# Statistical Substring Reduction in Linear Time

**Xueqiang Lü**
Institute of Computational Linguistics
Peking University, Beijing
lxq@pku.edu.cn

**Le Zhang**
Institute of Computer Software & Theory
Northeastern University, Shenyang
ejoy@xinhuanet.com

**Junfeng Hu**
Institute of Computational Linguistics
Peking University, Beijing
Hujf@pku.edu.cn

## Abstract

We study the problem of efficiently removing equal frequency $n$-gram substrings from an $n$-gram set, formally called Statistical Substring Reduction (SSR). SSR is a useful operation in corpus based multi-word unit research and new word identification task of oriental language processing. We present a new SSR algorithm that has linear time ($O(n)$), and prove its equivalence with the traditional $O(n^2)$ algorithm. In particular, using experimental results from several corpora with different sizes, we show that it is possible to achieve performance close to that theoretically predicated for this task. Even in a small corpus the new algorithm is several orders of magnitude faster than the $O(n^2)$ one. These results show that our algorithm is reliable and efficient, and is therefore an appropriate choice for large scale corpus processing.

## 1 Introduction

Multi-word unit has received much attention in corpus oriented researches. Often the first step of multi-word unit processing is to acquire large $n$-gram set (word or character $n$-gram ) from raw corpus. Then various linguistic and statistical methods can be employed to extract multi-word units from the initial $n$-grams . (Chang, 1997) applied a two stage optimization scheme to improve the overall accuracy of an English Compound Word Extraction task. (Merkel and Andersson, 2000) used two simple statistical filters ( frequency-based and entropy-based) to remove ill-formed multi-word units (MWUs) in a terminology extraction task. (Moon and Lee, 2002) investigated the use of multi-word translation units in a Korean-to-Japanese MT system. These efforts, while varied in specifics, can all benefit from a procedure called $n$-gram *Statistical Substring Reduction*. The notation of "Statistical Substring Reduction" refers to the removal of equal frequency $n$-gram substrings from an $n$-gram set. For instance, if both $n$-grams "the people's republic" and "the people's republic of China" occur ten times in a corpus, the former should be removed from the $n$-gram set, for it being the substring of the latter $n$-gram with the same frequency. The same technique can be applied to some oriental languages (such as Chinese, Japanese, Korean etc.) of which the basic processing unit is single character rather than word. In the case of Chinese, say the two character $n$-grams "华人民共和国" and "中华人民共和国" have the same frequency in corpus, the former should be removed.

While there exists efficient algorithm to acquire arbitrary $n$-gram statistics from large corpus (Nagao and Mori, 1994), no ideal algorithm for SSR has been proposed to date. When the initial $n$-gram set contains $n$ $n$-grams , traditional SSR algorithm has an $O(n^2)$ time complexity (Han et al., 2001), and is actually intractable for large corpus. In this paper, we present a new linear time SSR algorithm.

The rest of this paper is organized as follows, Section 2 introduces basic definitions used in this paper. Section 3 presents two SSR algorithms, the latter has an $O(n)$ time complexity. This is followed in Section 4 with the mathematical proof of the equivalence of the two algorithms. Experimental results on three data sets with different sizes are reported in Section 5. We reach our conclusion in Section 6.

## 2 Preliminaries

In the rest of this paper, we shall denote by $\mathbb{N}$ the set of all integers larger than 0 and denote by $\mathbb{N}^*$ the set of all non-negative integers.

**Definition 1** *The smallest counting unit in a corpus $\mathcal{C}$ is called a "statistical unit", denoted by lowercase letters. All other symbols in $\mathcal{C}$ are called "non-statistical unit". We denote by $\Phi$ the set of all statistical units in $\mathcal{C}$ .*

Viewed in this way, a corpus $\mathcal{C}$ is just a finite sequence of statistical units and non-statistical units. When dealing with character $n$-grams , the statistical units are all characters occur in corpus $\mathcal{C}$ ; similarly, the statistical units of word $n$-grams are all words found in $\mathcal{C}$ . In previous example "中", "人", "国" are statistical units for character $n$-grams and "the", "people's", "China" are statistical units for word $n$-grams . A particular application may include other symbols in a corpus as statistical units (such as numbers and punctuations).

