International Journal of Innovative
Computing, Information and Control
Volume **8**, Number **7(B)**, July **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 5055–5065

A NOVEL DISCRIMINATIVE PROBABILISTIC MODEL FOR AUTOMATIC WORD SPACING ON MOBILE DEVICES

Shinil Kim, Seon Yang and Youngjoong Ko*

Department of Computer Engineering
Dong-A University
840 Hadan 2-dong, Saha-gu, Busan 604-714, Korea
{ pirate2003; seony.yang }@gmail.com
*Corresponding author: yjko@dau.ac.kr

Received May 2011; revised November 2011

Abstract. *This paper presents a new method for automatic word spacing in Korean. Our major challenge is to reduce the memory usage and processing time to be suitable for mobile devices while ensuring a high performance. Since mobile devices are more difficult to type letters into than general PCs with current technology, spacing errors such as writing with no spaces can occur relatively frequently in mobile devices. As text typing to connect to Social Network Service via mobile devices continues to increase, more and more people can be confused in communication due to spacing errors. In this paper, we propose a novel "discriminative probabilistic model" in which syllables are classified into eight classes of "syllable-based space patterns". As baseline systems, we employ two different types of Conditional Random Field models which are regarded as one of the highest-performance systems in sequence labeling like word spacing. Our experimental results show that the proposed system performs better than the baseline while using significantly less memory and Spacing time.*

Keywords: Word spacing, Mobile device, Discriminative probabilistic model, Syllable-based space pattern

1. **Introduction.** Spacing is a very important factor to accurately understand the meaning of sentences. Assume that a Twitter user wanted to say "(a) *gi-ta-ga bang an-e iss-da*: The guitar is in the room (*gi-ta* means a guitar, *ga* is a subjective suffix, and *bang* means a room[1])" but actually wrote "*gi-ta-ga-bang-an-e-iss-da*" with no spaces. Some readers may understand this sentence correctly. However, many other readers can misinterpret it as "(b)*gi-ta ga-bang an-e iss-da*: It is in the guitar case (*gi-ta* still means a guitar, and *ga-bang* means a bag)" which is quite different from what the speaker wanted to convey. The difference of the above two sentences is the location of a syllable *ga*; it is attached to the preceding word, *gi-ta*, in (a), while attached to the following word, *bang*, in (b). If *ga* is attached to the preceding word and a space follows it, it is regarded as a subjective suffix like in (a). On the other hand, if there is a space before *ga*, *ga* is regarded as the first syllable of another word like in (b). As we can see from this example case, the word spacing is very important and complex in Korean. Moreover, the Korean word spacing is one of the most difficult tasks for the mobile voice search technology development, which is the focus of tremendous interest recently. An engineer who participated in a press conference held in *Google Korea* said that the most difficult part in the development of Korean mobile voice search was "word spacing".[2] He explained the complexity of Korean word spacing problem by comparing with those of English and Japanese. In English,

[1](continued description) *an*: in, *e*: adverbial suffix, *iss-da*: be/have/exist
[2]http://www.asiae.co.kr/news/view.htm?idxno=2011071214070729683

the spacing rule is very simple; all the words in a sentence are separated by spaces. In Japanese, it does not require any spacing rule; there is no space in a sentence. In Korean, however, spacing rules are very complicated as previously stated in [1,2]. In some cases, two (or more) words must be attached while, in some other cases, each word should be separated by a space. In addition, the meaning of a sentence can vary depending on the location of spaces as we can see in the above two example sentences, (a) and (b).

In order to solve the communication problem caused by spacing errors, many researchers have studied about the automatic word spacing technique. In particular, most Korean word processors, e.g., "*A-re-a-han-geul*" and "*Hun-min-word*", already have a function that suggests appropriate spacing. In other words, the spacing problem is recently deemed not very serious in a normal PC environment. However, due to the complex spacing rules, the automatic word spacing system tends to be heavy for mobile devices. Although the computing power of mobile devices is rapidly increasing, it is still difficult to load applications for general PC onto mobile devices without carrying out a lightweight process. Furthermore, since mobile devices are more difficult to type letters into than general PCs with current technology, spacing errors can occur relatively frequently in mobile devices. As text typing on Social Network Service (SNS) via mobile devices continues to increase, more and more people can become confused in communication due to spacing errors.

