N-gram Based Two-Step Algorithm for Word Segmentation

Dong-Hee Lim
Dept. of Computer Science
Kookmin University
Seoul 136-702, Korea
nlp@cs.kookmin.ac.kr

Kyu-Baek, Hwang
School of Computing
Soongsil University
Seoul 156-743, Korea
kbhwang@ssu.ac.kr

Seung-Shik Kang
Dept. of Computer Science
Kookmin University
Seoul 136-702, Korea
sskang@kookmin.ac.kr

Abstract

This paper describes an n-gram based reinforcement approach to the closed track of word segmentation in the third Chinese word segmentation bakeoff. Character n-gram features of unigram, bigram, and trigram are extracted from the training corpus and its frequencies are counted. We investigated a step-by-step methodology by using the n-gram statistics. In the first step, relatively definite segmentations are fixed by the tight threshold value. The remaining tags are decided by considering the left or right space tags that are already fixed in the first step. Definite and loose segmentation are performed simply based on the bigram and trigram statistics. In order to overcome the data sparseness problem of bigram data, unigram is used for the smoothing.

1 Introduction

Word segmentation has been one of the very important problems in the Chinese language processing. It is a necessary in the information retrieval system for the Korean language (Kang and Woo, 2001; Lee et al, 2002). Though Korean words are separated by white spaces, many web users often do not set a space in a sentence when they write a query at the search engine. Another necessity of automatic word segmentation is the index term extraction from a sentence that includes word spacing errors.

The motivation of this research is to investigate a practical word segmentation system for the Korean language. While we develop the system, we found that ngram-based algorithm was exactly applicable to the Chinese word segmentation and we have participated the bakeoff (Kang and Lim, 2005). The bakeoff result is not satisfiable, but it is acceptable because our method is language independent that does not consider the characteristics of the Chinese language. We do not use any language dependent features except the average length of Chinese words.

Another advantage of our approach is that it can express the ambiguous word boundaries that are error-prone. So, there are a good possibility of improving the performance if language dependent functionalities are added such as proper name, numeric expression recognizer, and the postprocessing of single character words.[1]

2 N-gram Features

The n-gram features in this work are similar to the previous one in the second bakeoff. The basic segmentation in (Kang and Lim, 2005) has performed by bigram features together with space tags, and the trigram features has been used as a postprocessing of correcting the segmentation errors. Trigrams for postprocessing are the ones that are highly biased to one type of the four tag features of "A_iB_jC".[2] In addition, unigram features are used for smoothing the bigram, where bigram is not found in the training corpora. In this current work, we extended the n-gram features to a trigram.

(a) trigram: A_iB_jC
(b) bigram: ${}_iA_jB_k$
(c) unigram: ${}_iA_j$

In the above features, AB and ABC are a Chinese character sequence of bigram and trigram, respectively. The subscripts i, j, and k

[1] Single character words in Korean are not so common, compared to the Chinese language. We can control the occurrence of them through an additional processing.
[2] We applied the trigrams for error correction in which one of the trigram feature occupies 95% or more.

Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing, pages 197–200,
Sydney, July 2006.

denote word space tags, where the tags are marked as 1(space tag) and 0(non-space tag). For the unigram ${}_iA_j$, four types of tag features are calculated in the training corpora and their frequencies are stored. In the same way, eight types of bigram features and four types of trigram features are constructed. If we take all the inside and outside space-tags of ABC, there are sixteen types of trigram features ${}_hA_iB_jC_k$ for *h,i,j,k = 0 or 1*. It will cause a data sparseness problem, especially for small-sized training corpora. In order to avoid the data sparseness problem, we ignored the outside-space tags *h* and *k* and constructed four types of trigram features of A_iB_jC.

