Automated Extraction of Lexicon Applied both to Chinese and Japanese Corpora

Shujing Ke, Simon C. K. Shiu
Department of Computing
The Hong Kong Polytechnic University, Hong Kong
shujingke@gmail.com, dr.simonshiu@gmail.com

Benjamin Goertzel
Novamente LLC
MD, USA
ben@goertzel.org

Gino Yu
School of Design
The Hong Kong Polytechnic University, Hong Kong
phusikoi@gmail.com

Xiaodong Shi, Changle Zhou
Cognitive Science Department
Xiamen University,Xiamen, China
mandel@xmu.edu.cn, dozero@xmu.edu.cn

***Abstract*—A novel statistical approach is described, enabling the automated extraction of large word lists from unsegmented corpora without reliance on existing dictionaries. The main contribution of this approach includes the following two points: First, it's very generic and has been successfully applied separately to both Chinese and Japanese; Second, it doesn't take any use of punctuation information, so compared to most of the existing methods, it doesn't need to pre-process the corpora to remove the punctuations or to pre-segment the corpora by punctuations. Our experiment results in the extraction of 14,087 Chinese words and 15,553 Japanese words. Precision achieved is over 80% for two-character Chinese words, over 90% for one-character Japanese words and over 70% for two-character Japanese words. And we've also successfully extracted most of single-character words including common functional characters, such as "在" (in), "和" (and), "或" (or), "的"('s), "也" (also), "林" (a family name) in Chinese; hiragana such as "を","も","へ" in Japanese; and punctuations such as "，","。","？".**

Keywords- Word extraction; Statistics; Punctuation; Combination Degree

I. INTRODUCTION

Detection of word boundaries in Chinese, Japanese and other similar languages is a nontrivial issue, for multiple reasons: Texts are written without a space between words; words often comprise several characters, typically two, three, or four; and many characters can be used alone as words; characters can combine with each other in tens of thousands of different patterns. Addressing this complexity, many algorithms for automated word segmentation exist[1-18].

Almost all of these approaches require an existing large lexicon or manually segmented corpora or both. The lexicon is used to find segmentation candidates, and the segmented corpora are used as training resources to calculate the probabilities of the combination patterns to solve segmentation ambiguities [10]. As Jin and Wong point out [11], manual lexicon construction and Preparation of large manually segmented corpora are usually time-consuming, so, automatically acquiring a good lexicon from unsegmented raw corpora is very important [12], which leads to the goal of our paper.

A number of approaches already exist for automatically extracting words:

In 1990, Sproat and Shih grouped Chinese characters into two-character words and calculate the mutual information between two characters. The character pairs with high mutual information are considered to be words. This approach can only process words with a length of two characters [13].

In 1997, Chang and Su developed an iterative approach for new word extraction [14].

Sun presented an algorithm to do Chinese word segmentation, taking use of the difference of t-score between characters instead of pure mutual information [10].

In 1999, Ge used "soft-counting" to estimate the probability of words, which takes the left and right neighboring segments into account. But the results could not exclude the errors from about 20 single-character auxiliary words (such as"的(of)", "和(and)", "或(or)") that occur together with other words too often to tell them apart (such as "我们的(of us)", "和你(with you)") [15]. To avoid this problem, Jin and Wong pre-define a list of such auxiliary characters (including Chinese digit numbers) as filters [17].

Lin and Yu propose a simple way to excludes all the error patterns which appear due to the appearance of a longer pattern. For example, "然科学" will be excluded, because it's the substring of "自然科学 (natural science)", and "然科学" never appears outside "自然科学" [16].

In 2002, Jin and Wong made use of local statistical information (data within a document) to extracts the longest repeated string patterns as candidates, but there are many occasions when the longest repeated strings are not real words [11].

In 2004, Feng applied "Accessor Variety" to Chinese new words extraction. Accessor Variety is an approach to evaluating the independence of a string, which is similar to

978-0-7695-4959-0/13 $25.00 © 2013 IEEE
DOI 10.1109/ACSAT.2012.15

Ge's "soft-counting" [15]. This approach involves extraction of out-of-vocabulary (OOV) words, so it also requires use of an existing dictionary as system dictionary [18].

