Eighth International Conference on Intelligent Systems Design and Applications

Suffix Tree Based Approach for Chinese Information Retrieval

Jin Hu Huang and David Powers
School of Computer Science, Engineering and Mathematics
Flinders University of South Australia
SA 5042, Australia
{jin.huang, powers}@flinders.edu.au

Abstract

With the widespread of the Internet, great research interests are being shown in Chinese language information retrieval in recent years. The absence of word boundaries in Chinese language makes Chinese information retrieval(IR) different to European IR. In order to apply traditional IR approaches to Chinese language, sentences have to be segmented into words first. Word segmentation is playing a key role in Chinese IR. As word segmentation is not straightforward and the results are sometime ambiguous, n-grams are used as an alternative. Several experimental studies have been conducted to compare words and n-grams[5, 6], word segmentation and its effect on information retrieval[3]. These studies show that using either words or n-grams leads to comparable performances. Higher word segmentation accuracy does not necessarily result in better retrieval performance. In this paper we propose a suffix tree based approach for Chinese information retrieval without word segementation.

1. Introduction

The number of electronic documents other than European languages available in the Internet is growing enormously. Traditional information retrieval systems for European languages such as English use words as indexing units can not apply directly to Asian languages such as Chinese because English text is written with delimiters and words can be easily recognized. Chinese text is written as continuous strings of ideographs (or characters). Thus there is major difference between Chinese IR and IR in European language. A pre-processing called segmentation has to be done to determine the boundaries of words before traditional IR approaches based on words can be adapted to Chinese language. Because text segmentation is not straightforward and the process itself can have ambiguous outcomes, n-grams are used as an alternative indexing units instead of words. Several studies have been carried out to compare these two kinds of indexing approaches. It turns out that IR based on words gives slightly better results than bi-grams. Further studies have revealed that the accuracy of word segmentation have an impact on IR performance but higher word segmentation accuracy does not necessarily result in better retrieval performance. In this paper, first we will introduce to use suffix arrays to compute term frequency and document frequency for all n-grams in documents. Then we propose a filter algorithm to limit the size of n-grams used for indexes. Finally we present results on information retrieval conducted on TREC Chinese corpus.

2 Chinese Information Retrieval

For Chinese IR systems, choosing what kind of indexing units is more problematic than those that dealing with European languages only.

2.1 Character-based Indexing

Typical character-based indexing uses single Chinese characters as index terms. In Chinese language, the Chinese character is the element unit of their writing system. The definition of Chinese character does not cause any controversy because visually a character is an isolated symbol. Although the majority of Chinese words are compound words consisting of free or bound morphemes, the meanings of most compound words can be derived from the meanings of their constituents. Each character has its own semantic and syntactic properties. It can ensure no information loss and is quite easy to implement. Any document contains the same characters as the query will be retrieved. It causes high recall but low precision as some single characters are polysemantic and homonymic.

2.2 N-grams-based Indexing

This indexing method uses chunks of n consecutive characters as the index term. Neither dictionary nor other linguistic knowledge is required in the processing. Bi-grams have been often used as indexing terms form Chinese IR as most Chinese words are composed of two characters. In addition to the ease of word identification, bi-grams bear more semantic information than single characters. Bi-gram indexing is exhaustive and avoids the difficult problem of word segmentation. Bi-grams can consider unknown words and abbreviations in a better way than words do. The drawbacks of using bi-grams as indexes are meaningless character chunks are abundant among bi-grams, leading to noisy matching between queries and documents.

2.3 Word-based Indexing

Using single characters and n-grams as index terms makes it difficult to incorporate linguistic knowledge into the retrieval processing because both of them are not ideal conceptual units. In this aspect, words are better index candidates.

As mentioned above Chinese words are not readily recognizable because Chinese orthography fails to represent word boundaries. Therefore, it is necessary and important that word segmentation has to be carried out to break the original Chinese text into a series of words. Segmentation of Chinese text into words is a very difficult task. It requires linguistic knowledge and coverage of dictionaries. Many characters form one-character words by themselves, but these characters can also form multi-character words when used with other characters. Chinese words have variable lengths, the same character may occur in many different words. The question of what constitutes a Chinese "word" cannot be answered without any controversy. There is poor agreement on word segmentation amongst human annotators and at least three relative widespread conventions (Penn Treebank, China, Taiwan).

