Investigating an Effective Character-level Embedding in Korean Sentence Classification

Won Ik Cho, Seok Min Kim, and Nam Soo Kim
Human Interface Laboratory
Department of Electrical and Computer Engineering and INMC
Seoul National University
1 Gwanak-ro, Gwanak-gu, Seoul, Korea, 08826
{wicho,smkim}@hi.snu.ac.kr, nkim@snu.ac.kr

Abstract

Different from the writing systems of many Romance and Germanic languages, some languages or language families show complex conjunct forms in character composition. For such cases where the conjuncts consist of the components representing consonant(s) and vowel, various character encoding schemes can be adopted beyond merely making up a one-hot vector. However, there has been little work done on intra-language comparison regarding performances using each representation. In this study, utilizing the Korean language which is character-rich and agglutinative, we investigate an encoding scheme that is the most effective among *Jamo*[1]-level one-hot, character-level one-hot, character-level dense, and character-level multi-hot. Classification performance with each scheme is evaluated on two corpora: one on binary sentiment analysis of movie reviews, and the other on multi-class identification of intention types. The result displays that the character-level features show higher performance in general, although the *Jamo*-level features may show compatibility with the attention-based models if guaranteed adequate parameter set size.

1 Introduction

Ever since an early approach exploiting the character features for the neural network-based natural language processing (NLP) (Zhang et al., 2015), character-level embedding[2] has been widely used for many tasks such as machine translation (Ling et al., 2015), noisy document representation (Dhingra et al., 2016), language correction (Xie et al., 2016), and word segmentation (Cho et al., 2018a). However, little consideration was done for intra-language performance comparison regarding variant representation types. Unlike English, a Germanic language written with an alphabet comprising 26 characters, many languages used in East Asia are written with scripts whose characters can be further decomposed into sub-parts representing individual consonants or vowels. This conveys that (sub-)character-level representation for such languages has the potential to be managed with more than just a simple one-hot encoding.

In this paper, a comparative experiment is conducted on Korean, a representative language with a featural writing system (Daniels and Bright, 1996). To be specific, the Korean alphabet *Hangul* consists of the letters *Jamo* denoting consonants and vowels. The letters comprise a morpho-syllabic block that refers to *character*, which is resultingly equivalent to the phonetic unit *syllable* in terms of Korean morpho-phonology. The conjunct form of a character is {Syllable: CV(C)}; this notation implies that there should be at least one consonant (namely *cho-seng*, the first sound) and one vowel (namely *cwung-seng*, the second sound) in a character. An additional consonant (namely *cong-seng*, the third sound) is auxiliary. However, in decomposition of the characters, three slots are fully held to guarantee a space for each component; an empty cell comes in place of the third entry if there is no auxiliary consonant. The number of possible sub-characters, or (compos-

[1] Letters of Korean alphabet *Hangul*.

[2] Throughout this paper, the terms *embedding* and *encoding* are parallelly used depending on the context.

33rd Pacific Asia Conference on Language, Information and Computation (PACLIC 33), pages 10-18, Hakodate, Japan, September 13-15, 2019

ite) consonants/vowels, that can come for each slot is 19, 21, and 27. For instance, in a syllable '간 (*kan*)', the three clock-wisely arranged characters ㄱ, ㅏ, and ㄴ, which sound *kiyek* (stands for *k*; among 19 candidates), *ah* (stands for *a*; among 21 candidates), and *niun* (stands for *n*; among 27 candidates), refers to the *first*, the *second* and the *third* sound respectively.

To compare five different *Jamo*/character-level embedding methodologies that are possible in Korean, we first review the related studies and the previous approaches. Then, two datasets are introduced, namely binary sentiment classification and multi-class intention identification, to investigate the performance of each representation under recurrent neural network (RNN)-based analysis. After searching for an effective encoding scheme, we demonstrate how the result can be adopted in combating other tasks and discuss if a similar approach can be applied to the languages with the complex writing system.

