A Novel Word Segmentation Approach for Written Languages with Word Boundary Markers

Han-Cheol Cho[†], Do-Gil Lee[§], Jung-Tae Lee[§], Pontus Stenetorp[†], Jun'ichi Tsujii[†] and Hae-Chang Rim[§]
[†]Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, Japan
[§]Dept. of Computer & Radio Communications Engineering, Korea University, Seoul, Korea
{hccho,pontus,tsujii}@is.s.u-tokyo.ac.jp, {dglee,jtlee,rim}@nlp.korea.ac.kr

Abstract

Most NLP applications work under the assumption that a user input is error-free; thus, word segmentation (WS) for written languages that use word boundary markers (WBMs), such as spaces, has been regarded as a trivial issue. However, noisy real-world texts, such as blogs, e-mails, and SMS, may contain spacing errors that require correction before further processing may take place. For the Korean language, many researchers have adopted a traditional WS approach, which eliminates all spaces in the user input and re-inserts proper word boundaries. Unfortunately, such an approach often exacerbates the word spacing quality for user input, which has few or no spacing errors; such is the case, because a perfect WS model does not exist. In this paper, we propose a novel WS method that takes into consideration the initial word spacing information of the user input. Our method generates a better output than the original user input, even if the user input has few spacing errors. Moreover, the proposed method significantly outperforms a state-of-the-art Korean WS model when the user input initially contains less than 10% spacing errors, and performs comparably for cases containing more spacing errors. We believe that the proposed method will be a very practical pre-processing module.

1 Introduction

Word segmentation (WS) has been a fundamental research issue for languages that do not have word boundary markers (WBMs); on the contrary, other languages that do have WBMs have regarded the issue as a trivial task. Texts segmented with such WBMs, however, could contain a human writer's intentional or un-intentional spacing errors; and even a few spacing errors can cause error-propagation for further NLP stages.

For written languages that have WBMs, such as for the Korean language, the majority of recent research has been based on a traditional WS approach (Nakagawa, 2004). The first step of the traditional approach is to eliminate all spaces in the user input, and then re-locate the proper places to insert WBMs. One state-of-the-art Korean WS model (Lee et al., 2007) is known to achieve a performance of 90.31% word-unit precision, which is comparable with other WS models for the Chinese or Japanese language.

Still, there is a downside to the evaluation method. If the user input has a few or no spacing errors, traditional WS models may cause more spacing errors than it correct because they produce the same output regardless the word spacing states of the user input.

In this paper, we propose a new WS method that takes into account the word spacing information from the user input. Our proposed method first generates the best word spacing states for the user input by using a traditional WS model; however the method does not immediately apply the output. Secondly, the method estimates a threshold based on the word spacing quality of the user input. Finally, the method uses the new word spacing states that have probabilities that are higher than the threshold.

The most important contribution of the proposed method is that, for most cases, the method generates an output that is better than the user input. The experimental results show that the proposed method produces a better output than the user input even if the user input has less than 1% spacing errors in terms of the *character-unit precision*. Moreover, the proposed method outperforms (Lee et al., 2007) significantly, when the

Proceedings of the ACL-IJCNLP 2009 Conference Short Papers, pages 29–32,
Suntec, Singapore, 4 August 2009. ©2009 ACL and AFNLP

user input initially contains less than 10% spacing errors, and even performs comparably, when the input contains more than 10% errors. Based on these results, we believe that the proposed method would be a very practical pre-processing module for other NLP applications.

The paper is organized as follows: Section 2 explains the proposed method. Section 3 shows the experimental results. Finally, the last section describes the contributions of the proposed method.

2 The Proposed Method

The proposed method consists of three steps: a baseline WS model, confidence and threshold estimation, and output optimization. The following sections will explain the steps in detail.

2.1 Baseline Word Segmentation Model

We use the tri-gram Hidden Markov Model (HMM) of (Lee et al., 2007) as the baseline WS model; however, we adopt the Maximum Likelihood (ML) decoding strategy to independently find the best word spacing states. ML-decoding allows us to directly compare each output to the threshold. There is little discrepancy in accuracy when using ML-decoding, as compared to Viterbi-decoding, as mentioned in (Merialdo, 1994).[1]

Let $o_{1,n}$ be a sequence of n-character user input without WBMs, x_t be the best word spacing state for o_t where $1 \leq t \leq n$. Assume that x_t is either 1 (space after o_t) or 0 (no space after o_t). Then each best word spacing state $\hat{x}_t$ for all t can be found by using Equation 1.

$$\hat{x_t} = \underset{i \in (0,1)}{\operatorname{argmax}} P(x_t = i | o_{1,n}) \quad (1)$$

$$= \underset{i \in (0,1)}{\operatorname{argmax}} P(o_{1,n}, x_t = i) \quad (2)$$

$$= \underset{i \in (0,1)}{\operatorname{argmax}} \sum_{x_{t-2}, x_{t-1}} P(x_t = i | x_{t-2}, o_{t-1}, x_{t-1}, o_t) \times \sum_{x_{t-1}} P(o_{t+1} | o_{t-1}, x_{t-1}, o_t, x_t = i) \times \sum_{x_{t+1}} P(o_{t+2} | o_t, x_t = i, o_{t+1}, x_{t+1}) \quad (3)$$

Equation 2 is derived by applying the Bayes' rule and by eliminating the constant denominator. Moreover, the equation is simplified, as is Equation 3, by using the Markov assumption, and by eliminating the constant parts. Every part of Equation 3 can be calculated by adding the probabilities of all possible combinations of x_{t-2}, x_{t-1}, x_{t+1} and x_{t+2} values.

