A Generalized Approach to Word Segmentation Using Maximum Length Descending Frequency and Entropy Rate

Md. Aminul Islam, Diana Inkpen, and Iluju Kiringa

School of Information Technology and Engineering,
University of Ottawa, Ottawa, ON, Canada, K1N 6N5
{mdislam, diana, kiringa}@site.uottawa.ca

Abstract. In this paper, we formulate a generalized method of automatic word segmentation. The method uses corpus type frequency information to choose the type with maximum length and frequency from "desegmented" text. It also uses a modified forward-backward matching technique using maximum length frequency and entropy rate if any non-matching portions of the text exist. The method is also extendible to a dictionary-based or hybrid method with some additions to the algorithms. Evaluation results show that our method outperforms several competing methods.

1 Introduction

Word segmentation is an important problem in many natural language processing tasks; for example, in speech recognition where there is no explicit word boundary information given within a continuous speech utterance, or in interpreting written languages such as Chinese, Japanese and Thai where words are not delimited by white-space but instead must be inferred from the basic character sequence. We differentiate the terms *word breaking* and *word segmentation*. Word breaking refers to the process of segmenting known words that are predefined in a lexicon. Word segmentation refers to the process of both lexicon word segmentation and unknown word or new word[1] detection. Automatic word segmentation is a basic requirement for unsupervised learning in morphological analysis. Developing a morphological analyzer for a new language by hand can be costly and time consuming, requiring a great deal of effort by highly-specialized experts.

In databases, word segmentation can be used in schema matching to solve semantic heterogeneity, a key problem in any data sharing system whether it is a federated database, a data integration system, a message passing system, a web service, or a peer-to-peer data management system [16]. The name of an element in a database typically contains words that are descriptive of the element's semantics. N-grams

[1] New words in this paper refer to out-of-vocabulary words that are neither recognized as named entities or factoids, nor derived by morphological rules. These words are mostly domain-specific and / or time-sensitive.

A. Gelbukh (Ed.): CICLing 2007, LNCS 4394, pp. 175 – 185, 2007.

have been shown to work well in the presence of short forms, incomplete names and spelling errors that are common in schema names [10].

Also, extracting words (word segmentation) from a scanned document page or a PDF is an important and basic step in document structure analysis and understanding systems; incorrect word segmentation during OCR leads to errors in information retrieval and in understanding the document.

One of the common approaches involving an extensive word list combined with an informed segmentation algorithm can help achieve a certain degree of accuracy in word segmentation, but the greatest barrier to accurate word segmentation is in recognizing unknown words, words not in the lexicon of the segmenter. This problem is dependent both on the source of the lexicon as well as the correspondence between the text in question and the lexicon. Fung and Wu [11] reported that segmentation accuracy is significantly higher when the lexicon is constructed using the same type of corpus as the corpus on which it is tested.

The term *maximum length descending frequency* means that we choose maximum length *n*-grams that have a minimum threshold frequency and then we look for further *n*-grams in descending order based on length. If two *n*-grams have same length then we choose the *n*-gram with higher frequency first and then the *n*-gram with next higher frequency if any of its characters are not a part of the previous one. If we follow this procedure, after some iterations, we can be in a state with some remaining character(s) (we call it *residue*) that is not matched with any type in the corpus. To solve this, we use the *leftMaxMatching* and *rightMaxMatching* algorithms presented in Section 3 along with entropy rate.

This paper is organized as follow: Section 2 presents a brief overview of the related work. The proposed method is described in Section 3. A walk-through example of the method is presented in Section 4. Evaluation and experimental results are discussed in Section 5. We address the potential applications of the proposed method and conclude in Section 6.

2 Related Work

Word segmentation methods can be roughly classified as either dictionary-based or statistically-based methods, while many state-of-the-art systems use hybrid approaches. In dictionary-based methods, given an input character string, only words that are stored in the dictionary can be identified. The performance of these methods thus depends to a large degree upon the coverage of the dictionary, which unfortunately may never be complete because new words appear constantly. Therefore, in addition to the dictionary, many systems also contain special components for unknown word identification. In particular, statistical methods have been widely applied because they use a probabilistic or cost-based scoring mechanism rather than a dictionary to segment the text [12].