II. CONTRIBUTION OF OUR APPROACH

The existing approaches for extracting words all rely on huge segmented training corpora, or existing dictionary, or a list of predefined punctuations and auxiliary characters. And as far as we know, none of them have applied in multiple languages because they have taken use of the language features of the specific language. Compared to these existing approaches, the unique assets of our approach, are as follows:

(1) Our approach relies on neither any existing dictionary nor manual segmented training corpora. We process the raw corpora directly, involving no manual work.

(2) We process punctuations as ordinary characters. All existing approaches require pre-processing of raw corpora to remove punctuations or to separate strings by punctuations [10,11,13-18]. Our underlying assumption has been that, since different languages have different punctuations, we shouldn't assume that we know which characters are punctuations. Thus, in our approach, punctuations are treated as ordinary characters, and are expected to be successfully extracted as single-character words.

(3)Language-generic, rather than applying to one specific language. Our approach is expected to process any unsegmented language, and has been successfully applied separately in Chinese and Japanese.

III. OUR APPROACH

Our approach to word list extraction involves five steps:

Step1: Extract high frequency strings from the corpus;

Step2: Exclude the non-independent strings;

Step3: Extract functional characters (including punctuations) from the corpus;

Step4: Exclude the words containing functional characters with a lower frequency;

Step5: Find the missing words.

In this section, we'll describe these five steps in detail, illustrating them by reference to experiments we have conducted applying the methodology to a Chinese corpus consisting of 80megabytes of texts from the People's Daily newspaper, and a Japanese corpus consisting of 23 megabytes of modern Japanese novels.

A. Extract High Frequency Strings from Corpus

In this step, we traverse the original corpus Φ literally to count the number of times each string appears, including two-character to ten-character strings (of course, the upper limit of ten could be increased if needed for some language; but for the majority of known languages such a limit seems appropriate). For example, suppose T(s) is the occurrence number of string sin this sentence:

"明天的明天的明天是大后天。"

(The tomorrow of tomorrow of tomorrow is three days from now.)

The occurrence numbers of the n-character strings in this sentence for example are :

T("明天") = 3, T("的明") = 2, T("明天的") = 2

T("明天的明天") = 2

T("明天的明天的明天是大") = 1

... ...

Overall this step yields a series of records: $T(s_2)$、$T(s_3)$… $T(s_{10})$, $s_n \in \Phi$, $n \in \{2,3,4,5,6,7,8,9,10\}$, s_2 is a two-character string and s_3 is a three-character string, and so on.

Next, we assign a threshold I(n) to distinguish the strings with adequately high frequency; chosen strings constitute the MayBe word list K_1. Because the longer strings often appear fewer times than shorter strings, we define the threshold function below:

$$I(n) = \frac{M}{n-1}, n \in \{2,3,4,5,6,7,8,9,10\}$$

M is a constant, whose value is dependent on the size of corpus. In this threshold function, we can see that I is inversely proportional to n, so it fulfills the rule that the longer strings generally appear fewer times than shorter strings. And then we filter strings as follows: If $T(s_n) > I(n)$, add s_n into K_1.

B. Exclude the Non-independent Strings

In step 1 we have obtained a MayBe word list K_1, but there still remain many strings with a high frequency but not serving as true words in K_1, for example:

TABLE I. NON-INDEPENDENT STRING EXAMPLES

Chinese example	Japanese example
T("会主义") = 11725	T("苦しそ") = 73
T("社会主义") = 11709	T("苦しそう") = 73

We can see from Table I that the string "会主义" appears 11725 times; but it's not a true word. In this step, we'll use the "Inclusion Rules" below to exclude the non-independent strings from K_1, and then get K_2.

Definition 1. ***Inclusion Rule 1****:* If there is a string s_{N+1} that makes $T(s_N) \approx T(s_{N+1})$, then that s_N is included in s_{N+1}.So s_N hardly appears alone; it always appears due to the appearances of s_{N+1}, suggesting that s_N is "non-independent" or does not have an independent meaning. Therefore, s_N cannot be a true word and should be excluded from K_1. For example:

T("会主义") = 11725 , T("社会主义") = 11709

$$\delta = \frac{T(\text{"会主义"}) - T(\text{"社会主义"})}{T(\text{"会主义"})} = 0.00136 \qquad (1)$$

δ , the difference between these two strings (see Equation (1)) is very small, so we can almost believe that every time "会主义" appears , "社会主义" appears too. "会主义" doesn't appear alone, so "会主义" is unable to express a complete meaning. Therefore, it's not a true word.