1. The segmentation guidelines for the Penn Chinese Treebank [10]
2. The guidelines for the Beijing University Institute of Computational Linguistics Corpus [12]
3. The ROCLING standard developed at Academia Sinica in Taiwan [7].

2.4 Retrieval Models

A retrieval model specifies how the content of a document and user's information need is represented in an IR system, how the documents and the information needs are matched so the relevant items can be retrieved. We used a vector space model, which view documents and queries as vectors in an n-dimension vector space and use distance as a measure of similarity.

Suppose there is a document D_i in collection D and a query Q_j, then the vector D_i and Q_j can be represented respectively as follows:

$$D_i = (d_{i1} \cdots d_{im}) \tag{1}$$

$$Q_j = (q_{j1} \cdots q_{jm}) \tag{2}$$

where d_{ik} is the weight of term t_k in the document D_i, q_{ik} is the weight of term t_k in the query Q_i,and m is the size of the vector space (the number of different terms, words or ngrams). Our weighting scheme is same as the SMART system [1]. The weight d_{ik} of a term in a document is calculated according to its occurrence frequency in the document (term frequency) and its distribution in the entire collection. We used the following formula as the Smart system.

$$d_{ik} = \frac{(\log(f_{ik}) + 1.0) * \log(\frac{N}{n_k})}{\sqrt{\sum_j((\log(f_{jk}) + 1.0) * \log(\frac{N}{n_k}))^2}} \tag{3}$$

where f_{ik} is the occurrence frequency of the term t_k in the document D_i, N is the total number of documents in the collection. n_k is the number of documents that contain the term t_k.

Similary between D_i and Q_j is calculated as the inner product of their vectors as following:

$$Sim(D_i, Q_j) = \sum_k (d_{ik} * q_{jk}) \tag{4}$$

3 Suffix Trees and Arrays

A suffix tree[8] is a data-structure that allows to solve many problems on strings. If $str = s_1 s_2 \ldots s_i \ldots s_n$ is a string, then $Si = s_i s_{i+1} \ldots s_n$ is the suffix of str that starts at position i to the end of the string, e.g. $str =$ "$to\_be\_or\_not\_to\_be$" illustrated in Table 1.

If the suffixes are sorted, some of them may share common prefixes shown in Table 2. These prefixes share a common path from the root as in a PATRICIA tree. Thus the sorted suffixes can be represented by a Trie-like or PATRICIA-like data structure called suffix tree. A given suffix tree can be used to search for a substring, $substr[1..m]$ in $O(m)$ time. There are $n(n+1)/2$ substrings in $str[1..n]$. A substring must be a prefix of a suffix of str, if it occurs in str.

Suffix arrays provide the same function as suffix trees and occupy much less space. A suffix array is simply an array containing all the pointers to the suffixes of a text

Position	0	1	2	3	4	5	6	7	8
Characters	t	o	_	b	e	_	o	r	_
Position	9	10	11	12	13	14	15	16	17
Characters	n	o	t	_	t	o	_	b	e

Suffix Array	Indexes	S_i	Suffixes
s[0]	0	S_0	to_be_or_not_to_be
s[1]	1	S_1	o_be_or_not_to_be
s[2]	2	S_2	_be_or_not_to_be
s[3]	3	S_3	be_or_not_to_be
⋮	⋮	⋮	⋮
s[13]	13	S_{13}	to_be
s[14]	14	S_{14}	o_be
s[15]	15	S_{15}	_be
s[16]	16	S_{16}	be
s[17]	17	S_{17}	e