Conditional Random Field (CRF) is known as one of the state-of-the-art systems in sequence labeling tasks such as word spacing, and Lee et al. [2] obtained the highest performance in word spacing for Korean by using two kinds of trigram patterns including syllable trigrams and spacing state trigrams. We conducted various experiments by using two CRF versions, CRF++[3] and the CRF linear model[4], for our evaluations, but they did not show better performance than the proposed method in both of two evaluation measures, accuracy and memory/time usage. In the work of [2], Lee et al. calculated probabilities from combinations of all the syllable and spacing state trigrams. Even though it is the state-of-the-art method for Korean with an excellent performance, an accuracy of 97.48%, their method is not suitable for mobile devices because a considerable amount of memory is required to employ the combinations of all the syllable and spacing state trigrams. (See Section 5)

In this paper, we propose a novel word spacing approach with low memory usage as well as high performance. Because we employ state trigram patterns and syllable bigrams instead of the combination of all the syllable and spacing state trigrams used in [2], we can reduce computational costs and memory usage costs dramatically. Since we use state emission probabilities instead of object emission probabilities, the proposed method belongs to a discriminative probabilistic model, not a generative probabilistic model such as Hidden Markov Model (HMM). The state trigrams used in the method are called *syllable-based space patterns* (hereafter, *SP-patterns*). An SP-pattern consists of three consecutive bits of "0" and "1". Each bit represents the spacing state after the second preceding (w_{-2}), preceding (w_{-1}), and current (w_0) syllables. "0" means that the syllable is not followed by a space, and "1" implies that the syllable is followed by a space. These SP-patterns have only eight different types of SP-pattern from 0 (000) to 7 (111) because each state has a binary value ("0" or "1"). In addition, we use syllable bigrams instead of syllable trigrams. These factors make the proposed method to use a small amount of memory. In summary, our model searches the best sequence of SP-patterns given a sentence by using SP-pattern emission probabilities and syllable bigram probabilities. Table 1 shows the SP-patterns and the syllable bigrams in an example sentence.

[3]http://crfpp.sourceforge.net/

[4]http://wapiti.limsi.fr/

TABLE 1. The SP-patterns and the syllable bigrams in a sentence "*i yeon-gu-ui mok-pyo* (the purpose of this research)"; each underline part in each SP-pattern indicates the spacing state of the current syllable. *We assume that the SP-pattern of the first syllable in a sentence starts with "01".

Syllable	SP-pattern	Syllable bigram
i	$01\underline{1}$	$< i, yeon >$
$yeon$	$11\underline{0}$	$< yeon, gu >$
gu	$10\underline{0}$	$< gu, ui >$
eui	$00\underline{1}$	$< ui, mok >$
mok	$01\underline{0}$	$< mok, pyo >$
pyo	$10\underline{1}$	$-$

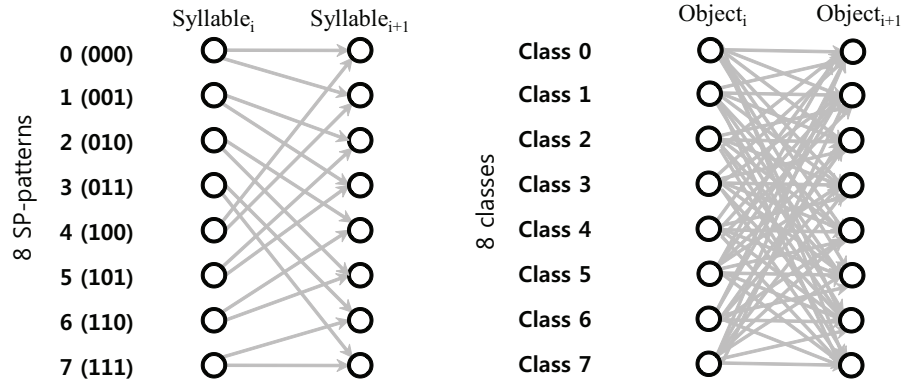

FIGURE 1. Comparison of the proposed model and typical searching

Note that, each SP-pattern has only two possible paths to the next one. For example, if the $syllable_i$ (i-th syllable in a sentence) is classified into a "001" SP-pattern, the possible SP-patterns which the $syllable_{i+1}$ can be classified into are only "010" and "011". We assume that $w_{-2}w_{-1}$ of $syllable_{i+1}$ must be the same as $w_{-1}w_0$ of $syllable_i$. By this reason, it is needed only $N * 2$ computations (N means the total number of classes; it is 8 in this study) when we go through every single step in sequence searching as shown in Figure 1(a). In typical sequence searching, it is needed $N * N$ computations as shown in Figure 1(b). The experiment section will show that the proposed method significantly reduced memory/time usage enough for mobile devices while ensuring the high performance. The rest of the paper is organized as follows. Section 2 briefly describes the related work. Section 3 describes our proposed system in detail. Section 4 presents our experimental results, and Section 5 is devoted to the additional experiments and analysis of our system. In Section 6, we summarize and conclude our paper.

