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

Makoto Nagao, Shinsuke Mori
Department of Electrical Engineering
Kyoto University

Abstract

In the process of establishing the information theory, C. E. Shannon proposed the Markov process as a good model to characterize a natural language. The core of this idea is to calculate the frequencies of strings composed of n characters (n-grams), but this statistical analysis of large text data and for a large n has never been carried out because of the memory limitation of computer and the shortage of text data. Taking advantage of the recent powerful computers we developed a new algorithm of n-grams of large text data for arbitrary large n and calculated successfully, within relatively short time, n-grams of some Japanese text data containing between two and thirty million characters. From this experiment it became clear that the automatic extraction or determination of words, compound words and collocations is possible by mutually comparing n-gram statistics for different values of n.

category: topical paper, quantitative linguistics, large text corpora, text processing

1 Introduction

Claude E. Shannon established the information theory in 1948 [1]. His theory included the concept that a language could be approximated by an n-th order Markov model by n to be extended to infinity. Since his proposal there were many trials to calculate n-grams (statistics of n character strings of a language) for a big text data of a language. However computers up to the present could not calculate them for a large n because the calculation required huge amount of memory space and time. For example the frequency calculation of 10-grams of English requires at least $26^{10} \approx 10^5 \sim 10^6$ giga word memory space. Therefore the calculation was done at most for $n = 4 \sim 5$ with modest text quantity.

We developed a new method of calculating n-grams for large n's. We do not prepare a table for an n-gram. Our methods consists of two stages. The first stage performs the sorting of substrings of a text and finds out the length of the prefix parts which are the same for the adjacent substrings in the sorted table. The second stage is the calculation of an n-gram when it is asked for a specific n. Only the existing n character combinations require the table entries for the frequency count, so that we need not reserve a big space for n-gram table. The program we have developed requires $7l$ bytes for an l character text of two byte code such as Japanese and Chinese texts and $6l$ bytes for an l character text of English and other European languages. By the present program n can be extended up to 255. The program can be changed very easily for larger n if it is required.

We performed n-gram frequency calculations for three different text data. We were not so much interested in the entropy value of a language but were interested in the extraction of varieties of language properties, such as words, compound words, collocations and so on. The calculation of frequency of occurrences of character strings is particularly important to determine what is a word in such languages as Japanese and Chinese where there is no spaces between words and the determination of word boundaries is not so easy. In this paper we will explain some of our results on these problems.

2 Calculation of n-grams for an arbitrary large number of n

It was very difficult to calculate n-grams for a large number of n because of the memory limitation of a computer. For example, Japanese language has more than 4000 different characters and if we want

to have 10-gram frequencies of a Japanese text, we must reserve 4000^{10} entries, which exceed 10^{35}. Therefore only 3 or 4-grams were calculated so far.

A new method we developed can calculate n-grams for an arbitrary large number of n with a reasonable memory size in a reasonable calculation time. It consists of two stages. The first stage is to get a table of alphabetically sorted substrings of a text string and to get the value of coincidence number of prefix characters of adjacently sorted strings. The second stage is to calculate the frequency of n-grams for all the existing n character strings from the sorted strings for a specific number of n.

2.1 First stage

(1) When a text is given it is stored in a computer as one long character string. It may include sentence boundaries, paragraph boundaries and so on if they are regarded as components of text. When a text is composed of l characters it occupies $2l$ byte memory because a Japanese character is encoded by 16 bit code. We prepare another table of the same size (l), each entry of which keeps the pointer to a substring of the text string. This is illustrated in Figure 1.

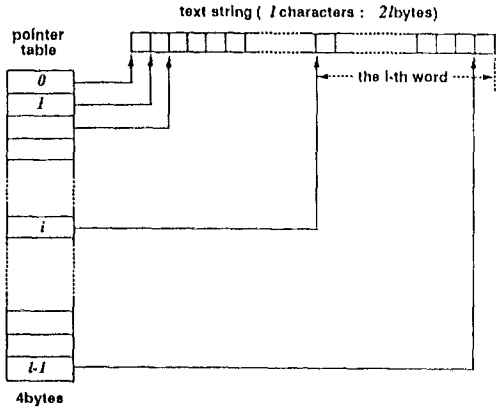

Figure 1: Text string and the pointer table to substrings.

