Proceedings of Machine Learning Research 77:561–573, 2017 ACML 2017

Radical-level Ideograph Encoder for RNN-based Sentiment Analysis of Chinese and Japanese

Yuanzhi Ke ENSHIKA8811.A6@KEIO.JP
Department of Information and Computer Science, Faculty of Science and Engineering, Keio University

Masafumi Hagiwara HAGIWARA@KEIO.JP
Department of Information and Computer Science, Faculty of Science and Engineering, Keio University

Editors: Yung-Kyun Noh and Min-Ling Zhang

Abstract

The character vocabulary can be very large in non-alphabetic languages such as Chinese and Japanese, which makes neural network models huge to process such languages. We explored a model for sentiment classification that takes the embeddings of the radicals of the Chinese characters, i.e, hanzi of Chinese and kanji of Japanese. Our model is composed of a CNN word feature encoder and a bi-directional RNN document feature encoder. The results achieved are on par with the character embedding-based models, and close to the state-of-the-art word embedding-based models, with 90% smaller vocabulary, and at least 13% and 80% fewer parameters than the character embedding-based models and word embedding-based models respectively. The results suggest that the radical embedding-based approach is cost-effective for machine learning on Chinese and Japanese.

Keywords: Natural Language Processing, Sentiment Analysis

1. Introduction

Word embeddings have been widely used for natural language processing (NLP) tasks (Collobert et al., 2011). However, the large word vocabulary makes word embeddings expensive to train. Some people argue that we can model languages at the character-level (Kim et al., 2016). For alphabetic languages such as English, where the characters are much fewer than the words, the character embeddings achieved the state-of-the-art results with much fewer parameters.

Unfortunately, for the other languages that use non-alphabetic systems, the character vocabulary can be also large. Moreover, Chinese and Japanese, two of the most widely used non-alphabetic languages, especially contain large numbers of ideographs: hanzi of Chinese and kanji of Japanese. The character vocabulary can be as scalable as the word vocabulary (e.g., see the datasets introduced in Section 3.1). Hence the conventional character embedding-based method is not able to give us a slim vocabulary on Chinese and Japanese.

For convenience, let us collectively call hanzi and kanji as Chinese characters. Chinese characters are ideographs composed with semantic and phonetic components called radicals, both of which are available for character recognition, and the semantic information may be embedded in Chinese characters by the semantic radicals (Williams and Bever, 2010).

© 2017 Y. Ke & M. Hagiwara.

Besides, though the character vocabulary is huge, the number of the radicals is much fewer. Accordingly, we explored a model that represents the Chinese characters by the sequence of the radicals. We applied our proposed model to sentiment classification tasks on Chinese and Japanese and achieved the follows:

- The results achieved by our proposed model are close to the state-of-the-art word embedding-based models, with approximately 90% smaller vocabulary, 91% and 82% fewer parameters for Chinese and Japanese respectively.
- The results are on par with the character embedding-based models, with approximately 13% fewer parameters, and 90% smaller vocabulary for both Chinese and Japanese.

2. Methodology

The architecture of our proposed model is as shown in Fig. 1. It looks similar to the character-aware neural language model proposed by Kim et al. (2016), but we represent a word by the sequence of radical embeddings instead of character embeddings. Besides, unlike the former model, there are no highway layers in the proposed model, because we find that highway layers do not bring significant improvements to our proposed model (see Section 5.3).

2.1. Representation of Characters: Sequences of Radical-level Embeddings

Radicals are the orthographic units of Chinese characters. Psychological researches have indicated that they are functional in character recognition of human beings, comparable to the letters in alphabetic languages (Chen, 1996). Inspired by them, we propose the representation on the basis of the radicals. For every character, we use a sequence of n radical-level embeddings to represent it. They are not treated as in a bag because the position of each radical is related to how it is informative (Hsiao et al., 2007). For a Chinese character, it is the sequence of the radical embeddings. When it comes to the other characters, including kanas of Japanese, alphabets, digits, punctuation marks, and special characters, it is the sequence comprised of the corresponding character embedding and $n-1$ zero vectors. We zero-pad the radical sequences of all the characters to align the lengths.