2. **Related Work.** Word segmentation is very important in many NLP tasks. It is regarded as a relatively minor problem in many western languages, because every word is basically separated by white space. However, it is problematic in isolating languages such as Chinese, and in agglutinative languages such as Korean, Japanese and Finnish.

In Chinese texts, words, which are composed of single or multiple characters, are not separated by spaces. [3] presented a method for segmenting Chinese texts without using a manually segmented corpus or a predefined dictionary. [4] presented a Chinese word

segmentation system submitted to the closed track of Sighan bakeoff 2005. [5] studied for Chinese word segmentation and named entity recognition based on CRF.

For Korean, most researchers have regarded the spacing issues between words as a classification problem. [6] presented a self-organizing word spacing model, and [1] presented a spacing approach which employs an improved Viterbi segmentation. To the best of our knowledge, [2] achieved the most outstanding performance, an accuracy of 97.48%. They regarded the spacing problem as the tagging problem and employed syllable trigrams and spacing state trigrams. [7] conducted a word spacing task by using the method of [2] for refining SMS text messages. [8] focused on a light-weight word spacing system. They employed a modified HMM based on a character unigram and then applied 100,000 rules to improve the performance. Their experimental result showed a relatively high accuracy of 94.14% in spite of a low memory usage of approximately 1 MB.

In this study, CRF and HMM are compared with the proposed model. [9] introduced CRF. We used CRF tookits given in [10,11]. [12] introduced HMM. [13] also presented the tutorial on HMM in speech recognition.

3. The Word Spacing System Based on SP-Patterns and Syllable Bigrams.

This section describes two types of probabilities which are key elements of our system configuration: state emission probability (or SP-pattern emission probability, PA) and syllable bigram probability (PB).

3.1. PA: SP-pattern emission probability.

PA indicates the probability of a syllable to be classified into a certain SP-pattern. It can be calculated as given in Equation (1):

$$\mathbf{PA}: \ a_{iy} = P(pat_y \mid syl_i), \ \ 0 \le y \le 7, \ \ syl_i \in S_1 \tag{1}$$

here, S_1 is the set of all the syllables in the training corpus, and syl_i is an elements in S_1, pat_y is one of the eight SP-patterns. If the training corpus consists of only three sentences given in Table 2, PA values of a syllable "*ui*" can be calculated as listed in Table 3.

Table 2. The SP-patterns of a syllable "*ui*"

Sentence	SP-pattern
i yeon-gu-**ui** *mok-pyo* (The purpose of this research)	001
geu-nyeo-neun geu-deul-**ui** *eo-meo-ni-da* (She is their mother)	001
nae **ui**-*gyun-eun byeon-ham-eop-da* (My opinion remains unchanged)	110

Table 3. PA values of the syllable "*ui*" based on Table 2

SP-pattern	0 (000)	1 (001)	2 (010)	3 (011)	4 (100)	5 (101)	6 (110)	7 (111)
PA	0	0.67	0	0	0	0	0.33	0

3.2. PB: Syllable bigram probability.

PB indicates the spacing probability between two consecutive syllables as given in Equation (2):

$$\mathbf{PB}: \ b_{ij} = P(space_{ij} \mid \ < syl_i, syl_j >), \quad < syl_i, syl_j > \in S_2 \tag{2}$$

here, $< syl_i, syl_j >$ is a syllable bigram, $space_{ij}$ indicates the space between syl_i and syl_j, and S_2 is the set of all syllable bigrams in the training corpus. It is based on the assumption that a spacing state after a syllable is affected by the following syllable. For example, a syllable "*i*" is used as a subjective suffix. It means that the word preceding "*i*" is a subject and a space follows the syllable, e.g., "*dal-i ddeun-da*[5] (the moon rises)".