2 Related Work

Inter-language comparison with word and character embedding was thoroughly investigated in Zhang and LeCun (2017), for Chinese, Japanese, Korean, and English. The paper investigates the languages via representations including *character, byte, romanized character, morpheme*, and *romanized morpheme*. The observation of tendency for Korean suggests that adopting the raw characters outperforms utilizing the romanized character-level features, and moreover both the performance are far beyond the morpheme-level features. However, to be specific on the experiment, decomposition of the morpho-syllabic blocks was not conducted, and the experiment did not make use of the dense embedding methodologies which can project the distributive semantics onto the representation. We concluded that more attention is to be paid to different character embedding methodologies of Korean. Here, to reduce ambiguity, we denote a morpho-syllabic block which consists of consonant(s) and a vowel by *character*, and the individual components by *Jamo*. A *Jamo* sequence is spread in the order of the *first* to the *third* sound if a *character* is decomposed.

There has been little study done on an effective text encoding scheme for Korean, a language that has distinguished character structure which can be decomposed into sub-characters. A comparative study on the hierarchical constituents of Korean morpho-syntax was first organized in Lee and Sohn (2016), in the way of comparing the performance of *Jamo, character, morpheme*, and *eojeol (word)*-level embeddings for the task of text reconstruction. In the quantitative analysis using edit distance and accuracy, the *Jamo*-level feature showed a slightly better result than the character-level one. The (sub-)character-level representations presented the outcome far better than the morpheme or *eojeol*-level cases, as in the classification task of Zhang and LeCun (2017). The results show the task-independent competitiveness of the character-level features.

In a more comprehensive viewpoint, Stratos (2017) showed that *Jamo*-level features combined with word and character-level ones display better performance with the parsing task. With more elaborate character processing, especially involving *Jamo*s, Shin et al. (2017) and Cho et al. (2018c) made progress recently in the classification tasks. Song et al. (2018) aggregated the sparse features into multi-hot representation successfully, enhancing the output within the task of error correction. In a slightly different manner, Cho et al. (2018a) applied dense vectors for the representation of the characters, obtained by skip-gram (Mikolov et al., 2013), improving the naturalness of word segmentation for noisy Korean text. To figure out the tendency, we implement the aforementioned *Jamo*/character-level features and discuss the result concerning classification tasks. The details on each approach are to be described in the following section.

3 Experiment

In this section, we demonstrate the featurization of five (sub-)character embedding methodologies, namely (i) ***Jamo*** (Shin et al., 2017; Stratos, 2017) (ii) **modified *Jamo*** (Cho et al., 2018c), (iii) **sparse *character* vectors**, (iv) **dense *character* vectors** (Cho et al., 2018a) trained based on fastText (Bojanowski et al., 2016), and (v) **multi-hot *character* vectors** (Song et al., 2018). We featurize only *Jamo*/character and no other symbols such as numbers and special letters is taken into account.

	Representation	Property	Dimension	Feature type
(i) *Shin2017*	ㄱ ··· ㅎ / ㅏ ··· ㅢ / ㄱ ··· ㉥	*Jamo*-level	67	one-hot
(ii) *Cho2018c*	(i) + ㄱ ··· ㅏ ··· ㉥	*Jamo*-level	118	one-hot
(iii) *Cho2018a*-Sparse	··· 간 ··· 밤 ··· 햇 ···	character-level	2,534	one-hot
(iv) *Cho2018a*-Dense	··· 간 ··· 밤 ··· 햇 ···	character-level	100	dense
(v) *Song2018*	··· 간 ··· 밤 ··· 햇 ··· + α	character-level	67	multi-hot

Table 1: A description on the *Jamo*/character-level features (i-v).

For (i), we used one-hot vectors of dimension 67 (= 19 + 21 + 27), which is smaller in width than the ones suggested in Shin et al. (2017) and Stratos (2017), due to the omission of special symbols. Similarly, for (ii), 118-dim one-hot vectors are constructed. The different point of (ii) regarding (i) is that it considers the cases that *Jamo* is used in the form of single (or composite) consonant or vowel, as frequently observed in the social media text. The cases make up an additional array of dimension 51.

