Is Character Glyph Useless? Improving Neural Chinese Word Segmentation with Character Glyph Embedding

Zexue He* [1], Jingle Xu [1], Mairgup Mansur [2], and Baobao Chang [3]
[1] Computer Science and Technology, College of CIST, Beijing Normal University
[2] Sogou Technology Inc, Beijing, China
[3] Key Lab of Computational Linguistics, School of EECS, Peking University
{zexueh, jinglex}@mail.bnu.edu.cn
maerhufu@sogou-inc.com
chbb@pku.edu.cn

Abstract

There is rich information hidden in the *glyph*[1] of Chinese characters, which consist of many small picture-like components. However, there are few works aware of the importance of overall glyph information and even some draw negative conclusion on it. Based on the idea of utilizing the overall glyph information in Chinese word segmentation (CWS) task, we propose a model by introducing autoencoder before BiLSTM with CRF on our synthetic Chinese Character Image Datasets to generate character glyph embeddings. Our experimental results show that the model performs quite well without any extra external dictionaries, word features or resources on several standard datasets including Simplified Chinese and Traditional Chinese, whose glyph is more regular with less evolutionary simplifications. These verify the feasibility of Chinese character glyph for Chinese word segmentation, especially its impressive support in solving the out-of-vocabulary(OOV) words and its great help for Traditional Chinese Word Segmentation.

Introduction

Chinese characters can be seen as vivid pictures. The earliest Chinese ancestor painted graphic symbols on animal bones (called *Oracle Bones Script*), which is the origin of Chinese characters. Then the script evolves over time, keeping the semantic information implicated in its varying graphics. The following is the evolution of the Simplified Chinese character "鸟" (bird).

Figure 1: The evolution of Chinese character "鸟" (bird) over time.

To be specific, Chinese characters are composed of several small graphic components with semantic or pronounced function. Thus, these characters with the same components will have related semantics or pronunciation. Obviously, the more of the same component these characters have, the more similar they may look like. Characters "鸡" (chicken), "鸭" (duck), and "鸦" (crow) are very similar in graphics with the same component "鸟" (bird). Also, they are all birds.

Moreover, not only the components but also the spatial structure between them may reflect the meaning and the function of a character. For example, the *closed structure* always implies *surrounding* or *encircling*, such as "囚" (imprison), which gets its meaning by putting a "人" (person) into a "口" (a picture-like component for *block*). The similar case is "圃" (garden). The "甫" is a component responsible for pronunciation and the "口" also represents a closed block, which contributes to meaning "garden" finally (a closed piece of land with plants). However, the character "哺" (feed) has the same components and the different structure. Then, it means completely different with the "口" (mouth) on the left of the character.

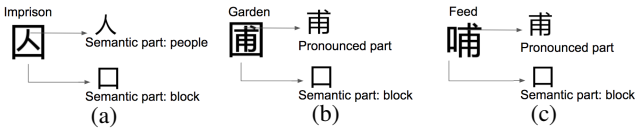


Figure 2: Structures of Chinese characters. (a) and (b) are of the same structure while (b) and (c) are different.

Instead of getting the component from retrieving or fragmenting, the overall information including not only the meaningful components but also their relationship in spacial structures should be taken into account. However, (Su and Lee 2017) once learned word representations from the average of all character glyph image vectors that appeared in one word, and showed that these representations were not better than those of the original embeddings obtained from word2vec[2]. Is glyph really useless? On the belief that the special meaning hidden in overall character's glyph could help Chinese language processing tasks and noticing that the characters with same components always appear in one word, such as "黝黑" (very dark), "鹧鸪" (partridge) and "霜露" (frost), we add character glyph representations into Chinese word segmentation (CWS) architecture and get im-

[1] *Glyph*: (typography, computing) A visual representation of a letter, character, or symbol, in a specific font and style.

[2] https://code.google.com/archive/p/word2vec/

proved performance on five standard CWS datasets, including Traditional Chinese corpus and Simplified Chinese corpus. Especially, we notice the impressive support of character glyph in solving out-of-vocabulary (OOV) words and the special usefulness in finding logical and semantic relationship of Traditional Chinese. In general, our experimental results verify the usefulness of character's glyph in CWS, which may be deeply hidden in its components' features.

Related Work

Chinese Word Segmentation Chinese word segmentation (CWS) is a fundamental task among the Chinese language processing tasks. (Xue 2003) proposed a character-based method that regarded CWS as a sequence tagging problem, which gained its great popularity. Then (Ng and Low 2004; Peng, Feng, and McCallum 2004) introduced the maximum entropy and linear-chain CRF as improvements. They also investigated Chinese part-of-speech (POS) tagging and CWS task based on word and character with one-at-a-time and all-at-once.