A substring pointed by i-1 is defined as composed of the characters from the i-th position to the end of the text string (see Figure 1). We call this substring a word. The first word is the text string itself, and the second word is the string which starts from the second character and ends at the final character of the text string. Similarly the last word is the final character of the text string.

As the text size is l characters a pointer must have at least p bits where $2^p \geq l$. In our program we set $p = 32$ bits so that we can accept the text size up to $2^{32} \approx 4$ giga characters. The pointer table represents a set of l words.

We apply the dictionary sorting operation to this set of l words. It is performed by utilizing the pointers in the pointer table. We used comb sort[2] which is an improved version of bubble sort. The sorting time is the order of $O(l \log l)$. When the sorting is completed the result is the change of pointer positions in the pointer table, and there is no replacement of actual words. As we are interested in n-grams of n less than 255, actual sorting of words is performed for the leftmost 255 or less characters of words.

(2) Next we compare two adjacent words in the pointer table, and count the length of the prefix parts which are the same in the two words. For example when "extension to the left side ..." and "extension to the right side ..." are two words placed adjacent, the number is 17. This is stored in the table of coincidence number of prefix characters. This is shown in Figure 2. As we are interested in $1 \leq n \leq 255$, one byte is given to an entry of this table. The total memory space required to this first stage operation is $2l+4l+l = 7l$ bytes. For example when a text size is 10 mega Japanese characters, 70 mega byte memory must be reserved. This is not difficult by the present-day computers.

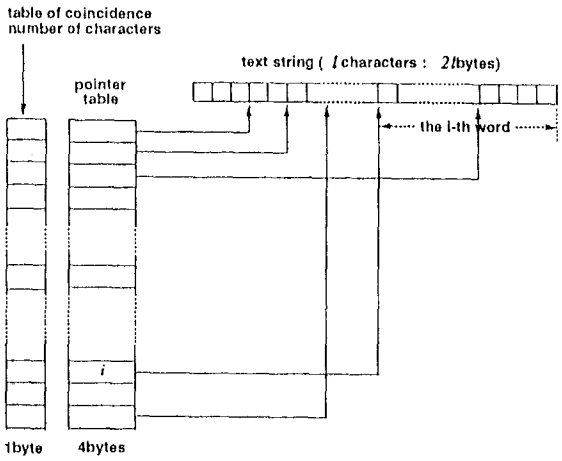

Figure 2: Sorted pointer table and table of coincidence number of characters

We developed two software versions, one by using main memory alone, and the other by using a disc memory where the software has the additional operations of disc merge sort. By the disc version we can handle a text of more than 100 mega character Japanese text. The software was implemented on a

SUN SPARC Station.

2.2 Second stage

The second stage is the calculation of n-gram frequency table. This is done by using the pointer table and the table of coincidence number of prefix characters. Let us fix n to a certain number. We first read out the first n characters of the first word in the pointer table, and see the number in the table of coincidence number of prefix characters. If this is equal to or larger than n it means that the second word has at least the same n prefix characters with the first word. Then we see the next entry of the coincidence number of prefix characters and check whether it is equal to or larger than n or not. We continue this operation until we meet the condition that the number is smaller than n. The number of words checked up to this is the frequency of the n prefix characters of the first word. At this stage the first n prefix characters of the next word is different, and so the same operation as the first n characters is performed from here, that is, to check the number in the coincidence number of prefix characters to see whether it is equal to or larger than n or not, and so on. In this way we get the frequency of the second n prefix characters. We perform this process until the last entry of the table. These operations give the n-gram table of the given text. We do not need any extra memory space in this operation when we print out every n-gram string and its frequency when they are obtained.

We calculated n-grams for some different Japanese texts which were available in electronic form in our laboratory. These were the followings.

1. Encyclopedic Dictionary of Computer Science (3.7 M bytes)
2. Journalistic essays from Asahi Newspaper (8 M bytes)
3. Miscellaneous texts available in our laboratory (59 M bytes)

The first two texts were not large and could be managed in the main memory. The third one was processed by using a disc memory by applying a merge sort program three times. The first two texts were processed within one and two hours by a standard SUN SPARC Station for the first stage mentioned above. The third text required about twenty four hours. Calculation of n-gram frequency (the second stage) took less than an hour including print-out.