Table 1. Suffixes and suffix arrays before sorting

Array	i	S_i	Suffixes	Lcp	v
s[0]	15	S_{15}	_be	lcp[0]	0
s[1]	2	S_2	_be_or_not_to_be	lcp[1]	3
s[2]	8	S_8	_not_to_be	lcp[2]	1
s[3]	5	S_5	_or_not_to_be	lcp[3]	1
s[4]	12	S_{12}	_to_be	lcp[4]	1
s[5]	16	S_{16}	be	lcp[5]	0
s[6]	3	S_3	be_or_not_to_be	lcp[6]	2
s[7]	17	S_{17}	e	lcp[7]	0
s[8]	4	S_4	e_or_not_to_be	lcp[8]	1
s[9]	9	S_9	not_to_be	lcp[9]	0
s[10]	14	S_{14}	o_be	lcp[10]	0
s[11]	1	S_1	o_be_or_not_to_be	lcp[11]	4
s[12]	6	S_6	or_not_to_be	lcp[12]	1
s[13]	10	S_{10}	ot_to_be	lcp[13]	1
s[14]	7	S_7	r_not_to_be	lcp[14]	0
s[15]	11	S_{11}	t_to_be	lcp[15]	0
s[16]	13	S_{13}	to_be	lcp[16]	1
s[17]	0	S_0	to_be_or_not_to_be	lcp[17]	5
				lcp[18]	0

Table 2. Suffixes and suffix arrays after sorting

sorted in lexicographical (alphabetical) order. Each suffix is a string starting at a certain position in the text and ending at the end of the text. Searching a text can be performed by binary search using the suffix array.

The algorithm, $suffix\_array$, presented below takes a string and its length N as input, and outputs the suffix array, s.

```
$suffix\_array \leftarrow function(string, N)\{$
    Initialize $s$ to be a vector of integers from 0 to $N-1$.
    Let each integer denote a suffix starting at $s[i]$.
    Sort $s$ so that the suffixes are in alphabetical order.
    return $s$; }
```

In order to compute the frequency, an auxiliary array is defined to store LCPs (longest common prefixes). The lcp array is a vector of $N+1$ integers. Each element, $lcp[i]$, denotes the length of the common prefix between the suffix $s[i-1]$ and the suffix $s[i]$. As mentioned above there are $N(N+1)/2$ substrings for a document with the length N. Instead of computing the statistics for all substrings directly the set of all substrings is partitioned by the classes with the same statistics (term frequency tf and document frequency df)[11]. The set of substrings in a class can be constructed from the lcp vector:

$$\begin{aligned} class(<i,j>) &= \{s[i]_m | LBL(<i,j>) < m \leq \\ &\quad SIL(<i,j>)\} \quad (5) \\ LBL(<i,j>) &= max(lcp[i], lcp[j+1]) \\ SIL(<i,j>) &= min(lcp[i+1], lcp[i+2], \ldots, lcp[j]) \\ tf_{class(<i,j>)} &= j-i+1 \quad (6) \end{aligned}$$

where LBL is longest bounding lcp, SIL is shortest interior lcp and $s[i]_m$ denotes the first m characters of suffix $s[i]$. A $class <i,j>$ exists between interval $<i,j>$ if $LBL < SIL$. Then this interval $<i,j>$ is lcp-delimited. It means all the suffixes in lcp-delimited interval $<i,j>$ share the same longest common prefix and no other suffixes outside the interval shares it. Thus it is not possible for two lcp-delimited intervals to overlap but it is possible to be nested. Table 2 shows that lcp-delimited interval $<0,1>$ with lcp "_be" is nested in interval $<0,4>$ with lcp "_". The term frequency for $class(<i,j>)$ is equal to $j-i+1$. $class(<i,i>)$ has term frequency 1 called trivial class. We are more interested in the nontrivial class with term frequency greater than 1. The number of substrings in a nontrivial class is

$$|class(<i,j>)| = SIL(<i,j>) - LBL(<i,j>). \quad (7)$$

The substrings in the nontrivial class $class(<i,j>)$ are the first $LBL(<i,j>)+1, \ldots, SIL(<i,j>)$ characters of suffix $s[i]$, total $SIL(<i,j>) - LBL(<i,j>)$ prefixes of suffix $s[i]$. All substrings in the same class have the same term frequency and document frequency if all suffixes