Recent year, the neural network has developed greatly, which became the dominant approach of CWS. (Zheng, Chen, and Xu 2013; Mansur, Pei, and Chang 2013) used the structure of (Collobert et al. 2011) to prove the feasibility of applying neural network method to Chinese word segmentation task. Following this work, quite a number of neural network methods sprung up on character-level (Pei, Ge, and Chang 2014; Chen et al. 2015a; Xu and Sun 2016) and word-level (Cai and Zhao 2016; Zhang, Zhang, and Fu 2016; Liu et al. 2016) .

Among those neural network architectures, recurrent neural network (RNN) performs outstandingly when dealing with sequential data, especially the Long Short-Term Memory Neural Networks (LSTM)(Hochreiter and Schmidhuber 1997). (Chen et al. 2015b; Yao and Huang 2016; Chen et al. 2017) utilized LSTM to capture the important information hidden in the words at long distance away off in context, which boosted the performance in CWS task.

Chinese Composition Structure learning Chinese character is composed of many meaningful radicals. A radical consists of one or more picture-like components. This unique structure of Chinese may be helpful for Chinese Natural Language Processing. From this view (Shi et al. 2015) once got the character and word embeddings from sentence radical sequence and validated its feasibility through CWS task. (Yu et al. 2017) used the word sub-character components sequence as the sentence representation of JWE model, which proved the semantic effectiveness of sub-character. However, cutting words into linear component sequence may lose their spatial relationship within the character. Compared with the whole character, the information just got from radical was incomplete. That might explain their passable results.

Being aware of the importance of the overall glyph information like us, (Su and Lee 2017) provided a new idea of getting overall structure features from Chinese character images. But they utilized the average of the image feature vectors of characters that appear in a word. Mixing images directly might confuse single character information conversely. (Liu et al. 2017) considered the whole character image features and found them work well on a novel text classification task, which is not commonly considered as a classical NLP task.

Method

With the purpose of revealing the utility of character glyph information and making sure the improvement only comes from character glyph itself instead of other tricks or mechanisms of complex network functions, we build a neural network model by adding autoencoder before bidirectional LSTM (shorten as BiLSTM in the following) with CRF to investigate CWS task, which is straightforward and unpretentious. The input of our model $\boldsymbol{x} = [x_0, x_1, \ldots, x_t, \ldots, x_n]$ is the sentence with length n which needs to be segmented. Then each character x_t in the sentence will be mapped to its corresponding image index id_t. The character image sequence $\boldsymbol{i} = [\boldsymbol{i_0}, \boldsymbol{i_1}, \ldots, \boldsymbol{i_t}, \ldots, \boldsymbol{i_n}]$ obtained from image index sequence $\boldsymbol{id} = [id_0, id_1, \ldots, id_t, \ldots, id_n]$ is fed into an autoencoder which is responsible for extracting the useful overall image features and generating character glyph representation $\boldsymbol{r_t}$ for each character at time t. In order to distinguish it against the conventional word embedding or character embedding in NLP field, we consider this representation $\boldsymbol{r_t}$ as the glyph embedding in our paper. The glyph embedding vector $\boldsymbol{r_t}$ of each character is concatenated with the typical character-embedding (introduced detailed in Experiments) $\boldsymbol{e_t}$ before being fed into the two BiLSTM layers for word segmentation. Then the next CRF layer learns to predict the segmentation tags $\boldsymbol{y} = [y_0, y_1, \ldots, y_t, \ldots, y_n]$ to complete CWS task, where $y_t \in \{B, M, E, S\}$. Figure 3 shows the overall architecture, with the input example as "人生价值" (life value) and the output as "人生 (life) 价值 (value)".

Glyph Embedding Generation

The autoencoder part is responsible to generate character glyph embedding with the image representation at each time t. The idea of autoencoder consists with two parts: an encoder ϕ and a decoder φ. The encoder part will extract useful features from overall character glyph and get the glyph representation, which will be used for the latter BiLSTM architecture for our CWS task. The decoder is plugged for reconstructing this representation to its original image. The idea of autoencoder can be described at Equation 1, where $L(i_t, \varphi(\phi(i_t))$ is the loss function of autoencoder.

$$\phi, \varphi = \underset{\boldsymbol{\phi}, \boldsymbol{\varphi}}{\arg\min}\ L(i_t, \varphi(\phi(i_t)) \tag{1}$$