A simple word segmentation algorithm is to consider each character a distinct word. This is practical for Chinese because the average word length is very short, usually between one and two characters, depending on the corpus [11], and actual words can be recognized with this algorithm. Although it does not assist in task such as parsing, part-of-speech tagging, or text-to-speech systems [24], the character-as-

word segmentation algorithm has been used to obtain good performance in Chinese information retrieval, a task in which the words in a text play a major role in indexing.

One of the most popular methods is *maximum matching* (MM), usually augmented with heuristics to deal with ambiguities in segmentation. Another very common approach to word segmentation is to use a variation of the *maximum matching algorithm*, frequently referred to as the *greedy algorithm*. The greedy algorithm starts at the first character in a text and, using a word list for the language being segmented, attempts to find the longest word in the list starting with that character. If a word is found, the maximum-matching algorithm marks a boundary at the end of the longest word, then begins the same longest match search starting at the character following the match. If no match is found in the word list, the greedy algorithm simply segments that character as a word and begins the search starting at the next character. A variation of the greedy algorithm segments a sequence of unmatched characters as a single word; this variant is more likely to be successful in writing systems with longer average word lengths. In this manner, an initial segmentation can be obtained that is more informed than a simple character-as-word approach. As a demonstration of the application of the character-as-word and greedy algorithms, consider an example of "desegmented" English, in which all white spaces has been removed: the "desegmented" version of the text *the most favourite music of all time* would thus be *themostfavouritemusicofalltime*, Applying the character-as-word algorithm would result in the useless sequence of tokens *t h e m o s t f a v o u r i t e m u s i c o f a l l t i m e*, which is why this algorithm only makes sense for languages such as Chinese. Applying the greedy algorithm with a "perfect" word list containing all known English words would first identify the word *them*, since that is the longest sequence of letters starting at the initial *t* which forms an actual word. Starting at the *o* following *them*, the algorithm would then find no match. Continuing in this manner, *themostfavouritemusicofalltime* would be segmented by the greedy algorithm as *them o s t favourite music of all time*. A variant of the maximum matching algorithm is the *reverse maximum matching* algorithm, in which the matching proceeds from the end of the string of characters, rather than the beginning. In the foregoing example, *themostfavouritemusicofalltime* would be segmented as *the most favourite music o fall time* by the reverse maximum matching algorithm. Greedy matching from the beginning and the end of the string of characters enables an algorithm such as *forward-backward matching*, in which the results are composed and the segmentation optimized based on the two results [7].

Many unsupervised methods have been proposed for segmenting raw character sequences with no boundary information into words [1, 2, 4, 5, 8, 14, 15]. Brent [1] gives a good survey of these methods. Most current approaches are using some form of EM to learn a probabilistic speech-or-text model and then employing Viterbi decoding procedures [19] to segment new speech or text into words. One reason that EM is widely adopted for unsupervised learning is that it is guaranteed to converge to a good probability model that locally maximizes the likelihood or posterior probability of the training data. For the problem of word segmentation, EM is typically applied by first extracting a set of candidate multi-grams from a given training corpus [8], initializing a probability distribution over this set, and then using the standard iteration to adjust the probabilities of the multi-grams to increase the posterior probability of the training data. Somewhat similar tasks of segmenting

words into morphemes, where methods use minimal length description were shown to give good results [13].

Saffran et al., [21] proposed that word segmentation from continuous speech may be achieved by using transitional probabilities (TP) between adjacent syllables A and B, where, TP($A \rightarrow B$) = P(AB)/P(A), with P(AB) being the frequency of B following A, and P(A) the total frequency of A. Word boundaries are postulated at local minima, where the TP is lower than its neighbors.

