Detect ‘protein word’ based on unsupervised word segmentation

Wang Liang[1], Zhao KaiYong[2]

[1]Sogou Tech, Beijing, 100080, P.R. China. [2] Department of Computer Science, Hong Kong Baptist University, HK, 999077, P.R. China. Correspondence to Wang Liang:wangliang.f@gmail.com

ABSTRACT

Unsupervised word segmentation methods were applied to analyze the protein sequence. Protein sequences, such as ‘MTMDKSELVQKA…..’, were used as input to these methods. Segmented ‘protein word’ sequences, such as ‘MTM DKSE LVQKA’, were then obtained. We compare the ‘protein words’ produced by unsupervised segmentation and the protein secondary structure segmentation. An interesting finding is that the unsupervised word segmentation is more efficient than secondary structure segmentation in expressing information. Our experiment also suggests there may be some ‘protein ruins’ in current noncoding regions.

Introduction

Word segmentation mainly refers to the process dividing a string of written language into its component words. For some East-Asian languages, such as Chinese, no spaces or punctuations are placed between letters. The ‘letter’ sequences must be segmented into word sequences to process the text of these languages. For example, ‘iloveapple’ is segmented into ‘I love apple’.

The main idea of segmentation method is very simple. For letter sequence ‘iloveapple’, it could be segmented into many forms of word sequences like ‘I love apple’, ‘ilove apple’, ‘il ove apple’, ‘iloveapple’, ‘ilo vea pple’,etc. We select ‘word sequences’ having the highest probability as its segmentation. For example,

P(‘I love apple’) = 0.8,

P(‘ilove apple’) = 0.3,

P(‘iloveapple’) = 0.03.

We will select ‘I love apple’ as the segmentation of letter sequence ‘iloveapple’.

The probability of a word sequence is calculated by the probability of multiplications of each word.

For example:

P(‘I love apple’)= P(i) *P(love) *P(apple).

P(‘ilove apple’)= P(ilove) *P(apple).

So the key for segmentation is to obtain the probability of every word.

If we have many segmented sequences like ‘i love apple’ , we could simply calculate the probability of every word according to the division of its occurrence divided by all words occurrences. For example, we have two sequences ‘i love apple’ and ‘i love mac’, there are 6 words occurrences. ‘i’ appears 2 times, so its probability P(‘i’)=2/6. ‘apple’ appears 1 times, its probability P(‘apple’)=1/6. This method is called supervised segmentation methods. For the words like ‘ilove’ which doesn’t appear in corpus, we give them a very small probability.

If we have no any segmented sequence, but many letter sequences like ‘iloveapple’, we could still calculate the probability of all possible words and build the word segmentation method. Its main idea is described as follows:

First, it gives the random or equal probability for every possible word. Normally, we need set a maximal word length. For example, for ‘iloveapple’, if we set 4 as the maximal word length, all possible words are ‘ilov’,’love’,’ovea’,etc.

Secondly, because we have had the word list with probability, we could segment all letter sequences into ‘word sequences’.

Thirdly, we calculate the new word probability according to the ‘word sequences’ in the second step.

We repeat the second and third steps, until the probabilities of all words reach a stable value. Then we could use this word list with probability to segment letter sequences. This method is called unsupervised segmentation method [1].

The basic theory of word segmentation is shown in Fig.1.