The detailed architecture of our autoencoder part can be described as Figure 4, where the input is the character image $\boldsymbol{i_t}$ and the output is reconstructed image $\boldsymbol{i_t^{'}}$. The more similar the reconstructed image $\boldsymbol{i_t^{'}}$ to the original one $\boldsymbol{i_t}$ is, the more confidence we have to claim that this presentation $\boldsymbol{r_t}$ has obtained enough information from the character glyphs, then the more powerful the glyph embedding $\boldsymbol{r_t}$ should be. To be specific, for each character in the sentence at time t, as

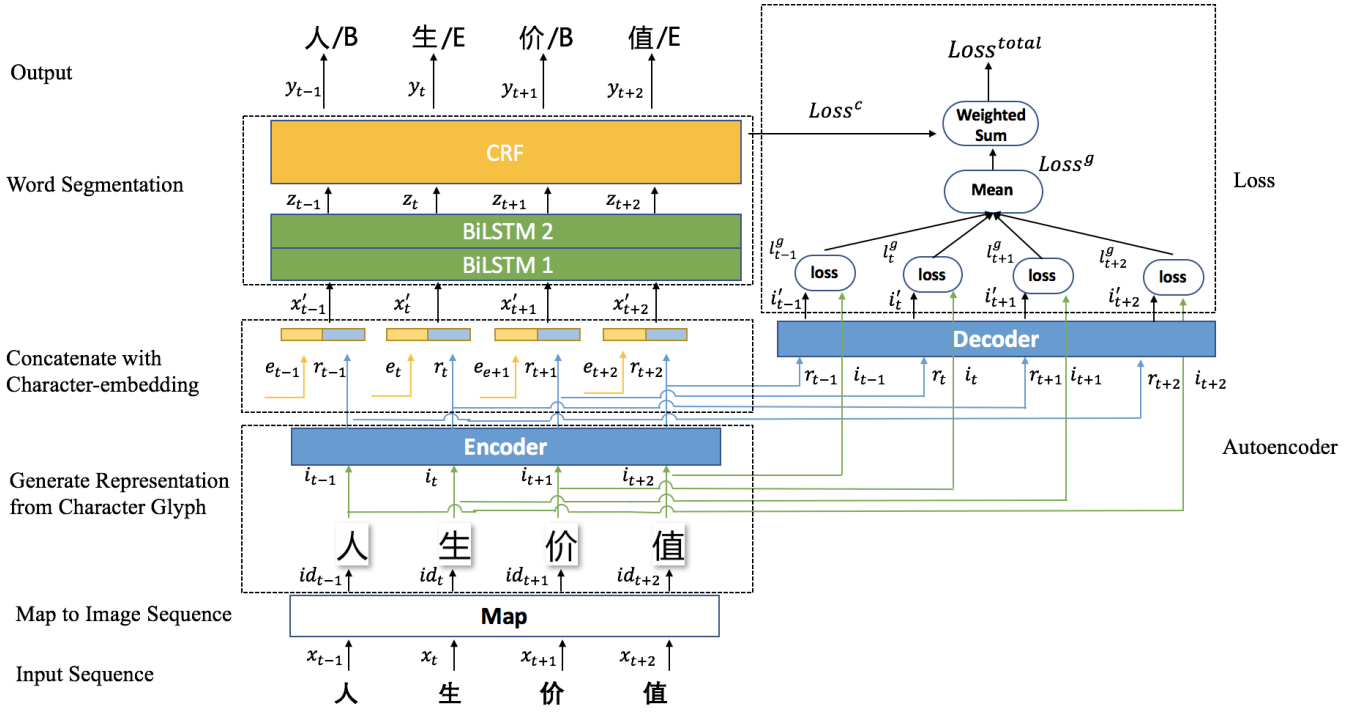


Figure 3: The overall architecture

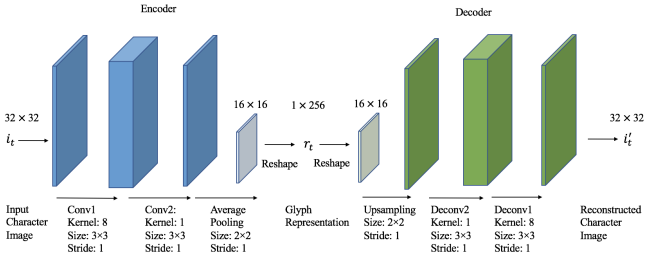


Figure 4: The detailed architecture of the autoencoder.

Equation 2 shows, we compute the character-level glyph loss l_t^g with Mean Squared Logarithmic Error(MSLE).

$$l_t^g = \frac{1}{m}\sum_{j=1}^{m} \log((i_{tj}-1) - \log(i'_{tj}-1))^2 \quad (2)$$

where m is the total number of pixels in raw image i_t and reconstructed image i'_t at time t.