Table 1 shows the number of n-gram features for each corpora. The total number of unique trigrams for CITYU corpus is 1,341,612 in which 104,852 trigrams occurred more than three times. It is less than one tenth of the total number of trigrams. N-gram feature is a compound feature of <character, space-tag> combination. Trigram classes are distinguished by the space-tag context, trigram class ${}_hA_iB_jC_k$ is named as t4-trigram or C3T4.[3] It is simplified into four classes of C3T2 trigrams of A_iB_jC, in consideration of the memory space savings and the data sparseness problem.

Table 1. The number of n-gram features

	Trigram				Bigram	Unigram
	freq≥1	freq≥2	freq≥3	freq≥4	freq≥1	freq≥1
cityu	1341612	329764	165360	104852	404411	5112
ckip	2951274	832836	444012	296372	717432	6121
msra	986338	252656	132456	86391	303443	4767
upuc	463253	96860	45775	28210	177140	4293

3 Word Segmentation Algorithm

Word segmentation is defined as to choose the best tag-sequence for a sentence.

$$\hat{T} = \arg\max_{T \in \tau} P(T \mid S)$$

where

$$T = t_1, t_2, \ldots, t_n \text{ and } S = c_1, c_2, \ldots, c_n$$

More specifically at each character position, the algorithm determines a space-tag '0' or '1' by using the word spacing features.

3.1 The Features

We investigated a two step algorithm of determining space tags in each character position of a sentence using by context dependent n-gram features. It is based on the assumption that space tags depend on the left and right context of characters together with the space tags that it accompanies. Let t_ic_i be a current <space tag, character> pair in a sentence.[4]

$$\ldots\ t_{i-2}c_{i-2}\ t_{i-1}c_{i-1}\ \mathbf{t_ic_i}\ t_{i+1}c_{i+1}\ t_{i+2}c_{i+2}\ \ldots$$

In our previous work of (Lim and Kang, 2005), n-gram features (a) and (b) are used. These features are used to determine the space tag t_i. In this work, core n-gram feature is a C3T2 classes of trigram features $c_{i-2}t_{i-1}c_{i-1}\mathbf{t_ic_i}$, $c_{i-1}\mathbf{t_ic_i}t_{i+1}c_{i+1}$. In addition, a simple character trigram with no space tag "$\mathbf{t_ic_i}c_{i+1}c_{i+2}$" is added.

(a) unigram:
$t_{i-1}c_{i-1}\mathbf{t_i}$, $\mathbf{t_ic_i}t_{i+1}$
(b) bigram:
$t_{i-2}c_{i-2}t_{i-1}c_{i-1}\mathbf{t_i}$, $t_{i-1}c_{i-1}\mathbf{t_ic_i}t_{i+1}$, $\mathbf{t_ic_i}t_{i+1}c_{i+1}t_{i+2}$
(c) trigram:
$c_{i-2}t_{i-1}c_{i-1}\mathbf{t_ic_i}$, $c_{i-1}\mathbf{t_ic_i}t_{i+1}c_{i+1}$, $\mathbf{t_ic_i}c_{i+1}c_{i+2}$

Extended n-gram features with space tags are effective when left or right tags are fixed. Suppose that t_{i-1} and t_{i+1} are definitely set to 0 in a bigram context "$t_{i-1}c_{i-1}\mathbf{t_ic_i}t_{i+1}$", then a feature "$0c_{i-1}\mathbf{t_ic_i}0$"($t_i$ = 0 or 1) is applied, instead of a simple feature "$c_{i-1}\mathbf{t_ic_i}$". However, none of the space tags are fixed in the beginning that simple character n-gram features with no space tag are used.[5]

3.2 Two-step Algorithm

The basic idea of our method is a cross checking the n-gram features in the space position by using three trigram features. For a character sequence "$c_{i-2}c_{i-1}\mathbf{t_ic_i}c_{i+1}c_{i+2}$", we can set a space mark '1' to t_i, if $P(t_i=1)$ is greater than $P(t_i=0)$ in all the three trigram features $c_{i-2}c_{i-1}\mathbf{t_ic_i}$, $c_{i-1}\mathbf{t_ic_i}c_{i+1}$, and $\mathbf{t_ic_i}$ $c_{i+1}c_{i+2}$. Because no space tags are determined in

[3] 'Cn' refers to the number of characters and 'Tn' refers to the number of spae-tag. According to this notation, ${}_iA_jB_k$ and ${}_iA_j$ are expressed as C2T3 and C1T2, respectively.

