Glyce: Glyph-vectors for Chinese Character Representations

Yuxian Meng*, **Wei Wu***, **Qinghong Han, Muyu Li, Xiaoya Li,**
Jie Mei, Ping Nie, Xiaofei Sun and Jiwei Li

Shannon.AI

{yuxian_meng, wei_wu, qinghong_han, muyu_li, xiaoya_li,
jie_mei, ping_nie, xiaofei_sun, jiwei_li}@shannonai.com

Abstract

It is intuitive that NLP tasks for logographic languages like Chinese should benefit from the use of the glyph information in those languages. However, due to the lack of rich pictographic evidence in glyphs and the weak generalization ability of standard computer vision models on character data, an effective way to utilize the glyph information remains to be found.

In this paper, we address this gap by presenting the Glyce, the glyph-vectors for Chinese character representations. We make three major innovations: (1) We use historical Chinese scripts (e.g., bronzeware script, seal script, traditional Chinese, etc) to enrich the pictographic evidence in characters; (2) We design CNN structures tailored to Chinese character image processing; and (3) We use image-classification as an auxiliary task in a multi-task learning setup to increase the model's ability to generalize.

For the first time, we show that glyph-based models are able to consistently outperform word/char ID-based models in a wide range of Chinese NLP tasks. Using Glyce, we are able to achieve the state-of-the-art performances on 13 (almost all) Chinese NLP tasks, including (1) character-Level language modeling, (2) word-Level language modeling, (3) Chinese word segmentation, (4) name entity recognition, (5) part-of-speech tagging, (6) dependency parsing, (7) semantic role labeling, (8) sentence semantic similarity, (9) sentence intention identification, (10) Chinese-English machine translation, (11) sentiment analysis, (12) document classification and (13) discourse parsing.[1] [2] [3]

1 Introduction

Chinese characters are logograms and can be decomposed into smaller and more basic phono-semantic compounds: pictographs (象形xiàngxíng), the graphical depiction of the object (e.g., 亻rén "person/human", 日rì "sun", 木mù "tree/wood"); and phonetic complexes (形声xíngsheng) which are used for pronunciations (e.g., 青 in 晴). Dating back to the Han dynasty in the 2nd century AD, the dictionary *shuo-wen-jie-zi* used graphic components to index characters, the tradition of which is still followed today. Because many Chinese characters evolved from pictures (examples shown in Figure 1), the logograms of Chinese characters encode rich information for meanings.

[1] Wei Wu and Yuxian Meng contribute equally to this work. The rest of the authors are in alphabetical order.

[2] All code will be released upon publication.

[3] Contributions: Jiwei Li (李纪为) proposed the original idea of using glyph information to model Chinese characters, which dates back to the summer of 2014, when he was brainstorming with Dr. Xinlei Chen, who is now a researcher at Facebook AI Research. Jiwei tried to implement this idea in Apr 2015, when he was a Ph.D student at Stanford with Prof. Dan Jurafsky. But he but did not succeed in obtaining consistent performance improvements. Wei Wu (吴炜) designed and implemented the first Glyce-char model on the char-level language modeling task. This is the first time that glyph-embeddings consistently outperform the charID embeddings. Yuxian Meng (孟昱先) proposed the tianzige-CNN structure, the image-classification auxiliary objective and the decaying λ. Jiwei then proposed using historical Chinese scripts. Based on these modifications, consistent performance boosts started being observed. Yuxian is responsible for the results of word-level language modeling and the intention classification; Wei is responsible for the results in the segmentation, NER and POS. Qinghong Han (韩庆宏) is responsible for the results in semantic role labeling; Xiaoya Li (李晓雅) for the results in Chinese-English MT; Muyu Li (李慕宇) for dependency parsing and POS; Jie Mei (梅杰) for discourse parsing; Ping Nie (聂平) for semantic similarity; and Xiaofei Sun (孙晓飞) is responsible for the results in text classification and sentiment analysis.