Then the average of them is considered as the sentence-level glyph loss $loss^g$ with sentence length n.

$$Loss^g = \frac{1}{n}\sum_{t=1}^{n} l_t^g \quad (3)$$

We generate the objective function of this autoencoder part with Equation 1 and 3, which forces the encoder to extract useful features from overall glyph as much as possible.

$$\phi, \varphi = \arg\min_{\phi,\varphi} Loss^g \quad (4)$$

Chinese Word Segmentation

We use two BiLSTM layers with CRF for CWS. For each character in the sentence, after getting its glyph representation $\boldsymbol{r_t}$ from encoder ϕ, we concatenate $\boldsymbol{r_t}$ with its character embedding $\boldsymbol{e_t}$ and get the input sequence $\boldsymbol{x}' = [\boldsymbol{x'_0}, \boldsymbol{x'_1}, \ldots, \boldsymbol{x'_t}, \ldots, \boldsymbol{x'_n}]$ of next BiLSTM1 layer for word segmentation, where $\boldsymbol{x'_t}$ can be described as $\boldsymbol{x'_t} = [\boldsymbol{e_t}; \boldsymbol{r_t}]$. Before being fed into the next BiLSTM2 layer and CRF layer, the hidden layer state vectors of the forward and backward direction in BiLSTM1 are concatenated together. This concatenation operation is same in BiLSTM2. Then the output $\boldsymbol{z} = [\boldsymbol{z_0}, \boldsymbol{z_1}, \ldots, \boldsymbol{z_t}, \ldots, \boldsymbol{z_n}]$ of BiLSTM2 layer is fed into the next CRF layer.

Finally, the CRF layer defines a family of conditional probability $p(\boldsymbol{y}|\boldsymbol{z}; \boldsymbol{W}, \boldsymbol{b})$ over each possible label sequence $\boldsymbol{y}$ given the generic input sequence $\boldsymbol{z}$, where $\boldsymbol{W}$ and $\boldsymbol{b}$ represent the weights and biases of the BiLSTM hidden layers.

As for the loss function $Loss^c$ of this CWS part, to make coding easy, different from (Ma and Hovy 2016), we minimize the Negative Conditional Log-likelihood Estimation, which can be defined as:

$$Loss^c(\boldsymbol{W}, \boldsymbol{b}) = -\sum_t \log p(\boldsymbol{y}|\boldsymbol{z}; \boldsymbol{W}, \boldsymbol{b}) \quad (5)$$

The predicted label sequence $\boldsymbol{y}^*$ is the one with the highest conditional probability in the set of $\boldsymbol{z}$:

$$\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{z}; \boldsymbol{W}, \boldsymbol{b}) \quad (6)$$

Total Loss Function

After getting the losses form glyph embedding generation part ($Loss^g$) and Chinese Word Segmentation part ($Loss^c$), we define the total loss $Loss^{total}$ of our architecture by taking the weighted sum of them, shown in Equation 7.

$$Loss^{total} = \lambda Loss^c + (1 - \lambda) Loss^g \tag{7}$$

where λ ($\lambda \in (0, 1)$) is the hyper-parameter to adjust the importance of Antoencoder part and CWS part.

Finally, the overall objective function of our model can be described as Equation 8.

$$\phi, \varphi, \boldsymbol{W}, \boldsymbol{b} = \underset{\phi, \varphi, \boldsymbol{W}, \boldsymbol{b}}{\arg\min}\ Loss^{total} \tag{8}$$

where ϕ, φ are encoder and decoder of autoencoder part to generate glyph embeddings, $\boldsymbol{W}, \boldsymbol{b}$ are parameters in BiLSTM layers to complete Chinese Word Segmentation task.

Experiments

We implement our idea on Five standard Chinese Word Segmentation corpus (MSR, PKU, CTB, AS, CityU) with our synthetic Chinese character image datasets.

Datasets

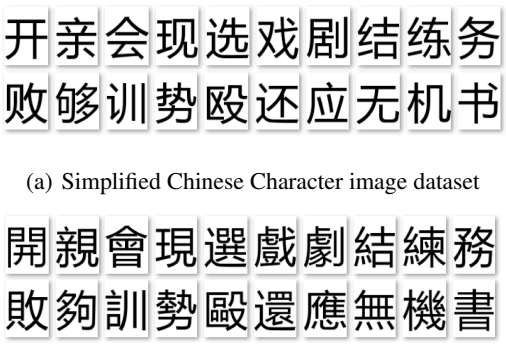
Corpus for Chinese Word Segmentation. We use the MSR, PKU, CityU and AS datasets from the 2^{nd} SIGHAN Bakeoff[3]. We get CTB from Stanford CoreNLP[4]. Among those five datasets, MSR, PKU, CTB are Simplified Chinese datasets; AS and CityU are Traditional Chinese datasets. As for training, we first shuffle each of the five training datasets and divide each of them into two sets. 90% is used as training data and the rest is used as development data. We test with provided test datasets and evaluation scripts.