**Definition 2** *A string is a sequence of one or more statistical units, denoted by uppercase letters. The set of all strings is denoted by $\Psi$. If $X \in \Psi$, then there exists an integer $n \in \mathbb{N}$ such that $X = x_1x_2 \ldots x_n$, where $x_i \in \Phi, (i = 1, 2, \ldots, n)$. We denote the $i$th statistical unit in $X$ as* `Char(X,i)`. *Then* `Char(X,i)` $= X_i$. *The length of $X$ is defined to be the number of statistical units in $X$, denoted by* `Len(X)`. *If* `Len(X)=n`, *then $X$ is called an $n$-gram .*

**Definition 3** *Let $Y \in \Psi$, and $Y = y_1y_2 \cdots y_n$ $(n \in \mathbb{N}, n \geq 2)$, then any $p$ $(p \in \mathbb{N}, p < n)$ consecutive statistical units of $Y$ comprise a string $X$ that is called the substring of $Y$. Equally, we call $Y$ the super-string of $X$. We denote this relationship by $X \propto Y$. The left most $p$ consecutive statistical units of $Y$ make up of string $X_{left}$ that is called the left substring of $Y$, denoted by $X_{left} \propto_L Y$. Similarly, the right most $p$ consecutive statistical units of $Y$ constitute string $X_{right}$, the right substring of $Y$, written as $X_{right} \propto_R Y$. We use* `Left(Y,p)` *and* `Right(Y,p)` *to denote $Y$'s left substring with length $p$ and right substring with length $p$, respectively.*

**Definition 4** *For $X \in \Psi, X = x_1x_2 \ldots x_n (n \in \mathbb{N})$, if $X$ occurs at some position in the finite sequence of statistical units in $\mathcal{C}$ , we say $X$ occurs in $\mathcal{C}$ at that position, and call $X$ a statistical string of $\mathcal{C}$ . The set of all statistical strings in $\mathcal{C}$ is denoted by $\Psi^C$. Obviously, we have $\Psi^C \subset \Psi$.*

**Definition 5** *For $X \in \Psi^C$, the number of different positions where $X$ occurs in $\mathcal{C}$ is called the frequency of $X$ in $\mathcal{C}$ , denoted by $f(x)$.*

**Definition 6** *A high-frequency string is a statistical string in $\Psi^C$ whose frequency is no less than $f_0$ $(f_0 \in \mathbb{N})$. We denote by $\Psi^C_{f_0}$ the set of all high-frequency strings in $\Psi^C$. The set of all strings in $\Psi^C_{f_0}$ such that $m_1 \leq$* `Len(X)` *$\leq m_2, (m_1, m_2 \in \mathbb{N}$ and $m_1 < m_2)$ is written as $\Psi^C_{m_1m_2f_0}$. For convenience, we use $\Omega$ as a shorthand notation for $\Psi^C_{m_1m_2f_0}$. Obviously, we have $\Omega \subset \Psi^C_{f_0} \subset \Psi^C$.*

**Definition 7** *For $X, Y \in \Omega$, if $X \propto Y$ and $f(X) = f(Y)$, then we say $X$ can be reduced by $Y$, or equally, $Y$ can reduce $X$. If $X$ can be reduced by some $Y$ then we say $X$ can be reduced. Let $\Omega' = \{X \in \Omega | \exists\, Y \in \Omega, X \text{ can be reduced by } Y\}$. $\Omega_0 = \Omega \setminus \Omega'$. Then $\Omega'$ denotes the set of strings in $\Omega$ that can be reduced, $\Omega_0$ denotes the set of strings in $\Omega$ that can not be reduced. Obviously $\Omega_0 \subset \Omega$.*

**Definition 8** *An algorithm that accepts $\Omega$ as input and outputs $\Omega_0$ is a Statistical Substring Reduction algorithm.*

## 3 Two Statistical Substring Reduction Algorithms

### 3.1 An $O(n^2)$ SSR Algorithm

Suppose $|\Omega| = n$, then $\Omega$ has $n$ statistical strings. The $i$th $(1 \leq i \leq n)$ statistical string in $\Omega$ can be represented as a 3-tuple $< X_i, f_i, M_i >$, where $X_i$ denote the $i$th statistical string, $f_i = f(X_i)$ is the frequency of $X_i$ in corpus $\mathcal{C}$ and $M_i$ is a merging flag. $M_i = 0$ means $X_i$ is not reduced and $M_i = 1$ indicates $X_i$ being reduced by its super-string. The initial value of all $\{M_i\}'s$ are set to 0. The first SSR algorithm is given in Algorithm 1.