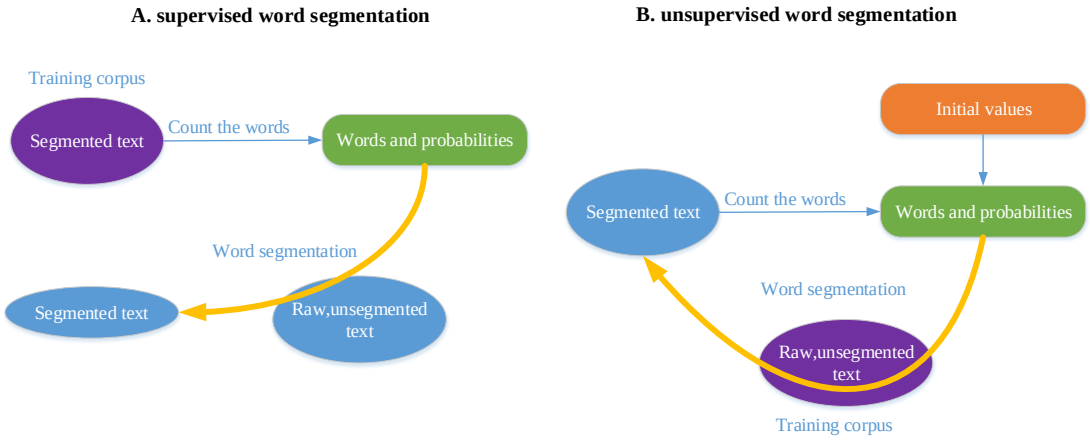

Fig1.A, supervised word segmentation. B. unsupervised word segmentation.

For English text, the accuracy of supervised segmentation is more than 95%. The unsupervised segmentation method could reach about 75%. Supervised methods are better than unsupervised methods; however, unsupervised approaches are highly adaptive to relatively unfamiliar ‘languages’

for which we do not have enough linguistic knowledge [2].

Unsupervised segmentation could be described as 'Inputting letter sequences, outputting meaningful word sequences'. If the input is protein sequences like 'MTMDKSELVQKA…..' , what will we get? This is just the topic of this paper.

In the following sections, we will run the unsupervised segmentation for protein sequences. Identification of functional equivalents of 'words' in protein sequences is a fundamental problem in bioinformatics. The protein secondary structure elements is normally regarded as the functional building blocks of protein sequences. So we compare our 'protein word' sequences and related secondary structure segmentation in section 3. We find our 'protein word segmentation' seems a better encoding method than secondary structure segmentation in expressing information. In section 4, we discuss this confusing result. The last section is a short summary.

Word segmentation experiment

The soft-counting method is a representative unsupervised segmentation methods, which use the standard EM algorithm to estimate the probability of any possible substring [1]. This operation produces a vocabulary with probability. Then to segment a sequence, a Viterbi-style algorithm is applied to search for the segmentations with the highest probability.

Unsupervised segmenting methods require only raw letter sequences. Here we use the amino acid letter sequences. Data are selected from the PDB structure database. The PDB dataset contains about 100,000 protein sequences. These protein sequences are used as input to the unsupervised segmentation method. Then we have:

Definition 1: **'protein word',** the segments produced by word segmentation method.

The unsupervised segmentation only need select a maximal word length. Here we set 9 as the maximal protein word length. This value could be adjusted according to the data size.

There are about 26 millions of possible protein words in our data. The frequency and border information features are used to filter these words [3-6]. After running the soft counting for these protein sequences, we get a final protein vocabulary containing about 630,598 words. Then we could use this vocabulary containing word probability to segment protein sequence. A protein word segmentation example is shown as follows:

Original protein sequence:

DADLAKKNNCIACHQVETKVVGPALKDIAAKYADKDDAATYLAGKIKGGSSGVWG QI PMPPNVNVSDADAK LADWILTLK

Corresponding segmented protein word sequence:

D ADLAKK NNCI ACH QVET KV VGPAL KDIAAK YADK D DAATYL AGKIK GGSSGV WGQI PMPPN VNVSD ADAKA LADWI LTLK

Comparing with secondary structure segmentation

For most human language texts, output of segmentation is meaningful words sequence. As we expected, so does for protein sequence. We compare protein word sequence with the protein secondary structure elements, which act as the functional building blocks of protein.

Protein secondary structure mainly refers to the local conformation of the polypeptide backbone of proteins that is often discretely classified into a few states. The word segmentation is very similar to the protein secondary structure assignment process. For protein sequences, such as "MTMDKSELVQKA", the corresponding consecutive amino acids of the same secondary structure can also be regarded as a 'structure word'. This process could be called secondary structure segmentation (**Fig. 2**). There are 437,537 distinct 'structure word' in our data set.

