Radical Embedding: Delving Deeper to Chinese Radicals

Xinlei Shi, Junjie Zhai, Xudong Yang, Zehua Xie, Chao Liu
Sogou Technology Inc., Beijing, China
{shixinlei, zhaijunjie, yangxudong, xiezehua, liuchao}@sogou-inc.com

Abstract

Languages using Chinese characters are mostly processed at word level. Inspired by recent success of deep learning, we delve deeper to character and radical levels for Chinese language processing. We propose a new deep learning technique, called "radical embedding", with justifications based on Chinese linguistics, and validate its feasibility and utility through a set of three experiments: two in-house standard experiments on short-text categorization (STC) and Chinese word segmentation (CWS), and one in-field experiment on search ranking. We show that radical embedding achieves comparable, and sometimes even better, results than competing methods.

1 Introduction

Chinese is one of the oldest written languages in the world, but it does not attract much attention in top NLP research forums, probably because of its peculiarities and drastic differences from English. There are sentences, words, characters in Chinese, as illustrated in Figure 1. The top row is a Chinese sentence, whose English translation is at the bottom. In between is the pronunciation of the sentence in Chinese, called PinYin, which is a form of Romanian phonetic representation of Chinese, similar to the International Phonetic Alphabet (IPA) for English. Each squared symbol is a distinct Chinese character, and there are no separators between characters calls for Chinese Word Segmentation (CWS) techniques to group adjacent characters into words.

In most current applications (*e.g.*, categorization and recommendation *etc.*), Chinese is represented at the word level. Inspired by recent success of delving deep (Szegedy et al., 2014; Zhang and LeCun, 2015; Collobert et al., 2011), an interesting question arises then: *can we delve deeper than word level representation for better Chinese language processing? If the answer is yes, how deep can it be done for fun and for profit?*

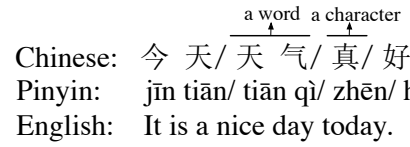

Figure 1: Illustration of Chinese Language

Intuitively, the answer should be positive. Nevertheless, each Chinese character is semantically meaningful, thanks to its pictographic root from ancient Chinese as depicted in Figure 2. We could delve deeper by decomposing each character into character radicals.

The right part of Figure 2 illustrates the decomposition. This Chinese character (meaning "morning") is decomposed into 4 radicals that consists of 12 strokes in total. In Chinese linguistics, each Chinese character can be decomposed into no more than four radicals based on a set of preset rules[1]. As depicted by the pictograms in the right part of Figure 2, the 1st radical (and the 3rd that happens to be the same) means "grass", and the 2nd and the 4th mean the "sun" and the "moon", respectively. These four radicals altogether convey the meaning that "the moment when sun arises from the grass while the moon wanes away", which is exactly "morning". On the other hand, it is hard to decipher the semantics of strokes, and radicals are the minimum semantic unit for Chinese. Building deep mod-

[1] http://en.wikipedia.org/wiki/Wubi_method

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers), pages 594–598, Beijing, China, July 26-31, 2015.

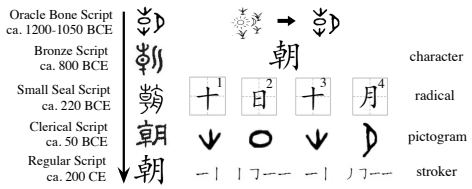

Figure 2: Decomposition of Chinese Character

els from radicals could lead to interesting results.

In sum, this paper makes the following three-fold contributions: (1) we propose a new deep learning technique, called "radical embedding", for Chinese language processing with proper justifications based on Chinese linguistics; (2) we validate the feasibility and utility of radical embedding through a set of three experiments, which include not only two in-house standard experiments on short-text categorization (STC) and Chinese word segmentation (CWS), but an in-field experiment on search ranking as well; (3) this initial success of radical embedding could shed some light on new approaches to better language processing for Chinese and other languages alike.

The rest of this paper is organized as follows. Section 2 presents the radical embedding technique and the accompanying deep neural network components, which are combined and stacked to solve three application problems. Section 3 elaborates on the three applications and reports on the experiment results. With related work briefly discussed in Section 4, Section 5 concludes this study. For clarity, we limit the study to Simplified Chinese in this paper.

