A Statistical Approach to Extract Chinese Chunk Candidates from Large Corpora

ZHANG Le, LÜ Xue-qiang, SHEN Yan-na, YAO Tian-shun
Institute of Computer Software & Theory.
School of Information Science & Engineering, Northeastern University
Shenyang, 110004 China
Email: ejoy@xinhuanet.com, studystrong@sohu.com, neu_syn@sohu.com, tsyao@mail.neu.edu.cn

Abstract

The extraction of Chunk candidates from real corpora is one of the fundamental tasks of building example-based machine translation model. This paper presents a statistical approach to extract Chinese chunk candidates from large monolingual corpora. The first step is to extract large N-grams (up to 20-gram) from raw corpus. Then two newly proposed Fast Statistical Substring Reduction (FSSR) algorithms can be applied to the initial N-gram set to remove some unnecessary N-grams using their frequency information. The two algorithms are efficient (both have a time complexity of $O(n)$) and can effectively reduce the size of N-gram set up to 50%. Finally, mutual information is used to obtain chunk candidates from reduced N-gram set.

Perhaps the biggest contribution of this paper is that it is the first time to apply Fast Statistical Substring Reduction algorithm to large corpora and demonstrate the effectiveness and efficiency of this algorithm which, in our hope, will shed new light on large scale corpus oriented research. Experiments on three corpora with different sizes show that this method can extract chunk candidates from corpora of giga bytes efficiently under current computational power. We get an extraction accuracy of 86.3% from People Daily 2000 news corpus.

Key Words: Chunk extraction, N-gram, Substring Reduction, Corpus

1 Introduction

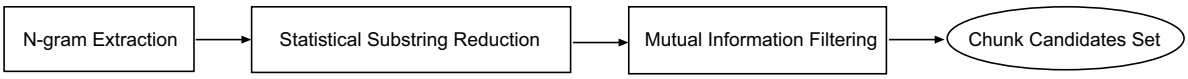
With the rapid development of computational power and the availability of large online corpora (BNC (Clear, 1993), People Daily (YÜ et al, 2002)), there has been a dramatic shift in computational linguistics from manually construction knowledge bases to partially or totally automatic knowledge acquisition by applying statistical learning methods to large corpora (see SU, 1996, for an overview). The concept of chunk was first raised by (Abney, 1991) in the early nineties to make the task of language parsing easier. He suggested to develop a parser based on chunk that decomposes sentences into chunks with each chunk being a syntactic unit for easier parsing. Chunks, especially bilingual chunk pairs, can be a useful ingredient in example based machine translation (EBMT) system to help obtain better translation result (YAO et al, 2000). Other NLP tasks such as information retrieval, knowledge discovery and the construction of semantic dictionary can also benefit from such kind of resources. Most Text Chunking methods applied to English require either a parsing stage to parse raw corpus into parsed trees or a corpus which already has syntactical information (such as Treebank) (Erik and Sabine, 2000). Unfortunately, neither kind of resources is largely available for Chinese. Some language specific properties of Chinese further impose challenges on Chinese chunking. Unlike western languages that have explicit word boundary, Chinese has no word separation in sentence and a sentence must be segmented into words before further processing. Segmentation is a field that has been researched for decades with more than two dozen methods tried in literature (Ponte and Croft, 1996; Palmer, 1997; Teahan, 2000). Even today no segmentation method totally satisfies native speakers. In addition, Chinese language enjoys more freedom in sentence structure than English, making even shallow parsing a formidable task.

