Shuffling Character: A Self-supervised Method Enhancing Representation Of Chinese

Chen Xingyuan

uvork@hotmail.com

2020/12/23

Abstract

Most studies on Natural Language Processing of Chinese overlook the difference between this analytical language and other synthetic languages like English. Common deep learning Natural Language Processing models emerged in recent years, like BERT and its derivations, usually treat segmentation of Chinese at a character-wise level. This may cause an issue on some downstream tasks, specifically, Named Entity Recognition, which is sensitive to word segmentation. However, Chinese requires more utilization of compound other than derivation on word formation, and thus the order of characters within a word usually conveys more information. On the ground of this feature, a language model emphasizing on describing the character order of words in Chinese is needed. As found in some linguistics studies, a shuffled order of Chinese characters within a compound word has limited disturbance on understanding a sentence, which is attributed to reader processing word at both word-wise and character-wise levels, followed by a restoration of appropriate character order. Inspired by this idea, we propose a self-supervised method based on BERT, by shuffling characters within a whole word segment and labelling each character with its correct position in a word. A Query-Key-Value attention mechanism is introduced for a sequence tagging task aiming at finding the correct position of each character, which is equivalent to the intuition of restoring the original Chinese word like human readers.

Introduction

NLP, known as Natural Language Processing, had great breakthroughs in recent years after the proposal of BERT [1], a transformer-based pretrained language model, which significantly improved performance of almost all major tasks in this aspect. In Chinese, the original version of BERT tokenizes a whole word into characters, neglecting the traditional segmentation at a word-wise level. BERT-wwm for Chinese [2] reconsidered this issue by creating a mask over the whole word after segmenting, which accorded more with the nature of Chinese and achieved a better performance in downstream NLP tasks. However, differing from Recurrent Neural Network, BERT uses a pre-defined positional encoding to describe the absolute position of tokens, which weakens the capability to capture the sequential relationship. In Chinese, this drawback is further exposed, as positional features exist not only among words, but also among characters wrapped by a single word, hence when dealing with sequence tagging task like NER, recognition of entity boundaries might be unclear. To relieve the mentioned shortcoming, it is needed to help the language model learn the position of characters within a word.

It was found in some linguistics studies that shuffled order of Chinese characters has limited disturbance on understanding a sentence, while a similar conclusion had been drawn on English with shuffled order of letters. However, differing from English being a synthetic language, Chinese, an analytic language, requires more utilization of compound other than derivation on word formation, whose single character usually convey its own meanings opposing to letter carrying less information in English. As discussed in [3], when reading a Chinese sentence with reverse order of characters within words, the reader first processes the whole word inputted, and secondly separates the word into characters, followed by combining which into words with both correct and reverse character order. The former procedure could be regarded as a shortcut for comprehending, while the latter consists of some exogenous understanding on the whole sentence so that the correct character order of word could be restored.

Motivated by [3], we designed a self-supervised task based on BERT to model the position of characters within words into pretrained embedding. Our model introduced two similar structures of KQV attention concerning *characters with the original order*, ***characters with shuffled order***, *original word,* and ***shuffled word*** (with the same boundary as the original one), and a residual connection to illustrate the brain activity of initially processing the whole inputted word when reading a word with shuffled characters.

Related Work

Natural Language Processing has been regarded as one of the most challenging aspects among all applications of artificial intelligence. Languages are purely made by humans,

reflecting human knowledge on describing the world, and thus it has been difficult to make computers understand human language.

Distributional hypothesis was proposed in 1954 [4], indicating that words appearing in similar or same context tend to have similar meanings, and has been the foundation of statistical semantics ever since. Word2Vec proposed in 2013 [5] well received the distributional hypothesis, representing word segment by means of dense vector opposing to sparse one-hot vector. Although a static embedding generated by Word2Vec could not tackle with words having multiple meanings, it was still a great leap for NLP, as it proved to be successful in many cases with a relatively simple and quick yet intuitive approach. With the rapid development of computation hardware and neural network, recurrent neural network and its derivations like long-short term memory [6], gated recurrent units [7], as well as convolutional neural network have been widely utilized in the aspect of NLP. As word, sentence and paragraph are sequential, using RNN and its derivations are intuitive and straight forward, while CNN could also well illustrate the N-gram model of natural language. However, both kinds of neural networks face some issues respectively: for RNNs, sequential computation requires significant time costs, being the major constraint; CNNs' capability of capturing long term features when dealing with long sentences is relatively weak. A transformer-based pretrained language model BERT [1] was proposed in 2017. BERT utilizes multiheaded KQV self-attention mechanism, with two self-supervised tasks, *Masked Language Model* and *Next Sentence Prediction*, and thus long-distance features could be extracted while enabling parallelization.