Character Image Database. We generate 20901 Simplfied and Traditional Chinese character images by converting[5] 64pt *Microsoft Yahei Font* of *Word* into corresponding monochrome images with 32×32 pixels. We map the special symbols, such as "©", "△" and "$" , to blank images. Part of the dataset looks like Figure 5.

Setup

Traditional Chinese characters are more complex than Simplified ones. They experienced less simplifications. Thus they keep more information in their glyph. In order to investigate the potential of glyph which may be different in Traditional and Simplified Chinese, our experiments can be divided into two groups according to Simplified Chinese dataset and Traditional Chinese dataset.

In each group, we have three models for CWS. We first implement a BiLSTM architecture as baseline with character embeddings. Then we add our autoencoder (introduced in Glyph Embedding Generation) before the BiLSTM to see the potential of character glyph embedding without any character embedding. Finally, we introduce in the character embedding and concatenate it with the encoder ouput of autoencoder, then feed them into the BiLSTM layers for CWS.



(a) Simplified Chinese Character image dataset

(b) Traditional Chinese Character image dataset

Figure 5: Preview of synthetic Chinese character image datasets.

The character embeddings mentioned above can be pretrained or random initialized. As for Simplified Chinese experiment group, we get the pre-trained character embeddings by word2vec on Simplified Chinese Wikipedia corpus. As for the Tradtional Chinese experiment group, in order to get enough character embeddings in vocabulary, we first convert Chinese Wikipedia corpus into Tradtional Chinese raw text with *Open Chinese Convert* (OpenCC)[6]. Then we use word2vec to get the pre-trained embeddings like what we do for Simplified Chinese.

Results and Discussion

Table 1 shows the our experiment results on MSR, PKU, CTB, AS and CityU datasets. We use vertical double-line to distinguish the datasets of Simplified CWS task and Traditional CWS task. Because (Shi et al. 2015) work is most similar to us, our results can be compared directly and fairly, which is shown in Table 2. Finally, we add some other methods for CWS task into our comparison pool (Table 3).

Table 6 is the hyper-parameter settings of our model. What need to mention is that because the purpose of our work is to introduce the idea of character glyph embedding and then verify its feasibility directly, we don't pay much effort on tuning hyper-parameters and use the same architecture and hyper-parameters settings in both Simplified CWS and Traditional CWS experiments on different datasets.

Results of Simplified Chinese Experiments We conduct Simplified Chinese experiments on MSR, PKU and CTB datasets, whose results are shown on the left of the vertical double-line in Table 1.

The first row and second row indicate the results of our baseline BiLSTM model("RandomE"and "PretrainedE" in Table 1) with random initialized character embeddings or pretrained character embeddings. No glyph embeddings are

[3]http://sighan.cs.uchicago.edu/bakeoff2005/

[4]https://stanfordnlp.github.io/CoreNLP/index.html

[5]We will open-source our code and synthetic image datasets for public use (link omitted for double-blind review).

[6]https://github.com/BYVoid/OpenCC

Model	MSR				PKU				CTB				CityU				AS			
	P	R	F	OOV	P	R	F	OOV	P	R	F	OOV	P	R	F	OOV	P	R	F	OOV
RandomE	95.8	95.5	95.6	69.7	93.2	92.4	92.8	64.9	93.8	93.6	93.7	72.3	93.7	93.6	93.6	65.2	94.1	93.4	93.7	66.2
PretrainedE	96.0	95.7	95.8	72.9	94.0	93.2	93.6	69.0	94.7	94.8	94.7	75.7	94.3	93.8	94.0	67.2	95.1	93.6	94.3	67.3
Auto	95.8	95.5	95.6	74.4	93.7	92.9	93.3	74.4	94.4	94.9	94.6	78.1	94.1	94.2	94.1	76.9	94.8	93.9	94.3	74.3
Auto+RandomE	96.0	95.8	95.9	73.4	94.2	92.8	93.5	71.3	94.9	94.8	94.8	77.2	94.4	94.4	94.4	75.4	95.0	94.1	94.5	72.6
Auto+PretrainedE	96.9	96.7	96.8	74.6	94.8	93.7	94.2	74.4	95.1	94.9	95.0	78.8	94.7	94.8	94.7	77.3	95.5	94.3	94.9	75.7