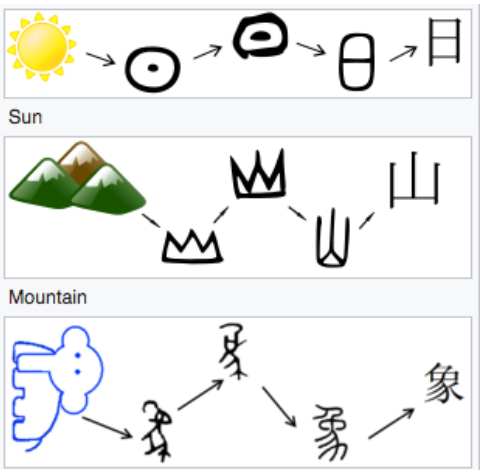

Figure 1: Many Chinese characters evolved from pictures (copied from Wikipedia https://en.wikipedia.org/wiki/Chinese_characters.)

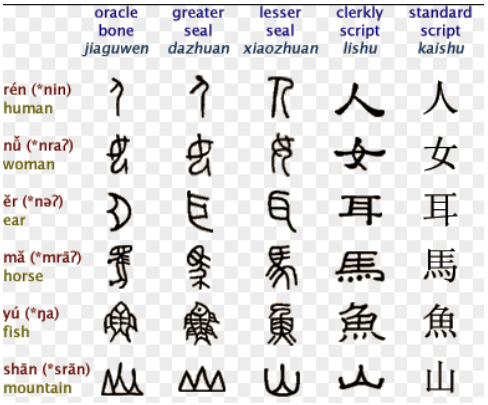

Figure 2: Evolution of Chinese characters. Picture borrowed from web.

The emergence of distributed representations (Mikolov et al., 2013; Pennington et al., 2014) provides an elegant way to represent text semantics, and have been widely applied to Chinese natural language processing. The mainstream deep learning algorithms mostly use words or characters as basic semantic units (Zheng et al., 2013; Chen et al., 2015b,a; Xu et al., 2016; Cai and Zhao, 2016), and learn embedding representations at the word/character level. Glyph representations have rarely been used.

It is intuitive that taking into account logographic information should help semantic modeling. Recent studies indirectly support this argument: Radical representations have proved to be useful in a wide range of language understanding tasks (Shi et al., 2015; Li et al., 2015b; Yin et al., 2016; Sun et al., 2014; Shao et al., 2017). Using the Wubi scheme —- a Chinese character encoding method that mimics the order of typing the sequence of radicals for a character on the computer keyboard —- is reported to improve performances on Chinese-English machine translation (Tan et al., 2018). Although radical and Wubi representations encode some information about character structures and help to build better character representations to some degree, both radicals and wubi representations are encoded in arbitrary IDs, and thus do not touch upon the deeper logographic information.

Recently, there have been some efforts applying CNN-based algorithms on the visual features of characters. Unfortunately, they do not show consistent performance boosts (Liu et al., 2017; Zhang and LeCun, 2017), and some even yield negative results (Dai and Cai, 2017). For instance, Dai and Cai (2017) run CNNs on char logos to obtain Chinese character representations and used them in the downstream language modeling task. They reported that the incorporation of glyph representations actually worsen the performance and concluded that CNN-based representations do not provide extra useful information for language modeling. Using similar strategies, Liu et al. (2017) and Zhang and LeCun (2017) test the idea on text classification tasks, and performance boosts were observed only in very limited number of settings.