3 Extraction of useful linguistic information from n-gram frequency data

3.1 Entropy

Everybody is interested in the entropy value of a language. Shannon's theory says that the entropy is calculated by the formula [3]

$$H_n(L) = \sum P(w) \log P(w)$$

where $P(w)$ is the probability of occurrence of w, and the summation is for all the different strings w of n characters appearing in a language. The entropy of a language L is

$$H(L) = \lim_{n \to \infty} H_n(L)$$

We calculated $H_n(L)$ for the texts mentioned in Section 2 for $n = 1, 2, 3, \ldots$ The results is shown in Figure 3. Unlike our initial expectation that the entropy will converge to a certain constant value between 0.6 and 1.3 which C. E. Shannon estimated for English, it continued to decrease to zero. We checked in detail whether our method had something wrong, but there was nothing doubtful. Our conclusion for this strange phenomenon was that the text quantity of a few mega characters were too small to get a meaningful statistics for a large n because we have more than 4000 different characters in the Japanese language. For English and many other European languages which have alphabetic sets of less than fifty characters the situation may be better. But still the text quantity of a few giga bytes or more will be necessary to get a meaningful entropy value for $n = 10$ or more.

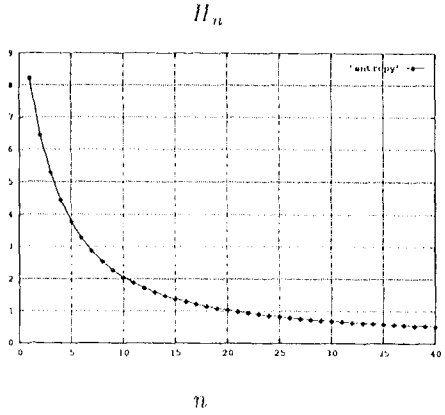

Figure 3: Entropy curve by n-gram

3.2 Obtaining the longest compound word

From the n-gram frequency table we can get many interesting information. When we have a string w (length n) of high frequency as shown in Figure 4, we can try to find out the longest string w' which includes w by the following process by using the n-gram frequency table.

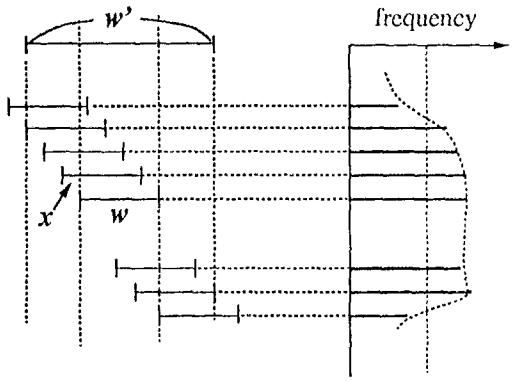

Figure 4: Obtaining the longest word w' from a high frequency word fragment w

(1) extension to the left: We cut off the last character of w and add a character x to the left of w. We call this a cut-and-pasted word. We look for the character x which will give the maximum frequency to the cut-and-pasted word. Repeat the same operation step by step to the left and draw a frequency curve for these words. This operation will be stopped when the frequency curve drops to a certain value. This process is performed by seeing the n-gram frequency table alone.

(2) extension to the right: The same operation as (1) is performed by cutting the left character and adding a character to the right.

(3) extraction of high frequency part: From the frequency curve as shown in Figure 4 we can easily extract a high frequency part as the longest string. An example is shown in Figure 5

The strings extracted in this way are very often compound words of postpositions in Japanese. Postpositional phrases are usually composed of one to three words, and are used as if they are compound postpositions. Some extracted examples are,

partial strings	frequencies
定すること	101
すること	1689
ることが	1310
ことがで	784
とができ	784
ができる	770
できるよ	147

Figure 5: Frequencies of partial strings and obtaining the longest word "することができる"

... しなければならない (must do ...)
... ことが知られている (it is known that ...)
... 行なうことができる (can do ...)
... 求めることができる (can ask ...)