**Algorithm 1** An $O(n^2)$ Statistical Substring Reduction Algorithm

Input: $\Omega$
Output: $\Omega_0$
**for** $i = 1$ to $n$ **do**
  **for** $j = 1$ to $n$ **do**
    **if** $X_i \propto X_j$ and $f_i = f_j$ **then**
      $M_i = 1$
**for** $i = 1$ to $n$ **do**
  **if** $M_i = 0$ **then**
    output $X_i$

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

### 3.2 An $O(n)$ SSR Algorithm

Algorithm 1 tries to find a string's super-strings by comparing it with all strings in $\Omega$. Since only a small portion of strings in $\Omega$ can be potential super-strings of any given string, a great deal of time will be saved if we restrict the searching space to the possible super-string set. Based on this motivation we now describe a faster SSR algorithm.

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

**Definition 9** *Let* $X \in \Psi$, $X = x_1 x_2 \ldots x_n (n \in \mathbb{N})$, *then* $X_R = x_n x_{n-1} \ldots x_1$ *is called the reversed string of* $X$. *All reversed strings of statistical units in* $\Omega$ *comprise the reversed string set* $\Omega_R$. `Reverse(X)` *returns the reversed string of* $X$.

**Algorithm 2** An $O(n)$ Statistical Substring Reduction Algorithm

1: Input: $\Omega$
2: Output: $\Omega_0$
3: sort all statistical strings in $\Omega$ in ascending order according to $X_i$'s value
4: **for** $i = 1$ to $n - 1$ **do**
5:   **if** $X_i \propto_L X_{i+1}$ and $f_i = f_{i+1}$ **then**
6:     $M_i = 1$
7: **for** $i = 1$ to $n$ **do**
8:   $X_i$=`Reverse(`$X_i$`)`
9: sort all statistical strings in $\Omega$ 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 $f_i = f_{i+1}$ **then**
12:     $M_i = 1$
13: **for** $i = 1$ to $n$ **do**
14:   $X_i$=`Reverse(`$X_i$`)`
15:   **if** $M_i = 0$ **then**
16:     output $X_i$

In this algorithm, all steps have a time complexity of $O(n)$ except step 3 and 9, which perform sorting on $n$ statistical strings. It is worth mention that sorting can be implemented with `radix sort`, an $O(n)$ operation, therefore this algorithm has an ideal time complexity of $O(n)$. For instance, if the maximum length of statistical unit in $\Omega$ is $m$, we can perform a `radix sort` by an m-way statistical unit collection (padding all strings to length $m$ with empty statistical unit). 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 reduction* and *right reduction* of algorithm 2. In algorithm 1, each string must be compared with all strings in $\Omega$ whereas in algorithm 2, each string is only required to be compared with two strings. This is why algorithm 2 reduces the number of comparison tremendously compared to algorithm 1.

## 4 The equivalence of the Two Algorithms

While it is obvious to see that algorithm 1 is an SSR algorithm, it is unclear how can algorithm 2 have the same function, despite its lower time complexity. In this section we will give a mathematical proof of the equivalence of the two algorithms: they yield the same output given the same input set (not considering element order).

For a given corpus $\mathcal{C}$ , $\Phi$ is a finite set, the finity of which is determined by the finity of $\mathcal{C}$

. Since any two statistical units can be assigned an ordering (either by machine code representation or specified manually) such that the two statistical units are ordered from less to greater one. We can denote this ordering by $\preceq$. It is obvious that this ordering satisfies reflexivity, antisymmetry and transitivity. For any given $a, b \in \Phi$, either $a \preceq b$ or $b \preceq a$ holds, therefore $< \Phi, \preceq>$ is a finite well-ordered set. Here we introduce the symbol $\prec$ and write the condition $a \neq b$ and $a \preceq b$ as $a \prec b$.