To address these problems, this paper presents a novel approach to extract Chinese chunk candidates from large unsegmented raw corpora. The originality of our approach resides in the statistical substring reduction, a procedure aims at efficiently removing unpoetical N-grams from extracted N-gram set. The first step is to acquire large N-gram (up to 20-gram) statistics from raw corpus. This initial N-gram set contains a vast amount of "garbage strings" which do not have any meaning at all. The work of (Fung, 1994) showed: without the help of a machine-readable dictionary, the extracted trigrams and 4-grams from Chinese raw corpus contain only 31.3% and 36.75% valid phrases

ICCPOL2003, pp. 109-117, ShenYang (China)

respectively. In other words, meaningful chunk candidates only bear a small portion. In order to rule out the majority "garbage strings" from the initial N-gram set, a *statistical substring reduction* algorithm need to be employed to reduce some "garbage substrings" to their super strings using frequency information. For instance, if both "亚太经合组织 (Asia-Pacific Economic Cooperation)" and "亚太经合组 (Asia-Pacific Economic)" occur 10 times in corpus, the latter should be removed from the N-gram set, since it is the substring of the former with the same frequency. This procedure is called *Statistical Substring Reduction*. When the initial N-gram set contains n N-grams, traditional substring reduction algorithm has a time complexity of $O(n^2)$ (Han et al, 2001), making it infeasible to handle large amount of real world corpora. To solve this computational bottleneck, we propose two Fast Statistical Substring Reduction (FSSR) algorithms, both having an $O(n)$ time complexity under ideal condition. Experiment shows it only take minutes to do substring reduction even on N-gram set of gigabytes corpus. Finally, mutual information criterion is used to filter chunk candidates from reduced N-gram set. Figure 1 gives an overview of the whole process.

Figure 1: The procedure of Chinese chunk candidates extraction



In what follows, Section 2 briefly reviews the large N-gram extraction algorithm. Section 3 describes the two Fast Statistical Substring Reduction algorithms in detail. Mutual information filtering strategy is discussed in Section 4. And Section 5 presents the experimental results on three corpora. Finally, we draw in Section 6 a conclusion and some perspectives.

2 Large N-gram Extraction

Traditional N-gram acquisition methods (hash table, sparse matrix) can not handle even a modest size of n (say $n = 5$) because of the huge amount of memory space and time required. In the case of calculating Chinese trigram statistics, assuming there are 50000 Chinese words with an average length of 2 (4 bytes[1]), and we use a 16 bit integer to store frequency (2 bytes), we need $(5 \times 10^4)^3 \times (4 + 2) = 7.5 \times 10^{14}$ *bytes* ≈ 700000 *Gb* memory to store all the trigram statistics. Even in real corpus the sparse data phenomenon requires less storage, the time and space burden is quite severe to current hardware. (Nagao, 1994) proposed a new method to extract N-gram statistics with arbitrary size of n. The basic idea of this method is to transform the N-gram acquisition problem into extracting adjacent substrings with same prefix in a sorted prefix table. By treating the whole corpus as a huge character string S, we can build a pointer table with each entry keeping a pointer to a substring of S. Then a sorting operation is carried out on the pointer table based on the comparison of the substrings it points to, resulting a sorted prefix pointer table.

Once the prefix table is ready, extracting N-gram with arbitrary n is straightforward: First let us read out the first n characters of the first substring in the pointer table and see if the next substring has at least the same n prefix characters with the first one. If so, let us continue to check the next substring until we find a string which does not have the same n prefix characters. The number of words checked up so far is just the frequency of the n prefix characters (N-gram) of the first string. To extract all the N-gram statistics in corpus, we only need to repeatedly perform this operation until the last entry of prefix table. The reader is referred to (Nagao, 1994) for a detail description of this method.

This method is also space efficient: for a corpus containing n Chinese characters, an entry in prefix pointer table needs at least p bits ($2^p \geq n$) which means: for a 32 bit pointer entry (4 bytes), it can process at most $2^{32} \approx 4GB$ characters corpus. Considering the memory needed to store corpus (a long text string), the N-gram extraction operation needs a storage of $2n + 4n = 6n$ bytes for a corpus with n characters. For the PeopleDaily 2000 corpus which contains roughly 25 million characters, the procedure needs $6 \times 25 \times 10^6 \approx 150MB$ memory, which can be easily fitted into today's desktop computer.

