Subcharacter Embeddings' Preference on Neural Networks

Yuanzhi Ke Masafumi Hagiwara
Keio University
{enshika8811.a6, hagiwara}@keio.jp

1 Introduction

A Chinese character can be decomposed into ideographic radicals ("部首"). The radical-enriched word and character embeddings outperform original word and character embeddings in various tasks in Chinese [7, 13, 14, 11] and in Japanese [4]. The subcharacter-based language models achieved much less perplexity than the character-based language model [9]. The subcharacter-based text classification models are more robust for the unseen characters and save the computational cost with the smaller vocabulary [5].

However, the subcharacter embeddings failed to outperform the word and character embeddings in the classification tasks [5, 4].

The inconsistent performance motivated our study. Because the previous works are under the different datasets and the different neural networks, we collected a pair of topic classification dataset and a pair of sentiment classification dataset in Chinese and Japanese and perform experiments with the different neural networks and embeddings (Section 3).

We find that the performance of the subcharacter embeddings highly depends on the model architecture. There is a significant gap between the long short-term memory neural network (LSTM) [2] and the other models for the subcharacter embeddings. The subcharacter embeddings achieve nearly state-of-the-art performance with LSTM, while they cooperate badly with the other models including fastText [3], the multilayer perceptron (MLP), the convolutional neural network (CNN), the quasi-recurrent neural network (QRNN) [1] and Transformer [12]. However, at the same time, the results for the word embeddings are much more close. We also find that the gaps between the models are larger for the Japanese datasets than the Chinese datasets (Section 5).

2 Models

FastText [3] contains an embedding layer and a softmax regression head, developed from the continuous bag-of-words [8]. Embeddings of words, characters and subcharacters are summed together as the input of the regression head. It is simple but reported efficient for word embeddings.

The multilayer perceptron (MLP) is a stack of fully connected layers. In the experiment, we employed a three-layer MLP, which contains the embedding layer, the fully connected layer and the softmax regression head.

The convolutional neural network (CNN) employs the convolutional filters and pooling layers to extract the features with relatively less parameters. In the experiment, we used a deep CNN of 10 layers whose filter width is 3 and pooling width is 2.

The long short-term memory network (LSTM) is the recurrent neural networks whose recurrent unit has an input gate, a forget gate and an output gate. The gates controls the hidden state to avoid vanishing gradient and learn long-range dependencies.

The quasi-recurrent neural network (QRNN) [1] is a kind of recurrent neural network which replace the matrix production in the recurrent units with masked convolution [10] for parallel computation and wide windows. This architecture allows the units to explicitly get information from larger windows. In the experiment, we follow the original implementation [1]. The filter width is set to 2.

[1] https://github.com/salesforce/pytorch-qrnn

For LSTM and QRNN, we implemented four layer models in the experiments comprised of the embedding layer, the recurrent or quasi-recurrent layer, the fully connected layer and the softmax regression head. We input the embeddings from the embedding layer to the recurrent layer, and then feed the hidden state of the last unit to the softmax regression head.

Transformer [12] employs masked multi-head self-attention to encode the context. The multi-head self-attention and the position embedding make the feed-forward neural network able to compute the contextual dependency and the positional information without recurrence and convolution. In the experiment, the input text embeddings and positional embeddings are encoded by six normalized eight-head masked self-attention layers in the experiments. The encoder's output is fed to a classification head to predict the label.

3 Datasets

We tested the models by two tasks: the topic classification task and the sentiment classification task. We prepare a Chinese dataset and a Japanese dataset for each of the tasks.

For the Chinese tasks, we use the "Chinanews" dataset and the "JD" dataset, as the topic classification dataset and the sentiment classification dataset, respectively. Both of the datasets are publicly shared by Zhang and LeCun [15].

For the Japanese topic classification task, we randomly collected samples of four topics from the Rakuten Data Release [2]: the product reviews from a online shopping website, Rakuten Ichiba [3]; the hotel reviews from a travel guide website, Rakuten Travel [4]; the golf course reviews from a golf information website, Rakuten GORA [5]; and recipes from a cooking website, Rakuten Recipe [6]. Each topic contains 200,000 training samples and 16,000 testing samples. There are totally 864,000 samples. It is marked as 'Rakuten T.'.

For the Japanese sentiment classification task, we firstly labeled the reviews from Rakuten Ichiba in the Rakuten Data Release in the same way as the 'JD' dataset. Then we collected 2,180,000 positive reviews and 2,180,000 negative reviews randomly. For each category, 2,000,000 samples are for training, and the other 180,000 samples are for testing. We marked this dataset as 'Rakuten I.'.

4 Setup

For the subcharacter-based representations for the Chinese characters, we replace them with their radicals according to the character structure information database, "CHISE" [7]. For the other characters, we do not decompose them. The subcharacter-based models use the radical embeddings of the Chinese characters and the character embeddings of the other characters in the experiments.

For each model above in the experiments, the dimension of the embedding layer and the other hidden layers is set to 256. In each epoch, the training set of each dataset is further randomly divided into a smaller training set (90%) and a validation set (10%). If the loss of the validation set does not decrease during the recent 5 epochs, the training ends. The parameters that achieved the best validation loss are saved. We use the ADAM algorithm [6] to update the parameters.

5 Results and Discussion

We provide the F1 scores, together with their variance for each of the embedding type in Table 1. The character and the subcharacter embeddings show significant preference on the neural network types. The improvement for the subcharacter embeddings can only be observed from the CNN and the LSTM, in comparison with fastText. Moreover, the improvement from the LSTM for the character and subcharacter embeddings is obviously much higher than the word emeddings. Furthermore, in comparison between the variances of the models (Column "$\mathcal{V}$" in Table 1), we find that the performance of the subcharacter embeddings depends on the model type more than the character embeddings.