[5] *dal*: the moon, *i*: subjective suffix, *ddeun-da*: rise

Hence, if we focus each syllable separately, we are likely to think that "*i*" will be followed by a space. However, if "*ron*" is the next syllable after "*i*", these two syllables are likely to belong to one word "*i-ron* (theory)". As seen in this case, spacing state after a syllable is dependent on the following syllable in many cases. Assume that syl_i is "*i*", syl_j is "*ron*", and the syllable bigram $< i, ron >$ is found ten times in the training corpus. If only one case contains a space between "*i*" and "*ron*", PB value becomes 0.1. If the testing corpus contains a certain syllable bigram which does not occur in the training corpus, we set the unbiased probability value of 0.5.

3.3. **Searching the best sequence.** We compute the best path through the entire trellis consisting of syllables and SP-patterns by using dynamic programming. In order to compute the best path more efficiently, we use an adaptation of the Viterbi algorithm as shown in Equations (3) and (4):

$$\text{Using only PA} \ : \delta_y(t+1) = \max_{0 \le x \le 7} \delta_x(t) a_{iy}, \ 0 \le y \le 7 \tag{3}$$

$$\text{Using both PA and PB} : \ \delta_y(t+1) = \begin{cases} \max_{0 \le x \le 7} \delta_x(t) a_{iy} b_{ij}, & y \in \{1,3,5,7\} \\ \max_{0 \le x \le 7} \delta_x(t) a_{iy}(1 - b_{ij}), & y \in \{0,2,4,6\} \end{cases} \tag{4}$$

here, t means the position in the sentence whose lengh is T, and $\delta_x(t)$ indicates the probability value of the most probable path that leads to node whose SP-pattern is x at position t. Finally, the most likely state sequence Seq_T is worked out from the right backwards as given in Equation (5):

$$Seq_T = \arg\max_{0 \le y \le 7} \delta_y(T) \tag{5}$$

The Viterbi algorithm is mainly used in the decoding step of HMM, which is a well-known generative model. In order to clarify our discriminative model, Figure 2 compares the original HMM and the proposed model in finding the best state at position $t+1$.

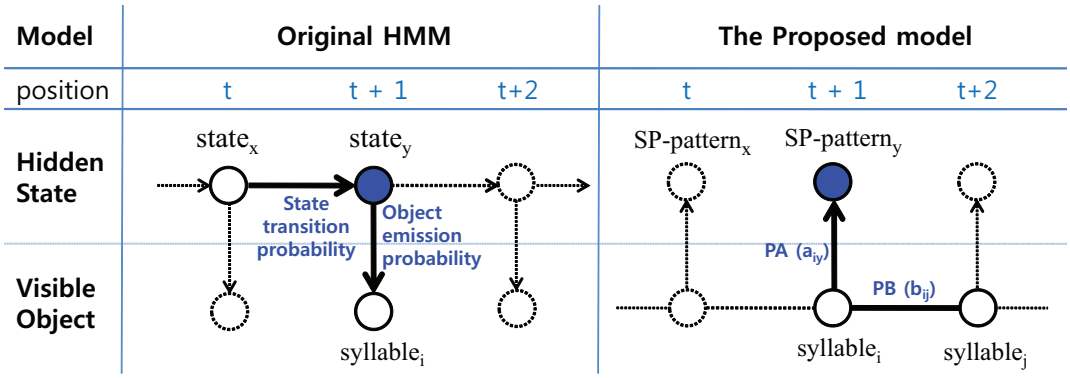

FIGURE 2. Original HMM vs. the proposed model

As shown in Figure 2, the proposed model employing SP-patterns and syllable bigrams is completely different from the original HMM. Although the SP-pattern is a sort of trigrams, the number of different SP-patterns is up to eight. As we can see in the Experiment section, our model reduces a considerable amount of memory. In addition, it can reduce the number of computations. In order to explain this, Figure 3 presents the best path searching in an example sentence "*hak-gyo ga-ja* (Let us go to school)". The PA and PB are given in Table 4 and Table 5, respectively.