**Definition 10** *For $X, Y \in \Psi, X = x_1x_2 \dots x_n (n \in \mathbb{N}), Y = y_1y_2 \dots y_m (m \in \mathbb{N})$. If $m = n$ and $\forall i (1 \leq i \leq m)$ such that $x_i = y_i$, then we say $X$ is equal to $Y$, denoted by $X = Y$. If $X \propto_L Y$, or $\exists\ p(1 \leq p \leq \min(n, m))$ such that $x_1 = y_1, x_2 = y_2, \dots, x_{p-1} = y_{p-1}$ and $X_p \prec Y_p$, then we say $X$ is less than $Y$. Whenever it is clear from context it is denoted by $X \prec Y$. If either $X = Y$ or $X \prec Y$ then we write $X \preceq Y$.*

Under these definitions it is easy to check that $< \Psi, \preceq>$ , $< \Psi^C, \preceq>$, $< \Omega, \preceq>$ and $< \Omega_R, \preceq>$ are all well-ordered sets.

**Definition 11** *Suppose $X, Y \in \Omega$ (or $\Omega_R$), and $X \prec Y$, $\forall Z \in \Omega$ (or $\Omega_R$) whenever $X \prec Z$ we have $Y \prec Z$. Then we say $X$ is the proceeder of $Y$ in $\Omega$ (or $\Omega_R$) and $Y$ is the successor of $X$ in $\Omega$ (or $\Omega_R$)*

Algorithm 1 compares current statistical string ($X_i$) to all statistical strings in $\Omega$ in order to decide whether the statistical string can be reduced or not. By comparison, algorithm 2 only compares $X_i$ with its successors in $\Omega$ (or $\Omega_R$) to find its super-strings.

The seemingly *in-equivalence* of the two algorithms can be illustrated by the following example: Suppose we have the following four statistical strings with `f(X1)=f(X1)=f(X3)=f(X4)=`$f_0$:

| | |
|---|---|
| X1="中华人民共和国" | (the people's republic of China) |
| X2="华人民共和国" | (people's republic of China) |
| X3="中华人民共和" | (the people's republic of) |
| X4="人民共" | (people's republic) |

According to definition 7, $X_2, X_3, X_4$ will all be reduced by $X_1$ in algorithm 1. In algorithm2, $X_2$ is the right substring of $X_1$, it will be reduced by $X1$ in right reduction. Similarly, $X_3$ can be reduced by $X_1$ in left reduction for being left substring of $X_1$. However, $X_4$ is neither the left substring of $X_1$ nor $X_1$'s right substring. It will not be reduced *directly* by $X_1$ in algorithm 2. As a matter of fact, $X_4$ will be reduced *indirectly* by $X_1$ in algorithm 2, the reason of which will be explained soon.

To prove the equivalence of algorithm 1 and 2, the following lemmas need to be used. Because of the space limitation, the proofs of some lemmas are omitted.

**Lemma 1** *If $X \in \Psi$ and $X \propto Y \in \Psi^C$ then $X \in \Psi^C$ and $f(X) \geq f(Y)$.*

Explanation: a statistical string's substring is also a statistical string, whose frequency is no less than its super-string's.

**Lemma 2** *For $X, Z \in \Omega$, $Y \in \Psi$. If $X \propto Y \propto Z$ and $f(X) = f(Z)$ then $f(Y) = f(X) = f(Z)$ and $Y \in \Omega$.*

**Proof:** Since $Y \in \Psi, Y \propto Z \in \Omega \subset \Psi^C$. by Lemma 1 we have $Y \in \Psi^C$ and $f(Y) \geq f(Z)$. Considering $X \in \Omega \subset \Psi, X \propto Y \in \Psi^C$, by Lemma 1 we get $f(X) \geq f(Y)$. Since $f(X) = f(Z)$ it follows that $f(Y) = f(X) = f(Z)$. Moreover $X, Z \in \Omega$, by definition 6 we get $m_1 \leq$ `Len(X)` $\leq m_2, m_1 \leq$ `Len(Z)` $\leq m_2$ and $f(X) \geq f_0, f(Y) \geq f_0$. Considering $X \propto Y \propto Z$, from definition 3, we conclude that `Len(X) < Len(Y) < Len(Z)`. Therefore $m_1 <$ `Len(Y)` $< m_2$. Since $f(Y) = f(X) = f(Z) \geq f_0$. From definition 6 $Y \in \Omega$. □

Lemma 2 is the key to our proof. It states that the substring sandwiched between two equal frequency statistical strings must be a statistical string with the same frequency. In the above example both "中华人民共和国" and "人民共" occur $f_0$ times. By Lemma 2 "华人民共和国", "中华人民共和" and all other string sandwiched between $X_1$ and $X_4$ will occur in $\Omega$ with the frequency of $f_0$. Therefore $X_4 =$"人民共" can be reduced by $X_1$="中华人民共和国" *indirectly* in algorithm 2.