Definition 2. ***Inclusion Rule 2****:* When $T(s_N) >> $ all the $T(s_{N+1})$, then s_N is capable of expressing a meaning independently; so s_N is high likely to be a true word. For example:

TABLE II. INDEPENDENT STRING EXAMPLES

Chinese example	Japanese example
T("目的") = 4516	T("もの") = 10368
T("目的是") = 671 T("目的。") = 446	T("いもの") =1346 T("たもの") =1646

We can see from Table Ⅱ that the Chinese string"目的" appears many more times than any three-character string which includes "目的", so "目的" is very likely to be an independent word. And it's the same situation to the Japanese string "もの".

According to Rule 1 and Rule 2 described above, and we set a threshold of 2%, then when there exists an s_{N+1} which makes d < 2%, we remove s_N from K_1, the left constructs K_2.

By this step, we totally discard 30580 non-independent Chinese strings and 23917 Japanese strings. Here we list a few typical non-independent strings that are successfully discarded by this step:

TABLE III. NON-INDEPENDENT STRING EXAMPLES OF CHINESE

s_N	s_{N+1}
T("播电台") = 357	T("广播电台") =357
T("轻人") =496	T("年轻人") =491
T("迟浩") =348	T("迟浩田") =348
T("有中国特") = 1588	T("有中国特色") = 1585

TABLE IV. NON-INDEPENDENT STRING EXAMPLES OF JAPANESE

s_N	s_{N+1}
T("転車") = 84	T("自転車") = 84
T("お嬢さ") = 257	T("お嬢さん") = 254
T("もしろい") =119	T("おもしろい") =119
T("お願) = 425	T("お願い") =425

C. Extracting Functional Characters

In section B we described how to exclude non-independent shorter strings from longer strings, but we also need to solve the opposite situation. We must find a method to exclude the strings like "的面貌" (appearance of), which is not a true word, while including a part of it, "面貌" (appearance), which is a true word. This situation is called redundancy combination by Yuan and Liang [Yuan and Liang 2006]; and they also proposed a method to solve this problem. Their method can solve situations like "用户" which is a part of "给用户", because T("用户")is much higher than T("给用户"). But it cannot solve similar situations like "优越", which is a part of "优越性", in which T("优越") is also much higher than T("优越性"), but in which "优越性" is a true word.

In this paper, to solve this problem, we do the following: We notice that most of the redundant combinational strings are composed of a true word combined with functional characters - such as "和" (and), "为"(for), "是"(is), "是"(is), "を"(a hiragana)," も"(a hiragana)or punctuations. Examples of this are: "和澳门"(and Macao), "あの女"(that woman)". So we extract a common functional character list, and then we can exclude the strings redundantly combined with the functional characters as shown above.

1) Combination Degree

Feng has extracted a single-character list using "Accessor Variety" [18], but doesn't take punctuations into account, while we argue that punctuations should be extracted too. Functional characters are single-character words including punctuations, such as "！","」","我", "中", "の", which are usually segmented as single-character words. Extraction of functional characters can help us to understand a language better (especially an unknown language), but also plays a key role in further excluding false words, as will be described in section D.

We observe that functional characters have high "Combination degrees".

Definition 3. ***Combination Degree (CD)***: A character can be combined with other characters, before or after it. For a character c, suppose Before(c) is the collection of all the characters that can be used before c; After(c) is the collection that can be used after c. So bc, ca, and bca, are possible patterns, where b∈Before(c), a∈After(c). Suppose Sizeof(Before(c)) is the number of characters in Before(c), and Sizeof(After(c)) is the number of characters in After(c), define that CT(c) = Sizeof(Before(c)) + Sizeof(After(c)), so CT(c) stands for how many different characters can be next to (before or after) c, which is similar to Accessor Variety [18]. But different from Accessor Variety we argue that, to evaluate the combination ability of a character, we should divide CT(c) by a transformed version of the total occurrences of the character itself in the test corpus, a number that indicate show many different characters this character combines with every time it appears. The total occurrences is a large number and if used un-transformed as a normalizing factor, it would have too much influence on the combination degree; so in our experiments we have opted to use the fourth root of the occurrence times as the divisor:

$$CD(c) = \frac{CT(c)}{\sqrt[4]{T(c)}} = \frac{Sizeof(Before(c)) + Sizeof(After(c))}{\sqrt[4]{T(c)}}$$

Other slowly increasing monotone transformations could of course be used in place of the fourth root here; experimentation with Chinese and Japanese corpora suggests the results of our method are not highly sensitive to this choice.