3 Fast Statistical Substring Reduction Algorithm

After getting the initial N-gram set, a *statistical substring reduction* procedure is required to rule out some unpoetical substrings. The substring reduction problem can be described as follows: given a super string $S_1 = x_1x_2 \ldots x_n$ and one of its substring $S_2 = x_i \ldots x_j$ ($i \geq 1, j \leq n, S_2 \neq S_1$), we remove S_2 from N-gram set if S_2 has a frequency near

[1]Each Chinese character needs 2 bytes to store.

to S_1 in corpus. More precisely, if the two strings have the same frequency in corpus, we call it substring reduction with *equal frequency*; if the two frequencies f_1 and f_2 are not equal but the condition $|f_1 - f_2| < k$ holds, we call it substring reduction with *k-nearest frequency*.

For the rest of the paper, we shall use the following notational conventions and definitions:

Definition 1 *Let $\Omega = \{X_1, X_2, \ldots, X_i, \ldots, X_n\}$ be the N-gram set to be reduced, where X_i denotes the ith N-gram. Each X_i is defined as: $X_i = x_{i1}x_{i2}\ldots x_{ij}\ldots x_{im}$, where x_{ij} is a character. Let $Len(X)$ denote the length of string X, then we have $Len(X) = m$.*

Definition 2 *Let $f(X)$ denote the frequency of X occurs in corpus. If X never occurs in corpus, $f(X) = 0$.*

Definition 3 *Let Y be a N-gram such that $Y \in \Omega$ and $Y = y_1y_2\ldots y_n$ $(n \geq 2)$. Then string X that consists of p $(p < n)$ continual characters of Y that is called the substring of Y, denoted by $X \propto Y$. The left most p continual characters of Y make up of string X_L which is called **left substring** of Y, denoted by $X_L \propto_L Y$. Similarly, the right most p continual characters of Y constitute string X_R, which is called the **right substring** of Y, is denoted by $X_R \propto_R Y$.*

Definition 4 *Let $X, Y \in \Omega$, if $X \propto Y$ and $f(X) - f(Y) = 0$ then we say X can be reduced by Y with **equal frequency** or Y can reduce X with **equal frequency**; If $X \propto Y$ and $f(X) - f(Y) < k (k > 0)$ then we say X can be reduced by Y with **k-nearest frequency** or Y can reduce X with **k-nearest frequency**. If X can be reduced by a statistical string with equal (k-nearest) frequency we say X can be reduced with equal (k-nearest) frequency. Whenever it is clear from context, we say X can be reduced for short.*

3.1 Traditional Statistical Substring Reduction Algorithm

Let $|\Omega| = n$, then Ω has n N-grams. The ith$(1 \leq i \leq n)$ N-grams in Ω is represented as a 3-tuple $< X_i, f_i, M_i >$, where X_i denote the ith N-gram, $f_i = f(X_i)$ is the frequency of X_i in corpus. M_i is a merging flag such that: $M_i = 0$ means X_i is not reduced, while $M_i = 1$ indicates X_i being reduced by its super string. The initial value of M_i is set to 0. This algorithm is given in Algorithm 1.

Algorithm 1 k-nearest Frequency Statistical Substring Reduction Algorithm (Traditional Algorithm)

```
1: Input:  Ω, k(k > 0)
2: Output: reduced N-gram set Ω'
3: for i = 1 to n do
4:   for j = 1 to n do
5:     if X_i ∝ X_j and f_i − f_j < k then
6:       M_i = 1
7: for i = 1 to n do
8:   if M_i = 0 then
9:     output X_i
```

Obviously, this algorithm has a time complexity of $O(n^2)$, making it infeasible to handle large corpora.

3.2 Two Fast Statistical Substring Reduction Algorithm

To describe algorithm 2, we need to introduce the notation of *reversed string* first:

Definition 5 *Let $X = x_1x_2\ldots x_n$ be a N-gram, then $X_R = x_nx_{n-1}\ldots x_1$ is defined as the **reversed string** of X. All the reversed strings in Ω comprise the reversed string set Ω_R.*

In this algorithm, all steps have a time complexity of $O(n)$ except step 3 and 9, which perform sorting on n N-grams. It is worth mention that sorting can be implemented with *radix sort* which is an $O(n)$ operation, therefore this algorithm has an ideal time complexity of $O(n)$. When special requirement on memory usage or speed is not very important, one can use *quick sort* to avoid additional space requirement imposed by radix sort. Quick sort is an $O(n \log n)$ operation, so the overall time complexity of algorithm 2 is $O(n \log n)$.

