(SUR|s1) * \mathcal{L}(GIV|s2) \}$ for every substring $s1$ and $s2$, where $s = s1 \cdot s2$, “ \cdot ” denotes the string concatenation, and:

$$\mathcal{L}(SUR|s) = \begin{cases} \mathcal{L}(SUR|c1) & \text{when } s \text{ is constituted of one character} \\ \text{Max}\{\text{GAvg}(\mathcal{L}(SUR|c1), \mathcal{L}(SUR|c2)), \mathcal{L}(SUR|c1c2)\} & \text{when } s \text{ is constituted of two characters} \\ 0 & \text{when } s \text{ is longer than two characters} \end{cases}$$

$$\mathcal{L}(GIV|s) = \begin{cases} \mathcal{L}(GIV|c1) & \text{when } s \text{ is constituted of one character} \\ \text{GAvg}(\mathcal{L}(GIV|c1), \mathcal{L}(GIV|c2)) & \text{when } s \text{ is constituted of two characters} \\ 0 & \text{when } s \text{ is longer than two characters} \end{cases}$$

$\text{GAvg}()$ returns geometric means.

- ◆ $\mathcal{L}(TRA|s) = \text{HAvg}(\mathcal{L}(TRA|c_k))$ where $s = c_1 \dots c_n$ and $\text{HAvg}()$ returns harmonic means.
- ◆ $\mathcal{L}(LOC|s) = \begin{cases} \mathcal{L}(LOCL|c1) * \mathcal{L}(LOCF|c2) & \text{when } s = c1c2 \\ \mathcal{L}(LOCL|c1) * \mathcal{L}(LOCF|c2) * \mathcal{L}(LOCF|c3) & \text{when } s = c1c2c3 \\ 0 & \text{elsewhere} \end{cases}$
- ◆ $\mathcal{L}(ORG|s) = \begin{cases} \mathcal{L}(ORGL|c1) * \mathcal{L}(ORGF|c2) & \text{when } s = c1c2 \\ \mathcal{L}(ORGL|c1) * \mathcal{L}(ORGL|c2) * \mathcal{L}(ORGF|c3) & \text{when } s = c1c2c3 \\ 0 & \text{elsewhere} \end{cases}$

3.3. Supplementary Mechanism

Besides the above models, there are three supplementary mechanisms designed to relieve over-generation problems of stochastic models:

1. If some candidate is constituted of two multisyllabic words or one multisyllabic word and one often seen monosyllabic word, this candidate would be removed from the candidate pool.
2. If the first or the last character of some three-character-long organization name candidate is a monosyllabic word that often appears adjacent to a name, as “前遠東” and “東鼎興”, this candidate will be removed from the candidate pool.
3. With transliteration names, sometimes a common word might be wrongly attached by a transliteration candidate. In this situation, maximal-matching-rule-driven lexical analyzer cannot filter it out properly.

A concept called “*team*” based on reoccurrences is introduced to solve the attaching problem. Basically, all substrings of possible transliteration name candidates are also possible candidates. Hence all transliteration name candidates can be grouped into *teams* according to their longest common superstring candidate. For example, a *team* can be represented as:

$$T_{\text{leader}=\text{麥可喬丹}} = \{ \text{麥可}(5), \text{可喬}(5), \text{喬丹}(6), \text{麥可喬}(4), \text{可喬丹}(5), \text{麥可喬丹}(4) \}$$

Where all appearance times of candidates are marked up, and superstring “麥可喬丹” is called the “*leader*” of the *team*.

The following algorithm is then applied:

- I. Subtract leader’s appearance times from each team member
- II. If the *leader* could be split into candidates with non-zero appearance times after subtraction and multisyllabic common words or frequently used monosyllabic words, discard the *leader* and members whose appearance times being subtracted to zero
- III. Form new teams comprised of remaining candidates with new leaders
- IV. Repeat step I-III, until no candidates could be discarded

4. Lexical Analysis

The lexical analyzer is responsible for verifying candidates generated by the candidate generator. Heuristic rules are adopted to filter out false named entity candidates and resolve ambiguities caused by false candidates. There are six heuristic rules applied in order precedence:

Rule 1: *Tri-word maximal matching*, which is proposed by Chen & Liu (1992) [1]. The rule follows below three steps:

1. From the segmenting point, look forward for all possible tri-word combinations.
2. Take the first word of the longest sequence of all, segment this word.
3. Move to the next segmenting point.

For example, with the sentence “張大春天天說”, “張大春” would be picked instead of “張大” because “張大春 天天 說” is longer than “張大 春天 天”.

Rule 2: *Least number of NEs first*, which would pick the tri-word sequence with the least number of named entities among all sequences of the same length.

Rule 3: *Most frequently appearing NEs first*, which would pick the tri-word sequence with the most appearing times of component NEs in the input document.