[4] Tag t_i is located before the character, not after the character that is common in other tagging problem like POS-tagging.

[5] Simple n-grams with no space tags are calculated from the extended n-grams.

the beginning, word segmentation is performed in two steps. In the first step, simple n-gram features are applied with strong threshold values (t_{low1} and t_{high1} in Table 2). The space tags with high confidence are determined and the remaining space tags will be set in the next step.

Table 2. Strong and weak threshold values[6]

	t_{low1}	t_{high1}	t_{low2}	t_{high2}	t_{final}
cityu	0.36	0.69	0.46	0.51	0.48
ckip	0.37	0.69	0.49	0.51	0.49
msra	0.33	0.68	0.46	0.47	0.46
upuc	0.38	0.69	0.45	0.47	0.47

In the second step, extended bigram features are applied if any one of the left or right space tags is fixed in the first step. Otherwise, simple bigram probability will be applied, too. In this step, extended bigram features are applied with weak threshold values t_{low2} and t_{high2}. The space tags are determined by the final threshold t_{final}, if it was not determined by weak threshold values. Considering the fact that average length of Chinese words is about 1.6, the threshold values are lowered or highered.[7]

In the final step, error correction is performed by 4-gram error correction dictionary. It is constructed by running the training corpus and comparing the result to the answer. Error correction data format is 4-gram. If a 4-gram $c_{i-2}c_{i-1}c_ic_{i+1}$ is found in a sentence, then tag t_i is modified unconditionally as is specified in the 4-gram dictionary.

4 Experimental Results

We evaluated our system in the closed task on all four corpora. Table 3 shows the final results in bakeoff 2006. We expect that R_{oov} will be improved if any unknown word processing is performed. R_{iv} can also be improved if lexicon is applied to correct the segmentation errors.

Table 3. Final results in bakeoff 2006

	R	P	F	R_{oov}	R_{iv}
cityu	0.950	0.949	0.949	0.638	0.963
ckip	0.937	0.933	0.935	0.547	0.954
msra	0.933	0.939	0.936	0.526	0.948
upuc	0.915	0.896	0.905	0.565	0.949

[6] Threshold values are optimized for each training corpus.
[7] The average length of Korean words is 3.2 characters.

4.1 Step-by-step Analysis

In order to analyze the effectiveness of each step, we counted the number of space positions for sentence by sentence. If the number of characters in a sentence is *n*, then the number of words positions is *(n-1)* because we ignored the first tag t_0 for c_0. Table 4 shows the number of space positions in four test corpora.

Table 4. The number of space positions

	# of space positions	# of spaces	# of non-spaces
cityu	356,791	212,662	144,129
ckip	135,123	80,387	54,736
msra	168,236	95,995	72,241
upuc	251,418	149,747	101,671

As we expressed in section 3, we assumed that trigram with space tag information will determine most of the space tags. Table 5 shows the application rate with strong threshold values. As we expected, around 93.8%~95.9% of total space tags are set in step-1 with the error rate 1.5%~2.8%.

Table 5. N-gram results with strong threshold

	# of applied (%)	# of errors (%)
cityu	342,035 (95.9%)	5,024 (1.5%)
ckip	128,081 (94.8%)	2,818 (2.2%)
msra	160,437 (95.4%)	3,155 (2.0%)
upuc	235,710 (93.8%)	6,601 (2.8%)

Table 6 shows the application rate of n-gram with weak threshold values in step-2. The space tags that are not determined in step-1 are set in the second step. The error rate in step-2 is 24.3%~30.1%.