In corpus-based word segmentation, there is either no explicit model learnt, as when neural networks [20] or lazy learning [6] are used, or the derived models are less sophisticated and do not use any abstractions of the word constituents found in data [3, 17]. Using annotated corpora greatly facilitates learning. However, there are situations in which one is interested in Unsupervised Learning (UL), that is, from unannotated corpora. Motivation for UL can vary from purely pragmatic, such as the high cost or unavailability of annotated corpora, to theoretical, when language is modelled as yet another communication code within the framework of Information Theory [22].

3 Proposed Method

Let $S = l_1 l_2 l_3 \ldots l_m$ denotes a text of m consecutive characters without any space in between them for which we need to segment and $C = \{c_1, c_2, \ldots, c_\tau\}$ denotes a large corpus of text containing τ words (tokens). Also, let $T^p = \{t_1, t_2, \ldots, t_p\}$ be the set of all (p) unique words (types) which occur in the corpus C and $T^f = \{f_1, f_2, \ldots, f_p\}$ be the set of frequencies of all the corresponding types in T^p i.e. f_x is the frequency of type t_x. Unlike the corpus C, which is an ordered list containing many occurrences of the same words, T^p is a set containing no repeated words. Again, let n be the maximum length of any possible words in the segmented words list where $n \quad m$ and $N^p = \{l_1, l_2, .., l_n, l_1 l_2, l_2 l_3, .. , l_1 l_2 .. l_n, \ldots\}$ be the set of all possible n-grams where $\eta = |N^p|$ is the total number of n-grams in N^p. We can also consider N^p as $N^p = \{w_1, w_2 \ldots, w_\eta\}$. And $N^f = \{f_1^{'}, f_2^{'} \ldots, f_\eta^{'}\}$ be the set of frequencies of all the corresponding n-grams of N^p taken from T^f, i.e. $f_x^{'}$ is the frequency of w_x. To get rid of the noise types of the corpus, we assign a set of minimum frequencies for each possible length from 1 to n to be considered as a valid word. $M^f = \{\alpha_1, \alpha_2 \ldots, \alpha_n\}$, where α_x is the minimum frequency required to be a valid word of length x. The steps of the method are as follows:

Step 1: Sort all the elements of N^p in descending order based on length (in characters). Again sort in descending order for same length words of the sorted N^p (say $\overline{N^p}$) based on the frequencies of N^f. For each element in $\overline{N^p}$ do the next steps:

Step 2: If $S \neq \emptyset$ and the current maximum length n-gram (say w_n) in $\overline{N^p}$ satisfies $f_n^{'} \geq \alpha_{|w_n|}$ and $w_n \in S$ (i.e., $S \cap w_n = w_n$) then add w_n to segmented word list, $S^{'}$ (i.e., $S^{'} \leftarrow S^{'} \cup w_n$) and remove w_n from S (i.e., $S \leftarrow S \setminus w_n$) and add a blank space as a boundary mark.

Step 3: If $S \neq \emptyset$ and not all elements in $\overline{N^p}$ are done then update w_n by the next maximum length n-gram from $\overline{N^p}$ and go to step 2.
Step 4: Rearrange all the words of $S^{'}$ in accordance with S. If $S = \emptyset$, then output $S^{'}$ and exit. Otherwise, for each remaining chunks[2], r in S call *matchResidue*(r), output $S^{'}$ and exit.