Rule 4: *Words of even lengths first*, which would choose the sequence with most words of even lengths. There are several exceptions to this rule. First, personal names, transliteration names, and numerical expressions are not concerned in this rule. Second, the often seen monosyllabic words, like “的”, “之”, “也”, etc., are viewed as words of even lengths instead. For example, “張宇 的 成功” is regarded as totally having two words of even lengths, one is “成功” and another one is “的”, not “張宇”. Third, the suffix part of a whole named entity is not considered into the length of it. For example, “揚昇高爾夫球場” is viewed as a word of even lengths, not of odd ones.

Rule 5: *Often seen monosyllabic words first*, which is also proposed by [1], would pick the sequence with the most often seen monosyllabic words.

Rule 6: *Forward precedence*, which would choose the tri-word sequence with longer forward words. For example, with two ambiguous tri-word sequence “決戰 爭 勝負” and “決 戰爭 勝負”, the former would be picked since “決戰” is longer than “決”.

In order to measure the performance of our lexical analyzer on ambiguity resolution, the test samples of our system (61 news articles from United Daily News and Central News Agency, which will be further discussed later) are examined. The following measurements are adopted:

- Ambiguous Tri-Word Sequences: # of all possible tri-word sequences which could not be discriminated by the prior rules
- Resolved: # of tri-word sequences which could be filtered by the corresponding rule
- Errors: # of correct words which are wrongly filtered
- Applying Rate: $\text{Resolved} / \text{Ambiguous Tri-Word Sequences}$
- Accuracy: $1 - \text{Errors} / \text{Resolved}$

The experimental results are listed in Table 4.1:

Table 4.1. The performance of heuristic rules in ambiguity resolution

	Ambiguous Tri-Word Sequences	Resolved	Errors	Applying Rate	Accuracy
Heuristic Rule 1	81273	78263	265	96.30%	99.66%
Heuristic Rule 2	3010	1935	10	64.29%	99.48%
Heuristic Rule 3	1075	225	5	20.93%	97.78%
Heuristic Rule 4	850	603	20	70.94%	96.68%
Heuristic Rule 5	247	9	1	3.64%	88.89%
Heuristic Rule 6	238	238	49	100.00%	79.41%

5. Recovery

The recovery mechanism would revive obvious incorrect results of segmentations which are not suitable to be solved by priority-style rules. These anomalies mainly comprise two situations: over-segmentations caused by under-generation, and under-segmentations caused by over-generation. The segmentation checker would find suspect segmentation sequences and try to recover them.

To deal with over-segmentations, sequences of three or more seldom used monosyllabic words in a row are suspected. These suspects are checked to see if any fragments of them could constitute NE candidates with $\angle(TYP|s)$ over a predefined suspect threshold of the corresponding type.

For example, since $\angle(TRA|“龐畢度”) = 0.43 < 0.51$, the candidate threshold of $\angle(TRA|s)$, the string is usually segmented to “龐畢度” in the first two phases. This suspect sequence will be detected by the segmentation checker. Because $\angle(TRA|“龐畢度”) is larger than the suspect threshold of $\angle(TRA|s)$, which is set to 0.2 in our system, “龐畢度” is added into the candidate list of transliteration names.$

With personal names, there is another special case. Let us consider the personal name “陳水扁”. $\angle(PER|“陳水扁”) = 0.23 < 0.26$, the candidate threshold of $\angle(PER|s)$. However, $\angle(PER|“陳水”) is larger than the candidate threshold. When this situation happens, the personal name is usually incorrectly segmented into a personal name of two characters and a monosyllabic word, such as “陳水扁” in this case. To cope with this situation, the following sequence is also viewed as suspects of over-segmentations:$

two-character-long personal name candidate + seldom used monosyllabic word

On the other hand, to deal with under-segmentations, segmentation sequences constituted of interlaced appearances of transliteration, location, organization names, and seldom used monosyllabic words, are suspected. These sequences are attempted to be re-segmented into a new sequence containing one more word than the original sequences. For example, if “群中” is incorrectly recognized as a location name, the phrase “台北人群中” would be wrongly segmented into a suspect sequence “台北人 群中”. This sequence would be detected and re-segmented into the right sequence “台北 人群中”. If the re-segmenting cannot be performed, the original sequence will be kept.

The procedure of segmentation checker is as follows:

1. Check over-segmented sequences
2. Check under-segmented sequences
3. Repeat step 2, until no new suspect sequences appear
4. Check over-segmented sequences again

6. Evaluation

To measure the performance of our system, a corpus which is balanced and well-tagged according to our standard is needed. The most popular standard test corpus, MET-2 data, is biased on some special topics and uses a different tagging standard from ours. Therefore, instead of a standard testing corpus, we obtain 61 articles from United Daily News and Central News Agency as our test bed. These articles are segmented and tagged by our system and corrected manually.