2.2. From Character Radicals to Word Features: CNN Encoder

CNNs (LeCun et al., 1989) have been used for various NLP tasks and shown effective (Collobert et al., 2011; Kim, 2014; Kim et al., 2016). For NLP, CNNs are able to extract the temporal features, reduce the parameters, alleviate over-fitting and improve generalization ability. We also take advantage of the weight sharing technology of CNNs to learn the shared features of the characters.

Let $\mathcal{C}$ be the radical-level vocabulary that contains Chinese character radicals, kanas of Japanese, alphabets, digits, punctuation marks, and special characters, $\mathbf{Q} \in \mathbb{R}^{d \times |\mathcal{C}|}$ be the matrix of all the radical-level embeddings, d be the dimension of each radical-level

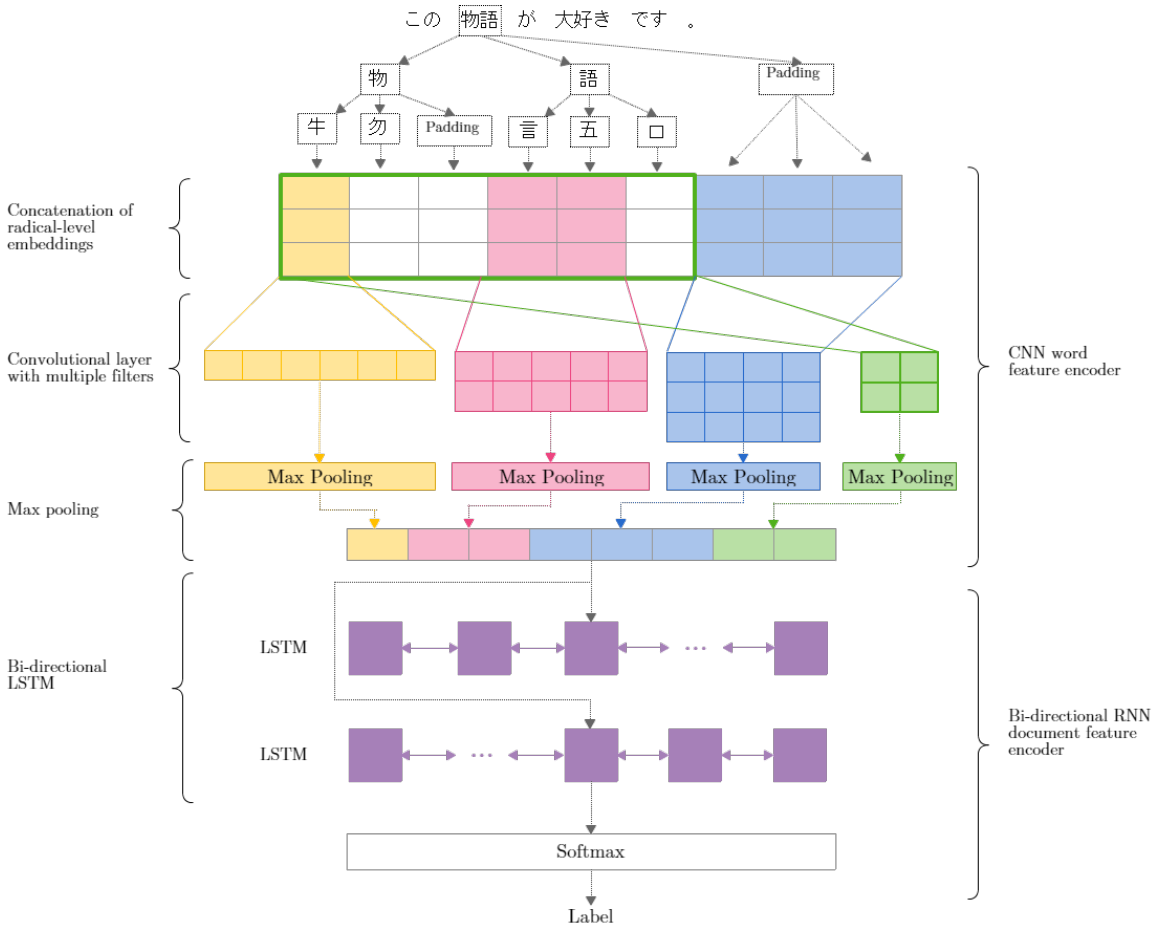

Figure 1: Architecture of our proposed model. The Chinese characters are split into radicals from the left to the right, the top to the bottom. Each character is represented by a sequence of radical-level embeddings. The radical embedding sequences of the characters in a word are input into a convolutional layer with multiple filters to encode the word features, which are then pooled max-over-time and input into a bi-directional RNN. The bi-directional RNN encodes the document-level feature, and after which the estimation of the label is obtained by applying affine transformation and softmax on the document-level feature. In the figure, there are four filters denoted with different colors. Their widths, strides, and numbers of output channels are different. The filters with stride of one are for radical-level information. Those with stride of three are for character-level information.