A little trick of shuffling the letters of words emerged on the Internet 20 years ago, and readers may find that they could still read the whole sentence. A similar phenomenon exists in the case of Chinese when shuffling the characters of words in a sentence. There were academic studies on this phenomenon [3]. When reading a sentence with words wrapping reverse characters, the reader first accepts the whole word as input. If the word accords with orthography, the reader would simply take the inputted word as a correct one and move on reading, otherwise, the reader would separate the word into characters and rearrange them into all possible permutations. To the best of our knowledge, there has been no computational interpretation of this phenomena, hence in the following sections, we would describe the corresponding language model and preliminary result of the practice.

A Self-supervised Method

Shuffling characters

In order to design a self-supervised method, it is required to label the original text with only endogenous information within the text itself. As we are looking for ways to illustrate the process when reading words with shuffled characters, it is feasible to first cut the sentence into word segments, noted as W_i^O, where i indicating the index of the sentences, and for each i there are τ_i^O word segments. We then further tokenize

W_i^O into character, noted as C_{ij}^O, where j referring to the position of the word segment in the ith sentence. For each j there are t_{ij}^O characters, and a sum over t_{ij}^O with respect to j for i is as $\sum_1^{\tau_i^O} t_{ij}^O = T_i^O$, hence T_i^O indicating the number of characters in the ith sentence. On the ground of the above segmentation on the original sentences at both word-wise and character-wise level, we label each character with its position of the wrapping word. An instance is as follows:

Char	患	者	8	月	无	明	显	诱	因
Labl	I-0	I-1	I-0	I-1	O	I-0	I-1	I-0	I-1

We use a labelling method similar to sequence tagging of NER task. I is used for a prefix of a character which wrapped by a word with length larger than 1, while the suffix integer starting from 0 indicating the character's position of its word segment. Characters which wrapped by a word with length equals to 1 are all labelled as O.

After labelling the original sentences, we shuffle the characters within word segments so that the original boundaries of word segments could be saved. An instance after shuffling is as follows:

Char	者	患	8	月	无	显	明	诱	因
Labl	I-1	I-0	I-0	I-1	O	I-1	I-0	I-0	I-1

It is noticed that not all word segments have shuffled characters. The fact is that for word segments with different length l_{ij}, there are $l_{ij}!$ permutations. With whose length equals to 2 in the mentioned instance, there is a chance of 0.5 to make the word segment remain the same. We simply note the ith processed sentence having a list of word segments W_i^S with a length of τ_i^S, and for the jth word in W_i^S there are t_{ij}^S characters, noted as C_{ij}^S whose a sum over j for i is T_i^S.

	Original Order	Shuffled Order
Word-wise	W_i^O (τ_i^O words in sentence)	W_i^S (τ_i^S words in sentence)
Character-wise	C_{ij}^O (T_i^O characters in sentence	C_{ij}^S (T_i^S characters in sentence

Following relationship hold: $\tau_i^O = \tau_i^S \leq T_i^O = T_i^S$.

It is also worth noting that a policy can be chosen in different ways, deciding the number of word segments whose wrapped characters are required to be shuffled. Reinforcement learning may also help when it comes to selecting word segments for processing.

Model Architecture

Embedding

After segmenting at both word-wise and character-wise levels, respective embeddings could be constructed. For word-wise level, we choose a fixed length τ for padding and truncating, while for character-wise level, a fixed length T is selected. In accord with the attention mechanism we use in the following step, we choose a fixed dimension of embedding for both word-wise and character-wise levels of segments, noted as D.