For (iii) and (iv), we adopted a vector set that is distributed publicly in Cho et al. (2018a), reported to be extracted from a drama script corpus of size 2M. Constructing the vectors of (iii) is intuitive; for N characters in the corpus, a N-dimensional one-hot vector is assigned for each. Case of (iv) can be considered awkward in the sense of using characters as a meaningful token, but we concluded that the Korean characters can be handled as a word piece[3] (Sennrich et al., 2015) or subword n-gram (Bojanowski et al., 2016) concerning the nature of their composition. All the characters are reported to be treated as a separate token (subword) in the training phase that uses skip-gram (Mikolov et al., 2013). Although the number of possible character combinations in Korean is precisely 11,172 (= 19 * 21 * 28), the number of ones that are used in real-life reaches about 2,500 (Kwon et al., 1995). Since the description says that the corpus is removed with punctuation and consists of around 2,500 Korean syllables, we exploited the dictionary of 100-dim fastText-embedded vectors which is provided in the paper, and extracted the list of the characters to construct a one-hot vector dictionary[4].

[3]The word piece models were not investigated in this study since here we concentrate on the (sub-)character-level embeddings.

[4]Two types of embeddings were omitted, namely the *Jamo*-based fastText and the 11,172-dim one-hot vectors; the former was considered inefficient since there are only 118 symbols at most and the latter was assumed to require a huge computation.

(v) is a hybrid of *Jamo* and character-level features; three vectors indicating the first to the third sound of a character, namely the ones with dimension 19, 21, and 27 each, are concatenated into a single multi-hot vector. This preserves the conciseness of the *Jamo*-level one-hot encodings and also maintains the characteristics of conjunct forms. In summary, (i) utilizes 67-dim one-hot vectors, (ii) 118-dim one-hot vectors, (iii) 2,534-dim one-hot vectors, (iv) 100-dim dense vectors, and (v) 67-dim multi-hot vectors (Table 1).

3.1 Task description

For evaluation, we employed two classification tasks that can be conducted with the character-level embeddings. Due to a shortage of reliable open source data for Korean, we selected the datasets that show a clear description of the annotation. One, a sentiment analysis corpus, is expected to display how well each character-level encoding scheme conveys the information regarding lexical semantics. The other, an intention analysis corpus, is expected to show how comprehensively each character-level encoding scheme deals with the syntax-semantic task that concerns sentence form and content. The details on each corpus are stated below.

3.1.1 Naver sentiment movie corpus

The corpus NSMC[5] is a widely used benchmark for evaluation of Korean language models. The annotation follows Maas et al. (2011) and incorporates 150K:50K documents for the train and test set each. The authors assign a positive label for the reviews with a score > 8 and negative for the ones with a score < 5 (in 10-scale), adopting a binary labeling system. To prevent confusion that comes from gray-zone data, neutral reviews were removed. The posi-

[5]https://github.com/e9t/nsmc

tive and negative instances are equally distributed in both train and test set.

3.1.2 Intonation-aided intention identification for Korean

The corpus 3i4K[6] (Cho et al., 2018b) is a recently distributed open-source data for multi-class intention identification. The labels, in total seven, include *fragment* and five clear-cut cases (*statement, question, command, rhetorical question (RQ), rhetorical command (RC)*). The remaining class is for the intonation-dependent utterances whose intention mainly depends on the prosody assigned to underspecified sentence enders, considering head-finality of the Korean language. Since the labels are elaborately defined and the corpus is largely hand-labeled (or hand-generated), the corpus size is relatively small (total 61K) and some classes possess a tiny volume (e.g., about 1.7K for RQs and 1.1K for RCs). However, such challenging factors of the dataset can show the aspects of the evaluation that can be overlooked. The train-test ratio is 9:1.

3.2 Feature engineering

In the first task, to manage with the document size, the length of *Jamo* or character sequence was fixed to the maximum of (i-ii) 420 and (iii-v) 140[7]. Similarly, in the second task, (i-ii) 240 and (iii-v) 80[8]. The length regarding (i-ii) being three times as long as that of (iii-v) comes from the spreadings of the sub-characters for each character.