[2] https://rit.rakuten.co.jp/data_release/
[3] https://www.rakuten.co.jp/
[4] https://travel.rakuten.co.jp/
[5] https://gora.golf.rakuten.co.jp/
[6] https://recipe.rakuten.co.jp/
[7] http://www.chise.org/ids/

Table 1. The F1 scores of all the combinations of the input embeddings and the neural networks (in percentage). The number in the brackets is the absolute change in percentage, in comparison with the word embedding-based fastText model. Column "$\mathcal{V}$" provides the variance of the results by the different models for each embedding type.

		fastText	MLP	CNN	LSTM	QRNN	TF.	$\mathcal{V}$
Chinanews	Word	85.65	85.88	86.92	88.84	84.82	89.11	3.11
	Char.	84.70	83.76	87.99	90.48	84.84	74.61	**29.28**
	Subchar.	81.05	79.59	87.09	90.35	80.45	83.19	18.08
JD	Word	89.89	90.03	90.59	91.38	89.96	90.14	0.07
	Char.	87.85	87.84	89.74	91.48	88.76	87.88	0.55
	Subchar.	85.94	86.24	89.68	91.20	87.63	83.22	**3.42**
Rakuten T.	Word	98.81	98.91	98.64	99.19	98.48	99.08	0.33
	Char.	97.90	97.85	98.23	99.20	98.00	96.90	2.13
	Subchar.	97.27	97.16	97.63	99.18	97.41	93.58	**8.12**
Rakuten I.	Word	87.22	87.23	90.67	91.79	88.04	87.35	4.01
	Char.	85.06	84.82	91.23	93.50	88.64	82.83	14.43
	Subchar.	83.74	84.03	90.61	93.35	84.48	78.35	**19.72**

Table 2. The error overlap between different neural networks over the **Chinese** tasks for the subcharacter embeddings (in percentage). 'TF.' refers to the Transformer Model here.

	fastText	MLP	CNN	LSTM	QRNN
TF.	42.74	43.04	33.40	33.46	44.21
QRNN	58.13	58.58	41.98	41.41	/
LSTM	38.95	39.01	50.83	/	/
CNN	37.97	38.53	/	/	/
MLP	68.02	/	/	/	/

Table 3. The error overlap between different neural networks over the **Japanese** tasks for the subcharacter embeddings (in percentage). 'TF.' refers to the Transformer Model here.

	fastText	MLP	CNN	LSTM	QRNN
TF.	36.13	37.78	20.11	15.72	33.78
QRNN	51.85	55.69	35.31	32.71	/
LSTM	31.38	31.50	45.37	/	/
CNN	32.91	33.32	/	/	/
MLP	71.09	/	/	/	/

To further clarify the gaps between the models, we provide the overlap of the wrongly predicted samples for the Chinese datasets and the Japanese datasets in Table 2, and Table 3, respectively. The mistakes made by fastText, MLP and QRNN are relatively highly overlapped, while LSTM, CNN and Transformer are more independent. The gaps between the models are larger for the Japanese datasets than the Chinese datasets.

6 Conclusion

The experimental results have shown the subcharacter embeddings' strong preference on the LSTM. The subcharacter embedding-based LSTM outperforms the word and character embedding-based fastText model, MLP models, CNN models, QRNN models and Transformer models, outperforms the word embedding-based LSTM models for one Japanese dataset and one Chinese dataset among the four datasets, but slightly underperform the character embedding-based LSTM. The gap between the models are larger for the Japanese datasets. It explains why the fully connected feed-forward models such as Skip-gram, CBOW and fastText failed to get

the expected results for the Japanese datasets, while they achieved relatively good results for the Chinese datasets in the previous works.

References

[1] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-recurrent neural networks. In *Proceedings of the 5rd International Conference on Learning Representations*, 2017.

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, Vol. 9, No. 8, pp. 1735–1780, November 1997.

[3] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 427–431. Association for Computational Linguistics, April 2017.

[4] Marzena Karpinska, Bofang Li, Anna Rogers, and Aleksandr Drozd. Subcharacter information in japanese embeddings: When is it worth it? In *Proceedings of the Workshop on the Relevance of Linguistic Structure in Neural Architectures for NLP*, pp. 28–37, 2018.

[5] Yuanzhi Ke and Masafumi Hagiwara. Radical-level ideograph encoder for rnn-based sentiment analysis of chinese and japanese. In *Proceedings of the 9th Asian Conference on Machine Learning*, pp. 561–573, 2017.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

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

[8] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

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

[10] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[11] Yan Shao, Jörg Tiedemann, Christian Hardmeier, and Joakim Nivre. Character-based joint segmentation and pos tagging for chinese using bidirectional rnn-crf. In *Proceedings of the 8th International Joint Conference on Natural Language Processing*, pp. 173–183. Asian Federation of Natural Language Processing, 2017.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. Curran Associates, Inc., 2017.

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

[14] Jinxing Yu, Xun Jian, Hao Xin, and Yangqiu Song. Joint embeddings of chinese words, characters, and fine-grained subcharacter components. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 286–291, 2017.

[15] Xiang Zhang and Yann LeCun. Which encoding is the best for text classification in chinese, english, japanese and korean? *CoRR*, Vol. abs/1708.02657, , 2017.