Differing from character-wise segment sharing the same mapping to tokens, here, we note that the original order word segments have a fixed mapping to tokens, while the shuffled order ones have a dynamically appended mapping since new word segments will possibly emerge during shuffling. An exhaustive method may be theoretically feasible, but due to the length of word segment l_{ij} varying, we might not have the computation capability to handle all permutations of long word segment.

The labels of the padding at both levels are required to be additionally noted, and appending to three mappings respectively: *original order word-wise*, *shuffled order word-wise*, and *character-wise.*

A summary table of the embedding shapes are as follows:

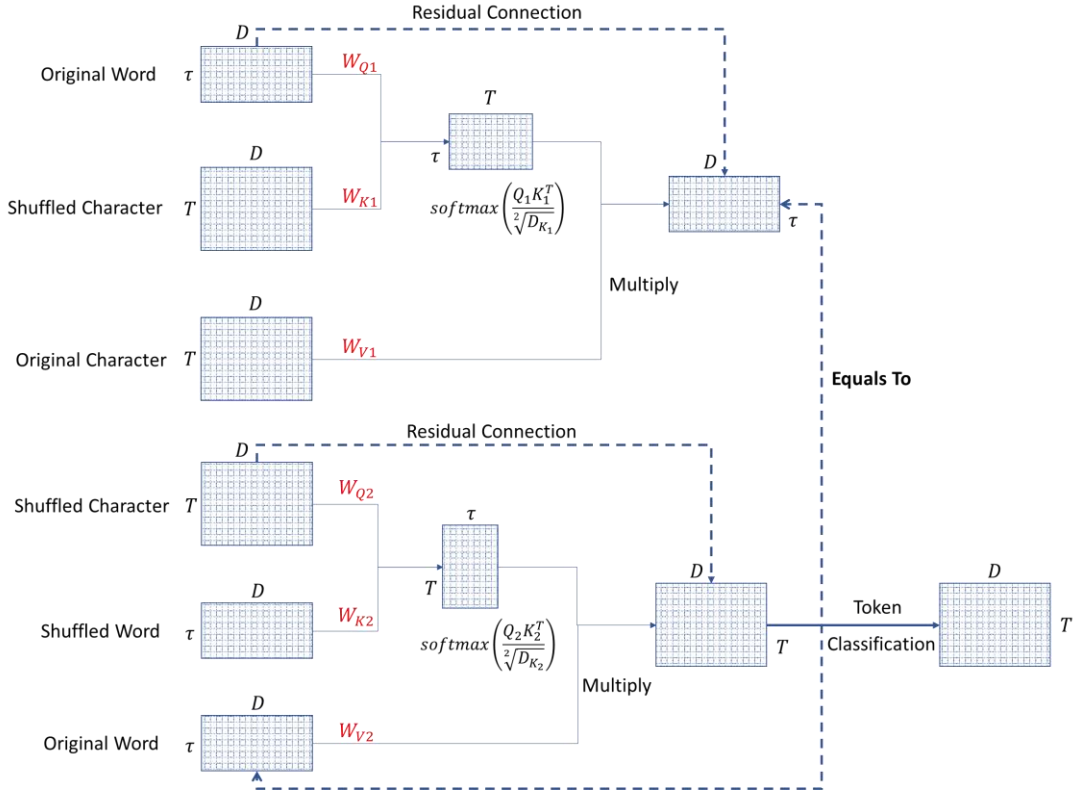
	Original Order	Shuffled Order
Word-wise	$\tau \times D$	$\tau \times D$
Character-wise	$T \times D$	$T \times D$

QKV Attention

Of the QKV attention, there exists three input matrices Query, Key and Value. Query is a matrix related to the task, extracting weights from Key matrix by computing attention. Here, we follow [1] using the scaled dot-product attention mechanism, which is as follows:

$$\text{attn}(\text{Q}, \text{K}, \text{V}) = softmax\left(\frac{QK^T}{\sqrt[2]{D_K}}\right) \times V$$

We design two attention structures to predict the label of each character:



The first structure simulates the process when reading a normal sentence with the correct order of characters within words. A reader will first process the whole word as input, which can be regarded as a *shortcut* and is illustrated by a residual connection from the original word to the weighted average result computed by scaled dot-product attention. This attention with respect to original word and shuffled character mimics a reader's subconscious. Although there is a shortcut mentioned above to help the reader activate the knowledge of the word verified by orthography, the reader may still separate words into characters without explicit cognition. An attention based on original word and shuffled character helps to find the inner connection between words and their characters, while a product concern original order of characters could pack the shuffled characters into words with correct order.