Table 1: The experimental results for CWS. "E" is shorten for character embeddings, which can be pre-trained or randomly initialized. "Auto" is shorten for autoencoder to extract character image features and generate glyph embeddings.

model	MSR				PKU			
	P	R	F	OOV	P	R	F	OOV
(Shi et al. 2015)	93.4	93.3	93.3	-	92.6	92.1	92.3	-
Auto	**95.8**	**95.5**	**95.6**	**74.4**	**93.7**	**92.9**	**93.3**	**74.4**

Table 2: Comparison with similar work. (Shi et al. 2015) considered character component features which are part of glyph to get radical embeddings in CWS task. They also use other features like sliding window of size 3.

used here. As expected, "PretrainedE" works better, since the embeddings have contained context information. Following the baseline is the model that has been plugged with autoencoder before BiLSTM.

When looking into the performance of different models with various kind of embeddings, we can notice the following findings:

Firstly, the model at third row("Auto" in Table 1) works with glyph embeddings and without character embeddings. The better performance of it over the first row ("RandomE") indicates that the glyph embeddings do actually contain some useful information, which is more helpful than randomly initialized embeddings for CWS task.

Secondly, the second row ("Pretrained") is the model with pre-trained character embeddings before BiLSTM and without Glyph embeddings. Compared with the "Auto" model where glyph embeddings are generated for BiLSTM (third row in Table 1), we can notice the variety between these two models is only the initialization difference for BiLSTM layers. And the better performance of "PretrainedE" model shows that the benefit of only using glyph is not enough, because the pre-trained character embeddings may contain more context information in their representations which can directly help to improve the performance.

However, when the information in both the character embedding and its glyph embedding is utilized at the same time, the performance of "Auto + PretrainedE" (last row in Table 1) is boosted to best.

What is remarkable is the OOV recall rate. It is impressive that glyph embeddings from autoencoder have a strong ability on solving OOV problem. Since the large context knowledge has been learned into the pre-trained character embeddings, the OOV recall rate of "PretrainE" is better than "RandomE" as expectation. However, with only the overall glyph information, the OOV of "Auto" model reaches much higher than "pretrainE", which means there does exist the useful semantic and logical similarity that resides in similar characters. And that similarity can guide our model to recognize the "unformiliar" words.

Besides what have been metioned above, we compute the cosine similarity of some fine-tuned character embeddings from the baseline "PretraineE" model and "Auto + PretrainedE" model, as shown in Table 4. We can notice that the cosine similarity increases as expected, which means the glyph similarity has successfully fused into their semantic representations.

Table 5 shows some segmentation cases where baseline BiLSTM with pretrained character embeddings fails but our "Auto + PretrainedE" works correctly after using generated character glyph embeddings obtained from autoencoder. For example, it is hard to judge how to segment the phrase "冰冷寂然" (cold and quiet) with traditional context-based model. But "冰" (ice) and "冷" (cold) are similar in glyph and semantics. Also, they look much different to "寂然" (quiet). The semantic relationship between these characters will be expressed through glyph embeddings. Thus, this additional information from glyph embeddings helps our model to make decisions correctly.

Results of Traditional Chinese Experiment The Traditional Chinese Experiments are conducted on AS and CityU datasets, whose results can be found on right of the vertical double-line in Table 1.

Similar to Simplified Chinese experiments, the baseline BiLSTM model with pretrained character embeddings ("PretraineE") outperform the baseline with random initiliazed character embeddings ("RandomE"). And the "Auto" model which only have glyph embeddings wins the baseline BiLSTM with random initialization ("RandomE"). Also, with both pretrained character embeddings and glyph embeddings, "Auto + PretrainedE" model works best.

But what is different from Simplified Chinese experiments is that only with glyph embeddings and without character embeddings, the model ("Auto") performs no less than baseline BiLSTM with pretrained character embeddings ("PretraineE") on CityU and AS datasets, which reveals the glyph embeddings are able to contain useful information for word segmentation no less than what pretrained embeddings have learned from context if the glyph features are strong enough. What's more, together with pretrained character embeddings, the "Auto + PretraineE" still performs better than "Auto" only with glyph embeddings, which indicates the complementary effect of glyph information and context information.