embedding. We have introduced that each character is represented by a sequence of radical-level embeddings of length n. Thus a word k composed of m characters is represented by a matrix $\mathbf{C}^k \in \mathbb{R}^{d \times (m \times n)}$, each column of which is a radical-level embedding.

We apply convolution between $\mathbf{C}^k$ and several filters (convolution kernels). For each filter h, we apply a nonlinear activation function g and a max-pooling on the output to obtain a feature vector. Let r be the stride, w be the window, $\mathbf{H}_h$ be the hidden weight of a filter, respectively. The feature vector of k obtained by h is given by:

$$\mathbf{x}_h^k = \text{maxpool}(g(\mathbf{C}^k) \star \mathbf{H}_h + \mathbf{b}) \tag{1}$$

where $\star$ is the convolution operator. The pooling window is $\frac{m \times n}{r} - w + 1$ to obtain the most important information of word k.

We have two kinds of filters: (1) the filters with stride $r = 1$ to obtain radical-level features; (2) the filters with stride $r = n$ to obtain character-level features.

After the max-pooling layer, we concatenate and flatten all of the outputs through all of the filters as the feature vector of the word. Let a be the number of the output channel of each filter. If we use totally h filters, each output of which is $\mathbf{x}_h^k = [x_{h1}\ x_{h2}\ x_{h3}\ ...\ x_{ha}]$, the output feature of k is $\mathbf{x}_h^k = [x_{11}\ x_{22}\ x_{33}\ ...\ x_{1a}\ x_{21}\ x_{22}\ x_{23}\ ...\ x_{2a}\ x_{h1}\ x_{h2}\ x_{h3}\ ...\ x_{ha}]$. Here, we assume that the number of the output channels of every filter is the same, but we tailor it for each filter in the experiments following Kim et al. (2016).

2.3. From Word Features to Document Features: Bi-directional Long Short-term Memory RNN Encoder

An RNN is a kind of neural networks designed to learn sequential data. The output of an RNN unit at time t depends on the output at time $t-1$. Bi-directional RNNs (Schuster and Paliwal, 1997) are able to extract the past and future information for each node in a sequence, have shown effective for Machine Translation (Bahdanau et al., 2015) and Machine Comprehension (Kadlec et al., 2016).

A Long Short-term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) Unit is a kind of unit for RNN that keeps information from long range context. We use a bi-directional RNN of LSTM to encode the document feature from the sequence of the word features.

An LSTM unit contains a forget gate $\mathbf{f}_t$ to decide whether to keep the memory, an input gate $\mathbf{i}_t$ to decide whether to update the memory and an output gate $\mathbf{o}_t$ to control the output. Let $\mathbf{h}_t$ be the output of a LSTM unit at time t, $\tilde{\gamma}_t$ be the candidate cell state at time t, γ_t be the cell state at time t. They are given by:

$$\begin{aligned}
\mathbf{f}_t &= \sigma\left(\mathbf{W}_f\left[\mathbf{h}_{t-1}, \mathbf{x}_t\right] + \mathbf{b}_f\right) \\
\mathbf{i}_t &= \sigma\left(\mathbf{W}_i\left[\mathbf{h}_{t-1}, \mathbf{x}_t\right] + \mathbf{b}_i\right) \\
\tilde{\gamma}_t &= \tanh\left(\mathbf{W}_c\left[\mathbf{h}_{t-1}, \mathbf{x}_t\right] + \mathbf{b}_c\right) \\
\gamma_t &= \mathbf{f}_t * \gamma_{t-1} + \mathbf{i}_t * \tilde{\gamma}_t \\
\mathbf{o}_t &= \sigma\left(\mathbf{W}_o\left[\mathbf{h}_{t-1}, \mathbf{x}_t\right] + \mathbf{b}_o\right) \\
\mathbf{h}_t &= \mathbf{o}_t * \tanh\left(\gamma_t\right)
\end{aligned} \tag{2}$$

where, $\sigma(\cdot)$ and $*$ are the element-wise sigmoid function and multiplication operator.