For both tasks, the document was numericalized in the way that the tokens are placed on the right end of the feature, to preserve *Jamo*s or characters which may incorporate syntactically influential components of the phrases in a head-final language. For example, in a sentence "배고파 (pay-ko-pha, *I'm hungry*)", a vector sequence is arranged in the form of [0 0 $\cdots$ 0 0 $v1$ $v2$ $v3$], where $v1$, $v2$, and $v3$ each denotes the vector embeddings of the characters *pay*, *ko*, and *pha*. Here, *pha* encompasses the head of the phrase with the highest hierarchy in the sentence, which assigns the sentence a speech act of *statement*. The spaces between *eojeol*s were represented as zero vector(s)[9].

To look into the content of the corpora, the first dataset (NSMC) contains many incomplete characters such as solely used sub-characters (e.g., ㅋ ㅋ, ㄲㄲ) and non-Korean symbols (e.g., Chinese characters, special symbols, punctuations). The former ones were treated as characters, whereas the latter ones were ignored in all features. Although (i, iii, iv) do not represent the symbols regarding the former as non-zero vector while (ii, v) do so, we concluded that this does not threaten the fairness of the evaluation, since a wider range of representation is own advantage of each feature. The second dataset (3i4K) contains only the full characters. Thus no disturbance or biasedness was induced in the featurization.

3.3 Implementation

The implementation was done with Hangul Toolkit[10], fastText[11], and Keras (Chollet and others, 2015), which were used for character decomposition, dense vector embedding and RNN-based training, respectively. For RNN models, bidirectional long short-term memory (BiLSTM) (Schuster and Paliwal, 1997) and self-attentive sentence embedding (BiLSTM-SA) (Lin et al., 2017) were applied.

In vanilla BiLSTM, an autoregressive system that is representatively utilized for time-series analysis, a fully connected layer (FC) is connected to the last hidden layer of BiLSTM, finally inferring the output with a softmax activation. In BiLSTM with a self-attentive embedding, the context vector whose length equals to that of the hidden layers of the BiLSTM, is jointly trained along with the network so that it can provide the weight assigned to each hidden layer. The weight is obtained by making up an attention vector via a column-wise multiplication of the context vector and the hidden layers. The model specification is provided as supplementary material.

3.4 Result

For both tasks, we split the train set into 9:1 to have a separate validation set. As a result, we achieved

[6] https://github.com/warnikchow/3i4k

[7] The data description says the maximum volume of the input characters is 140.

[8] The number of the utterances with the length longer than 80 were under 40 ($< 0.07\%$).

[9] *Eojeol* denotes the unit of spacing in the written Korean.

[10] https://github.com/bluedisk/hangul-toolkit

[11] https://pypi.org/project/fasttext/

Accuracy (F1-score)	NSMC		3i4K	
	BiLSTM	*BiLSTM-SA*	*BiLSTM*	*BiLSTM-SA*
(i) ***Shin2017***	0.8203	0.8316	0.8694 (0.7443)	**0.8769** (0.7692)
(ii) ***Cho2018c***	0.7895	0.7973	0.8688 (0.7488)	0.8728 (0.7727)
(iii) ***Cho2018a*-Sparse**	**0.8271**	**0.8321**	0.8694 (0.7763)	0.8722 (0.7741)
(iv) ***Cho2018a*-Dense**	**0.8312**	**0.8382**	**0.8799** (0.7887)	**0.8844** (0.7963)
(v) ***Song2018***	**0.8271**	0.8314	**0.8696** (0.7713)	0.8761 (0.7828)

Table 2: Performance comparison. Only the accuracy is provided for NSMC since the labels are equally distributed. Two best models regarding accuracy (and F1-score for 3i4K) are bold (and underlined) for both tasks, with each architecture (BiLSTM and BiLSTM-SA).

135K instances for the training of NSMC (15K for the validation) and 50K for the training of 3i4K (5K for the validation).

3.4.1 Performance

The evaluation phase displays the pros and cons of the conventional methodologies (Table 2). In both tasks, (iv) showed significant performance. It is assumed that the result comes from the distinguished property of (iv); it does not break up the syllabic blocks and at the same time provides the distributional semantics to the models, in the way of employing skip-gram (Mikolov et al., 2013). (v) also performs in a similar way, by using a multi-hot encoding that assigns own role to each vector representation, displaying a compatible performance using BiLSTM in both tasks.