In the second structure, Value matrix is transmitted from the result of the first attention structure. This attention makes attempt on simulating reading sentence with words wrapping shuffled characters. A reader may find it hard to read the confusing words, and will explicitly cut the words into characters. Since it is unlikely for a reader to recognize the correct words until reading the full context sometimes, a scaled dot-product attention here is meant to extract the relationship among the context words and their shuffled characters. A product of the attention weights and the result of the first attention structure wraps the semantical information of original word into the shuffled character, helping characters finding their original positions within a word.

After two attention structure, we obtain a matrix with a shape of $T \times D$. This matrix contains features at word-wise levels, character-wise levels and their interactions. A

self-supervised task is hence constructed to classify each character, or token, into its label indicating whose position within a word.

An Implementation on Medical Named Entity Recognition

Self-supervised Task Implementation

One promising result of the previous self-supervised task is that it could better distinguish the Chinese word boundary, which is significant for Chinese NER task as there has been no clear boundary like space between Chinese words in a sentence. We made attempt on implementing a trial on medical NER task.

The dataset is formed by over a thousand electronic medical records and corresponding entity mentions concerning symptoms, medicines, medical examinations and organs.

After certain steps of text cleansing, we used Jieba to segment the sentences into words. A dictionary for mapping was hence constructed for mapping the word segments to integers. At character-wise level, we use the vocabulary in RoBERTa-wwm for further tokenizing.

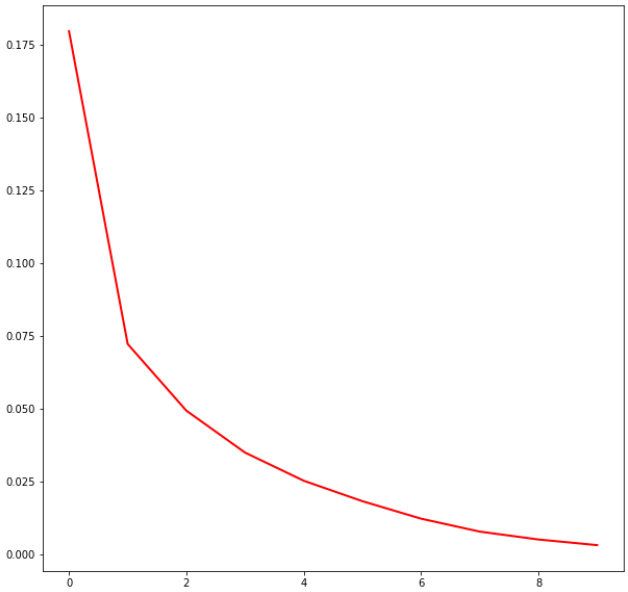
We pre-set the training epochs to be 10, and shuffled the characters after labelling with the positions in the wrapping word. It is worth noting here that the maximum length of the word segments is 11, which led to almost 40M possible permutations, so we chose to shuffle beforehand and record the result shuffled characters and the wrapping words for every epoch. Shuffled characters still remained in the same vocabulary as the normal order ones, while the wrapping words varied. And thus we took out all of the emerged word segments after shuffling and build a new vocabulary for them.

It is also worth noting that, we only shuffle over those word segments with length greater than 1 by a variable ratio. At the first epoch, the ratio was set to 0.6, which means that 60 percent of the word segments which had a length greater than 1 were processed. A minimum ratio was set to 0.1 at the last epoch, and the ratios at intermediate epochs decreased linearly.

After consideration on the length distribution, the maximum length of both original word segments and shuffled word segments were set to 256, while that of character segments were 512. Inheriting BERT, we set the embedding dimension to 768. We only masked padded tokens when computing attention other than backward masking, since we believed that posterior information of context was important for reading messy texts. Embedding layers of word segments were initialized randomly, while that of character segments were obtained by the hidden states of RoBERTa-wwm model.