Algorithm *matchResidue*
Input: r, $S^{'}$
1. // Take the prefix word, w_{n-1} and suffix
2. // word, w_n of r from $S^{'}$ according to the
3. // would be position of r in $S^{'}$.
4. $S^{'} \leftarrow S^{'} \setminus w_{n-1}$
5. $S^{'} \leftarrow S^{'} \setminus w_n$
6. $S_t \leftarrow w_{n-1} \cup r \cup w_n$
7. // $S_t = \{l_1 l_2 l_3 \ldots l_m\}$, where m is the length of S_t
8. $S_t^{'} \leftarrow leftMaxMatching(S_t)$
9. $S_t^{''} \leftarrow rightMaxMatching(S_t)$
10. **if** ($|S_t^{'}| > |S_t^{''}|$)
11. $S^{'} \leftarrow S^{'} \cup S_t^{''}$
12. **elseif** ($|S_t^{'}| < |S_t^{''}|$)
13. $S^{'} \leftarrow S^{'} \cup S_t^{'}$
14. **else**
15. find a $x \in \{S_t^{'}, S_t^{''}\}$ for which entropy
16. rate $\frac{1}{|x|}\sum_{i=1}^{|x|} \log_2(f_i)$ is maximum
17. $S^{'} \leftarrow S^{'} \cup x$
18. **end**

Output: $S^{'}$

Algorithm *leftMaxMatching*
// n is the maximum length of any possible valid words in S_t and $n \leq m$
Input: S_t
1. **while** $S_t \neq \emptyset$ do
2. $N^p \leftarrow \{l_1, l_1 l_2, l_1 l_2 l_3 .., l_1 l_2 \ldots l_n\}$
3. i.e., $N^p \leftarrow \{w_1, w_2 \ldots, w_n\}$
4. $N^f \leftarrow \{f_1^{'}, f_2^{'} \ldots, f_n^{'}\}$
5. $M^f \leftarrow \{\alpha_1, \alpha_2 \ldots, \alpha_n\}$
6. $i \leftarrow 1$
7. **while** ($i \leq n$ && $i \leq m$)
8. **if** ($f_i^{'} \geq \alpha_i$)
9. $max \leftarrow i$
10. **end**
11. increment i

[2] A single chunk may contain one or more characters.

12. **end**
13. $S_t^{'} \leftarrow S_t^{'} \cup w_{max}$
14. $S_t \leftarrow S_t \setminus w_{max}$
15. **end**

Output: $S_t^{'}$

Algorithm *rightMaxMatching*

// n is the maximum length of any possible valid words in S_t and $n \leq m$

Input: S_t

1. **while** $S_t \neq \emptyset$ do
2. $N^p \leftarrow \{l_m, l_{m-1}l_m, l_{m-2}l_{m-1}l_m, ..,$
3. $l_{m-n}l_{m-n+1}\ldots l_m\}$
4. i.e., $N^p \leftarrow \{w_1, w_2 \ldots, w_n\}$
5. $N^f \leftarrow \{f_1^{'}, f_2^{'} \ldots, f_n^{'}\}$
6. $M^f \leftarrow \{\alpha_1, \alpha_2 \ldots, \alpha_n\}$
7. $i \leftarrow 1$
8. **while** ($i \leq n$ && $i \leq m$)
9. **if** ($f_i^{'} \geq \alpha_i$)
10. $max \leftarrow i$
11. **end**
12. increment i
13. **end**
14. $S_t^{'} \leftarrow S_t^{'} \cup w_{max}$
15. $S_t \leftarrow S_t \setminus w_{max}$
16. **end**

Output: $S_t^{'}$

4 A Walk-Through Example

As a demonstration of the application of the proposed algorithms, consider the same example of "desegmented" English text, S = {*themostfavouritemusicofalltime*}. We have used the BNC[3] (British National Corpus) to calculate T^p and T^f. let, n=10 be the maximum length[4] of all possible words in S and M^f = {1000, 500, 50, 16, 15, 12, 10, 3, 2, 2}. Table 1 shows the sorted n-grams, $\overline{N^p}$ and their frequencies, N^f for this specific example.

For each element w_n (say, *favourite*) in $\overline{N^p}$,

Step 2: w_n satisfies $f_n^{'} \geq \alpha_{|w_n|}$ as 4671 $\geq$ 2 and w_n is a substring of S.

$S^{'}$={*favourite*} and S = {*themost musicofalltime*}.

Step 3: Not all elements in $\overline{N^p}$ are done, update w_n = {*alltime*} and go to step 2.