**Lemma 3** *If $X, Y, Z \in \Omega$. $X \prec Y \prec Z$ and $X \propto_L Z$ then $X \propto_L Y$.*

**Lemma 4** *If $X, Y \in \Omega$ (or $\Omega_R$), $X \propto_L Y$,* `Len(X)+1=Len(Y)`, $f(X) = f(Y)$*; then* $Y$ *is* $X$*'s successor in* $\Omega$ *or (*$\Omega_R$*).*

**Lemma 5** *If* $X, Y \in \Omega$ *and* $X \propto_R Y$ *then* $X_R, Y_R \in \Omega_R$ *and* $X_R \propto_L Y_R$*.*

**Lemma 6** *If* $X, Y \in \Omega, X \propto Y$, `Len(X)+1=Len(Y)`, $f(X) = f(Y)$ *then X will be reduced in algorithm 2.*

**Lemma 7** *If* $X, Y \in \Omega, X \propto Y, f(X) = f(Y)$*, then* $X$ *will be reduced in algorithm 2.*

**Proof:** If `Len(X)+1=Len(Y)` the result follows immediately after applying Lemma 6. We now concentrate on the situation when `Len(Y)>Len(X)+1`. Let $X = x_1x_2 \ldots x_n (n \in \mathbb{N})$. Since $X \propto Y$, from definition 3 there exists $k, m \in \mathbb{N}^*$, which can not be zero at the same time, such that $Y = y_1y_2 \ldots y_kx_1x_2 \ldots x_nz_1z_2 \ldots z_m$. If $k \neq 0$, let $M = y_kx_1x_2 \ldots x_n$; if $m \neq 0$, let $M = x_1x_2 \ldots x_nz_1$. In any case we have `Len(X) + 1 = Len(M) <Len(Y)`. Considering $X, Y \in \Omega, X \propto M \propto Y, f(X) = f(Y)$, by Lemma 2 we have $M \in \Omega$ and $f(M) = f(X)$, therefore `Len(X) + 1 = Len(M)`, by Lemma 6 $X$ will be reduced in algorithm 2. □

Now we arrive the main result of this paper:

**Theorem 1** *Algorithm 1 and 2 are equivalent, that is: given the same input* $\Omega$ *they both yield the same output* $\Omega_0$*.*

**Proof:** Suppose $X \in \Omega$. If $X$ can be reduced in algorithm 2, obviously $X$ can also be reduced in algorithm 1. If $X$ can be reduced in algorithm 1, then there exists a $Y \in \Omega$ such that $X \propto Y, f(X) = f(Y)$. By Lemma 7 $X$ will be reduced in algorithm 2. So given the same $\Omega$ as input, the two algorithms will output the same $\Omega_0$. □

## 5 Experiment

To measure the performances of the two SSR algorithms we conduct experiments on three Chinese corpora with different sizes (table 1). We first extract 2-20 $n$-grams from these raw corpora using Nagao's algorithm. In our experiments, the high-frequency threshold is chosen to be $f_0 = \lfloor \log_{10} n \rfloor$, and $n$ is the total number of characters in corpus, discarding all $n$-grams with frequency less than $f_0$. Then we run the two SSR algorithms on the initial $n$-gram set $\Omega$ and record their running times (not including I/O operation). All results reported in this paper are obtained on a PC with a single PIII 1G Hz CPU running GNU/Linux. Table 2 summarizes the results we obtained[1].