Let's compare two characters below:

(1) Character"。" is the Chinese period punctuation, which of course is a functional character. It can be combined by almost all the other characters. So CD("。") is very high.

(2) Character "灯" (light) is an ordinary character with a real meaning, not a functional character. It can only be combined by very few characters, such as "电灯"(electric

light), "日光灯"(fluorescent light), "灯泡"(light bulb) and so on, so CD("灯") is very low.

This high CD value for functional characters also exists in Japanese. For example, the character"を" in Japanese is a typical functional character, which is often used between a noun and a verb to suggest that this noun is the object of the verb. Therefore it can be combined with many nouns and verbs, so CD("を") is very high. The Japanese character "感"(sense) is not a functional character, it's only used in cases such as "感じ"(feeling), "感情"(emotion), "予感"(presentiment), "第六感" (sixth sense), "感謝する" (thank) and so on, so CD("感") is low.

Thus we conclude that high CD values for functional characters are common. While we have only systematically validated this conjecture in Chinese and Japanese, we suggest the same property will carry over to other non-segmented languages.

2) The Process and Results of Extracting Functional Characters

Next, we apply the above rules to extract a functional character list. We traverse the original corpus Φ literally again to count how many different characters can be combined with every character, that is, the Combination Degree. Take the Chinese character"幕" for example:

Here are all the combination strings with "幕" in the corpus we analyzed:

幕。　帷幕　幕，　幕式
屏幕　开幕　幕的　序幕　闭幕

We can see there are totally 9 different characters above that can be combined with character"幕", so we record that CT("幕") = 9. And we've count the total occurrence times of "幕": T("幕")= 4615, so $CD("幕") = \frac{9}{\sqrt[4]{4615}} = 41.7365$.

In this step, in our computational experiments, we calculated the Combination Degree of all the 6424 Chinese characters, and 4452 Japanese characters in our corpora. In these tests, the Chinese character with the highest Combination Degree is the punctuation"、", CD("、") =318.283, the Japanese character with the highest Combination Degree is the punctuation"(", CD("(") = 233.796. Fig.1 and Fig.2 show the distributions of Chinese and Japanese characters due to their CD value: (The cross shaft stands for the CD value interval, the ordinate axis stands for the how many characters in this interval.)

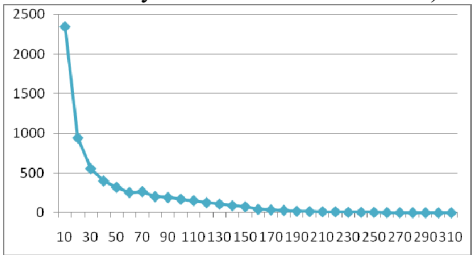

Figure 1. Distribution of Chinese character number due to CD.

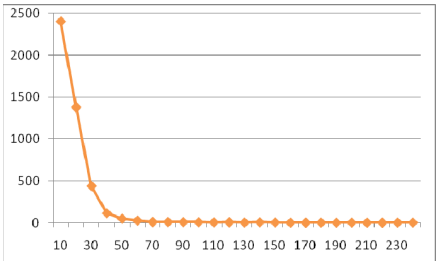

Figure 2. Distribution of Japanese character number due to CD.