Step 2: doesn't satify $f_n^{'} \geq \alpha_{|w_n|}$ as 6<10 though w_n is a substring of S.

[3] http://www.natcorp.ox.ac.uk/

[4] Though in BNC, the length of the longest valid word is 34.

Table 1. Sorted n-grams and their frequencies (the right-hand side continues the table)

$\overline{N^p}$	N^f	$\overline{N^p}$	N^f	$\overline{N^p}$	N^f	$\overline{N^p}$	N^f
favourite	4671	tfa	2	hem	305	ll	233
alltime	6	of	3052752	sic	292	ri	230
favour	6805	it	1054552	mus	269	ou	151
musico	10	he	641236	emu	247	ic	132
music	15134	me	131869	ico	95	vo	93
vouri	1	us	80206	uri	46	ur	77
them	167457	co	17476	fal	44	tf	11
time	164294	th	16486	ofa	36	a	2179299
most	98276	st	15565	mos	36	i	873059
fall	11202	al	7299	fav	33	l	59
item	3780	fa	2172	tem	31	c	46
rite	293	em	1641	emo	20	t	21
allt	28	os	1005	ost	18	s	19
emus	14	te	831	rit	13	e	17
musi	3	si	658	ite	11	r	14
hemo	3	mo	639	usi	8	h	12
emos	2	ti	615	ime	6	f	10
the	6057315	im	576	cof	5	v	9
all	282012	lt	485	avo	5	m	8
our	93463	av	291	lti	4	o	5
tim	3401	mu	276	vou	3	u	3

Step 3: Not all elements in $\overline{N^p}$ are done, update w_n = {*favour*} and go to step 2.
Step 2: Condition fails as w_n is not a substring of *S*.

Step 3: Not all elements in $\overline{N^p}$ are done, update w_n = {*musico*} and go to step 2.
Step 2: Condition fails as w_n does not satisfy $f_n^{'} \geq \alpha_{|w_n|}$ as 10<12.

Step 3: Not all elements in $\overline{N^p}$ are done, update w_n = {*music*} and go to step 2.
Step 2: w_n satisfies $f_n^{'} \geq \alpha_{|w_n|}$ as 15134 15 and w_n is a substring of *S*.

$S^{'}$={*favourite*, *music*} and
S = {*themost ofalltime*}.

We will only show the step 2 of all the remaining elements in $\overline{N^p}$ that satisfy the conditions.

Step 2: w_n = {*them*}, $S^{'}$={*favourite*, *music, them*} and *S* = { *ost ofalltime*}.
Step 2: w_n = {*time*}, $S^{'}$={*favourite*, *music, them, time*} and *S* = { *ost ofall* }.
Step 2: w_n = {*fall*}, $S^{'}$={*favourite*, *music, them, time, fall*} and *S* = { *ost o* }.

Step 4: Rearrange $S^{'}$ = { them, favourite, music, fall, time} and S ≠ Ø, so call **matchResidue**(ost) and then **matchResidue**(o).

Case 1: ***matchResidue***(*ost*) is called
$S^{'} = S^{'} \setminus \{ w_{n-1}, w_n \}$

$S^{'}$ = { *them, favourite*, *music, fall, time*}\{ *them, favourite* }
= {*music, fall, time*}
S_t = {*themostfavourite*}
$S_t^{'}$ = {*them*, *os*, *t, favourite*}← ***leftMaxMatching***(*themostfavourite*)
$S_t^{''}$ = {*the*, *most, favourite*}← ***rightMaxMatching***(*themostfavourite*)
As $|S_t^{'}| > |S_t^{''}|$, $S^{'}$ = {*music, fall, time*} U $S_t^{''}$
i.e., $S^{'}$ = {*the*, *most, favourite*, *music, fall, time*}