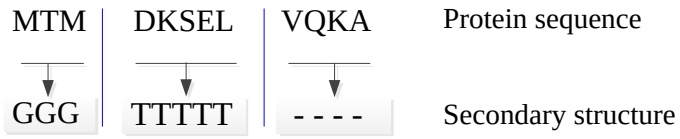

Fig.2 Secondary structure segmentation. The protein sequence is segmented by its secondary structure. This sequence contains three secondary structure words, 'MTM', 'DKSEL', and 'VQKA'.

Definition 2: **'structure word',** the protein segments produced by secondary structure segmentation.

The segmentation performance is normally evaluated using the boundary F-score measure, F = 2RP/(R + P). The recall R and precision P are the proportions of the correctly recognized word boundaries to all boundaries in the gold-standard and an output for word segmentation of a segmenter, respectively. Here we use the structure segmentation as gold-standard (**Fig.3**).

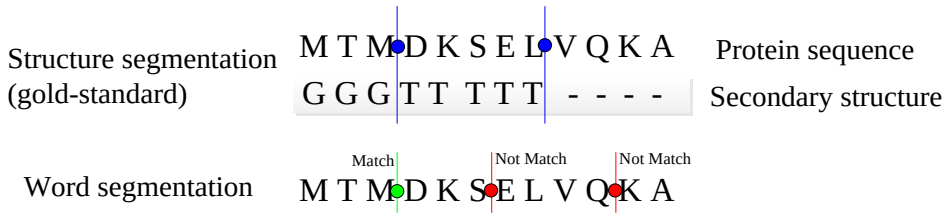

Fig.3 Evaluation of word segmentation. There are 2 segmentation points in structure segmentation and 3 segmentation points in unsupervised word segmentation, 1 point match. So the precision is 1/33≈3% and the recall are 1/2=50%. The F-score is 2*0.33*0.5/(0.33+0.5)≈39.8%

For soft-counting method, the boundary precision is 39.9%, the boundary recall is 28.0% and the boundary f-score is 32.9%. A segmentation example is shown in Fig.4.

MVLS EGEWQLVLHVWAKV EAD VAGHGQDILIRLFKS H PETLEK F DRVKH L Structure segmentation
- - - - HHHHHHHHHHHHHH GGG HHHHHHHHHHHHHHH - GGGGGG - TTTTT - Secondary structure

MV LSEGE WQLV LHVW A KVEAD VAGH GQDIL IRLF K SHPET LEKFD R VKHL Soft-counting word segmentation

Fig.4 structure segmentation and word segmentation of a protein sequence

The boundary f-score is about 85% for unsegmented English text using soft-counting method. For Chinese text, the boundary f-score is about 80%. For protein word segmentation, such value is only 32.9%. So protein word segmentation is different from secondary structure segmentation. This result doesn't meet our expectation. We will analyze their differences in the next section.

Encoding efficiency of segmentation

As a basic statistical feature, the words occurrence distribution may explain the differences of two kinds of segmentations.

In a typical English date set, if we regard the top 10% frequency word in vocabulary as high frequency word, occurrences of this words normally account for about 90% of all letters in date set. The low frequency word, for example, 1 frequency words account for about 50% of all words in vocabulary, but their occurrences only account for about 1% of letters of whole corpus. This is an 'efficient' law in most human language. It's also a fundamental assumption for most unsupervised segmentation methods. Related values of soft-counting segmentation and secondary structure segmentation are shown in Table 1：

Table 1: words occurrence distribution

	Letter percentage of high frequency words in data set	Letter percentage of low frequency words in data set
English text	80%	3%
Soft-counting segmentation	34%	4%
Structure segmentation	40%	58%

From Table.1, we find the secondary structure segmentation seems not so 'efficient'. Mainly because its vocabulary contains too many low frequency words. We could use the 'Description Length (DL)' to describe the 'efficient' of segmentation [7]. The segmentation process could be regarded as an encoding process, replacing the letters of a word with related word symbol. A codebook in which each word is represented by a unique string can be used to encode a corpus of

words as efficiently as possible. The total number of letters required to encode the corpus (sum of the lengths of the codebook and encoded corpus) using a well-designed codebook/vocabulary would be less than the original corpus. Smaller units, such as morphemes or phonemes, which require fewer code words and thus a shorter codebook, can be encoded further. However, efficiently encoding the corpus becomes more difficult using fewer code words. Meanwhile, some words may never be used when too many words are in the codebook. Thus the length of the codebook and the length of the encoded corpus must be balanced. The Description length principle states that a codebook that leads to the shortest total combined length must be chosen. So 'Description Length' can be used to describe segmentation efficiency.