As we can see in Figure 3, each SP-pattern $(w_{-2}w_{-1}w_0)$ has only two possible paths to the next one. For example, if the first chosen SP-pattern is 2 (0$\underline{10}$), we consider only two following SP-patterns: 4 ($\underline{10}$0) and 5 ($\underline{10}$1). That is because the last two bits $(w_{-1}w_0)$ of "*hak*" must be the same as the first two bits $(w_{-2}w_{-1})$ of "*gyo*". When we go through every

TABLE 4. PA of "*hak-gyo ga-ja* (Let us go to school)"

Syllable	PA							
	0 (000)	1 (001)	2 (010)	3 (011)	4 (100)	5 (101)	6 (110)	7 (111)
hak	0.19	0.10	0.20	0.05	0.23	0.12	0.10	0.01
gyo	0.23	0.18	0.08	0.02	0.18	0.21	0.09	0.01
ga	0.13	0.22	0.28	0.08	0.14	0.12	0.02	0.01
ja	0.20	0.15	0.18	0.09	0.22	0.09	0.05	0.02

TABLE 5. PB of "*hak-gyo ga-ja* (Let us go to school)"

Syllable bigram	$< hak, gyo >$	$< gyo, ga >$	$< ga, ja >$
PB	0.03	0.60	0.08

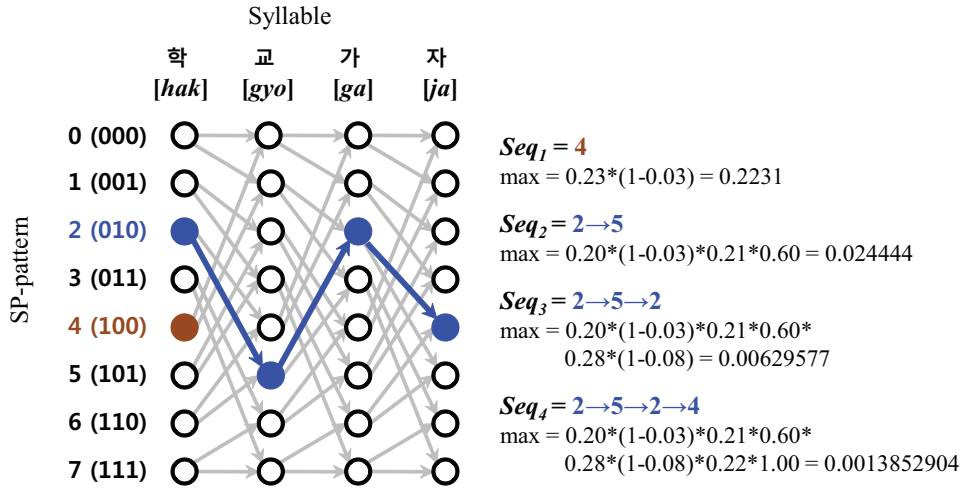

FIGURE 3. Finding the best path in "*hak-gyo ga-ja* (Let us go to school)"

single step, we require only $N * 2$ computations, where N is the number of hidden states (N is 8 in this study). In the original Viterbi algorithm, it is required $N * N$ computations for every step. In other words, when we find the best spacing sequence consisting of T syllables, we carry out only $T * 8 * 2$ computations and not $T * 8 * 8$ computations. This fact implies that we can reduce 75% of the computations in comparison with the original Viterbi algorithm.

4. Experimental Evaluations.

4.1. Experimental setting.

In order to compare our system with the previous studies under the same conditions, we used both *1) the* 21st *Century Sejong Project's raw corpus*[6] of 809,914 sentences and *2) the ETRI (Electronics and Telecommunications Research Institute) POS tagged corpus*[7] of 27,858 sentences.

The PC environment we conducted experiments are as follows:

- Processor: AMD Athlon(tm) II X4 630 Processor, 2.79 GHz
- OS: Windows 7 Ultimate K (64 bits), 2.00 GB RAM
- Programming Language: Java

[6]www.sejong.or.kr
[7]www.etri.re.kr

As an evaluation measure, we used *syllable accuracy* as the following Equation (6):

$$accuracy = \frac{\#\text{ of correctly spaced syllables}}{\#\text{ of syllables}} \tag{6}$$

4.2. **Feature selection.** This section describes our feature selection processes to explain why we finally selected the 3-bit SP-patterns and the syllable bigrams as our proposed features. We first conducted various experiments according to the different number of states, from 2-bit (4 states) to 5-bit (32 states), as shown in Table 6.

TABLE 6. Evaluation results according to the different SP-pattern lengths when using only PA. *ms/sen: millisecond per sentence.

Using only PA	2-bit	**3-bit**	4-bit	5-bit
Accuracy (%)	89.32	**91.52**	52.92	53.17
Memory (KB)	72	**161**	224	399
Time (ms/sen)	0.13	**0.14**	0.35	0.71

As a result, we obtained the best performance when employing the 3-bit SP-patterns. The 2-bit SP-patterns also showed quite good performance with a very small amount of memory/time usage. On the other hand, the patterns with 4 or more bits showed very low performance. We think that it is caused by a sparseness problem. For better performance, we determined to use the syllable bigram features. The results are listed in Table 7.