The optimizer used was AdamW, with a warmup maximum learning rate 0.0001. Loss

value curve is as follows:



After training for 10 epochs, the model obtained the capability of correcting the order of shuffled characters. Here we saved only the trained RoBERTa-wwm model, although we believed that the embedding layer of original words should also be meaningful.

Downstream Task: Medical Named Entity Recognition

The downstream medical NER task was performed on the basis of the saved RoBERTa-wwm and Conditional Random Field model. AdamW was used as optimizers of both models, and warmup maximum learning rates were 1e-5 and 8e-5 respectively.

We trained both vanilla RoBERTa-wwm + CRF model and saved RoBERTa-wwm + CRF model over 20 epochs. Evaluation of results was made by metrics provided by *seqeval*. The summary table is as follows:

Model	Micro-			Macro-		
	Precision	Recall	F1	Precision	Recall	F1
Vanilla RoBERTa-wwm + CRF	0.795	0.792	0.794	0.779	0.770	0.774
Saved RoBERTa-wwm + CRF (ours)	0.806	0.826	0.816	0.789	0.812	0.801

Our model showed a lead over the combination of vanilla RoBERTa-wwm and CRF model. Considering we were not careful enough to adjust the training parameters, and the batch size was strictly limited due to the weak computation capability, this result could be significantly improved with larger batch size and serious adjustment of hyperparameters, and thus the results here are only for reference. We would like to take some time looking into details of the predicted outcomes.

Some instances are shown below comparing the outcomes of the vanilla model and ours:

	胃	癌	皮	革	胃	）	侵	犯	贲	门
Label	B-3	I-3	I-3	I-3	I-3	I-3	O	O	B-5	I-5
Ours	B-3	I-3	I-3	I-3	I-3	I-3	O	O	B-5	I-5
Vanilla	B-3	I-3	I-3	I-3	I-3	I-3	**I-3**	**I-3**	B-5	I-5

	贲	门	、	胃	底	胃	壁
Label	B-5	I-5	O	B-5	I-5	B-5	I-5
Ours	B-5	I-5	O	B-5	I-5	B-5	I-5
Vanilla	B-5	I-5	O	B-5	I-5	**B-5**	**O**

Our model showed better recognition on the boundary of words. We attributed this advantage to the self-supervised task making model to learn from the position of characters, and hence the performance could be improved. This feature is especially useful in tasks which are highly sensitive to the word boundary.

Conclusion and Future Work

In Natural Language Processing, self-supervised learning has been rather important, as it packs human's prior knowledge on language into the model, and help the model learn the representation of characters, words, sentences and paragraphs. A well-performed self-supervised method should be in accordance with intuition and human's custom of reading. BERT's masked language model is representative as it somehow simulates CLOZE test, which requires a reader's full understanding of the context.

We introduced this self-supervised method of Chinese character shuffling, which was inspired by a little trick emerged on the Internet showing that the order of characters may have limited disturbance on reading Chinese texts. Our method focuses on restoring the original order of characters within words by the simulation of the process when a real person reads a sentence with shuffled characters, and expects the model to learn the representation of words, including their boundaries, and characters.

Although the result of our method showed some advantages, there are still some problems remains. Future work would mainly focus on:

1. Verifying the mechanism by investigating the two attention structures, including the obtained attention weights;
2. The policy of selecting word segments to be shuffled, and whether reinforcement learning could help;
3. The possibility of utilizing the embedding of word segments other than just extracting BERT's character-wise weights and bias.

References

[1] Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

[2] Cui, Yiming, et al. "Revisiting Pre-Trained Models for Chinese Natural Language Processing." *arXiv preprint arXiv:2004.13922* (2020).

[3] 彭聃龄, et al. "汉语逆序词的加工—词素在词加工中的作用." *心理学报* 31.1 (1999): 36-46.

[4] Harris, Zellig S. "Distributional structure." *Word* 10.2-3 (1954): 146-162.

[5] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in neural information processing systems* 26 (2013): 3111-3119.

[6] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

[7] Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." *arXiv preprint arXiv:1406.1078* (2014).