The description length for structure segmentation and soft-counting segmentation is shown in Fig.5:

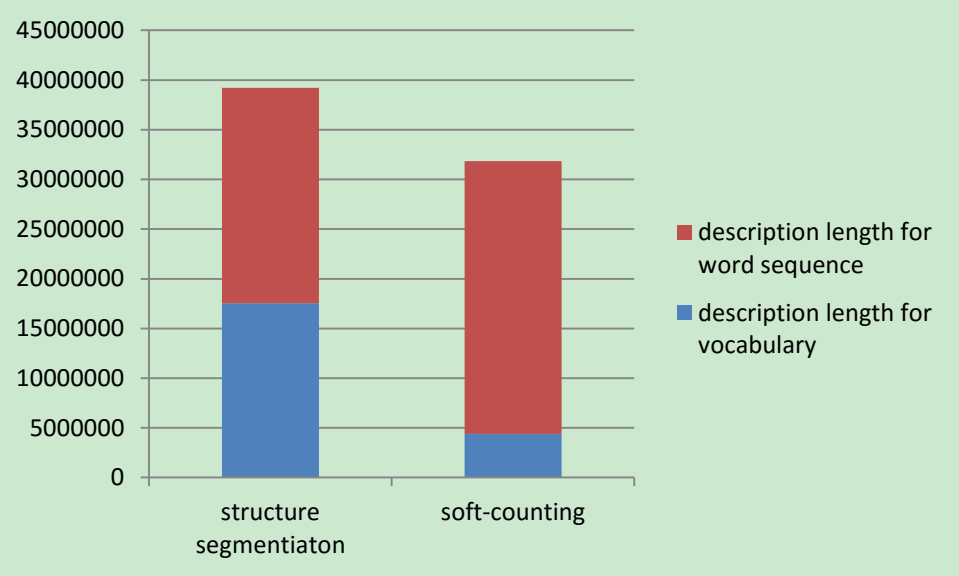

Fig.5 Description length value of structure segmentation and soft-counting segmentation

From Fig.5, we could find the whole description length of structure segmentation is more than that of soft-counting segmentation. In structure segmentation, about 45% description length value is used for vocabulary, in soft-counting segmentation, this value is only 16%. Structure segmentation designs a large 'codebook', but only few of its abilities are used. Thus, secondary structure segmentation maybe not a good encoding method. There may be two opposite explanations for this result:

First, current structure segmentation is 'really' not efficient. This means we can find a different structure assignment method, whose structure segments could express gene's functions better. Our data set uses the DSSP secondary structure definition. We try other kinds of secondary structure definitions like STRIDE [8], but get the similar low 'efficient' segmentation.

Secondly, the structure segmentation scheme is 'truly' efficient. Since the segmentation method is right, there must be some problems in data set. A very common problem in date set construction process is data unbalanced problem. For example, if we only select several paragraphs of a long

book to build the data set, this problem will appear.

We construct the experimental data by filter the similar protein sequences in PDB. The PDB protein data could also be regarded as randomly selected from all available protein data. There is also no clear problem in our data construction process. So if our data set is unbalanced, it means all current protein data is unbalanced. If so, there must be more 'protein words' in current non-coding regions. For example, some short DNA regions could be translated into 'protein words', but such 'protein words' covering region maybe not comply with the strict definition of 'protein' sequence. We have:

Definition 3: **'protein word covering region',** the DNA region which could be translated into 'protein words'.