TABLE 7. Evaluation results according to the different SP-pattern lengths when using both PA and PB

Both PA and PB	2-bit	**3-bit**	4-bit	5-bit
Accuracy (%)	93.16	**94.38**	79.28	77.25
Memory (KB)	420	**509**	572	747
Time (ms/sen)	0.14	**0.16**	0.38	0.93

Among these experiments, the 3-bit SP pattern also showed the best performance like Table 6. In addition, when compared with the method using only PA in Table 6, it achieved the improvement of 2.86% by adding only 348 KB for memory and 0.02 ms/sen for processing time. We finally selected the 3-bit SP-patterns added the syllable bigrams as the proposed features.

4.3. **Comparing the proposed method with other methods.** Now, we compare our model with two CRF models, CRF++ and a CRF linear model, and one previous study of [8] (hereafter, Song & Kim). We chose two CRF models because CRF is known to perform best in sequence labeling, and Song & Kim because it is the latest study related to light-weight word spacing. Although the study of [2] is the research we mainly referred to, we do not compare our method with theirs. The reason is that their main purpose is not to develop a light-weight system but to obtain only a high performance; we will compare our method with theirs in the discussion section. When conducting experiments with two CRF models, we used two feature options, (1) unigram and (2) bigram. Song & Kim also presented two results; (1) using only a modified HMM (Naïve Bayesian HMM, denoted by NB-HMM) and (2) additionally employing a rule-based system to re-correct the output of NB-HMM. Table 8 and Figure 4 summarize the final results.

As can be seen, our system leads to outstanding results as a light-weight system. In comparison with two CRF models, we significantly reduced the memory/time usage. In

TABLE 8. Final results; we tested with the 21st century Sejong project's corpus for an equal environment with Song & Kim

Method		Memory (KB)	Time (ms/sen)	Accuracy (%)
CRF++	(1) unigram	496	0.59	90.51
	(2) bigram	95,018	0.74	94.05
CRF linear model	(1) unigram	658	0.11	86.97
	(2) bigram	5,278	0.23	91.85
Song & Kim	(1) NB-HMM	266	0.26	89.28
	(2) NB-HMM + 100,000 rules	1,074	0.37	94.14
Proposed Method	(1) **PA**	**161**	**0.14**	**91.52**
	(2) **PA, PB**	**509**	**0.16**	**94.38**

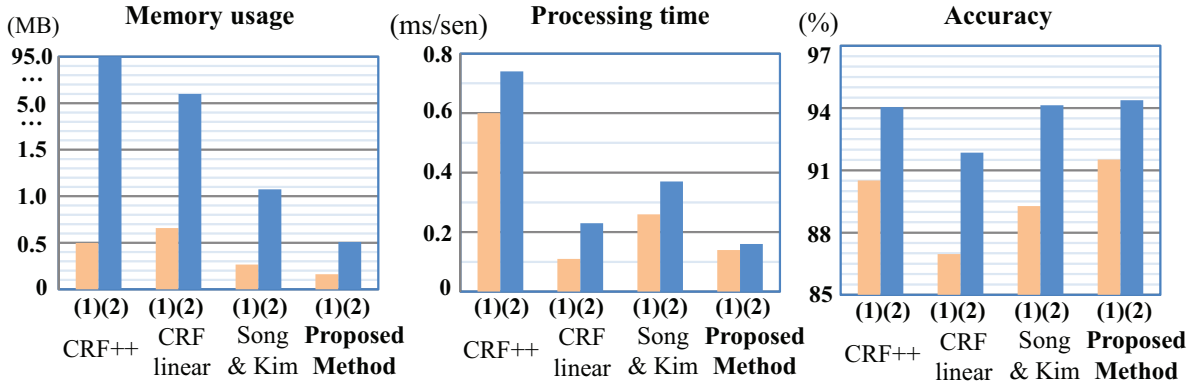

FIGURE 4. Final results; the proposed method uses considerably less memory and time while ensuring a similar or slightly higher performance

comparison with Song & Kim , we achieved a similar performance (technically, our method shows a slightly better performance) while using less than half the memory and processing time. Further, it is important to note that we implemented the proposed method using Java; we believe that Java is more suitable for mobile mounting. On the other hand, the two CRF models and Song & Kim's model are implemented by C++. We believe that if we implement our system using C++, we can reduce the processing time further.