In algorithm 2, only step 6 and 12 modify the merging flag, we call them *Left Merging* and *Right Merging* of algorithm 2. In algorithm 1, each string must be compared with all other strings in Ω whereas in algorithm 2, each string is only required to be compared with two strings. Therefore algorithm 2 reduces the number of comparison tremendously compared to algorithm 1.

Algorithm 2 Fast k-nearest Frequency Statistical Substring Reduction Algorithm

1: Input: $\Omega, k(k > 0)$
2: Output: reduced N-gram set Ω'
3: sort all N-grams in Ω in ascending order according to X_i's value
4: **for** $i = 1$ to n **do**
5: **if** $X_i \propto_L X_{i+1}$ and $f_i - f_{i+1} < k$ **then**
6: $M_i = 1$
7: **for** $i = 1$ to n **do**
8: reverse X_i
9: sort all N-grams in Ω in ascending order according to X_i's value
10: **for** $i = 1$ to $n - 1$ **do**
11: **if** $X_i \propto_L X_{i+1}$ and $f_i - f_{i+1} < k$ **then**
12: $M_i = 1$
13: **for** $i = 1$ to n **do**
14: reverse X_i
15: **for** $i = 1$ to n **do**
16: **if** $M_i = 0$ **then**
17: output X_i

Algorithm 3 is another Fast Statistical Substring Reduction algorithm based on hash table. For a N-gram $X \in \Omega$, we can design a hash function $hash(X)$ and a hash table HT with a capacity of T, which can store 2-tuple $< X, f(X) >$. Let $HT[i] \rightarrow X$ and $HT[i] \rightarrow f$ denote the N-gram and its frequency stored in the ith slot of HT respectively. If there is no N-gram stored in $HT[i]$, we say $HT[i]$ is *empty*. The operation of removing $< X, f(X) >$ from $HT[i]$ is called to *empty* $HT[i]$. To make our algorithm concise, we assume there is no hash collision here. The hash based FSSR algorithm is given in algorithm 3.

Algorithm 3 Fast k-nearest Frequency Statistical Substring Reduction Algorithm(2)

1: Input: $\Omega, k(k > 0), m_1$ $(m_1 \geq 1)$ m_1 denotes the minimal length of a substring.
2: Output: reduced N-gram set Ω'
3: hash all strings in Ω into HT based on $hash(X)$
4: **for** $i = 1$ to T **do**
5: **if** $HT[i]$ is not empty **then**
6: **for all** substring Y of $HT[i] \rightarrow X$ such that $Len(Y) > m_1$ and $Len(Y) \leq Len(X) - 1$ **do**
7: **if** $HT[hash(Y)]$ is not empty and $(HT[hash(Y)] \rightarrow f - HT[i] \rightarrow f) < k$ **then**
8: empty $HT[hash(Y)]$
9: **for** $i = 1$ to T **do**
10: **if** $HT[i]$ is not empty **then**
11: output $HT[i] \rightarrow X$

In this algorithm, step 3 and 9 have a time complexity of $O(T)$. Let m_2 denote the length of the longest N-gram in Ω, then for any given m_1, m_2 $(m_1 < m_2)$ the following conditions hold:

1. there are at most $(m_2 - m_1 + 3)(m_2 - m_1)/2$ substrings for any super string.
2. the iterations of inner loop (step 6) is less than the constant $(m_2 - m_1 + 3)(m_2 - m_1)/2$.

So the time complexity of step 6 is determined by outer loop, also $O(T)$. Consequently the overall time complexity of algorithm 3 is $O(T)$.