It is clear from Figure 1 and Figure 2 that the CD values of most Chinese characters are below 130, and the CD values of most Japanese characters are below 25. So for our experiments, we set the $Threshold_{Chinese} = 130$, $Threshold_{Japanese} = 25$. Similar tuning could be done for any language to which our method was applied. For character c, whose Combination Degree is CD(c), if CD(c)≥Threshold, we add c into the functional character list. During this step, in our experiments, we extracted a list of 280 Chinese functional characters and a list of 378 Japanese functional characters. The top 30 Chinese functional characters and Japanese functional characters that we found are:

TABLE V. THE TOP 30 CHINESE FUNCTIONAL CHARACTERS EXTRACTED

char	CD	char	CD	char	CD
、	318.283	，	247.557	了	230.090
；	283.051	着	246.923	）	229.572
和	274.563	之	245.472	到	228.543
”	273.834	而	245.248	上	227.782
或	264.400	是	241.715	以	226.891
“	262.486	。	238.263	又	224.949
等	259.813	被	237.471	地	222.527
与	255.213	及	232.775	将	218.975
的	253.771	（	232.358	：	217.933
在	248.160	为	230.200	就	217.546

TABLE VI. THE TOP 30 JAPANESE FUNCTIONAL CHARACTERS EXTRACTED

char	CD	char	CD	char	CD
(	233.796	、	136.402	だ	105.803
《	227.802	も	131.162	な	105.764
		194.600	と	126.138	へ	105.403
（	171.393		122.382	り	99.7896
を	145.822	。	119.948	る	99.4488
に	141.892	「	115.896	,	97.2028
が	140.276	や	114.800	か	92.5878
の	139.577	く	113.858	た	91.3141
は	139.532	ら	112.862	い	90.3571
で	136.411	て	107.889	ば	89.8805

The lists above indicate that we have found, via our automated analysis, almost all the common functional characters in these languages, including punctuations and Japanese hiragana and katakana. We call the function character list G.

D. *Excluding the Words Containing Functional Characters with a Lower Frequency*

Most of the redundant combination strings consist of a true word combined with a functional character. Since we have extracted a functional character list G in section C, in this step we can exclude the false words that contain any character in G.

But should we exclude all the strings containing functional characters? Let's see an example below:

Strings containing Chinese functional character "们":

T("们之") = 209　　T("人们") = 8510

T("们也") = 831　　T("他们") = 21928

We find that false words "们之" and "们也" have a low appearance frequency while true words "人们" and "他们" have a much higher frequency. So we need to set a threshold value Y to decide which strings containing functional characters should be excluded.

We take the character "们"as an example again: All the strings combined with "们" totally appear 154918 times in our research, so Tall("们") = 154918, we define threshold function Y(n) as below:

$$Y(n) = \frac{T_{all}(s_n)}{100.0 \times (n-1)^2}$$

S_n is a n-character string, and n, the length of a word, has a great influence on the appearance times (see section A), so we take the square of n.

In this step, we traverse the word list K_2. For every word $s_n \in K_2$, if there exists a character c, $c \in s_n$ and $c \in G$, making $T(s_n) < Y(n)$, then s_n is not a true word and we remove it from K_2. The remaining words constitute a new MayBe word list K_3.

In our experiments, we excluded 64988 Chinese non-words and 184262 Japanese non-words from K_2in this step. Some strings we excluded in this step were:

TABLE VII.　A PART OF REDUNDANT COMBINATION STRINGS EXCLUDED

Chinese example		Japanese example	
0个	我国是	で！	関係も
3天	智慧和	と。	陽子は
。那	更需要	遠くに	その子は
会，	节前夕	道路の	、彼女
的路	表示：	部分を	へ入る
该产品	特别是党	銀行へ	邪魔し
中写道:	他们都	開けよ	の動物

E. *Finding the Missing Words*

If we use the word list K_3 generated via the above steps to segment the corpus, we'll miss a lot of strings that cannot be found in K_3. For example, there is a sentence in the corpus:

"然而，就是这样一些沉默寡言、被困苦……"

In K_3 we can find "然而", "，", "就", "是", "这样", "一些", "、", "被", "困苦", but cannot find "沉默寡言" because it has been neglected in the first step (section A) due to its low appearance frequency, which means a true word is missing by this stage. So, in this final processing step, we find all the missing words in the corpus and add them to the missing word list L.