5. Discussion.

5.1. Additional experiments to improve the performance of the proposed system. This section employs a Transformation-Based Learning (TBL) technique for our additional step to improve the performance. TBL corrects the errors included in the output of the preceding step. We first generate a set of error-driven transformation rules. Table 9 describes the transformation templates, and Table 10 lists some example rules.

TABLE 9. Transformation templates

Insert (or remove) a space after **w** if ... where **w** is a wrongly spaced syllable.
1. The second preceding syllable of **w** is **x**, and the preceding syllable of **w** is **y**.
2. The preceding syllable of **w** is **x**, and the following syllable of **w** is **y**.
3. The following syllable of **w** is **x**, and the second following syllable of **w** is **y**.

TABLE 10. Transformation rules generated by an example sentence

Example Sentence	*i yeon-gu-***ui** *mok-jeok-eun ...* (The purpose of this research is ...)
wrongly spaced: *ui*	*i yeon-gu-***ui***-mok-jeok-eun ...*
Three generated rules	1. Insert a space after "*ui*" if the second preceding syllable is "*yeon*" and the preceding syllable is "*gu*".
	2. Insert a space after "*ui*" if the preceding syllable is "*yeon*" and the following syllable is "*mok*".
	3. Insert a space after "*ui*" if the following syllable is "*mok*" and the second following syllable is "*jeok*".

After the generation of approximately 70,000 transformation rules, a scoring function ranks these rules. We define our scoring function as $C_i - E_i$ where C_i is the number of corrected syllables after the i-th rule is applied and E_i is the number of the opposite case. This ranking process is iteratively executed. The highest scored rules are selected repeatedly. The iterations stop when the scoring function reaches a certain threshold. We finally set the threshold value as 1 after tuning. This implies that we use only the rules whose score is 2 or more. As a result, we have achieved an accuracy of 96.46% with a memory usage of 1.7 MB. This implies that we have improved the accuracy by approximately 2% while using a memory of approximately 1.2 MB more. From the viewpoint of memory usage, this performance can be evaluated positively.

We compare this result with that of [2] (hereafter, Lee et al.). As previously mentioned, their method showed the highest performance in Korean word spacing tasks. Equation (7) represents their method:

$$\begin{aligned} P(u_{1,n}, c_{1,n}) &= \prod_{i=1}^{n} (P(c_i|c_{1,i-1}, u_{1,i-1}) * P(u_i|c_{1,i}, u_{1,i-1})) \\ &\approx \prod_{i=1}^{n} (P(c_i|c_{i-K,i-1}, u_{i-J,i-1}) * P(u_i|c_{i-L,i}, u_{i-H,i-1})) \end{aligned} \tag{7}$$

here, n is the number of syllables in a sentence. For a given sentence of syllables $c_{1,n}$, they find the word-spacing tags $u_{1,n}$ where K and L refer to the number of tags relative to the number of syllables J and H in a sequence; i.e., K, J, L and H indicate the feature types ("0": unigram, "1": bigram, "2": trigram). They experimented with various (K, J, L, H) options and achieved the best performance using (2, 2, 1, 2). Table 11 shows the performance comparison.

As listed in Table 11, Lee et al.'s system shows approximately 1% higher performance than ours. It is clear that 1% is not a small difference. However, it is also clear that a

TABLE 11. Comparison with Lee et al.'s system; we used the 21st century Sejong project's corpus and the ETRI POS tagged corpus for an equal environment with theirs

Method	Memory (MB)	Accuracy (%)
Lee et al.: The combination of all syllable trigrams and state trigrams	About 70	**97.48**
Our expanded method: Discriminative model (with PA and PB) + TBL	**1.7**	96.46

considerable amount of memory is required for syllable trigrams. After we carefully implemented their method[8], we could find that it is required about 70 MB of memory usage. Therefore, we believe that our expanded method is competitive to the best performance system even when the focus is only on the system performance.