In the meanwhile, the glyph embeddings generated from autoencoder with character images provide a more prominent support to solve OOV words. Compared with Simplified characters, the Traditional character consists of more

model	MSR				PKU				CTB				CityU				AS			
	P	R	F	OOV	P	R	F	OOV	P	R	F	OOV	P	R	F	OOV	P	R	F	OOV
(Chen et al. 2015b)▲△	97.5	97.3	97.4	-	96.6	96.4	96.5	-	96.2	95.8	96.0	-	-	-	-	-	-	-	-	-
(Chen et al. 2017)*▲	95.7	95.99	95.84	66.28	93.67	92.93	93.3	66.09	95.19	95.42	95.30	76.47	94.00	94.15	94.07	65.79	93.64	94.77	94.20	70.07
(Cai and Zhao 2016)	96.3	96.8	96.5	-	95.8	95.2	95.5	-	-	-	-	-	-	-	-	-	-	-	-	-
(Shi et al. 2017)▲	96.17	96.34	96.26	69.20	94.16	93.75	93.95	67.45	95.35	95.45	95.40	77.21	93.91	94.11	94.01	65.40	94.29	95.23	94.76	71.40
(Han et al. 2017)▲	97.17	97.40	97.29	-	94.78	95.66	95.22	-	95.14	95.28	95.21	-	94.86	94.16	94.51	-	95.28	94.53	94.90	-
Our Auto + PretrainedE	96.9	96.7	96.8	**74.6**	94.8	93.7	94.2	**74.4**	95.1	94.9	95.0	**78.8**	94.7	**94.8**	**94.7**	**77.3**	**95.5**	94.3	**94.9**	**75.7**

Table 3: Comparison with other models.Results with ▲ use extra features like bigram and results with △ use external dictionaries. *(Chen et al. 2017) reported the replicated result of his 2015 BiLSTM model without extra dictionaries.

Cosine Similarity	PretrainE	Auto + PretrainE
鸡(chicken) v.s. 鸭(duck)	0.253	0.419
舰(ship) v.s. 船(boat)	0.219	0.453
霜(frost) v.s. 露(dew)	0.124	0.367

Table 4: Cosine similarity of final character representations before and after adding autoencoder.

PretrainE	Auto + PretrainE
神情/变/得/冰冷寂然	神情/变/得/冰冷/寂然
中国/边陲/城市/漠/河	中国/边陲/城市/漠河
避/地/时/忽/忽	避地/时/忽忽

Table 5: Some segmentation results before and after adding autoencoder.

Max sentence length	32
Character Embedding length	100
LSTM1 & LSTM2 hidden unit number	150
LSTM1 dropout	0.2
LSTM2 dropout	0.0
λ	0.9
batch size	256
Initial learning rate	0.003

Table 6: The settings of hyper-parameter.

components averagely which are informative and graphic. They are more likely to have a larger overlap of the same components among different characters. Therefore, autoencoder can find more semantic and logical similarities between them and generate more powerful glyph embeddings by fusing these similarities into them. As Table 1 shows, the OOV recall rate on AS has been improved by 10.1% over the baseline BiLSTM and 8.4% on CityU ("Auto + PretrainedE" than "PretrainedE"). The differences are more significant than those on Simplified Chinese datasests.

We also compute the cosine similarity of final character embeddings before and after adding autoencoder. Then we compute the similarity improvement by taking subtraction and compare these improvements of characters in Traditional format and Simplified format. From Table 7, we can notice the improvement in Traditional Chinese is more significant than it in Simplified Chinese, which indicates glyph information is more powerful in finding implicit relationship for Traditional Chinese.

Comparison with other works Since there is little work to investigate CWS with character glyph, we first compare our model with (Shi et al. 2015) in Table 2, whose work is the most similar to ours.

Traditional Chinese Format	Similarity Improvement	Simplified Chinese Format	Similarity Improvement
繼(continue) v.s. 續(continue)	0.086	继(continue) v.s. 续(continue)	0.009
書(writing) v.s. 畫(painting)	0.107	书(writing) v.s. 画(painting)	0.001

Table 7: The cosine similarity improvement before and after autoencoder in Traditional and Simplified Format.

They proposed a new radical embedding technique which is also inspired by the semantic meanings in character's components that are part of character glyph, then validated radical embeddings as the input of CWS task. In order to show the competitiveness of our generated glyph embeddings and to make the comparison fair, we use the "Auto" model only with the mapped character raw image sequence as input and without any pretrained character embeddings to compare with (Shi et al. 2015). Better performance of their model is not shown in Table 3, perhaps because they just used the separated component information from radical sequence and ignored the overall character glyph features.

Then we compare with other neural network CWS models with LSTM and its novel variants (Table 3). (Chen et al. 2015b) first introduced the BiLSTM model for CWS and got the impressive results. But it needs to be mentioned that they used external Chinese idiom dictionaries and other features like bigram. Later in their 2017 work, they replicated this model without any dictionaries and reported the results as the second row in Table 3, which decreased reasonably.