The motivation of algorithm 1 is to find all super strings starting from a given substring, whereas algorithm 3 tries to locate all substrings of a given super string. Since it is impossible to directly locate the super strings from its substring, while one can enumerate all substrings of a super string easily, the searching space of algorithm 3 is much smaller than algorithm 1. This explains why algorithm 3 has a lower time complexity compared to algorithm 1. Algorithm 2 also tries to locate substrings from super string but restricts the searching space in suffix strings, hence has a minimal searching space with a trade off of time spent on sorting. The reader is referred to (LÜ, 2003) for a mathematical proof on the equality of the three algorithms.

4 Post Processing

After performing N-gram extraction and statistical substring reduction, we get chunk candidates without the help of an online dictionary. Still, there are many nonsensical N-grams ("garbage string") in the result set. The purpose of post processing is to rule out these "garbage strings" from candidates set as many as possible while retaining meaningful chunks. Currently, a mutual information criterion is employed for filtering.

4.1 Mutual Information Criterion

Mutual Information(MI) is a statistical measurement of correlation of two random variables X and Y, which is defined as:

$$MI(x,y) = \log \frac{P(x,y)}{P(x) \times P(y)} \tag{1}$$

Where the probability x and y occur in corpus are denoted by $P(x)$ and $P(y)$, respectively. $P(x,y)$ denotes the probability of x and y co-occurs. For two adjacent characters x and y in corpus, their Mutual Information can be calculated as:

$$MI(x,y) = \log \frac{P(x,y)}{P(x) \times P(y)} \approx \log \frac{\frac{C(x,y)}{N}}{\frac{C(x)}{N} \times \frac{C(y)}{N}} = \log \frac{C(x,y) \times N}{C(x) \times C(y)} \tag{2}$$

Where N is the total number of characters in corpus, $C(x), C(y), C(x,y)$ denote the frequencies of x,y and (x,y) occur in corpus. Here we estimate $P(x), P(y)$, and $P(x,y)$ through $\frac{C(x)}{N}$, $\frac{C(y)}{N}$, $\frac{C(x,y)}{N}$ respectively. Mutual information reflects the degree of correlation of two characters:

1. Given two characters x and y, if $MI(x,y) \gg 0$, it means x and y are highly correlated.
2. if $MI(x,y) \approx 0$, it means there is no manifest correlation between x and y. Their cooccurence is only by chance.
3. if $MI(x,y) \ll 0$, then there is no correlation between x and y.

4.2 Filtering N-gram with Mutual Information

We can filter N-grams according to their mutual information. In our experiment, if the mutual information exceeds predefined thresholds (i.e. $MI(x,y) > f$ where f is an empirical threshold), we consider (x,y) is a valid chunk candidate. Considering some language specific properties of Chinese, different filtering strategies are used based on the length of N-gram:

1. For bigram (a,b): remove all (a,b) whose $MI(a,b) < f$ from result set.
2. For trigram (a,b,c): remove (a,b,c) if either $MI(a,b) < f$ or $MI(b,c) < f$.
3. For 4-gram (a,b,c,d): most Chinese 4-grams are compound words made up of two bigrams in the form $((a,b),(c,d))$, thus $MI(a,b)$ and $MI(c,d)$ tend to have a higher value. So our filtering strategy is to remove (a,b,c,d) when either $MI(a,b) < f$ or $MI(c,d) < f$.
4. N-gram with $n > 4$: by manually inspecting the reduced N-gram set we find most extracted large N-grams $(n > 4)$ in Chinese are compound words, collocations or proper names which are already valid chunk candidates, so we keep all N-grams longer than 4-gram without future filtering.

Some high frequency Chinese characters do not have strong cohesion when forming compound words. Most of them are function words, the possessive and the copula, such as "的,地,得", "我,你,它" and "个,日,月,年". Their existence adds *noisy* to the result set. A stop list is built to further filter out N-grams with these heading (tailing) characters.

5 Evaluation

To measure the performance of our model under different sizes of corpora, we choose three Chinese corpora with different sizes (Table 1). All experiments in this paper are carried out on a Dell PowerEdge 6000/700 with a PIII 700 MHz CPU and 2G RAM.