The model is trained by using the relative frequency information of the training data, and a smoothing technique is applied to relieve the data-sparseness problem which is the linear interpolation of n-grams that are used in (Lee et al., 2007).

2.2 Confidence and Threshold Estimation

We set a variable threshold that is proportional to the word spacing quality of the user input, *Confidence*. Formally, we can define the threshold T as a function of a confidence C, as in Equation 4.

$$T = f(C) \quad (4)$$

Then, we define the confidence as is done in Equation 5. Because calculating such a variable is impossible, we estimate the value by substituting the word spacing states produced by the baseline WS model, $x_{1,n}^{WS}$, with the correct word spacing states, $x_{1,n}^{correct}$, as is done in Equation 6. This estimation is based on the assumption that the word spacing states of the WS model is sufficiently similar to the correct word spacing states in the ***character-unit precision***.[2]

$$C = \frac{\text{\# of } x_t^{input} \text{ same to } x_t^{correct}}{\text{\# of } x_t^{input}} \quad (5)$$

$$\approx \frac{\text{\# of } x_t^{input} \text{ same to } x_t^{WS}}{\text{\# of } x_t^{input}} \quad (6)$$

$$\approx \sqrt[n]{\prod_{k=1}^{n} P(x_k^{input} | o_{1,n})} \quad (7)$$

To handle the estimation error for short sentences, we use the probability generating word spacing states of the user input with the length normalization as shown in Equation 7.

Figure 1 shows that the estimated confidence of Equation 7 is almost linearly proportional to the true confidence of Equation 5, thus suggesting that the threshold T can be defined as a function of the estimated confidence of Equation 7.[3]

[1] In the preliminary experiment, Viterbi-decoding showed a 0.5% higher word-unit precision.

[2] In the experiment with the development data, the baseline WS model shows about 97% character-unit precision.

[3] The development data is generated by randomly introducing spacing errors into correctly spaced sentences. We think that this reflects various intentional and un-intentional error patterns of individuals.

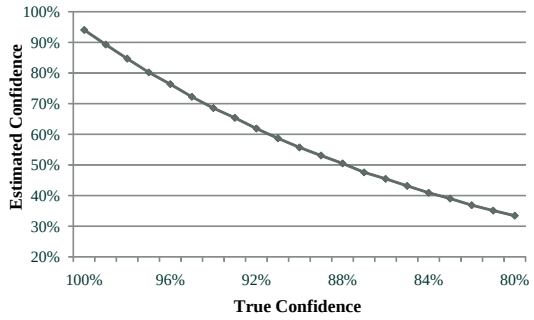

Figure 1: The relationship between estimated confidence and true confidence

To keep the focus on the research subject of this paper, we simply assume $f(x) = x$ as in Equation 8, for the threshold function f.

$$T \approx f(C) = C \tag{8}$$

In the experimental results, we confirm that even this simple threshold function can be helpful in improving the performance of the proposed method against traditional WS models.

2.3 Output Optimization

After completing the two steps described in Section 2.1 and 2.2, we have acquired the new spacing states for the user input generated by the baseline WS model, and the threshold measuring the word spacing quality of the user input.

The proposed method only applies a part of the new word spacing states to the user input, which have probabilities that are higher than the threshold; further the method discards the other new word spacing states that have probabilities that are lower than the threshold. By rejecting the unreliable output of the baseline WS model in this way, the proposed method can effectively improve the performance when the user input contains a relatively small number of spacing errors.

3 Experimental Results

Two types of experiments have been performed. In the first experiment, we investigate the level of performance improvement based on different settings of the user input's word spacing error rate. Because it is nearly impossible to obtain enough test data for any error rate, we generate pseudo test data in the same way that we generate development data.[4] In the second experiment, we attempt figuring out whether the proposed method really improves the word spacing quality of the user input in a real-world setting.

3.1 Performance Improvement according to the Word Spacing Error Rate of User Input

For the first experiment, we use the Sejong corpus[5] from 1998-1999 (1,000,000 Korean sentences) for the training data, and ETRI corpus (30,000 sentences) for the test data (ETRI, 1999). To generate the test data that have spacing errors, we make twenty one copies of the test data and randomly insert spacing errors from 0% to 20% in the same way in which we made the development data. We feel that this strategy can model both the intentional and un-intentional human error patterns.