Table 6. N-gram results with weak threshold

	# of applied (%)	# of errors (%)
cityu	14,756 (4.1%)	3,672 (24.9%)
ckip	7,042 (5.2%)	1,710 (24.3%)
msra	7,799 (4.6%)	2,349 (30.1%)
upuc	15,708 (6.3%)	4,565 (29.1%)

4.2 4-gram Error Correction

We examined the effectiveness of 4-gram error correction. The number of 4-grams that is extracted from training corpora is about 10,000 to 15,000. We counted the number of space tags that are modified by 4-gram error correction dictionary. Table 7 shows the number of modified space tags and the negative effects of 4-gram error correction. Table 8 shows the results before error correction. When compared with the final results in Table 3, F-measure is slightly lower than the final results.

Table 7. Modified space tags by error correction

	# of modified space tags (%)	Modification errors (%)
cityu	418 (0.1%)	47 (11.2%)
ckip	320 (0.2%)	94 (29.4%)
msra	778 (0.5%)	153 (19.7%)
upuc	178 (0.1%)	61 (34.3%)

Table 8. Results before error correction

	R	P	F
cityu	0.948	0.947	0.948
ckip	0.935	0.931	0.933
msra	0.930	0.930	0.930
upuc	0.915	0.895	0.905

5 Conclusion

We described a two-step word segmentation algorithm as a result of the closed track in bakeoff 2006. The algorithm is based on the cross validation of the word spacing probability by using n-gram features of <character, space-tag>. One of the advantages of our system is that it can show the self-confidence score for ambiguous or feature-conflict cases. We have not applied any language dependent resources or functionalities such as lexicons, numeric expressions, and proper name recognition. We expect that our approach will be helpful for the detection of error-prone tags and the construction of error correction dictionaries when we develop a practical system. Furthermore, the proposed algorithm has been applied to the Korean language and we achieved a good improvement on proper names, though overall performance is similar to the previous method.

Acknowledgements

This work was supported by the Korea Science and Engineering Foundation(KOSEF) through Advaned Information Technology Research Center(AITrc).

References

Asahara, M., C. L. Go, X. Wang, and Y. Matsumoto, Combining Segmenter and Chunker for Chinese Word Segmentation, Proceedings of the 2nd SIGHAN Workshop on Chinese Language Processing, pp.144-147, 2003.

Chen, A., Chinese Word Segmentation Using Minimal Linguistic Knowledge, SIGHAN 2003, pp.148-151, 2003.

Gao, J., M. Li, and C.N. Huang, Improved Source-Channel Models for Chinese Word Segmentation, ACL 2003, pp.272-279, 2003.

Kang, S. S. and C. W. Woo, Automatic Segmentation of Words using Syllable Bigram Statistics, Proceedings of NLPRS'2001, pp.729-732, 2001.

Kang, S. S. and D. H. Lim, Data-driven Language Independent Word Segmentation Using Character-Level Information, Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing, pp.158-160, 2005.

Lee D. G, S. Z. Lee, and H. C. Rim, H. S. Lim, Automatic Word Spacing Using Hidden Markov Model for Refining Korean Text Corpora, Proc. of the 3rd Workshop on Asian Language Resources and International Standardization, pp.51-57, 2002.

Maosong, S., S. Dayang, and B. K. Tsou, Chinese Word Segmentation without Using Lexicon and Hand-crafted Training Data, Proceedings of the 17th International Conference on Computational Linguistics (Coling'98), pp.1265-1271, 1998.

Nakagawa, T., Chinese and Japanese Word Segmentation Using Word-Level and Character-Level Information, COLING'04., pp.466-472, 2004.

Ng, H.T. and J.K. Low, Chinese Part-of-Speech Tagging: One-at-a-Time or All-at-Once? Word-Based or Character-Based, EMNLP'04, pp.277-284, 2004.

Shim, K. S., Automated Word-Segmentation for Korean using Mutual Information of Syllables, Journal of KISS: Software and Applications, pp.991-1000, 1996.

Sproat, R. and T. Emerson, The First International Chinese Word Segmentation Bakeoff, SIGHAN 2003.