Label	Source	Domain	Size	Characters
corpus1	People Daily of Jan, 1998	News	3.5M	1.8 million
corpus2	People Daily of 2000	News	48M	25 million
corpus3	Web pages from internet	Various topics (novel, sports, politics etc.)	1GB	520 million

Table 1: Summary of the three corpora

5.1 Experiment 1: k-nearest Frequency Statistical Substring Reduction

Our goal here is to measure the effect of the fast substring reduction algorithm on corpora with different sizes. To this end, we first extract 2-20 N-grams from raw corpora. Only N-grams with frequency larger than threshold f_0 are extracted to eliminate the influence of low frequency N-grams that are not reliable. Here we choose $f_0 = \lfloor \log_{10} n \rfloor$, and n is the total number of characters in corpus. We first run k-nearest substring reduction algorithm to reduce the size of the initial N-gram set by removing incomplete collocations (nonsensical substrings). We use $k = f_0$ in our experiments. Traditional substring reduction method (algorithm 1) serves as the baseline method to measure the performances of two fast statistical substring reduction algorithms (algorithm 2 and 3). The sorting operation in algorithm 2 is implemented as quick sort. Table 2 summarizes the results we obtained[2]:

Label / Time (Including I/O)	Algo 1	Algo 2	Algo 3	N-grams Before	N-grams After
corpus1	17 min 20 sec	3.3 sec	4.4 sec	110890	60458
corpus2	27 hours	48.8 sec	54.6 sec	1397264	672249
corpus3	N/A	8 min 23 sec	7 min 25 sec	19737657	10388240

Table 2: 2 - 20-gram k-nearest statistical substring reduction results

It can be shown from table 2 that the statistical substring reduction algorithm does reduce the N-gram result set size significantly: the reduced N-gram set is 45% - 50% smaller than the original N-gram set. And the data in table 2 indicates that the two newly proposed Fast Statistical Substring Reduction Algorithms are vastly superior to baseline algorithm in terms of time consuming: even on small corpus like corpus 1, the speeds of algorithm 2 and 3 are 200-300 times faster than algorithm 1. This difference is not surprising. Since algorithm 1 has a time complexity of $O(n^2)$, making it infeasible to handle even corpus of modest size, whereas the two fast algorithms both have an $O(n)$ time complexity, making even very large corpus tractable under current computational power: it takes less than ten minutes to reduce a 2-20 N-gram set from corpus of 1 Giga bytes.

To further investigate the performances of algorithm 2 and 3 on corpora with different sizes, we draw twenty samples from corpus 3 with sizes of: 50M,100M,...,1000M, and extract 2-20 N-grams ($f_0 = 6$) from these samples. After that, we run algorithm 2 and 3 ($k = 6$) on these N-gram sets, recording the time consumed (not including I/O operation). The result is illustrated in Figure 2:

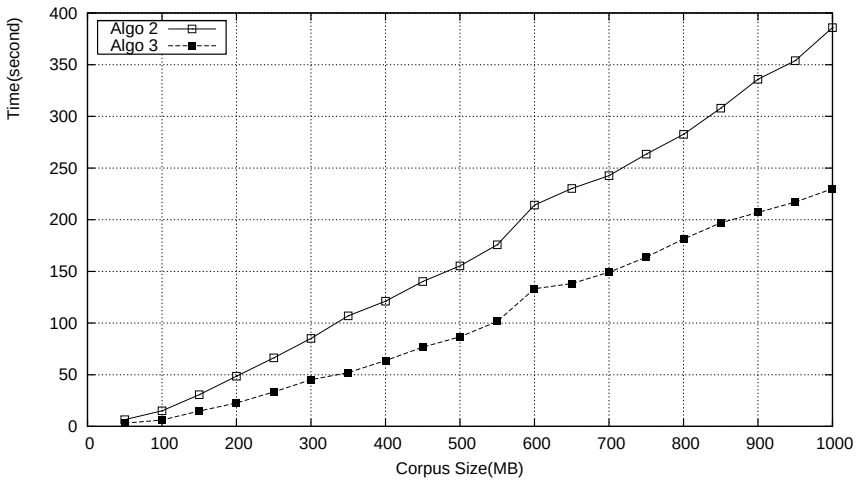
Figure 2 shows that when algorithm 2 and 3 are applied to small corpora (<100M), there are no manifest differences between them. As the size of corpus continues to grow, algorithm 3 becomes increasingly faster than algorithm 2. On the 1000M sample, algorithm 3 performs 40% times faster than algorithm 2. The possible reason is, although algorithm 3 has a time complexity of $O(n)$, the hash finding has extra cost that hinders its speed. Moreover, on small corpus the additional cost of loading hash table outweighs the speed gained by hashing, making algorithm 3 perform slightly worse than algorithm 2 (table 2). However, when dealing with large corpus, most computation of algorithm 2 is spent on quick sort ($O(n \log n)$), which costs more time than algorithm 3's hash finding. Consequently, algorithm 3 performs much better on large corpora. We therefore recommend that when processing corpora of median size (<100MB), algorithm 2 is fast enough, for large corpus of Giga bytes, algorithm 3 should be chosen for best performance.