Our proposed model contains two RNN layers that read document data from different directions. Let s be a document composed of l words. One of the RNN layers reads the document from the first word to the lth word, the other reads the document from the lth word to the first word. Let $\mathbf{h}_l$ be the final output of the former RNN layer and $\mathbf{h}_1$ be the final output of the latter. We concatenate $\mathbf{h}_l$ and $\mathbf{h}_1$ as the document feature. After that, we apply an affine transformation and a softmax to obtain the prediction of the sentiment labels:

$$\mathbf{Pr}\left(\hat{y}=i|s\right)=\frac{exp\left(\mathbf{W}_p^i\mathbf{z}+\mathbf{b}_p^i\right)}{\sum_{i'\in\mathcal{E}}exp\left(\mathbf{W}_p^{i'}\mathbf{z}+\mathbf{b}_p^{i'}\right)} \tag{3}$$

where $z=h_l^\frown h_1$, $^\frown$ is the concatenation operator. $\hat{y}$ is the estimated label of the document, i is one of the labels in the label set $\mathcal{E}$.

We minimize the cross entropy loss to train the model. Let $\mathcal{S}$ be the set of all the documents, and y be the true label of document s, the loss is given by:

$$\mathcal{L}=-\sum_{s\in\mathcal{S}}\left(\log\mathbf{Pr}(\hat{y}=y|s)\right) \tag{4}$$

3. Experimental Setup

3.1. Datasets

In the experiments, we used a Chinese dataset and a Japanese dataset. We used the publicly available Ctrip review data pack [1] for Chinese. They are comprised of travel reviews crawled from ctrip.com [2]. We used a subset of 10,000 reviews in the pack. We randomly select 8,000 and 2,000 from it for training and test, respectively. The Japanese dataset is provided by Rakuten, Inc. It contains 64,000,000 reviews of the products in Rakuten Ichiba [1]. The reviews are labeled with 6-point evaluation of 0-5. We labeled the reviews with less than 3 points as the negative samples, and the others as the positive samples. We randomly chose 10,000 reviews to align the size of the Chinese datasets, 8,000 and 2,000 from it for training and test, respectively.

Table 1: The statistics of the datasets. $|\mathcal{S}|$ = the number of the samples in each dataset; $|\mathcal{C}|$ = the size of the radical-level vocabulary; $|\mathcal{V}|$ = the size of the character vocabulary; $|\mathcal{W}|$ = the size of the word vocabulary; T_c = the number of Chinese characters; T_v = the number of all kinds of characters; T_w = the number of the words. The radical-level vocabularies contain Chinese characters, other CJK characters, alphabets, digits, punctuation marks and special characters.

Dataset	$\|\mathcal{S}\|$	$\|\mathcal{C}\|$	$\|\mathcal{V}\|$	$\|\mathcal{W}\|$	T_c	T_v	T_w	Language
Ctrip	10k	2k	21k	30k	1057k	1,256k	795k	Chinese
Rakuten	10k	2k	21k	18k	231k	1,005k	608k	Japanese

1. http://www.datatang.com/data/11936
2. http://www.ctrip.com
1. http://www.rakuten.co.jp/

The detailed information of the preprocessed datasets is shown in Table 1. The character vocabularies are as scalable as the word vocabularies but the radical-level vocabularies are much smaller. The character vocabulary of the Rakuten dataset is even larger than its own word vocabulary.

3.2. Baselines

We compared the proposed model with the follows:

- **The character-aware neural language model** (Kim et al., 2016): It is an RNN language model that takes character embeddings as the inputs, encodes them with CNNs and then input them to RNNs for prediction. It achieved the state-of-the-art as a language model on alphabetic languages. We let it predict the sentiment labels instead of words.
- **Bi-directional RNN (Schuster and Paliwal, 1997) with word embeddings**: It is a classical bi-directional RNN classifier, basic but effective. We also employed LSTM for it, and input the word embeddings.
- **Hierarchical attention networks** (Yang et al., 2016): It is the state-of-the-art RNN-based document classifier. Following their method, the documents were segmented into shorter sentences of 100 words, and hierarchically encoded with bi-directional RNNs.
- **FastText** (Joulin et al., 2016): It is the state-of-the-art baseline for text classification, which simply takes n-gram features and classifies sentences by hierarchical softmax. We used the word embedding version but did not use the bigram version because the other models for comparison do not use bigram inputs.