These 61 articles are gathered from five different domains. They are politics, society, business, sports, and entertainment. Because the quantity of politics news and society news is more than others, we obtain three different sub-topics (lawsuit, government, and election) from politics news and two (crime and local) from society news.

Table 6.1 draws the experimental results of our system. Standard measurements are estimated:

$$\text{Recall} = (\# \text{ of Ext.} - \# \text{ of False}) / (\# \text{ of True})$$

$$\text{Precision} = 1 - (\# \text{ of False}) / (\# \text{ of Ext.})$$

Notice that there are two special columns in the table, *number of words* and *excluding rate*. Because appearing frequencies of NEs are varied in different domains and have a great impact on precision, precision is thus less meaningful. We consider that excluding rate might be a better measurement of over-generation. Excluding rate is counted from:

$$\text{Excluding rate} = 1 - (\# \text{ of False}) / (\# \text{ of Words} - \# \text{ of True})$$

It stands for the percentage of non-NEs being correctly filtered by our system.

Table 6.1. Experimental results of our system

Topic	Articles	True	Extracted	False	Words	Recall	Precision	Excluding
Politics 1	7	211	224	34	3460	90.05%	84.82%	98.95%
Politics 2	10	465	444	32	4343	88.60%	92.79%	99.17%
Politics 3	7	158	155	18	2750	86.71%	88.39%	99.31%
Society 1	7	321	317	23	3599	91.59%	92.74%	99.30%
Society 2	10	372	378	39	5423	91.13%	89.68%	99.23%
Business	7	295	289	34	3392	86.44%	88.24%	98.90%
Sports	7	272	226	18	3690	76.47%	92.04%	99.47%
Entertainment	6	196	182	16	2742	84.69%	91.21%	99.37%
Total	61	2290	2215	214	29399	87.38%	90.34%	99.21%

Table 6.2 shows the recall of different types of NE. Because we do not focus on automatic classification, one NE might be recognized by many different models, it's hard to judge the precision of each type and only the recalls are listed here.

Table 6.2. Recall of our system with different types of NEs

Topic		PER	TRA	LOC	ORG	ABB	PO	LO	OO	AO	TITLE	MIS
Politics 1	True	104	2	14	9	6	0	0	2	1	66	5
	Detected	102	2	9	5	6	0	0	2	0	61	2
	Recall	98.08%	100.00%	64.29%	55.56%	100.00%	--	--	100.00%	0.00%	92.42%	40.00%
Politics 2	True	261	0	18	11	9	5	0	148	10	4	0
	Detected	250	0	17	5	6	5	0	128	2	2	0
	Recall	95.79%	0.00%	94.44%	45.45%	66.67%	100.00%	--	86.49%	20.00%	50.00%	--
Politics 3	True	43	0	24	0	2	0	0	42	46	2	0
	Detected	43	0	24	0	2	0	0	28	43	0	0
	Recall	100.00%	0.00%	100.00%	0.00%	100.00%	--	--	66.67%	93.48%	0.00%	--
Society 1	True	185	1	115	0	30	2	1	2	0	1	0
	Detected	180	1	98	0	29	1	1	2	0	0	0
	Recall	97.30%	100.00%	85.22%	0.00%	96.67%	50.00%	100.00%	100.00%	--	0.00%	--
Society 2	True	135	3	137	13	6	2	8	60	5	0	2
	Detected	133	2	128	11	6	2	7	43	3	0	2
	Recall	98.52%	66.67%	93.43%	84.62%	100.00%	100.00%	87.50%	71.67%	60.00%	--	100.00%
Business	True	72	2	65	131	0	0	9	4	5	1	0
	Detected	68	2	56	111	0	0	3	4	5	0	0
	Recall	94.44%	100.00%	86.15%	84.73%	0.00%	--	33.33%	100.00%	100.00%	0.00%	--
Sports	True	56	110	23	9	1	4	6	58	3	1	5
	Detected	50	99	19	8	1	0	6	20	2	1	4
	Recall	89.29%	90.00%	82.61%	88.89%	100.00%	0.00%	100.00%	34.48%	66.67%	100.00%	80.00%
Entertainment	True	126	12	18	4	4	12	1	0	5	12	4
	Detected	118	9	16	2	4	6	1	0	5	7	1
	Recall	93.65%	75.00%	88.89%	50.00%	100.00%	50.00%	100.00%	--	100.00%	58.33%	25.00%
Total	True	982	130	414	177	58	25	25	316	75	87	16
	Detected	944	115	367	142	54	14	18	227	60	71	9
	Recall	96.13%	88.46%	88.65%	80.23%	93.10%	56.00%	72.00%	71.84%	80.00%	81.61%	56.25%

Notice that the first five columns (PER, TRA, LOC, ORG, ABB) only include the focused types of our system. Column PER comprise only formal Chinese personal names and personal names with appellations. Other personal names, such as Japanese name “酒井光次郎” and pseudonym “老子”, are counted in the column PO instead. Monosyllabic place names without suffixes, like “粤” and “台”, are recognized by lexicon matching and counted in the column LO. Government and team names are also recognized by lexicon. They are viewed as OO. All other location names and organization names are included in the column LOC and ORG respectively. Column ABB contains only abbreviations with original reference in the input document, other abbreviations are considered as AO.