Case 2: ***matchResidue***(*o*) is called
$S^{'} = S^{'} \setminus \{ w_{n-1}, w_n \}$
$S^{'}$ = {*the*, *most, favourite*, *music, fall, time*}\{ *music, fall* }
= {*the*, *most, favourite*, *time*}
S_t = {*musicofall*}
$S_t^{'}$ = {*music*, *of*, *all*}← ***leftMaxMatching***(*musicofall*)
$S_t^{''}$ = {*mus*, *ico, fall*}←***rightMaxMatching***(*musicofall*)
As in this case $|S_t^{'}| = |S_t^{''}|$, we need to find whether $S_t^{'}$ or $S_t^{''}$ maximizes the entropy rate, $\frac{1}{|x|}\sum_{i=1}^{|x|} \log_2(f_i)$, where $x \in \{S_t^{'}, S_t^{''}\}$. The entropy rate for $S_t^{'}$ is (13.89 + 21.54 + 18.11) / 3 and for $S_t^{''}$, (8.07 + 6.57 + 13.45) / 3. So, $S^{'}$ = {*the*, *most, favourite*, *time*} $\cup$ $S_t^{'}$, as $\frac{1}{|S_t^{'}|}\sum_{i=1}^{|S_t^{'}|} \log_2(f_i) > \frac{1}{|S_t^{''}|}\sum_{i=1}^{|S_t^{''}|} \log_2(f_i)$. Finally, $S^{'}$ = {*the*, *most, favourite*, *music*, *of*, *all*, *time*}.

5 Evaluation and Experimental Results

An obstacle to high-accuracy word segmentation is that there are no widely accepted guidelines for what constitutes a word; therefore, there is no agreement on how to "correctly" segment a text in a "desegmented" language. Native speakers of a language do not always agree about the "correct" segmentation, and the same text could be segmented into several very different (and equally correct) sets of words by different native speakers. Such ambiguity in the definition of what constitutes a word makes it difficult to evaluate segmentation algorithms that follow different conventions, as it is nearly impossible to construct a "gold standard" against which to directly compare results [7]. As shown in [23], the rate of agreement between two human judges on this task is less than 80%.

The performance of word segmentation is usually measured using *precision* and *recall*, where recall is defined as the percent of words in the manually segmented text identified by the segmentation algorithm, and precision is defined as the percentage of words returned by the algorithm that also occurred in the hand-segmented text in the same position. In general, it is easy to obtain high performance for one of the two measures but relatively difficult to obtain high performance for both. *F-measure* (*F*) is the geometric mean of *precision* (*P*) and *recall* (*R*) and expresses a trade-off between those two measures. These performance measures are defined as follows:

$P = TP / (TP + FP)$
$R = TP / (TP + FN)$
$F = (1 + \beta)PR / (\beta P + R)$
$= 2PR / (P + R)$, with $\beta = 1$ such that *precision* and *recall* weighted equally.

For instance, if the target segmentation is "we are human", and the model outputs "weare human", then precision is 1/2 ("human" out of "weare" and "human", recall is 1/3 ("human" out of "we", "are", and "human") and F-measure is 2/5.

We used the type frequency from BNC and tested our segmentation method on part of the Brown corpus. Specifically, we converted a portion of the corpus to lowercase letters and removed all white space and punctuation. We used 285K characters, 57904 tokens as our test data. We obtained 84.28% word precision rate, 81.63% word recall rate, and 82.93% word F-measure.

In a second test, we used the type frequency from BNC and tested our segmentation method on the Brown corpus to make sure that we test on different vocabulary from the training data. This insures that some of the word in the test set were not previously seen (out-of-vocabulary words). There were 4,705,022 characters and 1,003,881 tokens in the Brown corpus. We obtained 89.92% word precision rate, 94.69% word recall rate, and 92.24% word F-measure. The average number of tokens per line could be the reason for obtaining better result when we tested on the Brown corpus, as 8.49 and 16.07 are the average number of tokens per line in the Brown corpus and the BNC corpus, respectively.