3.3. Hyperparameters

The setup of the hyperparameters in our experiments is shown in Table 2. They were tuned on the development set of 4,000 reviews, 2,000 from another subset in the public Ctrip data pack, and the other 2,000 randomly chosen from the review data of Rakuten Ichiba. We aligned the sizes of the feature vectors of the words and the documents in different models for a fair comparison. All the embeddings are initialized randomly with the uniform distribution.

All of the models were trained by RMSprop (Tieleman and Hinton, 2012) with mini-batches of 100 samples. The learning rate and decay term were set as 0.001 and 0.9 respectively, also tuned on the development set.

3.4. Text Preprocess

We segmented the documents into words by Jieba [2] and Juman++(Morita et al., 2015) [3], respectively for Chinese and Japanese. We zero-padded the length of the sentences, words, and radical sequences of the characters as 500, 4 and 3, respectively.

2. https://github.com/fxsjy/jieba
3. For some non-Japanese tokens that creep in the dataset, Juman++ throws errors In such cases, we used Janome (http://mocobeta.github.io/janome/) instead.

Table 2: The setup of the hyperparameters tuned for the experiments. w = the filter width; r = the filter stride; a = the number of the output channels, as a function of $\frac{w}{r}$; g = the nonlinear activation function; $d_{\mathbf{c}}$ = the dimensions of the radical-level embeddings or the character embeddings, in the proposed model and Kim et al. (2016)'s model respectively; $d_{\mathbf{x}}$ = the dimension of the word embedding, or the output of the CNN encoder in the proposed model and Kim et al. (2016); $d_{\mathbf{z}}$ = the dimension of the document feature vector. They were tuned on the development set of totally 4,000 reviews.

Non-word embedding-based models							
Model	w	r	a	$d_{\mathbf{c}}$	$d_{\mathbf{x}}$	$d_{\mathbf{z}}$	g
The proposed	$[1, 2, 3, 3, 6, 9]$	$[1, 1, 1, 3, 3, 3]$	$[50 \cdot \frac{w}{r}]$	15	600	300	ReLU
Kim et al. (2016)	$[1, 2, 3]$	$[1, 1, 1]$	$[100 \cdot \frac{w}{r}]$	15	600	300	ReLU

Word embedding-based models			
Model	$d_{\mathbf{x}}$	$d_{\mathbf{z}}$	g
Schuster and Paliwal (1997)	600	300	ReLU
Yang et al. (2016)	600	300	tanh
Joulin et al. (2016)	600	300	Linear

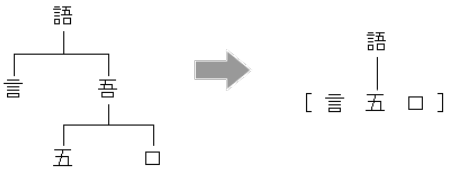

Figure 2: We split the Chinese characters into radicals from the left to the right, the top to the bottom.

We split the Chinese characters in CJK Unified Ideographs of ISO/IEC 10646-1:2000 character set, until there is no component can be split further, according to CHISE Character Structure Information Database [4]. Then the Chinese character is represented by the sequence of the radicals from the left to the right, the top to the bottom as shown in Fig. 2. The sequences are zero-padded to the same length. For an unknown Chinese character not in the set, we treat it as a special character.