Interval	Class {Substrings}	SIL	LBL	tf
$< 0, 1 >$	_be {_b,_be }	3	1	2
$< 0, 4 >$	_ {_}	1	0	5
$< 5, 6 >$	be {b,be}	2	0	2
$< 7, 8 >$	e {e}	1	0	2
$< 10, 11 >$	o_be {o_,o_b,o_be}	4	1	2
$< 10, 13 >$	o {o}	1	0	4
$< 16, 17 >$	to_be {to,to_,to_b,to_be}	5	1	2
$< 15, 17 >$	t {t}	1	0	3

Table 3. Nontrivial classes for string "to_be_or_not_to_be"

were terminated with the first end of document symbol in multi-document corpus. Table 3 shows nontrivial classes for string "to_be_or_not_to_be". For $class < 16, 17 >$ with SIL 5 and LBL 1 there are 4 substrings {to, to_, to_b, to_be} in the class with the same term frequency 2. In Chinese the longest substring in the class is most likely to be a word or a phase. We can use the longest substring to represent the class instead of using all the substring to reduce the size of all substrings.

As two lcp-delimited intervals are not possible to overlap and possible to be nested, there are at most $N - 1$ nontrivial classes with term frequency greater than 1. For trivial classes there are at most N classes with term frequency equal to 1. This significantly reduces the computation of various statistics over substrings ($N(N + 1)/2$)) to a computation over classes ($2N - 1$).

If we can identify all the classes of a corpus, we can use a straightforward method to compute the term frequency in a document and the document frequency.

4 Filtering Algorithm

Suffix tree based approach described above can reduce the computation term frequency and document frequency over substrings ($N(N + 1)/2$)) to a computation over classes ($2N - 1$). The following algorithm is used to filter out the substrings which are incomplete and lack of representative.

4.1 Stop Words

In modern information retrieval systems, effective indexing can be achieved by removal of stop words. Chinese language hardly has any grammatical inflections. It makes heavy use of grammatical particles to indicate aspect and mood. They carry no significant information to the document. They are used often enough to mislead the occurrences of string patterns. A stop word list about 100 Chinese words is used to filter out these noisy patterns.

4.2 Longest substring in the class

All the substrings in the same class have the same term frequency and document frequency. The longest substring in the class is most likely to be a word or a phase. We can use the longest substring to represent the class instead of using all the substring to reduce the size of the substrings. As suffix tree is built from one direction, all the substrings with the same attributes (same statistics and part of longest substring in the class) are not including in the same class. In Table 3 there are 4 substrings {to, to_, to_b, to_be} in the $class < 16, 17 >$ with the same term frequency 2. If we construct another suffix from the other direction (from the end to the start), we should get 5 substrings {e, be, _be, o_be, to_be} in the same class with longest substring "to_be". In this case, we can merge classes with same attributes.

4.3 Mutual Information

Mutual information is commonly used to evaluate the correlation of substrings. We adapted the same mutual information metric as [2] by observing mutual information of two overlapped patterns.

$$\begin{aligned} MI_{ab} &= \frac{Pr(c)}{Pr(a) + Pr(b) - Pr(c)} \\ &= \frac{\frac{f_c}{F}}{\frac{f_a}{F} + \frac{f_b}{F} - \frac{f_c}{F}} \\ &= \frac{f_c}{f_a + f_b - f_c} \end{aligned}$$

where c is the substring to be estimated, $c = c_1, c_2, \ldots, c_n$, a and b are the two longest composted substrings of c with the length $n - 1$, i.e. $a = c1, \ldots, c_{n-1}$, $b = c_2, \ldots, c_n$, f_a,f_b and f_c are the frequencies of a, b and c. If MI_{ab} is large, it can be found that more of the time substrings a and b have to occur together. It seems that c is more complete in semantics than either a or b.

4.4 Frequency

The iterative occurrences of substring in multi-documents is most likely to be a word or a phrase. Low frequency substrings are more likely to occur by chance. We filter the trivial classes with the the length of substring greater than 1.