5.2. **Analysis of the proposed method.** This section analyzes the advantages of our system and its future directions for improvement. The strong point of our system is that it shows high performance in spite of using only a small size of memory/time usage. We here try to summarize the reasons why our system can show such good properties as follows:

1. Since automatic word spacing is performed prior to any other preprocessing steps such as POS tagging, most spacing systems including the proposed system have employed the syllable-based features. The important point is that the previous high-performing systems require a large amount of memory/time usage in order to save (or calculate) the probabilities of the features. In other words, most of them are not suitable for mobile devices. However, the proposed system only requires a small amount of memory/time usage (see Table 8 in Section 4.3).
2. We employed syllable bigrams as our second feature; these bigrams also do not require much memory. Some previous researchers also used syllable bigram features. However, they usually calculated and utilized three different probability values per the position of a syllable bigram to detect the existence of a space; before the syllable bigram, in the middle of the syllable bigram, and after the syllable bigram. On the other hand, we are interested in only the middle of the syllable bigram. Thus, we could reduce a relatively much memory usage.
3. Since each SP-pattern provides only two possible paths to the next one, we can save a considerable amount of time when searching the best path of spacing states.
4. Furthermore, the proposed system shows high performance, an accuracy of 94.38%. Due to the above reasons, we introduce our system as a novel word spacing system for mobile devices, e.g., short message service, e-mailing and social network service.

We think there is still any room for improvement in our system. We are planning to develop the personalized system. A mobile device is generally used for only one person and each individual person differs in domains of interest, frequently used words, word abbreviation, space omission, etc. Hence, it could be very useful if a word spacing system were continuously upgraded according to the characteristics of each user. We are looking for a variety of ways to upgrade the proposed system to a user-customized one.

6. **Conclusion and Future Work.** This paper has presented a novel Korean spacing model for mobile systems. We used SP-patterns and syllable bigrams based on a discriminative model. The proposed approach significantly reduced both memory usage and processing time while ensuring that the performance was as much as that achieved in the latest studies.

As our future work, we have three plans. Firstly, we will apply our model to other languages. Secondly, we will improve the system's performance more. We attempted to expand the model using TBL and improved the performance; we are looking for a better alternative. Lastly, the application of the model to other application areas remains for further research.

[8]Since they did not focus on a light-weight system, they did not make mention of memory/time usage.

Acknowledgment. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2011-0003780).

REFERENCES

[1] G. Hong and H. Rim, Korean spacing by improving viterbi segmentation, *Proc. of the 6th ALPIT*, pp.75-80, 2007.
[2] D. Lee, H. Rim and D. Yook, Automatic word spacing using probabilistic models based on character *n*-grams, *IEEE Intelligent Systems*, vol.22, no.1, pp.28-35, 2007.
[3] J. Xu, R. Zens and H. Ney, Do we need Chinese word segmentation for statistical machine translation? *Proc. of the 3rd SIGHAN Workshop on Chinese Language Processing*, 2004.
[4] H. Tseng, P. Chang, G. Andrew, D. Jurafsky and C. Manning, A conditional random field word segmenter for sighan bakeoff 2005, *Proc. of the 4th SIGHAN Workshop on Chinese Language Processing*, 2005.
[5] X. Mao, S. He, S. Bao, Y. Dong and H. Wang, Chinese word segmentation and named entity recognition based on conditional random fields, *Proc. of the 6th SIGHAN Workshop on Chinese Language Processing*, 2008.
[6] S. Park, Y. Tae and S. Park, Self-organizing *n*-gram model for automatic word spacing, *Proc. of the 21st CL and the 44th ACL*, pp.633-640, 2006.
[7] J. Byun, S. Lee, Y. Song and H. Rim, Two phase model for SMS text messages refinement, *Proc. of AAAI Workshop on Enhanced Messaging*, 2008.
[8] Y. Song and H. Kim, An automatic Korean word spacing system for devices with low computer power, *KIPS*, vol.16(B), no.4, pp.333-340, 2009.
[9] J. Lafferty, A. McCallum and F. Pereira, Conditional random fields: Probabilistic models for segmenting and labeling sequence data, *Proc. of ICML*, 2001.
[10] T. Kudo, *CRF++: Yet Another CRF Toolkit*, gttp://crfpp.sourceforge.net/.
[11] T. Lavergne, O. Cappe and F. Yvon, Practical very large scale CRFs, *Proc. of ACL*, pp.504-513, 2010.
[12] L. R. Rabiner and B. Juang, An introduction of hidden Markov models, *IEEE ASSP Magazine*, pp.4-15, 1986.
[13] L. R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, *Proc. of IEEE Conf.*, vol.72, no.2, pp.257-286, 1989.