In this step, we use "greedy algorithm" to segment the corpus Φ - trying to match every string as long as possible according to the words list in K_3 , and then collect the strings left failed to be segmented as missing words. Take the above sentence as example: after the simple greedy segmentation, this sentence is like:

"然而 ， 就是 这样 一些 沉默寡言 、 被 困苦……"

The words "然而", "就是", "一些", "困苦" have been found in the two-character list in K_3, and ",", "，", "、" and "被" have been found in the functional character list in K_3, only the string "沉默寡言" cannot match any words in K_3, so we add "沉默寡言" into missing word list L. We use the above method to find out all the missing words, at the same time counting the appearance time of every missing word. But not every word in L is a real word, here we use the same method as 4.1 to pick up the words with higher appearance time in L and add them into our word list K_3. We set the threshold value according to the length of the strings, because longer strings often appear fewer times than shorter strings (see section A). We calculate the average occurrence numbers for the missing words of each length, and set the threshold value A(n) to be the average value thus calculated.

Most of the missing words longer than four characters appear fewer than 2 times, so we just neglect them. Therefore, in this step, for every missing word s_n in L, if $T(s_n) > A(n)$, $n \in \{2,3,4\}$, add s_n into K_3, and the final K_3 is the our resulting word list Ψ.

During this step, in our experiments, we extracted a total of 447 one-character Chinese words and 93 Japanese, which we added to the functional character list. Most of these are family names, quantifiers, punctuations and particles. Here are some examples:

TABLE VIII.　A PART OF RETRIEVED MISSING SINGLE-CHARACTER WORDS

Chinese example		Japanese example	
’	它	ぽ	べ
•	她	耳	ゆ
千	杨	車	ぞ
吨	陈	ぽ	笑
倍	则	ゅ	ひ

We find 537 two-character Chinese words and 4132 two-character Japanese words; 1490 three-character Chinese words and 375 three-character Japanese words, most of which are proper nouns; 1136 four-character Chinese words and 157 four-character Japanese words, most of which are names, proper nouns or correct idioms. See some examples in TABLE Ⅸ

TABLE IX.　A PART OF RETRIEVED MISSING WORDS

Chinese example		Japanese example	
落户	病房	泉夫	転職
愤怒	坎坷	夢想	佐助
餐馆	腐蚀	放題	万歳

弟弟	吃惊	服従	優美
NBA	IC卡	尿毒症	優越感
浪淘沙	乌托邦	警視庁	思春期
脑血栓	妈祖庙	ＢＭＷ	清掃員
脊梁骨	滕王阁	取調室	過敏症
花木兰	乌纱帽	ＵＦＯ	盂蘭盆
熙熙攘攘	波澜壮阔	ペラペラ	１９７９
狂轰滥炸	徇私舞弊	阿弥陀仏	メソメソ
牵线搭桥	浩浩荡荡	永井玄蕃	有価証券
死灰复燃	CCTV	ヒクヒク	竹岡美穂
释迦牟尼	CDMA	单身赴任	豊臣秀頼

IV. EVALUATION

We extracted a total of 14087 Chinese words and 15,553 Japanese words, including two-character to nine-character words. To evaluate, we adopted “ICTCLAS” dictionary (http://ictclas.org/) as Chinese standard dictionary, and "ipadic-2.7.0" (http://chasen-legacy.sourceforge.jp/) as Japanese standard dictionary. Compared to these two standard dictionaries, we calculated the precision rate of our experiment as follows:

TABLE X. THE PRECISION OF OUR EXPERIMENT

Precision	Chinese	Japanese
one-char	-	90.7121%
two-char	81.7322%	73.6567%
three-char	26.1605%	29.7162%
four-char	38.8921%	22.6493%

Note: There are no single-character Chinese words in any existing dictionary, so we cannot calculate the precision of the single-character Chinese words.

The precision of one-character and two-character words are high, and the precision of longer words is lower. Since most Chinese words are two-character words (75%), and words longer than four-character account for only 6%[28]; and Japanese words have the similar distribution due to the length of words, our results are acceptable.

We follow the method of Feng [18] to calculate the partial recall rate of our experiment. We randomly select 34.7KB Chinese text and 12.9KB Japanese text from our corpora, and extract all the words included in these small part texts manually, separately by Chinese and Japanese speakers. Then we calculate the partial recall rate of our experiment: 64.1869% for Chinese and 61.3315% for Japanese.

We cannot make any comparison with the results of other researches, since other researchers are conducted under much easier conditions like using an existing dictionary, or manually segmented training corpora, or pre-defined punctuation information (see section Ⅱ).