We propose the following explanations for negative results reported in the earlier CNN-based models (Dai and Cai, 2017): (1) not using the correct version(s) of script: Chinese character system has a long history of evolution, as shown in Figure 2. The most famous versions include Oracle bone script (2000 BC – 300 BC), Clerical script (200BC-200AD), Seal script (100BC - 420 AD), Tablet script (420AD - 588AD), etc. This evolution follows certain patterns. The characters started from being easy-to-draw, and slowly transitioned to being easy-to-write. Also, they became less pictographic and less concrete over time. The most widely used script version to date, the *Simplified Chinese*, is the easiest script to write, but inevitably loses the most significant amount of pictographic information. This potentially leads to the poor performances of the models trained only on simplified Chinese characters (Dai and Cai, 2017). (2) not using the proper CNN structures: unlike ImageNet images (Deng et al., 2009), the size of which is mostly at the scale of 800*600, character logos are significantly smaller (usually with the size of 12*12). It requires a different CNN architecture to capture the local graphic features of character

images; (3) no regulatory functions were used in previous work: unlike the classification task on the imageNet dataset, which contains tens of millions of data points, there are only about 10,000 Chinese characters. Auxiliary training objectives are thus critical in preventing overfitting and promoting the model's ability to generalize.

In this paper, we propose the GLYCE, the GLYph-vectors for Chinese character representations. We treat Chinese characters as images and use CNNs to obtain their representations. We resolve the aforementioned issues by using the following techniques:

- We use the ensemble of the historical and the contemporary scripts (e.g., the bronze-ware script, the clerical script, the seal script, the traditional Chinese etc), along with the scripts of different writing styles (e.g, the cursive script) to enrich pictographic information from the character images.
- We utilize the Tianzige-CNN (田字格) structures tailored to logographic character modeling for Chinese.
- We use multi-task learning methods by adding an image-classification loss function to increase the model's ability to generalize.

Using Glyce, we are able to achieve the state-of-the-art performances for 13 (almost all) Chinese NLP tasks, including: (1) character-Level language modeling, (2) word-Level language modeling, (3) Chinese word segmentation, (4) name entity recognition, (5) part of speech tagging, (6) dependency parsing, (7) semantic role labeling, (8) sentence semantic similarity, (9) intention classification, (10) Chinese-English machine translation, (11) sentiment analysis, (12) document classification and (13) discourse parsing.

The rest of this paper is organized as follows: we describe the related work in Section 2, model details in Section 3, experimental details in Section 4, followed by a conclusion in Section 5.

2 Related Work

Chinese characters can be decomposed into smaller and primitive components, which serve as more basic units for meanings than words. One line of work utilizes radical representations to build character representations (Shi et al., 2015; Li et al., 2015b; Yin et al., 2016; Sun et al., 2014). Li et al. (2015b) and Sun et al. (2014) learned radical representations based on skip-gram (Mikolov et al., 2013). Yin et al. (2016) found that radical representations improve word-similarity tasks. Nguyen et al. (2018) used radical embeddings to improve performances on word pronunciation tasks. Another way of representing Chinese characters is to use the Wubi encoding scheme, which is a shape-based encoding method for Chinese characters. Tan et al. (2018) showed that wubi-style encoding can help machine-translation tasks.

As far as we are concerned, running CNNs on character logos to obtain Chinese character representations was first explored in Dai and Cai (2017). The logo-based representations were tested on two tasks: language modeling and word segmentation. Unfortunately, the results reported in Dai and Cai (2017) are negative. Tan et al. (2018) incorporateed CNN structures into GloVe (Pennington et al., 2014) and skipgram (Mikolov et al., 2013) to learn Chinese character embedings. Significant improvements were reported on word semantic evaluation tasks such as analogy predictions. Unfortunately, Tan et al. (2018) did not take further steps to explore the performances on real-world Chinese NLP tasks. Another important work is from Liu et al. (2017), which comprehensively studied visual features of three logographic languages: Chinese, Japanese, and Korean. The models were evaluated on the task of classifying Wikipedia titles and models with logographic features. Similarly, Zhang and LeCun (2017) used logographic features to obtain characters embeddings for Chinese, Japanese, and Korean. The logo embeddings were tested on text classification tasks using online product reviews. Liu et al. (2017) and Zhang and LeCun (2017) tentatively explored the possibility of using CNNs to encode logographic features for Chinese in text classification tasks. As far as we are concerned, no widely-agreed conclusions have been make on whether and how treating Chinese characters as logo images can help semantic modeling,.