We could also detect these 'protein word covering region' by segmentation methods. We back translate the protein words into DNA forms and get a DNA vocabulary. Then we could use this vocabulary to segment the DNA sequences based on supervised segmentation methods. For example, a DNA sequence 'ATGGTTTTGTGTCTTGAGGGT' is segmented into:

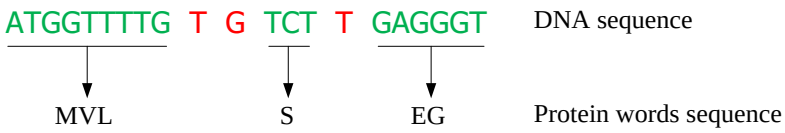

Fig.6 Segment DNA sequence based on protein word vocabulary. The green parts could be translated into protein words. The red part can't be translated. (Attention: this sequence is not a protein sequence. Only part regions could be covered by 'protein words')

Here is a typical example of DNA sequence segmentation:

TTGTGG G TTGTCTATTGATGTT TTTGGT C TATTCTAAG A A TTGGAG A GAG A GAGGTT A A AAT C TCT G ACT ATG A TTGTGG A TTGTCT GCTGAT GTT TTTGGT C TATTGTTCT AAGAAT TGG A GAG A GAG A GAGGTT A A A A TCT C C G ACT ATG A TTGTGG A TTGTCT ATTGCTGTTTTT GGT C TATTGTTCT AAGAAT

This sequence is part of "gi|157704448|ref|AC_000133.1| Homo sapiens chromosome 1, alternate assembly HuRef, whole genome shotgun sequence:307501-308000". The green segments could be translated into 'protein words'. Although its 87% region is protein word covering region, there is no distinct introns, exons and stop codon, etc. So such sequence can't be regarded as a coding sequence.

We use human genome to judge whether data unbalanced problem could be improved if adding all 'protein words covering regions'. The genome is divided into 500bps equal length sequences. We use two vocabularies, soft-counting words vocabulary and secondary structure words vocabulary, to segment these DNA sequences. Then we calculate their Description Length respectively. The results are shown in Fig.7:

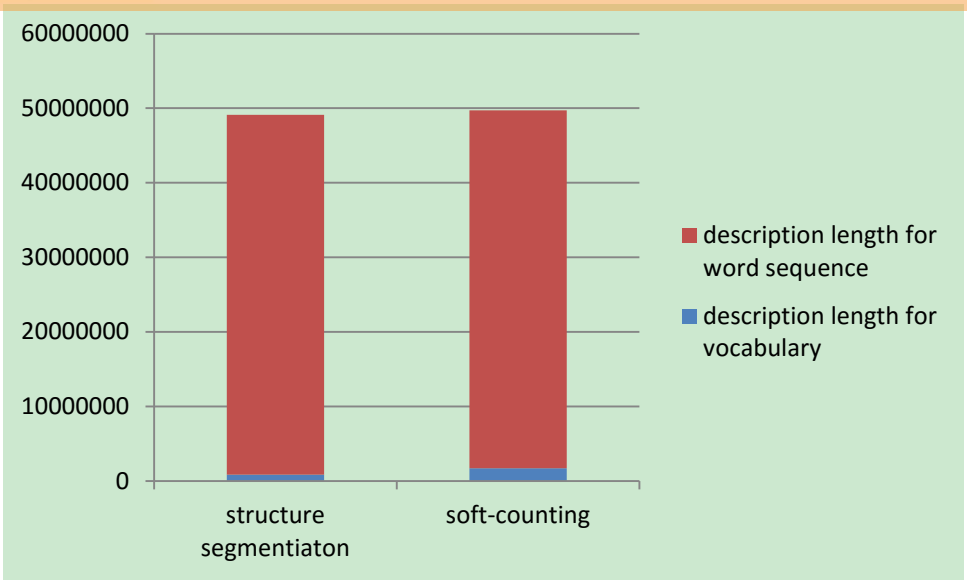

Fig.7 Description length value of structure segmentation and soft-counting segmentation

We find two kinds of segmentations have the similar Description Length. Then we convert all DNA 'protein word covering regions' into protein word sequences. These sequences is regarded as gold-standard segmentation. Then we delete the space between words and run the unsupervised segmentation test again. The boundary precision reaches 67%, the boundary recall reaches 60% and boundary f-score reaches 63%. This result complies with our conjecture. Adding 'protein word covering regions' could improve the data balance problem and make the structure segmentation more efficient.