One of the best known results on segmenting the Brown corpus is due to Kit and Wilks [15] who use a description-length gain method. They trained their model on the whole corpus (6.13M) and reported results on the *training* set, obtaining a boundary precision of 79.33%, a boundary recall of 63.01% and boundary F-measure of 70.23%. Peng and Schuurmans [18] trained their model on a subset of the corpus (4292K) and tested on *unseen* data. After the lexicon is optimized, they obtained 16.19% higher recall and 4.73% lower precision; resulting in an improvement of 5.2% in boundary F-measure. De Marcken [9] also used a minimum description length (MDL) framework and a hierarchical model to learn a word lexicon from raw speech. However, this work does not explicitly yield word boundaries, but instead recursively decomposes an input string down to the level of individual characters. As pointed out by Brent [1], this study gives credit for detecting a word if any node in the hierarchical decomposition spans the word. Under this measure [9] reports a word recall rate of 90.5% on the Brown corpus. However, his method creates numerous chunks and therefore only achieves a word precision rate of 17%. Christiansen *et al*. [5] used a simple recurrent neural network approach and report a word precision rate of 42.7% and word recall rate of 44.9% on spontaneous child-directed British English. Brent and Cartwright [2] used a MDL approach and reported a word precision rate of 41.3% and a word recall rate of 47.3% on the CHILDES collection. Brent [1] achieved about 70% word precision and 70% word recall by employing additional language modeling and smoothing techniques. Peng and Schuurmans [18] obtained 74.6% word precision rate, 79.2% word recall rate, and 75.4% word F-measure on the Brown corpus. A balance of high precision and high recall is the main advantage of our proposed method. However, it is difficult to draw a direct comparison between these results because of the different test corpora used by different authors.

Fig. 1 summarizes the result of different methods, which are tested on the Brown corpus based on precision, recall and F-measure. Though all the methods in Fig. 1 use the Brown corpus, the testing data sets in the Brown corpus are not exactly the same.

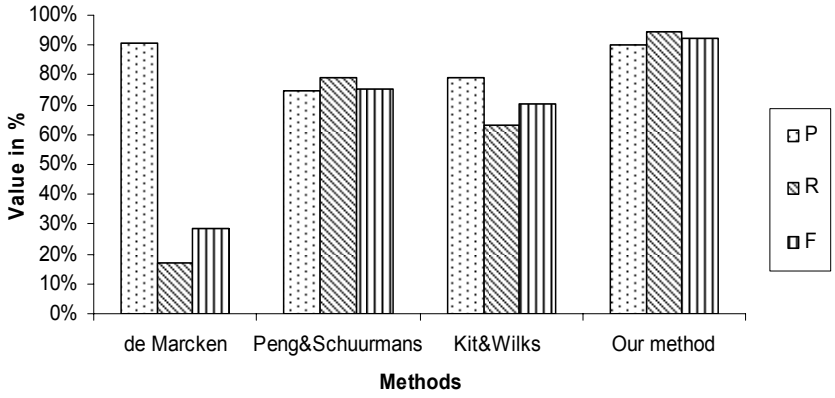

Fig. 1. Test result on the Brown corpus

6 Conclusion and Future Work

Actually the uses of *maximum length descending frequency* and entropy rate can effectively distill special terms and proper nouns when the corpus covers a huge collection of both domain-dependent and domain-independent words, and it can effectively avoid statistical errors on shorter strings which belong to a longer one. However, names are not always easy to exploit and contain abbreviations and special characters that vary between domains. This method can be used to address this issue, an important step of schema matching in databases. Top choices search engines segment the 'desegmented' part from a search text only if the 'desegmented' part contains two to three words. Even the popular search engine Google segments a 'desegmented' part of search text consisting of only two words and fails to provide any search result when the search text consists of more than two 'desegmented' words. Experimental results show that our method can segment words with high precision and high recall. Future directions also involve integrating the current algorithm into a larger system for comprehensive and context-based word analysis.