CNN based models are widely used in character-level encoding for alphabetical languages (Zhang et al., 2015; Kim et al., 2016; Chung et al., 2016; Luong and Manning, 2016). In Chinese, character level representations are combined with word-level representations (Yin et al., 2016; Dong et al., 2016; Yu et al., 2017). Also in Zhang and LeCun (2017), Chinese characters are transformed to pingyin, a romanization system for standard Chinese using alphabets.

3 Glyce

In this section, we describe the proposed Glyce model in detail.

3.1 Using Historical Scripts

As discussed in Section 1, pictographic information is heavily lost in the simplified Chinese script. We thus propose using scripts from various time periods in history and also of different writing styles. We collect the following major historical scripts:[4] 金文(Bronzeware script), 隶书(Clerical script), 篆书(Seal script) and 魏碑(Tablet script), 繁体中文(traditional Chinese) and 简体中文(Simplified Chinese), along with scripts of different writing styles: 草书(cursive script) and 仿宋. The details are shown in Table 1, with examples shown in Figure 2.

Scripts from different historical periods, which are usually very different in shape, help the model to integrate pictographic evidence from various sources; Scripts of different writing styles help improve the model's ability to generalize. Both strategies are akin to widely-used data augmentation strategies in computer vision.

3.2 The Tianzige-CNN Structure for Glyce

Deep CNNs (He et al., 2016; Szegedy et al., 2016; Ma et al., 2018b) are widely used in computer vision tasks like image classification and detection. Standard CNN structures consist of two major components: convolutional layers and pooling layers. For a three-dimensional image input $x \in \mathcal{R}^{H \times W \times C}$ (H, W, C are input width, height, channels respectively), a CNN model uses a kernel to sweep over the input grids. The $n \times n$ kernel maps each $n \times n \times C$ region from the original image to a single value. The max-pooling operation usually follows the convolutional operation, picking the maximum value from neighboring grids.

Unfortunately, directly using deep CNNs in our task results in very poor performances because of (1) relatively smaller size of the character images: the size of Imagenet images is usually at the scale of 800*600, while the size of Chinese character images is significantly smaller, usually at the scale of 12*12; and (2) the lack of training examples: classifications on the imageNet dataset utilizes tens of millions of different images. In contrast, there are only about 10,000 distinct Chinese characters.

To tackle these issues, we propose the Tianzige-CNN structure, which is tailored to Chinese character modeling as illustrated in Figures 3. Tianzige (田字格) is a traditional form of Chinese Calligraphy. It is a four-squared format (similar to Chinese character 田) for beginner to learn writing Chinese characters, shown in Figure 4. The input image x_{image} is first passed through a convolution layer with kernel size 5 and output channels 1024 to capture lower level graphic features. Then a max-pooling layer of kernel size 4 is applied to the feature map and reduces the resolution from 8×8 to the tianzige 2×2, . This 2×2 tianzige structure presents how radicals are arranged in Chinese characters and also the order by which Chinese characters are written. We find the tianzige structure of significant importance in extracting character meanings. Finally, we apply convolutional operations to map tianzige grids to the final outputs. Instead of using conventional convolutional operations, we use group convolutions (Krizhevsky et al., 2012; Zhang et al., 2017) as shown in Figure 6. Group convolutional filters are much smaller than their normal counterparts, and thus are less prune to overfitting.

It is fairly easy to adjust the model from single script to multiple scripts, which can be achieved by simply changing the input from 2D (i.e., $d_{\text{font}} \times d_{\text{font}}$) to 3D (i.e., $N_{\text{script}} \times d_{\text{font}} \times d_{\text{font}}$), where d_{font} denotes the font size.