The distribution of 'protein word covering regions' is show in Fig.8:

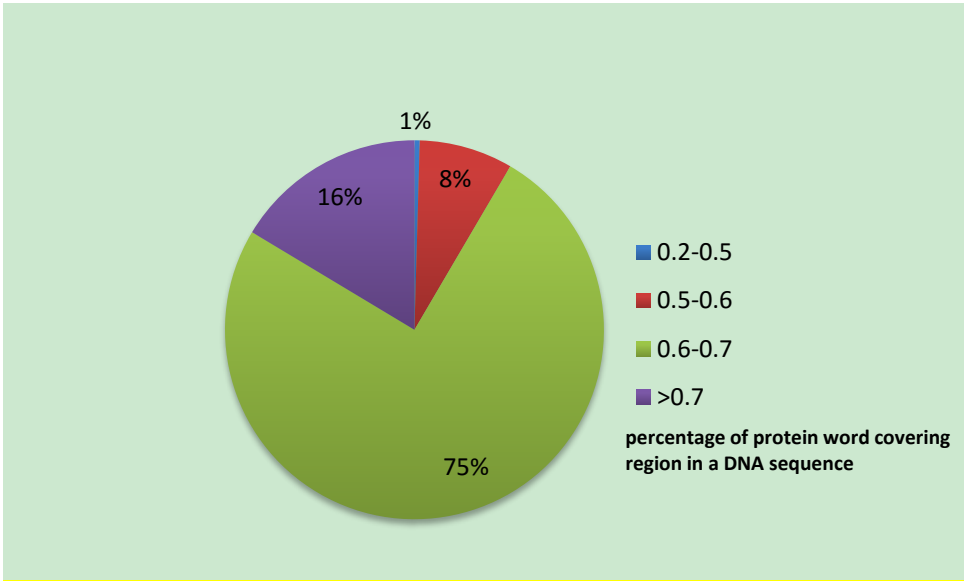

Fig.8 Distribution of protein word covering sequence in Homo sapiens chromosome 1. For example, there are about 75% DNA sequences whose percentage of protein word encoding region lies in [0.6, 0.7].

We find there are about 16% DNA sequences whose percentage of 'protein word covering region'

is more than 70%. Among these sequences, only about 10% are really gene area, the other sequences may be regarded as the 'protein ruins', which may be abandoned in evolution.

Conclusions

We use the protein sequences as input to run the word segmentation test. As we expected, the output should be some meaningful 'protein word' sequences. So we compare them with the secondary structure segments, which are normally regarded as the basic functional block of protein. But we find two kinds of segmentations are different. Then we use the "Description Length" to analyze their difference and find the word segmentation is even more efficient than structure segmentation. This result is in conflict with our common sense. So we guess this problem is mainly caused by the data unbalanced problem. That means the gene is a large book and current 'protein' sequences are only several paragraphs of this book. Our DNA segmentation experiments also support this conjecture.

All in all, the 'protein word' and 'protein word cover region' may have no any biological meaning. Word segmentation is a fundamental technology in processing Chinese and some other languages, which have no any space or punctuation between words, just like protein sequence. Word segmentation is the preliminary step for their search engine system, translation system and proofread system, etc. The main aim of this paper is also to spark more new ideas in applying these technologies in protein sequence analyzing. The word segmentation is just a bridge.

MATERIALS AND METHODS

Protein Data

Unsupervised segmenting methods only need raw letter sequence. We mainly use the data of PDB (http://www.rcsb.org/pdb/) as our experiment data. This dataset contain about 100,000 pieces of protein sequences. We use CD-HIT algorithms to delete the similar protein sequence. Its codes could be found in : http://weizhong-lab.ucsd.edu/cd-hit/download.php . We also use some protein sequence data of website "uniprot.org" (http://www.uniprot.org/downloads), which is a central repository of amino acid sequence. We select the UniProtKB/Swiss-Prot and UniProtKB/TrEMBL.