5.2 Experiment 2: Filtering Chunk Candidates with Mutual Information

Even after the operation of substring reduction, the result set still contains many meaningless N-grams, we further filter them with mutual information criterion in order to remove these superfluous "garbage strings". The reduced set

[2]We did not run algorithm 1 on corpus 3 for it being too large to be efficiently handled by traditional statistical substring algorithm.

Figure 2: Benchmark of Algorithm 2 and 3 under corpora with different sizes



of corpus 2 (People Daily of year 2000) is selected to conduct this experiment. After filtering, our chunk candidates set contains 111864 N-grams. To estimate the precision, we randomly choose 1000 N-grams out of the result set and do a manually check. There are 863 meaningful chunk candidates found out of the 1000 samples, so the precision of the extraction is about 86.3%. Some of the extraction results are given in table 3.

Meaningful Chunks	Nonsensical Chunks
富润 (Fu Run, company name)	无所
獭兔 (the rex)	线贯
安妮 (Annie, person name)	张床
被窃 (be stolen)	丽画
明确地表示 (to express explicitly)	院所属
可口可乐公司 (the Coca-Cola company)	处寻找
发展民族教育 (developing national education system)	著名女
语重心长地说 (to tell with great patience)	明确保
瓦斯爆炸事故 (gas explosion accident)	成社会主义
义务植树活动 (tree-planting action by volunteers)	量逐年增
遇到许多困难 (come across many difficulties)	通过了专家
增进了相互了解和友谊 (to improve the friendship and mutual understanding)	推动两岸人员往来和各

Table 3: Some chunk candidates extracted from People Daily 2000 corpus

From the result set we can find that most large N-grams ($n \geq 4$) are idioms, collocations or domain specific compounds, most of which are not included in traditional Chinese dictionary. The majority errors occur in bigrams and trigrams, possible reasons are:

1. some bigrams are wrongly identified as chunks because in Chinese these bigrams tend to occur frequently as common parts of many longer phrases. But these bigrams themselves are nonsensical. For example, “无所” is included in “无所适从,无所谓,无所不能,无所作为...”; “丽画” is included in “壮丽画卷,美丽画面,绚丽画卷...”

2. Occasionally, some long chunks extracted are incomplete because of the N-gram length limitation (20 characters in our experiment). For instance, the incomplete chunk: “设小康社会并加快推进现代化的新的发展阶段” has 20 characters. If we extract 21-gram we can correctly extract its super string (the complete chunk) as:“建设小康社会并加快推进现代化的新的发展阶段 (building a well-off society in an all-round way and speeding up modernization to the new stage)” (21 characters).

Currently, the MI criterion we used could not effectively rectify the first kind of error. This is probably because we only extract chunk with purely statistical measurement without even basic linguistic knowledge. Still, the result of this method is encouraging.