2 Deep Networks with Radical Embeddings

This section presents the radical embedding technique, and the accompanying deep neural network components. These components are combined to solve the three applications in Section 3.

Word embedding is a popular technique in NLP (Collobert et al., 2011). It maps words to vectors of real numbers in a relatively low dimensional space. It is shown that the proximity in this numeric space actually embodies algebraic semantic relationship, such as "Queen − Woman + Man ≈ King" (Mikolov et al., 2013). As demonstrated in previous work, this numeric representation of words has led to big improvements in many NLP tasks such as machine translation (Sutskever et al., 2014), question answering (Iyyer et al., 2014) and document ranking (Shen et al., 2014).

	input	output
Convolution	$\boldsymbol{f} \in \mathbb{R}^m$ $\boldsymbol{k} \in \mathbb{R}^n$	$\boldsymbol{y} \in \mathbb{R}^{m+n-1}$ $y_i = \sum_{s=i}^{i+n-1} f_s \cdot k_{s-i}$ $0 \leq i \leq m-n+1$
Max-pooling	$\boldsymbol{x} \in \mathbb{R}^d$	$y = \max(\boldsymbol{x}) \in \mathbb{R}$
Lookup Table	$\boldsymbol{M} \in \mathbb{R}^{d \times \lvert D \rvert}$ $\boldsymbol{I}_i \in \mathbb{R}^{\lvert D \rvert \times 1}$	$\boldsymbol{v}_i = \boldsymbol{M}\boldsymbol{I}_i \in \mathbb{R}^d$
Tanh	$\boldsymbol{x} \in \mathbb{R}^d$	$\boldsymbol{y} \in \mathbb{R}^d$ $y_i = \frac{e^{x_i} - e^{-x_i}}{e^{x_i} + e^{-x_i}}$ $0 \leq i \leq d-1$
Linear	$\boldsymbol{x} \in \mathbb{R}^d$	$\boldsymbol{y} = \boldsymbol{x} \in \mathbb{R}^d$
ReLU	$\boldsymbol{x} \in \mathbb{R}^d$	$\boldsymbol{y} \in \mathbb{R}^d$ $y_i = 0 \ if \ x_i \leq 0$ $y_i = x_i \ if \ x_i > 0$ $0 \leq i \leq d-1$
Softmax	$\boldsymbol{x} \in \mathbb{R}^d$	$\boldsymbol{y} \in \mathbb{R}^d$ $y_i = \frac{e^{x_i}}{\sum_{j=1}^{d} e^{x_j}}$ $0 \leq i \leq d-1$
Concatenate	$\boldsymbol{x}^i \in \mathbb{R}^d$ $0 \leq i \leq n-1$	$y = (\boldsymbol{x}^0, \boldsymbol{x}^1, ..., \boldsymbol{x}^{n-1})$ $\in \mathbb{R}^{d \times n}$
D: radical vocabulary $\boldsymbol{M}$: a matrix containing $\lvert D \rvert$ columns, each column is a d-dimensional vector represent radical in D. $\boldsymbol{I}_i$: a one hot vector stands for the ith radical in vocabulary		

Table 1: Neural Network Components

Radical embedding is similar to word embedding except that the embedding is at radical level. There are two ways of embedding: CBOW and skip-gram (Mikolov et al., 2013). We here use CBOW for radical embedding because the two methods exhibit few differences, and CBOW is slightly faster in experiments. Specifically, a sequence of Chinese characters is decomposed into a sequence of radicals, to which CBOW is applied. We use the word2vec package (Mikolov et al., 2013) to train radical vectors, and then initialize the lookup table with these radical vectors.

We list the network components in Table 1, which are combined and stacked in Figure 3 to solve different problems in Section 3. Each component is a function, the input column of Table 1 demonstrates input parameters and their dimensions of these functions, the output column shows the formulas and outputs.

3 Applications and Experiments

In this section, we explain how to stack the components in Table 1 to solve three problems: short-text categorization, Chinese word segmentation and search ranking, respectively.