4. Results

The number of parameters, test accuracy, and cross entropy loss of each model are as shown in Fig. 3. The proposed model has 13% fewer parameters than the character embedding-based model, 91% and 82% fewer parameters than the word embedding-based models for

4. http://www.chise.org/ids/

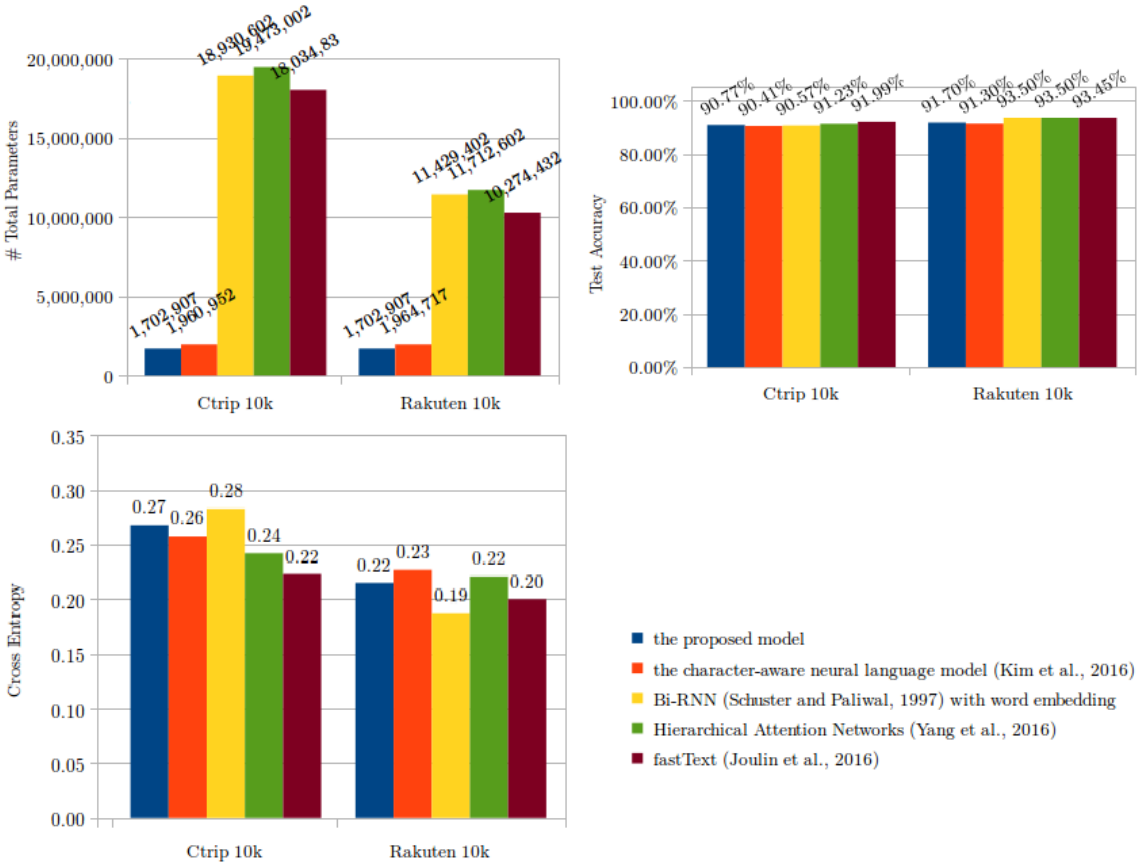

Figure 3: The number of parameters, test accuracy and cross entropy loss off each model. The proposed model has approximately 13% fewer parameters than the character embedding-based model, 91% and 82% fewer parameters than the word embedding-based models for Ctrip dataset and Rakuten dataset respectively. The performance of the proposed model is statistically the same as the character embedding-based model for all the datasets, approximately 99% of the word embedding-based model for Ctrip dataset, and 98% of the word embedding-based model for Rakuten dataset, with fewer parameters. The cross entropy losses are also close, especially on Rakuten dataset. The models are close to each other while the proposed model has the fewest parameters.

Ctrip dataset and Rakuten dataset, respectively. The accuracy is statistically the same as the character embedding-based model, approximately 98% of the word embedding-based model. The losses of the models are also close. The hierarchical attention networks and fastText achieved approximately 11% and 19% lower loss on Ctrip dataset. But on Rakuten dataset whose percentage of Chinese characters is higher, the differences between them and the proposed model drops to 0% and 9% respectively.

5. Discussions

5.1. The Proposed Model Is the Most Cost-effective

The performance of the proposed model is not significantly different from the character embedding-based baseline, and very close to the word embedding-based baselines, with a smaller vocabulary and fewer parameters. It indicates that radical-embeddings are at least as effective as the character-embeddings for Chinese and Japanese, but require less space. It suggests that for Chinese and Japanese, the radical embeddings are more cost-effective than the character embeddings.