6 Conclusion

In this paper, we present a purely statistical method to extract Chinese chunk candidates from large monolingual corpora. To address the redundant substring problem in extracted N-gram set, two Fast Statistical Substring Reduction algorithms are proposed, both of which have a time complexity of $O(n)$. Finally, mutual information criterion is applied to extract useful chunk candidates. Perhaps the biggest contribution of this paper is that it is the first time to apply Fast Statistical Substring Reduction algorithm to large corpora and demonstrate the effectiveness and efficiency of this algorithm which, in our hope, will shed new light on large scale corpus oriented research such as new word discovery, bilingual chunk extraction, long range collocation discovery etc. In future work, we would like to incorporate more linguistic knowledge in our method such as:

1. Use probabilistic distribution of proper names to help detect chunks contains Chinese proper name.
2. May resort to a POS tagger or even a shallow tree parser to obtain basic linguistic information of chunk to make the method more *linguistic viable*.
3. Use other criterions (such as a statistical language model) to help evaluate the result set.

Acknowledgments

We are grateful to YANG Er-bao and HUO Huan for helpful discussions and suggestions. This research was partially funded by the National Natural Science Foundation of China under Grant No. 60083006 and the National Grand Foundational Research 973 Program of China under Grant No. G19980305011.

References

Steven P. Abney. Parsing by Chunks. In Robert C. Berwick, Steven P. Abney, and Carol Tenny, editors, Principle-Based Parsing: Computation and Psycholinguistics, pages 257–278. Kluwer Academic Publishers, Boston, 1991.

Clear, J.H. The British National Corpus, The Digital Word : text-based computing in the humanities . Cambridge, Mass.: MIT Press,: 163-187, 1993.

Erik F. Tjong Kim Sang and Sabine Buchholz. Introduction to the CoNLL-2000 Shared Task: Chunking. Proceedings of CoNLL-2000 and LLL-2000, pages 127-132, Lisbon, Portugal, 2000.

Han, Ke-song, Wang, Yong-cheng, Chen, Gui-lin. Research on Fast High-Frequency strings Extracting and Statistics Algorithm with No Thesaurs. Journal of Chinese Information Processing, 2001.15(2):23-30. (in Chinese)

LÜ Xue-qiang. Research of E-Chunk Acquisition and Application in Machine Translation. Ph.D. dissertation, Northeastern University, Shen Yang, China, Jan, 2003.

Nagao, M., Mori,S. A New Method of N-gram Statistics for Large Number of N and Automatic Extraction of Words and Phrases from Large Text Data of Japanese. COLING-94, 1994.

David D. Palmer. A Trainable Rule-Based Algorithm for Word Segmentation. Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics, Somerset, New Jersey, 1997.

Fung P. , Wu D. Statistical Augmentation of A Chinese Mathine-Readable Dictionary. In WVLC-2,Second Annual Workshop on Very Large Corpora(COLING-94), Kyoto:Aug.94.

Ponte, J and Croft, W.B (1996). USeg: a retargetable word segmentation procedure for information retrieval. In: Symposium on document analysis and information retrieval (SDAIR '96), 1996.

Su, Keh-Yih, Tung-Hui Chiang and Jing-Shin Chang. An Overview of Corpus-Based Statistics-Oriented (CBSO) Techniques for Natural Language Processing. Intl. Journal of Computational Linguistics and Chinese Language Processing (CLCLP), vol. 1, no. 1, pp. 101-157, Taipei, August 1996.

W. J. Teahan and Yingying Wen and Rodger J. McNab and Ian H. Witten. A compression-based algorithm for Chinese word segmentation. Computational Linguistics, vol 26, No. 3, pages 375–393, 2000

YAO Tian-shun, LI Mu, GAO Wei-jun. E-Chunk Based Machine Translation in CETRAN. Proceedings of International Comference on Chinese Language Computing, Chicago, USA, July 8-9, 2000.

YÜ Shi-wen, DUAN Hui-ming,ZHU Xue-feng, SUN Bin. The Basic Processing of Contemporary Chinese Corpus at Peking University SPECIFICATION. Journal of Chinese Information Processing. Vol. 16,No. 5, Sep. 2002.