REFERENCES

[1] MARUYAMA, N., MOROHASHI, M., UMEDA, S. AND SUMITA, E. 1998. A Japanese sentence analyzer. IBM Journal of Research and Development. 32, 2, 238–250.

[2] WU, Z.-M. AND Tseng, G. 1993.Chinese text segmentation for text retrieval: achievements and problems. Journal of the American Society for Information Science. 44, 9, 532–542.

[3] K. Kita, Y. Kato, T. Omoto and Y. Yano, A comparative study of automatic extraction of collocations from Corpora: Mutual information vs. cost criteria, Journal of Natural Language Processing, vol.1, no.1, pp.21-29, 1992.

[4] PALMER, D.D. 1997.A Trainable Rule-basedAlgorithm for Word Segmentation.In Proceedings of the 8th conference on European chapter of the Association for Computational Linguistics (EACL '97). ACL, Madrid, Spain,321–328.

[5] DAI, Y.-B., LOH, T. E. AND KHOO, C. S. G. 1999. A new statistical formula forChinese text segmentation incorporatingcontextual information. In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR ’99). ACM, Berkeley, CA, 82–89.

[6] XUE, N. 2003.Chinese word segmentation as character tagging. International Journal of Computational Linguistics and Chinese Language Processing (IJCLCLP). 8, 1, 29–48.

[7] NAKAGAWA, T. AND MATSUMOTO, Y. 2004.Chinese and Japanese WordSegmentation Using Word-Level and Character-LevelInformation. In Proceedings of the 20th international conference on Computational Linguistics (COLING '04). ACL,Geneva, Switzerland, 466–472.

[8] ZHAO, H., HUANG, C.-N., LI, M. AND LU, B.-L. 2010. A Unified Character-Based Tagging Framework for Chinese Word Segmentation. ACM Transactions on Asian Language Information Processing (TALIP). 9, 2, 1–32.

[9] ZHAO, H. AND KIT, C. 2011.Integrating unsupervised and supervised word segmentation: The role of goodness measures. Information Sciences: an International Journal. 181, 1, 163–183.

[10] SUN, M.-T., SHEN, D.-Y. AND TSOU, B. K. 1998.Chinese word segmentation without using lexicon and hand-crafted training data. InProceedings of the 17th international conference on Computational linguistics (COLING '98).ACL, Montreal, QC,1265–1271.

[11] JIN, H. AND WONG, K.-F. 2002.A Chinese dictionary construction algorithm for information retrieval. ACM Transactions on Asian Language Information Processing (TALIP). 1, 4, 281–296.

[12] PONTE, J. M. AND CROFT, W. B. 1996.USeg: A retargetable word segmentation procedure for informationretrieval. In Symposium on Document Analysis and Information Retrieval 96 (SDAIR '96). University of Massachusetts, Amherst, MA.

[13] SPROAT, R. AND SHIH, C. 1990.A statistical method for finding word boundaries in Chinese text.Computer Processing of Chinese and Oriental Languages. 4, 4, 336–351.

[14] CHANG, J.-S. AND SU, K.-Y. 1997.Anunsupervised iterative method forChinese new lexicon extraction.International Journal of Computational Linguistics and Chinese Language Processing (IJCLCLP). 2, 2, 97–148.

[15] GE, X., PRATT, W. AND SMYTH, P. 1999.Discovering Chinese words from unsegmented text. In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in InformationRetrieval (SIGIR '99).ACM,Berkeley, CA,271–272.

[16] LIN, Y.-J. AND YU,M.-S.2001.Extracting Chinese Frequent Strings Without a Dictionary From a Chinese Corpus and its Applications. Journal of Information Science and Engineering (JISE). 17, 5, 805–824.

[17] YUAN, F.-Y. AND LIANG, S.-P. 2006. Analysis and Study on the Method of Extracting Words Without Dictionary. International Journal of Systems and Control. 1, 1, 96–103.

[18] H. Feng, K. Chen, X. Deng, and W. Zheng, “Accessor variety criteria for Chinese word extraction RID E-8607-2011”, Computational Linguistics, vol. 30, no. 1, pp. 75-93, Mar 2004.