5.2. The CNN Encoder Is Efficient

Even though the character vocabulary is as scalable as the word vocabulary on Chinese and Japanese, the character embedding-based method with CNN encoder can still reduce approximately 90% and 80% parameters for the Chinese and Japanese datasets, respectively. The CNNs allow low-dimension inputs, and share weights in the procedure of encoding the intpus to the high-dimension word features. It is probably the reason that it can save parameters although the sizes of the vocabularies are similar.

5.3. Highway Layers Are Not Effective For Us

Kim et al. (2016) reported that the highway networks (Srivastava et al., 2015) are effective for RNN language models. A highway layer is tailored to adaptively switch between a full-connected layer and a "highway" that directly outputs the input. We also studied that whether it is effective for our proposed model in the sentiment classification task. Following Kim et al. (2016), we attempted to input the flattened concatenated output of the max-pooling layer to a highway layer that employs ReLU before we input it to RNN. The change of the performance is as shown in Fig. 4.

We observed no significant improvement. The results with or without the highway layer is statistically the same (p-value $\rho = 0.860$ for Ctrip dataset, $\rho = 0.979$ for Rakuten dataset, tested by the Student's paired t-test). Probably for two-class sentiment classification, a full-connected layer with ReLU is not necessary between the CNN encoder and the bi-directional RNN encoder, hence the highway network learned to pass the inputs directly to the outputs all the time.

6. Related Works

The computational cost brought by the large word vocabulary is a classical problem when neural networks are employed for NLP. In the earliest works, people limited the size of

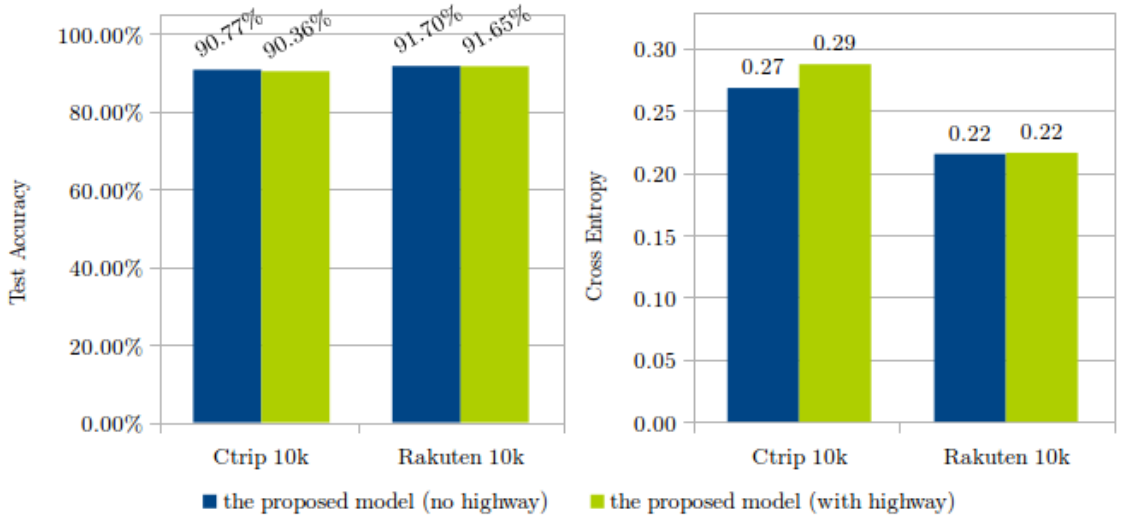

Figure 4: The test accuracy and cross-entropy loss with and without highway layers. No indicated difference is observed (p-value $\rho = 0.860$ for Rakuten dataset, $\rho = 0.979$ for Ctrip dataset, tested by the Student's paired t-test).