(iii) preserves the blocks as well, but one-hot encoding hardly gives any information on each character. It is assumable that such representation can be powerful for the dataset with a rich and balanced resource, as in NSMC, but is weak if the class volume is imbalanced, which led to an insignificant result for 3i4K. Although some compatible performance was achieved with BiLSTM, the models regarding (iii) reached saturation fast and displayed overfitting afterward, while the models with the other features showed a gradual increase in accuracy. The reason for fast saturation seems to be the limited flexibility coming from the vast parameter set size.

The unexpected point is that the models utilizing additional letters (ii) showed significant performance degeneration in NSMC task, where the solely used sub-characters (as ㅋ ㅋ implying joy or ㅠ ㅠ implying sadness) were expected to be aggregated into the featurization and yield a positive outcome. In the pilot research executed without validation set (that the model performing best with the test set was searched greedily), a comparable result as in (i) was shown. Thus, the reason for the degeneration seems to be the limitation of using a validation set, where the cutback in the training resource is inevitable[12]. Also, some solely used sub-characters might have caused the disorder in the inference of the sentiment, since not all the users employ the sentiment-related sub-characters in the same way. Supporting this observation, feature (ii) shows much less difference with (i) in 3i4K, where only the full characters are adopted.

3.4.2 Using self-attentive embedding

The advantage of using self-attentive embedding was the most emphasized in *Jamo*-level feature (i) for both tasks, and the least in (iii) (Table 2). We assume that relatively more significant improvement using (i) originates in the decomposability of the blocks. If a sequence of the blocks is decomposed into the sequence of sub-characters, the morphemes can be highlighted to provide more syntactically/semantically meaningful information to the system, especially the ones that could not have been revealed in the block-preserving environment (iii-v). For example, a Korean word '이상한 (*i-sang-han*, strange)' can be split into '이상하 (*i-sang-ha*, the root of the word)' and '-ㄴ (*-n*, a particle that makes the root an adjective)', making up the circumstances in which the presence and role of the morphemes is pointed out. This property is also reflected in the case of using the feature (ii), although the absolute score is not notable.

[12]It is highly recommended to use the cross-validation if one wants to boost the performance.

Trainable param.s & Training time	*BiLSTM*		*BiLSTM-SA*	
	Param.s	Time / epoch	Param.s	Time / epoch
(i) ***Shin2017***	34,178	13.5m	297,846	18m
(ii) ***Cho2018c***	47,234	16m	310,902	20.5m
(iii) ***Cho2018a*-Sparse**	665,730	33m	772,318	38.5m
(iv) ***Cho2018a*-Dense**	42,626	6.5m	149,214	6m
(v) ***Song2018***	34,178	6m	140,766	6m

Table 3: Computation burden for NSMC models.

3.4.3 Decomposability vs. Local semantics

The point described above is the disadvantage of character-level features (iii-v) in the sense that in such ones, characters cannot be decomposed, even for the sparse multi-hot encoding. The higher performance of (iv-v) compared to the *Jamo*-level features, which is currently displayed, can hence be explained as a result of preserving the cluster of letters. If the computation resource is sufficient so that exploiting deeper networks (e.g., Transformer (Vaswani et al., 2017) or BERT (Devlin et al., 2018)) is available, we infer that (i-ii) may also show compatible or better performance, since the modern self-attention-based mechanisms utilize the positional encodings to grasp the relation between the tokens, advanced from the location-based models we adopted. Nevertheless, it is still quite impressive that (iv) scores the highest even though the utilized dictionary does not incorporate all the available character combinations. It is suspected to be where the distributive semantics on the word pieces are engaged in.