4.5 Length of N-gram

The length of N-gram will dramatically increase the number of N-grams produced. The longer N-gram, the larger space and longer time required. Most Chinese words

N-gram	Average Precision
1-gram	0.3571
1,2-gram	0.4187
1,2,3-gram	0.4223
1,2,3,4-gram	0.4250
1,2,3,4+-gram	0.4206

Table 4. Test results on TREC Chinese corpus

are 2 character and less than 4 characters but some noun phrases can be more than 4 characters such as ”亚太经济合作组织” (APEC) in query 17 and ”菲律宾皮纳图博火山” (Philippine Mount Pinatubo Volcano) in query 47. Term and document frequency thresholds are used in our experiments to choose substring with length greater than 2.

5 Experiments and Future Work

The tests are conducted on TREC Chinese corpus. The documents in the collection consist of approximately 170 megabytes of articles drawn from the People's Daily newspaper from 1991 to 1993 and the Xinhua newswire in 1994 and 1995. There are 164,789 documents in the collection. A set of 54 queries (TREC 5 and 6) has been set up and used to evaluate Chinese information retrieval task.

We used SMART system to evaluate our results. The average precision is measured on 11 recall points $(0.0, 0.1, \ldots, 1.0)$. Table 4 shows the test results on TREC Chinese corpus. The results on 1-gram and 1,2-gram are comparable to other researches [5, 6]. Wu [9] applied suffix tree for Chinese information retrieval but he did not mention how to rank the documents retrieved. Longer n-gram slightly improve the average precision. N-gram longer than 4 yields worse performance maybe because of the nature of the queries (shorter phrases).

Long strings are more meaningful than short strings as more contexts are available. Short and long strings should be treated differently in Chinese information retrieval. Luk et al. [4] found that increasing the weight according to the length might improve retrieval effectiveness. In future we will apply a length-weighting scheme to our work.

6 Conclusions

This paper proposes a suffix tree based approach for Chinese information retrieval. It uses n-gram as indexes without word segmentation. Most previous studies only used uni-gram and bi-gram. We extend N-gram to any length. This suffix tree based approach can reduce the computation term frequency and document frequency over substrings $(N(N+1)/2)$ to a computation over classes $(2N-1)$. The results on TREC Chinese corpus are comparable to other researches. Longer n-gram slightly improve the average precision in our experiment.

References

[1] C. Buckley. Implementation of the smart information retrieval system. Technical Report #85-686, Cornell University, 1985.

[2] L.-F. Chien. Pat-tree-based keyword extraction for chinese information retrieval. In *Proceedings of the 1997 ACM SIGIR*, 1997.

[3] S. Foo and H. Li. Chinese word segmentation and its effect on information retrival. *Information Processing and Management*, 40:161–190, 2004.

[4] R. Luk and K. Kwok. A comparison of chinese document indexing strategies and retrieval models. *ACM Transactions on Asian Language Information Processing*, 1(3):225–268, 2002.

[5] G. J. Z. J. NIE, J.-Y. and M. ZHOU. On the use of words and n-grams for chinese information retrieval. In *In Proceedings of the Information Retrieval with Asian Languages*, 2000.

[6] J.-Y. Nie and F. Ren. Chinese information retrieval: using characters or words? *Information Processing and Management*, 35:443–462, 1999.

[7] ROCLING. Segmentation principle for chinese language processing, 1997.

[8] E. Ukkonen. On-line construction of suffix tree. *Algorithmica*, 14(3):249–260, 1995.

[9] L. Wu. A novel information search approach for languages without word delimiters. *International Journal of Computer Science and Network Security*, 6(5A):53–59, 2006.

[10] F. Xia. The segmentation guidelines for the penn chinese treebank (3.0). Technical report, University of Pennsylvania, 2000.

[11] M. Yamamoto and K. Church. Using suffix arrays to compute term frequency and document frequentcy for all substrings in a corpus. *Computational Linguistics*, 27(1):1–30, 2001.

[12] S. Yu. Guidelines for the annotation of contemporary chinese texts: word segmentation and pos-tagging. Technical report, Institute of Computational Linguistics, Beijing University, Beijing, 1999.