the vocabulary, which is not able to exploit the potential generalization ability on the rare words (Goodfellow et al., 2016, Chapter 12). It has made people explore alternative methods for the softmax function to efficiently train all the words, e.g., hierarchical softmax (Morin and Bengio, 2005), noise-contrastive estimation (Mnih and Teh, 2012) and negative sampling (Mikolov et al., 2013). However, the temporal complexity of the softmax function is not the only thing suffering the high-dimension vocabulary. Scalable word vocabulary leads to a large embedding layer, hence huge neural network with millions of parameters, which costs quite a few gigabytes to store. Zhang et al. (2015) proposed a convolutional neural network (CNN) that takes characters as the input for text classification and outperforms the previous models for large datasets. They showed the character-level CNNs are effective for text classification without the need for words. Kim et al. (2016) introduced a recurrent neural network (RNN) language model that takes character embeddings encoded by convolutional layers as the input. Their model has much fewer parameters than the models using word embeddings, and reached the performance of the state-of-the-art on English, and outperformed baselines on morphologically rich languages. However, for Chinese and Japanese, the character vocabulary is also large, and the character embeddings are blind to the semantic information of the radicals.

Psychological researches have shown that the radicals in Chinese characters are effective in word and character recognition for human beings, like the letters in alphabetic languages (Chen, 1996; Williams and Bever, 2010). Besides, enhancing the embeddings of word or character with the radical-level information was also reported effective in some other NLP tasks (Li et al., 2015; Yin et al., 2016). However, because the word or character

embeddings were still used, they were also suffering the issues of the vocabulary. On the other hand, in some other works, the characters were represented and modeled by images of them (Shimada et al., 2016; Liu et al., 2017). The authors showed that such approaches learned the radicals and were effective for text classification. However, the size of such model became larger than the character-based models. They are not for the storage issue.

7. Conclusion and Outlook

We have proposed a model that takes radicals of characters as the inputs for sentiment classification on Chinese and Japanese, whose character vocabulary can be as scalable as word vocabulary. Our proposed model is as powerful as the character embedding-based model, and close to the word embedding-based model for the sentiment classification task, with much smaller vocabulary and fewer parameters. The results show that the radical embeddings are cost-effective for Chinese and Japanese. They are useful for the circumstances where the storage is limited.

There are still a lot to do on radical embeddings. For example, a radical may be related to the meaning sometimes, but express the pronunciation at other times. We will work on dealing with such phenomena for machine learning in the future.

Acknowledgments

The authors would like to thank Rakuten, Inc. and the Advanced Language Information Forum (ALAGIN) for generously providing us the Rakuten Ichiba data.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations (ICLR2015)*, 2015.

Yi-Pung Chen. What are the functional orthographic units in chinese word recognition: The stroke or the stroke pattern? *The Quarterly Journal of Experimental Psychology: Section A*, 49(4):1024–1043, 1996.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997. ISSN 0899-7667.

Janet Hui-Wen Hsiao, Richard Shillcock, and Michal Lavidor. An examination of semantic radical combinability effects with lateralized cues in chinese character recognition. *Perception & psychophysics*, 69(3):338–344, 2007.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. Text understanding with the attention sum reader network. In *Proceedings of the 54th annual meeting of the Association for Computational Linguistics (ACL 2016)*, 2016.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. In *13th AAAI Conference on Artificial Intelligence (AAAI-16)*, pages 2741–2749, 2016.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Yanran Li, Wenjie Li, Fei Sun, and Sujian Li. Component-enhanced chinese character embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

Frederick Liu, Han Lu, Chieh Lo, and Graham Neubig. Learning character-level compositionality with visual features. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

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

Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. In *29th International Conference on Machine Learning (ICML 2012)*, 2012.

Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252, 2005.

Hajime Morita, Daisuke Kawahara, and Sadao Kurohashi. Morphological analysis for unsegmented languages using recurrent neural network language model. In *the 2015 Conference on Empirical Methods on Natural Language Processing (EMNLP 2015)*, pages 2292–2297, 2015.

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

Daiki Shimada, Ryunosuke Kotani, and Hitoshi Iyatomi. Document classification through image-based character embedding and wildcard training. In *Big Data (Big Data), 2016 IEEE International Conference on*, 2016.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS'15, pages 2377–2385, Cambridge, MA, USA, 2015. MIT Press.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Clay Williams and Thomas Bever. Chinese character decoding: a semantic bias? *Reading and Writing*, 23(5):589–605, 2010.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J Smola, and Eduard H Hovy. Hierarchical attention networks for document classification. In *the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2016)*, pages 1480–1489, 2016.

Rongchao Yin, Quan Wang, Peng Li, Rui Li, and Bin Wang. Multi-granularity chinese word embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

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