7. Conclusions and Future Works

Overall speaking, pure lexical information is employed to recognize named entities in our system. Only statistical features and internal structures of NE are utilized. Our statistical model and heuristic rules are simplified for easy implementation. However, our system gets a satisfied performance, and there are still many rooms for improvement.

First, statistical models could be refined. More training data could be collected. More elaborate candidate generating models could be adopted, such as bi-gram models. More internal features could be exploited, such as positional information of characters. Contextual information, such as word probability of being adjacent to some type of NEs, could be also added into our model.

Second, heuristic rules could be more completed or substituted by other mechanisms. Shortcomings of heuristic rules form an upper-bound barrier of performances. More rules could be introduced to cover the inadequacies of original ones. Other mechanism like statistical approaches could be used to replace rule-driven methods.

Third, more candidate generating models could be added. Many types of NEs have not been addressed in our system. We could find that these NEs occupy a great proportion of true negative errors. If these NEs could be recognized, the recall of our system is supposed to be boosted.

Fourth, more knowledge could be gathered and utilized. The suffix and appellation information used in our system is handcrafted at present. Bootstrapping or machine learning algorithm might help us automatically retrieve these kinds of information from the Internet or corpus. Part-of-speech tagging, syntactic checking and even semantic analysis might also be added into our future system.

References

- [1] Chen, Keh-Jiann and S. H. Liu, 1992, "Word Identification for Mandarin Chinese Sentences," Proceedings of COLING-92, Vol. 1, pp. 101-107
- [2] Chinchor, Nancy, 1998, "MUC-7 Test Score Reports for all Participants and all Tasks" in Proceedings of the MUC-7.
- [3] Chua, Tat-Seng and J. Liu, 2002, "Learning Pattern Rules for Chinese Named Entity Extraction," Proceedings of AAAI/IAAI 2002, pp. 411-418
- [4] Goh, Chooi Ling, M. Asahara, Y. Matsumoto, 2003, "Chinese Unknown Word Identification Using Character-based Tagging and Chunking," ACL-2003 Interactive Posters/Demo, pp. 197-200
- [5] Ji, Heng and Z. S. Luo, 2001, "Inverse Name Frequency Model and Rule Based Chinese Name Identification," (In Chinese) Natural Language Understanding and Machine Translation, Tsinghua University Press, pp. 123-128.
- [6] Mo, Ruo Ping, Y. J. Yang, K. J. Chen, and C. R. Huang, 1996, "Determinative- Measure Compounds in Mandarin Chinese Formation Rules and Parser Implementation," In C. R. Huang, K. J. Chen and B. K. Tsou (Eds.), Readings in Chinese natural language processing, pp. 123-146, Journal of Chinese Monograph Series Number 9.
- [7] Sekine, Satoshi, K. Sudo, and C. Nobata, 2002, "Extended named entity hierarchy," Proceedings of the LREC 2002 Conference, pp. 1818-1824.
- [8] Sun, Jian, J. F. Gao, L. Zhang, M. Zhou, and C. N. Huang, 2002, "Chinese Named Entity Identification Using Class-based Language Model," Proceedings of the 19th International Conference on Computational Linguistics, Taipei, pp. 967-973
- [9] Tan, Hong-Ye, 1999, "Chinese Place Automatic Recognition Research," Proceedings of Computational Language, C. N. Huang & Z.D. Dong, ed., Tsinghua Univ. Press, Beijing, China.
- [10] Wu, Xue-Jun, J. B. Zhu, H.Z. Wang, and N. Ye, 2003, "The Application of the Method of Co-Training in Identification of Chinese Organization Names," The 2003 National Joint Symposium on Computational Linguistics (JSCL-2003)
- [11] Xiao, Jing, J. M. Liu, and T. S. Chua, 2002, "Extracting pronunciation-translated names from Chinese texts using bootstrapping approach", Nineteenth International Conference on Computational Linguistics (COLING2002), Taipei, Taiwan, Aug 2002.
- [12] Yu, Shi-Hong, S. H. Bai, and P. Wu, 1998, "Description of the Kent Ridge Digital Labs System Used for MUC-7," Proceedings of the Seventh Message Understanding Conference (MUC-7).
- [13] Zheng, Chen, W. Y. Liu, and F. Zhang, 2002, "A New Statistical Approach to Personal Name Extraction," ICML 2002, pp. 67-74.