3.4.4 Computation efficiency

In this study, we investigate only on the classification tasks. Notwithstanding they take a short amount of time for training and inference, the measurement on parameter volume and complexity is meaningful (Table 3). It is observed that (v) yields a compatible or better performance with respect to the other schemes, accompanied by less burden of computation. Besides, we argue that the multi-hot encoding (v) has a significant advantage over the rest in terms of multiple usages; it possesses both conciseness of the sub-character (*Jamo*)-level features and local semantics (although not distributional) of the character-level features. Due to these reasons, the derived models are fast in training and also have potential to be effectively used in sentence reconstruction or generation, as shown in Song et al. (2018), where applying large-dimensional one-hot encoding has been considered challenging.

4 Discussion

The primary goal of this paper is to search for a *Jamo*/character-level encoding scheme that best resolves the given task in Korean NLP. Empirically, we found out that the fastText-embedded vectors outperform the other features if provided with the same environment (model architecture). It is highly probable that the distributive semantics plays a significant role in the NLP tasks concerning syntax-semantics, at least in the feature-based approaches (Mikolov et al., 2013; Pennington et al., 2014). However, we experimentally verified that even with traditional feature-based systems, the sparse encoding schemes also perform adequately with the dense one, especially displaying computation efficiency in the multi-hot case.

At this moment, we want to emphasize that the utility of the comparison result is not only restricted to Korean, in that the introduced character encoding schemes are also available in other languages. Although the Korean writing system is unique, the Japanese language incorporates several morae (e.g., small *tsu*) that approximately correspond to the third sound (*cong-seng*) of the Korean characters, which may let the Japanese characters be encoded in a similar manner with the cases of Korean. Also, each character of the Chinese language (and *kanji* in Japanese) can be further decomposed into sub-characters (*bushu* in Japanese) that have meanings as well, as suggested in Nguyen et al. (2017) (e.g., 鯨 "whale" to 魚 "fish" and 京 "capital city").

Besides, many other languages that are used in South Asia (India), such as Telugu, Devanagari, Tamil, and Kannada, have writing system type of Abugida[13] (Daniels and Bright, 1996), the composition of consonant and vowel. The cases are not the same as Korean in view of a writing system since featural decomposition of the Abugida characters is not represented in the way of segmentation of a glyph. However, for example, instead of listing all the CV combinations, one can simplify the representation by segmenting the property of the charac-

[13] https://en.wikipedia.org/wiki/Writing_system

ter into consonant and vowel and making up a two-hot encoded vector. The similar kind of character embedding can be applied to many native Philippine languages such as Ilocano. Moreover, we believe that the argued type of featurization is robust in combating the noisy user-generated texts.

5 Conclusion

In this study, we have reviewed the five different types of (sub-)character-level embedding for a character-rich language. It is remarkable that the dense and multi-hot representation perform best given the classification tasks, and specifically, the latter one has the potential to be utilized beyond the given tasks due to its conciseness and computation efficiency. The utility of the sub-character-level features is also noteworthy in the syntax-semantic tasks that require morphological decomposition. It is expected that the overall performance tendency may provide a useful reference for the text processing of other character-rich languages with conjunct forms in the writing system, including Japanese, Chinese, and the languages of various South and Southeast Asian regions. A brief tutorial on both datasets using embedding methodologies presented in this paper is available online[14].

Acknowledgement

This research was supported by Projects for Research and Development of Police science and Technology under Center for Research and Development of Police science and Technology and Korean National Police Agency funded by the Ministry of Science, ICT and Future Planning (PA-J000001-2017-101). Also, this work was supported by the Technology Innovation Program (10076583, Development of free-running speech recognition technologies for embedded robot system) funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea). The authors appreciate Yong Gyu Park for giving helpful opinions in performing validation and evaluation. After all, the authors want to send great thanks to the three anonymous reviewers for the insightful comments.

[14] https://github.com/warnikchow/kcharemb

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Won Ik Cho, Sung Jun Cheon, Woo Hyun Kang, Ji Won Kim, and Nam Soo Kim. 2018a. Real-time automatic word segmentation for user-generated text. *arXiv preprint arXiv:1810.13113*.

Won Ik Cho, Hyeon Seung Lee, Ji Won Yoon, Seok Min Kim, and Nam Soo Kim. 2018b. Speech intention understanding in a head-final language: A disambiguation utilizing intonation-dependency. *arXiv preprint arXiv:1811.04231*.