We can make several useful observations from table 2. First, the SSR algorithm does reduce the size of $n$-gram set significantly: the reduced $n$-gram set $\Omega_0$ is 30% - 35% smaller than $\Omega$, conforming the hypothesis that a large amount of initial $n$-gram set are superfluous "garbage substrings". Second, the data in table 2 indicates that the newly proposed SSR algorithm is vastly superior to algorithm 1 in terms of speed: even in small corpus like corpus1 the speed of algorithm 2 is 1500 times faster. This difference is not surprising. Since algorithm 1 is an $O(n^2)$ algorithm, it is infeasible to handle even corpus of modest size, whereas the algorithm 2 has an ideal $O(n)$ time complexity, making even very large corpus tractable under current computational power: it takes less than five minutes to reduce a 2-20 $n$-gram set from corpus of 1 Giga bytes.

## 6 Conclusion

Ever since the proposal of Nagao's $n$-gram extraction algorithm, the acquisition of arbitrary $n$-gram statistics is no longer a problem for large scale corpus processing. However, the fact that no efficient SSR algorithm has been proposed to deal with redundant $n$-gram substrings in the initial $n$-gram set has prevented statistical substring reduction from being used widely. Actually, almost all researches involving large $n$-gram acquisition (statistical lexicon acquisition, multi-word unit research, lexicon-free word segmentation, to name just a few) can benefit from SSR operation. We have shown that a simple fast SSR algorithm can effectively remove up to 30% useless $n$-gram substrings. SSR algorithm can also combine with other filtering methods to improve filter accuracy. In a Chinese multi-word unit acquisition task, a combined filter with fast SSR operation and simple mutual information achieved good accuracy (Zhang et al., 2003). In the future, we would like

[1]We did not run Algo 1 on corpus 3 for it is too large to be efficiently handled by algorithm 1.

| 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, politics etc.) | 1GB | 520 million |

Table 1: Summary of the three corpora

| Label | $m_1$ | $m_2$ | $f_0$ | $\|\Omega\|$ | $\|\Omega_0\|$ | Algo 1 | Algo 2 |
|---|---|---|---|---|---|---|---|
| corpus1 | 2 | 20 | 6 | 110890 | 75526 | 19 min 20 sec | **0.82 sec** |
| corpus2 | 2 | 20 | 7 | 1397264 | 903335 | 40 hours | **14.87 sec** |
| corpus3 | 2 | 20 | 8 | 19737657 | 12888632 | N/A | **185.87 sec** |

Table 2: 2 - 20-gram statistical substring reduction results.

to explore the use of SSR operation in bilingual multi-word translation unit extraction task.

In this paper, a linear time statistical substring reduction algorithm is presented. The new algorithm has an ideal $O(n)$ time complexity and can be used to rule out redundant $n$-gram substrings efficiently. Experimental result suggests the fast SSR algorithm can be used as an effective preprocessing step in corpus based multi-word research.

## Acknowledgements

This research was funded by a grant from the National High Technology Research and Development Program (863 Program, Beijing, No. 2002AA117010-08), a grant from the National Natural Science Foundation (Beijing, No. 60083006) and a grant from the Major State Basic Research Development Program (973 Program, Beijing, No. G199803050111).

## References

Jing-Shin Chang. 1997. *Automatic Lexicon Acquisition and Precision-Recall Maximization for Untagged Text Corpora*. Ph.D. thesis, National Tsing-Hua University, National Tsing-Hua University Hsinchu, Taiwan 300, ROC.

Kesong Han, Yongcheng Wang, and Guilin Chen. 2001. Research on fast high-frequency extracting and statistics algorithm with no thesaurs. *Journal of Chinese Information Processing (in Chinese)*.

M. Merkel and M. Andersson. 2000. Knowledge-lite extraction of multi-word units with language filters and entropy thresholds.

Kyonghi Moon and Jong-Hyeok Lee. 2002. Translation of discontinuous multi-word translation units in a korean-to-japanese machine translation system. *International Journal of Computer Processing of Oriental Languages*, 15(1):79–99.

Makoto Nagao and Shinsuke Mori. 1994. 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. In *COLING-94*.

Le Zhang, Xueqiang LÜ, Yanna Shen, and Tianshun Yao. 2003. A statistical approach to extract chinese chunk candidates from large corpora. In *Proceeding of 20th International Conference on Computer Processing of Oriental Languages (ICCPOL03)*, pages 109–117.