In Figure 2, the x-axis indicates the word spacing error rate of the user input in terms of the character-unit precision, and the y-axis shows the word-unit precision of the output. Each graph depicts the word-unit precision of the test corpus, a state-of-the-art Korean WS model (Lee et al., 2007), the baseline WS model, and the proposed method.

Although Lee's model is known to perform comparably with state-of-the-art Chinese and Japanese WS models, it does not necessarily suggest that the word spacing quality of the model's output is better than the user input. In Figure 2, Lee's model exacerbates the user input when it has spacing errors that are lower than 3%.

The proposed method, however, produces a better output, even if the user input has 1% spacing errors. Moreover, the proposed method shows a considerably better performance within the 10% spacing error range, as compared to Lee's model, although the baseline WS model itself does not outperforms Lee's model. The performance improvement in this error range is fairly significant because we found that the spacing error rate of texts collected for the second experiment was about 9.1%.

3.2 Performance Comparison with Web Text having Usual Error Rate

In the second experiment, we attempt finding out whether the proposed method can be beneficial under real-world circumstances. Web texts, which consist of 1,000 erroneous sentences from famous

[4] See Footnote 3.

[5] Details available at: http://www.sejong.or.kr/eindex.php

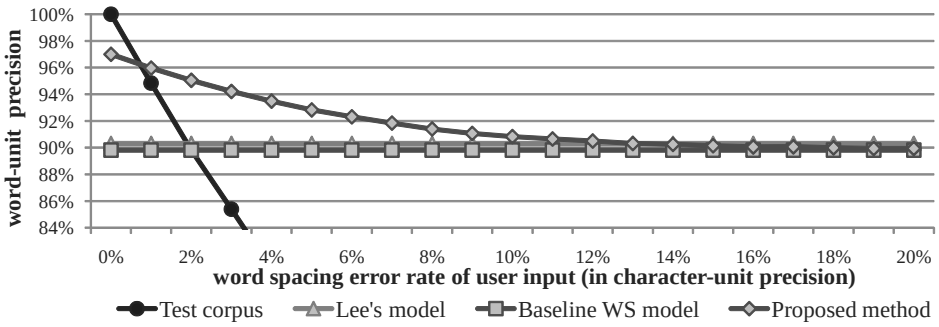

Figure 2: Performance improvement according to the word spacing error rate of user input

Method	Web Text
Test Corpus	70.89%
Lee's Model	70.45%
Baseline WS Model	69.13%
Proposed Method	73.74%

Table 1: Performance comparison with Web text

Web portals and personal blogs, were collected and used as the test data. Since the test data tend to have a similar error rate to the narrow standard deviation, we computed the overall performance over the average word spacing error rate, which is 9.1%. The baseline WS model is trained on the Sejong corpus, described in Section 3.1.

The test result is shown in Table 1. The overall performance of Lee's model, the baseline WS model and the proposed method decreased by roughly 18%. We hypothesize that the performance degradation probably results from the spelling errors of the test data, and the inconsistencies that exist between the training data and the test data. However, the proposed method still improves the word spacing quality of the user input by 3%, while the two traditional WS models degrades the quality. Such a result indicates that the proposed method is effective for real-world environments, as we had intended. Furthermore, we also believe that the performance can be improved if a proper training corpus is provided, or if a spelling correction method is integrated.

4 Conclusion

In this paper, we proposed a new WS method that uses the word spacing information of the user input, for languages with WBMs. By utilizing the user input, the proposed method effectively refines the output of the baseline WS model and improves the overall performance.

The most important contribution of this work is that it produces an output that is better than the user input even if it contains few spacing errors. Therefore, the proposed method can be applied as a pre-processing module for practical NLP applications without introducing a risk that would generate a worse output than the user input. Moreover, the performance is notably better than a state-of-the-art Korean WS model (Lee et al., 2007) within the 10% spacing error range, which human writers seldom exceed. It also performs comparably, even if the user input contains more than 10% spacing errors.

5 Acknowledgment

This work was partially supported by Grant-in-Aid for Specially Promoted Research (MEXT, Japan) and Special Coordination Funds for Promoting Science and Technology (MEXT, Japan).

References

ETRI. 1999. Pos-tag guidelines. Technical report. *Electronics and Telecomminications Research Institute*.

Do-Gil Lee, Hae-Chang Rim, and Dongsuk Yook. 2007. Automatic Word Spacing Using Probabilistic Models Based on Character n-grams. *IEEE Intelligent Systems*, 22(1):28–35.

Bernard Merialdo. 1994. Tagging English text with a probabilistic model. *Comput. Linguist.*, 20(2):155–171.

Tetsuji Nakagawa. 2004. Chinese and Japanese word segmentation using word-level and character-level information. In *COLING '04*, page 466, Morristown, NJ, USA. Association for Computational Linguistics.