Yong Woo Cho, Gyu Su Han, and Hyuk Jun Lee. 2018c. Character level bi-directional lstm-cnn model for movie rating prediction. In *Proceedings of Korea Computer Congress 2018 [in Korean]*, pages 1009–1011.

François Chollet et al. 2015. Keras. `https://github.com/fchollet/keras`.

Peter T Daniels and William Bright. 1996. *The world's writing systems*. Oxford University Press on Demand.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, and William W Cohen. 2016. Tweet2vec: Character-based distributed representations for social media. *arXiv preprint arXiv:1605.03481*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Hyuk-Chul Kwon, Ho-Jeong Hwang, Min-Jung Kim, and Seong-Whan Lee. 1995. Contextual postprocessing of a korean ocr system by linguistic constraints. In *icdar*, page 557. IEEE.

Jaeyeon Lee and Kyung-Ah Sohn. 2016. Comparison of decoder performance by representation for korean language in rnn encoder-decoder model. In *Proceedings of the KISS conference [in Korean]*, pages 609–611.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011.

Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Viet Nguyen, Julian Brooke, and Timothy Baldwin. 2017. Sub-character neural language modelling in japanese. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 148–153.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing*, pages 1532–1543.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Haebin Shin, Min-Gwan Seo, and Hyeongjin Byeon. 2017. Korean alphabet level convolution neural network for text classification. In *Proceedings of Korea Computer Congress 2017 [in Korean]*, pages 587–589.

Chisung Song, Myungsoo Han, Hoon Young Cho, and Kyong-Nim Lee. 2018. Sequence-to-sequence autoencoder based korean text error correction using syllable-level multi-hot vector representation. In *Proceedings of HCLT [in Korean]*, pages 661–664.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Karl Stratos. 2017. A sub-character architecture for korean language processing. *arXiv preprint arXiv:1707.06341*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y Ng. 2016. Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727*.

Xiang Zhang and Yann LeCun. 2017. Which encoding is the best for text classification in chinese, english, japanese and korean? *arXiv preprint arXiv:1708.02657*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Supplementary Material

BiLSTM and *BiLSTM-SA* model specification

Variables

- Sequence length (L) and the number of output classes (N) depend on the task. For NSMC, L = 420 for feature (i-ii) and 140 for (iii-v). For 3i4K, L=240 for feature (i-ii) and 80 for (iii-v). N equals 2 and 7 for NSMC and 3i4K, respectively.
- Character vector dimension (D) depends on the feature. For features (i-v), D equals 67, 118, 2534, 100, and 67, respectively.

BiLSTM

- Input dimension: (L, D)
- RNN Hidden layer width: 64 (=32×2)
- The width of FCN connected to the last hidden layer: 128 (Activation: *ReLU*)
- Output layer width: N (Activation: *softmax*)

BiLSTM-SA

- Input dimension: (L, D)
- The dimension of RNN hidden layer sequence output: (64 (= 32×2), L)
  >> each layer connected to FCN of width: 64 (Activation: *tanh*; equals to d_a in Lin et al. (2017)) [a]
- Auxiliary zero vector size: 64
  >> connected to FCN of width 64 (Activation: *ReLU*, Dropout (Srivastava et al., 2014): 0.3)
  >> connected to FCN of width 64 (Activation: *ReLU*) [b]
- Vector sequence [a] is column-wisely dot-multiplied with [b] to yield the layer of length L
  >> connected to an attention vector of size L (Activation: *softmax*)
  >> column-wisely multiplied to the hidden layer sequence to yield a weighted sum [c] of width 64
  >> [c] is connected to an FCN of width: 256 (Activation: *ReLU*, Dropout: 0.3) × 2 (multi-layer)
- Output layer width: N (Activation: *softmax*)

Settings

- Optimizer: Adam (Kingma and Ba, 2014) (Learning rate: 0.0005)
- Loss function: Categorical cross-entropy
- Batch size: 64 for NSMC, 16 for 3i4K (due to the difference in the corpus volume)
- For 3i4K, class weights were taken into account to compensate the volume imbalance.
- Device: Nvidia Tesla M40 24GB