Method and source code

All source codes and experiment instructions of this paper could be found in:
https://github.com/maris205/secondary_structure_detection

Maximal word length evaluation

We could use "zipf's laws" to evaluate the length of words. The "zipf's laws" states, in a long enough document, about 50% words only occur once. These words are called 'Hapax legomenon'.

Because we have no any structure information, we could construct words by intersecting segmenting the amino acid sequences and calculate the percentage of 'Hapax legomenon'. For instance, the sequence "AAAQL", assume the maximal word length is 2. All there intersecting segmentation is A, A, A, Q, L, AA, AA, RQ, RL. There are 6 different words, AA,AC,CG. Q, L,RQ and RL appears once, which are 'Hapax legomenon'. So there are 4 'Hapax legomenon'. Its percentage is 4/6, about 66%. If for a maximal word length, the percentage of "Hapax legomenon" near 50%. This length could be regarded as word length of most word in data set. The maximal word length could be set according to this length. For our protein sequence, the relation of word length 'n' of intersecting segmentation and the percentage of 'Hapax legomenon' is show in Fig.9:

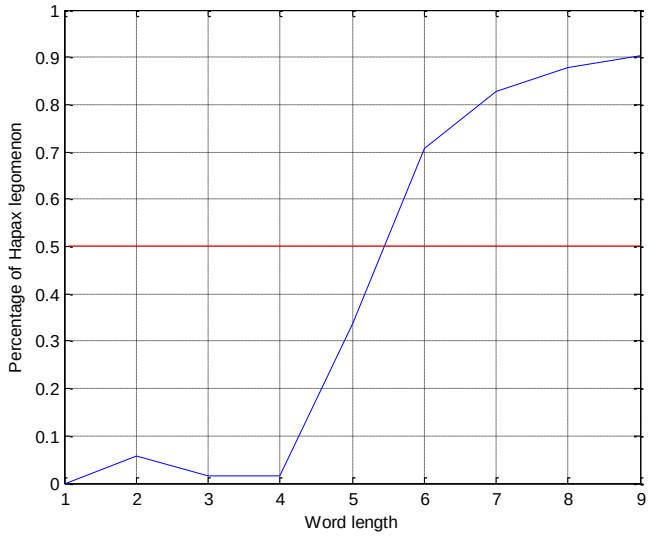

Fig.9 relation of word length and percentage of hapax legomenon (x axis is the word length, y axis is the percentage of hapax legomenon)

In Fig.9, we find 50% line of 'Hapax legomenon' near word length 6. So we set the 9 as the maximal word length for unsupervised methods. If we have more corpus, we could set a longer word length.

References and Notes

1. Xiaping Ge, Wanda Prat, Padhratic Smyth. Discovering Chinese words from unsegmented text. Proceedings on the 22 Annual International ACM SIGIR Conference On Research and Development in Information Retrieval. **USA**, 217-272(1999).
2. H Wang, J Zhu, S Tang, X Fan. A new unsupervised approach to word segmentation.

Computational Linguistics. **35**,421-454(2011).

3. Paul Cohen, Niall Adams, Brent Heeringa. Voting Experts: An unsupervised algorithm for segmenting sequences. Intell. Data Anal. **11**,607-625(2007).
4. Daniel Hewlett, Paul Cohen. Word segmentation as general Chunking. Proceedings of the Fifteenth Conference on Computational Natural Language Learning. **USA** , 39–47(2011).
5. Hai Zhao, Chunyu Kit. Exploiting unlabeled text with different unsupervised segmentation criteria for Chinese word segmentation. Proceedings of the Conference on Empirical Methods in Natural Language Processing. **Israel**, 17-23(2008).
6. Hai Zhao,Chunyu Kit. An empirical comparison of goodness measures for unsupervised Chinese word segmentation with a unified framework. The Third International Joint Conference on Natural Language Processing. **India 1**, 9-16(2008).
7. Chunyu Kit, Yorick Wilks. Unsupervised learning of word boundary with description length gain. CoNLL. **Norway**, 1-6(1999).
8. Zhang W, Dunker AK, Zhou Y. Assessing secondary structure assignment of protein structures by using pairwise sequence-alignment benchmarks.Proteins.**71**.(61-7)2008