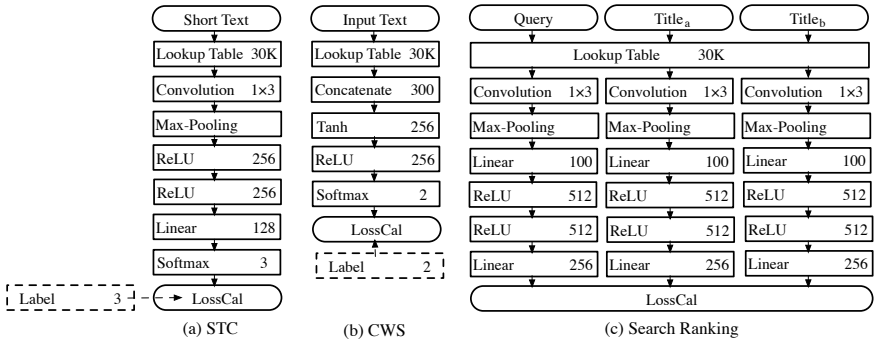

Figure 3: Application Models using Radical Embedding

Accuracy(%)	**Competing Methods**		**Deep Neural Networks with Embedding**				
	LR	SVM	wrd	chr	rdc	wrd+rdc	chr+rdc
Finance	93.52	94.06	94.89	**95.85**	94.75	95.70	95.74
Sports	92.40	92.83	95.10	95.01	92.24	95.87	**95.91**
Entertainment	91.72	92.24	94.32	94.77	93.21	**95.11**	94.78
Average	92.55	93.04	94.77	95.21	93.40	**95.56**	95.46

Table 2: Short Text Categorization Results

3.1 Short-Text Categorization

Figure 3(a) presents the network structure of the model for short-text categorization, where the width of each layer is marked out as well. From the top down, a piece of short text, *e.g.*, the title of a URL, is fed into the network, which goes through radical decomposition, table-lookup (*i.e.*, locating the embedding vector corresponding to each radical), convolution, max pooling, two ReLU layers and one fully connected layer, all the way to the final softmax layer, where the loss is calculated against the given label. The standard back-propagation algorithm is used to fine tune all the parameters.

The experiment uses the top-3 categories of the SogouCA and SogouCS news corpus (Wang et al., 2008). 100,000 samples of each category are randomly selected for training and 10,000 for testing. Hyper-parameters for SVM and LR are selected through cross-validation. Table 2 presents the accuracy of different methods, where "wrd", "chr", and "rdc" denote word, character, and radical embedding, respectively. As can be seen, embedding methods outperform competing LR and SVM algorithms uniformly, and the fusion of radicals with words and characters improves both.

3.2 Chinese Word Segmentation

Figure 3(b) presents the CWS network architecture. It uses softmax as well because it essentially classifies whether each character should be a segmentation boundary. The input is firstly decomposed into a radical sequence, on which a sliding window of size 3 is applied to extract features, which are pipelined to downstream levels of the network.

We evaluate the performance using two standard datasets: PKU and MSR, as provided by (Emerson, 2005). The PKU dataset contains 1.1M training words and 104K test words, and the MSR dataset contains 2.37M training words and 107K test words. We use the first 90% sentences for training and the rest 10% sentences for testing. We compare radical embedding with the CRF method[2], FNLM (Mansur et al., 2013) and PSA (Zheng et al., 2013), and present the results in Table 3. Note that no dictionary is used in any of these algorithms.

We see that the radical embedding (RdE) method, as the first attempt to segment words at radical level, actually achieves very competitive results. It outperforms both CRF and FNLM on both datasets, and is comparable with PSA.

[2] `http://crfpp.googlecode.com/svn/trunk/doc/index.html?source=navbar`

Data	Approach	Precision	Recall	F1
PKU	CRF	88.1	86.2	87.1
	FNLM	87.1	87.9	87.5
	PSA	**92.8**	92.0	**92.4**
	RdE	92.6	**92.1**	92.3
MSR	CRF	89.3	87.5	88.4
	FNLM	92.3	92.2	92.2
	PSA	92.9	**93.6**	93.3
	RdE	**93.4**	93.3	**93.3**