References

1. Brent, M.: An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning* 34, (1999) 71–106
2. Brent, M. and Cartwright, T.: Distributional regularity and phonotactics are useful for segmentation. *Cognition* 61, (1996) 93-125
3. Brill, E.: Some advances in transformation-based part of speech tagging. In: *Proc. of the Twelfth National Conference on Artificial Intelligence*, AAI Press/MIT Press, (1994) 748–753

4. Christiansen, M. and Allen, J.: Coping with Variation in Speech Segmentation. In *Proceedings of GALA 1997: Language Acquisition: Knowledge Representation and Processing*, (1997) 327-332
5. Christiansen, M., Allen, J. and Seidenberg, M.: Learning to Segment Speech Using Multiple Cues: A Connectionist Model. *Language and Cognitive Processes* 13, (1998) 221-268
6. Daelamans, W., van den Bosch, A. and Weijters, A.: IGTree: Using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, *11*, (1997) 407–423
7. Dale, R., Moisl, H. and Somers, H.: *Handbook of Natural Language Processing*. Marcel Dekker, Inc. New York (2000) 22-26
8. Deligne, S. and Bimbot, F.: Language Modeling by Variable Length Sequences: Theoretical Formulation and Evaluation of Multigrams. In *Proceedings ICASSP* (1995)
9. de Marcken, C.: The Unsupervised Acquisition of a Lexicon from Continuous Speech. *Technical Report AI Memo No. 1558, M.I.T., Cambridge, Massachusetts* (1995)
10. Do, H.H. and Rahm, E.: COMA – A System for Flexible Combination of Schema Matching Approaches. In *VLDB* (2002)
11. Fung, P. and Wu, D.: Improving Chinise tokenization with linguistic filters on statistical lexical acquisition. *Fourth Conference Applied Natural Language Processing*, Stuttgart (1994) 180-181
12. Gao, J., Li, M., Wu, A. and Huang, C.-N.: Chinese word segmentation and named entity recognition: a pragmatic approach. *Computational Linguistics*, 31(4) (2005)
13. Gelbukh, A., Alexandrov, M. and Han, S.Y.: Detecting Inflection Patterns in Natural Language by Minimization of Morphological Model. In *CIARP 2004*, LNCS 3287, (2004) 432-438
14. Hua, Y.: Unsupervised word induction using MDL criterion. In *Proceedings ISCSL2000*, Beijing (2000)
15. Kit, C. and Wilks, Y.: Unsupervised Learning of Word Boundary with Description Length Gain. In *Proceedings CoNLL99 ACL Workshop*. Bergen (1999)
16. Madhavan, J., Bernstein, P., Doan, A. and Halevy, A.: Corpus-based Schema Matching. In *International Conference on Data Engineering (ICDE-05)* (2005)
17. Mikheev, A.: Automatic rule induction for unknown word guessing. *Computational Linguistics*, *23*(3) (1997) 405–423
18. Peng, F. and Schuurmans, D.: A Hierarchical EM Approach to Word Segmentation, In *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium (NLPRS 2001)* Tokyo, Japan. (2001) 475-480
19. Rabiner, L.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Proceedings of IEEE*, 77(2) (1989)
20. Rumelhart, D.E. and McClelland, J.: On learning the past Tense of English verbs. In *Parallel distributed processing* Vol. II, Cambridge, MA: MIT Press (1986) 216–271
21. Saffran, J.R., Newport, E.L. and Aslin, R.N.: Word segmentation: The role of distributional cues. *Journal of Memory and Language 35*, (1996) 606–621
22. Shannon, C.E. and Weaver, W.: *The mathematical theory of communication*. Urbana: University of Illinois Press (1963)
23. Sproat, R., Shih, C., Gale, W. and Chang, N.: A stochastic finite-state word-segmentation algorithm for Chinese. *Computational Linguistics*, 22(3) (1996) 377–404
24. Sproat, R., Shih, C., Gale, W. and Chang, N.: A stochastic word segmentation algorithm for a Mandarin text-to-speech system. *32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, NM (1994) 66-72