3.3 Word extraction

After getting high frequency character strings by the above method we can make consultations with dictionaries for these strings. Then we find out many strings which are not included in the dictionaries.

Some are phrases(collocations, idiomatic expressions), some others are terminology words, and unknown (new) words. From the text data of Encyclopedic Dictionary of Computer Science we extracted many terminological words. In general the frequencies of n-grams become smaller as n becomes larger. But we had sometimes relatively high frequency values in n-grams of large n's. These were very often terminological words or terminological phrases. We extracted such terminological phrases as,

- (...) 言語で書かれたプログラム
  (programs written by (...) language)
- 人工知能における問題解決
  (problem solving in artificial intelligence)
- ページ置換アルゴリズム
  (page replacement algorithm)
- プログラムの部分的正当性
  (partial correctness of programs)

3.4 Compound word

We can get more interesting information when we compare data of different n's. When we have a character string (length n) of high frequency, which we may be able to define as a word (w), we are recommended to check whether two substrings (w_1 and w_2) of the length n_1 and n_2 ($n_1 + n_2 = n$) as

Table 1: Determination of compound word

Compound word		proper segmentation	improper segmentation
記憶装置 (280)	=	記憶 (1545)・装置 (1540)	記憶装 (280), 憶装置 (280), 憶装 (280)
情報処理 (166)	=	情報 (2058)・処理 (2698)	情報処 (166), 報処理 (166), 報処 (166)
集積回路 (188)	=	集積 (242)・回路 (1350)	集積回 (188), 積回路 (188), 積回 (188)

( ):frequency in Encyclopedic Dictionary of Computer Science

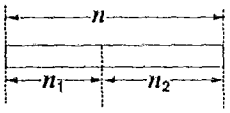

Figure 6: Possible segmentation of a word into two components

shown in Figure 6 have high frequency appearance in n_1-gram and n_2-gram tables. If we can find out such a situation by changing n_1 (and n_2) we can conclude that the original character string w is a compound word of w_1 and w_2. Some examples are shown in Table 1.

3.5 Collocation

We can see whether a particular word w has strong collocational relations with some other words from the n-gram frequency results. We can get an n-gram table where n is sufficiently large, w is the prefix of these n-grams, and some words (w', w'', ...) may appear in relatively high frequency. This is shown in Figure 7. We can find out easily that $w - w'$ and $w - w''$ are two allocational expressions from this figure. For example we have 「影響」(effect) and find out that 「影響を受ける」(receive effect) and 「影響を与える」(give effect) have relatively high frequencies and there are no other significant combinations in the n-gram table with 「影響」as the prefix. 「入退院」(in and out hospital) have almost all the time 「を繰り返す」(repeat) as the following phrase, and so we will be able to judge that 「入退院を繰り返す」is an idiomatic expression.

4 Conclusions

We developed a new method and software for n-gram frequency calculation for n up to 255, and calculated n-grams for some large text data of Japanese. From these data we could derive words, compound words and collocations automatically.

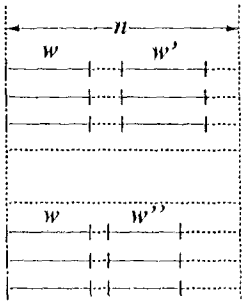

Figure 7: Finding collocational word pairs $w - w'$ and $w - w''$

We think that this method is equally useful for languages like Chinese where there is no word spaces in a sentence, and for European languages as well, and also for speech phoneme sequences to get more detailed HMM models.

Another possibility is that when we get a large text data with part-speech tags, we can extract high frequency part-of-speech sequences by this n-gram calculation over the part-of-speech data. These may be regarded as grammar rules of the primary level. By replacing these part-of-speech sequences by single non-terminal symbols we can calculate new n-grams, and will be able to get higher level grammar rules. These examples indicate that large text data with varieties of annotations are very important and valuable for the extraction of linguistic information by calculating n-grams for larger value of n.

References

[1] C. E. Shannon: A mathematical theory of communication, Bell System Tech.J., Vol.27, pp.379-423, pp.623-656, (1948).

[2] Stephen Lacey, Richard Box: Nikkei BYTE, November, pp.305-312, (1991).

[3] N. Abramson: Information theory and coding, McGraw Hill, (1963).