Table 3: CWS Result Comparison

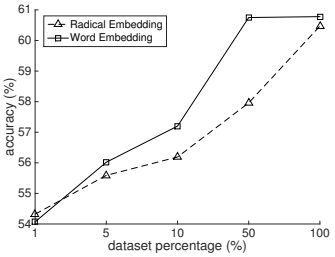

Figure 4: Search Ranking Results

3.3 Web Search Ranking

Finally, we report on an in-field experiment with Web search ranking. Web search leverages many kinds of ranking signals, an important one of which is the preference signals extracted from click-through logs. Given a set of triplets {query, title_a, title_b} discovered from click logs, where the URL title_a is preferred to title_b for the query. The goal of learning is to produce a matching model between query and title that maximally agrees with the preference triplets. This learnt matching model is combined with other signals, *e.g.*, PageRank, BM25F, *etc.* in the general ranking. The deep network model for this task is depicted in Figure 3(c), where each triplet goes through seven layers to compute the loss using Equation (1), where $\boldsymbol{q_i}$, $\boldsymbol{a_i}$, $\boldsymbol{b_i}$ are the output vectors for the query and two titles right before computing the loss. The calculated loss is then back propagated to fine tune all the parameters.

$$\sum_{i=1}^{m} \log\left(1 + \exp\left(-c * \left(\frac{\boldsymbol{q_i}^T \boldsymbol{a_i}}{|\boldsymbol{q_i}||\boldsymbol{a_i}|} - \frac{\boldsymbol{q_i}^T \boldsymbol{b_i}}{|\boldsymbol{q_i}||\boldsymbol{b_i}|}\right)\right)\right) \quad (1)$$

The evaluation is carried out on a proprietary data set provided by a leading Chinese search engine company. It contains 95,640,311 triplets, which involve 14,919,928 distinct queries and 65,125,732 distinct titles. 95,502,506 triplets are used for training, with the rest 137,805 triplets as testing. It is worth noting that the testing triplets are hard cases, mostly involving long queries and short title texts.

Figure 4 presents the results, where we vary the amount of training data to see how the performance varies. The x-axis lists the percentage of training dataset used, and 100% means using the entire training dataset, and the y-axis is the accuracy of the predicted preferences. We see that word embedding is overall superior to radical embedding, but it is interesting to see that word embedding saturates using half of the data, while ranking with radical embedding catches up using the entire dataset, getting very close in accuracy (60.78% vs. 60.47%). Because no more data is available beyond the 95,640,311 triplets, unfortunately we cannot tell if radical embedding would eventually surpass word embedding with more data.

4 Related Work

This paper presents the first piece of work on embedding radicals for fun and for profit, and we are mostly inspired by fellow researchers delving deeper in various domains (Zheng et al., 2013; Zhang and LeCun, 2015; Collobert et al., 2011; Kim, 2014; Johnson and Zhang, 2014; dos Santos and Gatti, 2014). For example, Huang *et al.*'s work (Huang et al., 2013) on DSSM uses letter trigram as the basic representation, which somehow resembles radicals. Zhang and Yann's recent work (Zhang and LeCun, 2015) represents Chinese at PinYin level, thus taking Chinese as a western language. Although working at PinYin level might be a viable approach, using radicals should be more reasonable from a linguistic point of view. Nevertheless, PinYin only represents the pronunciation, which is arguably further away from semantics than radicals.

5 Conclusion

This study presents the first piece of evidence on the feasibility and utility of radical embedding for Chinese language processing. It is inspired by recent success of delving deep in various domains, and roots on the rationale that radicals, as the minimum semantic unit, could be appropriate for deep learning. We demonstrate the utility of radical embedding through

two standard in-house and one in-field experiments. While some promising results are obtained, there are still many problems to be explored further, *e.g.*, how to leverage the layout code in radical decomposition that is currently neglected to improve performance. An even more exciting topic could be to train radical, character and word embedding in a unified hierarchical model as they are naturally hierarchical. In sum, we hope this preliminary work could shed some light on new approaches to Chinese language processing and other languages alike.

References

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

Cícero Nogueira dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers*, pages 69–78.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*, volume 133.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 633–644.

Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. *CoRR*, abs/1412.1058.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

Mairgup Mansur, Wenzhe Pei, and Baobao Chang. 2013. Feature-based neural language model and chinese word segmentation. In *Sixth International Joint Conference on Natural Language Processing, 2013, Nagoya, Japan, October 14-18, 2013*, pages 1271–1277.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Gregoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2014. Going deeper with convolutions. *CoRR*, abs/1409.4842.

Canhui Wang, Min Zhang, Shaoping Ma, and Liyun Ru. 2008. Automatic online news issue construction in web environment. In *Proceedings of the 17th International Conference on World Wide Web*, pages 457–466.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *CoRR*, abs/1502.01710.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 647–657.