(Cai and Zhao 2016) employed LSTM on language scoring model and got good performance, but they also used the external Chinese idiom dictionaries. (Shi et al. 2017) introduced a hyper-gated LSTM by adding sequential information on gates with pre-trained character embeddings on Chinese Wikipedia corpus and bigram features. (Han et al. 2017) also utilized LSTM to train a single joint model on multi-criteria corpora in CWS task. And he took the advantage of bigram features as well.

Because the glyph information is hidden in character's attributions itself, our model does not use any extra external dictionaries or resources. Also, in order to guarantee the improvements only come from character glyph itself and be convincing, our model are more straightforward and unpretentious with little tricks or complex network functions. But our results are still competitive and impressive.

So far, these discussions can verify the effectiveness of the overall character glyph information in CWS task.

Conclusions

In this paper, based on the new idea that the semantic features hidden in glyph of character components are of vital importance, we propose a model by adding autoencoder before BiLSTM with CRF to generate character glyph embeddings from the character images, which are verified to be useful in Chinese word segmentation (CWS). Since this additional implicit information is the inherent attribute of characters, our work does not use any extra external dictionaries or other resources. Our experiment results on Simplified Chinese datasets and Traditional Chinese datasets show the utility of character glyph, especially its impressive support for solving OOV and finding semantic relationship of Traditional Chinese characters. In future, we will apply our idea in other potential NLP tasks like Sentiment Analysis.

References

Cai, D., and Zhao, H. 2016. Neural word segmentation learning for chinese. *arXiv preprint arXiv:1606.04300*.

Chen, X.; Qiu, X.; Zhu, C.; and Huang, X. 2015a. Gated recursive neural network for chinese word segmentation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, 1744–1753.

Chen, X.; Qiu, X.; Zhu, C.; Liu, P.; and Huang, X. 2015b. Long short-term memory neural networks for chinese word segmentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1197–1206.

Chen, X.; Shi, Z.; Qiu, X.; and Huang, X. 2017. Adversarial multi-criteria learning for chinese word segmentation. *arXiv preprint arXiv:1704.07556*.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Han, H.; Lei, W.; Hua, Y.; Zhimin, G.; Yi, F.; and George, T. 2017. Effective neural solution for multi-criteria word segmentation. *arXiv preprint arXiv:1712.02856*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Liu, Y.; Che, W.; Guo, J.; Qin, B.; and Liu, T. 2016. Exploring segment representations for neural segmentation models. *arXiv preprint arXiv:1604.05499*.

Liu, F.; Lu, H.; Lo, C.; and Neubig, G. 2017. Learning character-level compositionality with visual features. *arXiv preprint arXiv:1704.04859*.

Ma, X., and Hovy, E. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Mansur, M.; Pei, W.; and Chang, B. 2013. Feature-based neural language model and chinese word segmentation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, 1271–1277.

Ng, H. T., and Low, J. K. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Pei, W.; Ge, T.; and Chang, B. 2014. Max-margin tensor neural network for chinese word segmentation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 293–303.

Peng, F.; Feng, F.; and McCallum, A. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the 20th international conference on Computational Linguistics*, 562. Association for Computational Linguistics.

Shi, X.; Zhai, J.; Yang, X.; Xie, Z.; and Liu, C. 2015. Radical embedding: Delving deeper to chinese radicals. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, 594–598.

Shi, Z.; Chen, X.; Qiu, X.; and Huang, X. 2017. Hyper-gated recurrent neural networks for chinese word segmentation. In *National CCF Conference on Natural Language Processing and Chinese Computing*, 443–455. Springer.

Su, T.-R., and Lee, H.-Y. 2017. Learning chinese word representations from glyphs of characters. *arXiv preprint arXiv:1708.04755*.

Xu, J., and Sun, X. 2016. Dependency-based gated recursive neural network for chinese word segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, 567–572.

Xue, N. 2003. Chinese word segmentation as character tagging. *International Journal of Computational Linguistics & Chinese Language Processing, Volume 8, Number 1, February 2003: Special Issue on Word Formation and Chinese Language Processing* 8(1):29–48.

Yao, Y., and Huang, Z. 2016. Bi-directional lstm recurrent neural network for chinese word segmentation. In *International Conference on Neural Information Processing*, 345–353. Springer.

Yu, J.; Jian, X.; Xin, H.; and Song, Y. 2017. Joint embeddings of chinese words, characters, and fine-grained subcharacter components. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 286–291.

Zhang, M.; Zhang, Y.; and Fu, G. 2016. Transition-based neural word segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 421–431.

Zheng, X.; Chen, H.; and Xu, T. 2013. Deep learning for chinese word segmentation and pos tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 647–657.