3.3 Using Image Classification as an Auxiliary Objective

To further prevent overfitting, we use the task of image classification as an auxiliary training objective. The glyph embedding h_{image} from CNNs will be forwarded to an image classification objective to predict its corresponding charID. Suppose the label of image x is z. The training objective for the image classification task $\mathcal{L}(\text{cls})$ is given as follows:

$$\begin{aligned} \mathcal{L}(\text{cls}) &= -\log p(z|x) \\ &= -\log \text{softmax}(W \times h_{\text{image}}) \end{aligned} \quad (1)$$

Let $\mathcal{L}(\text{task})$ denote the task-specific objective for the task we need to tackle, e.g., language modeling, word segmentation, Machine Comprehension, etc. We linearly combine $\mathcal{L}(\text{task})$ and $\mathcal{L}(\text{cl})$, making the final objective training function as follows:

$$\mathcal{L} = (1 - \lambda(t))\, \mathcal{L}(\text{task}) + \lambda(t) \mathcal{L}(\text{cls}) \quad (2)$$

[4] We did try the oracle-bone script (甲骨文). But the number of deciphered characters is fewer than 2,000 out of about 5,000 symbols, which means a large proportion of currently used Chinese characters do not have oracle-bone-script correspondence. We thus give up the oracle-bone script.

Chinese	English	Time Period
金文	Bronzeware script	Shang dynasty and Zhou dynasty (2000 BC – 300 BC)
隶书	Clerical script	Han dynasty (200BC-200AD)
篆书	Seal script	Han dynasty and Wei-Jin period (100BC - 420 AD)
魏碑	Tablet script	Northern and Southern dynasties 420AD - 588AD
繁体中文	Traditional Chinese	600AD - 1950AD (mainland China). still currently used in HongKong and Taiwan
简体中文(宋体)	Simplified Chinese - Song	1950-now
简体中文(仿宋体)	Simplified Chinese - FangSong	1950-now
草书	Cursive script	Jin Dynasty to now

Table 1: Scripts and writing styles used in Glyce.

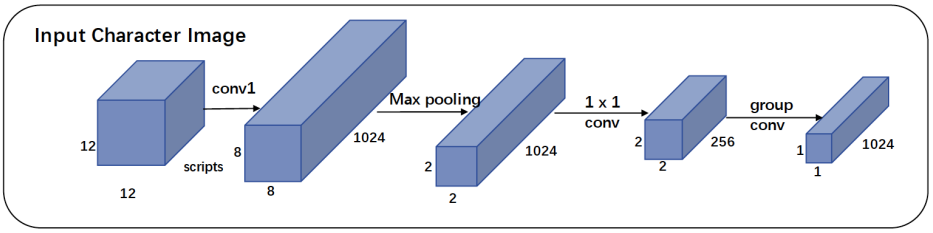

Figure 3: The CNN structure for Glyce.

layer	kernel size	feature size
input		n× 12× 12
conv	5	1024×8×8
relu		1024×8×8
maxpool	4	1024× 2×2
8-group conv	1	256×2× 2
16-group conv	2	1024×1× 1

Table 2: The tianzige-CNN structure in Glyce.

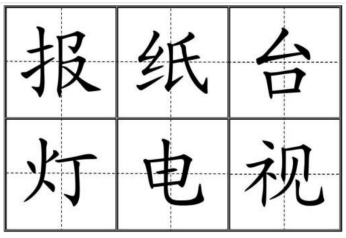

Figure 4: An illustration of the 2 × 2 structure of 田字格(tianzege), the pattern of which illustrates how radicals are arranged in Chinese characters and the order by which Chinese characters are written.

where $\lambda(t)$ controls the trade-off between the task-specific objective and the auxiliary image-classification objective. λ is a function of the number of epochs t: $\lambda(t) = \lambda_0 \lambda_1^t$, where $\lambda_0 \in [0, 1]$ denotes the starting value, $\lambda_1 \in [0, 1]$ denotes the decaying value. This means that the influence from the image classification objective decreases as the training proceeds, with the intuitive explanation being that at the earlier stage of training, we need more regulations from the image classification task. Adding image classification as a training objective mimics the idea of multi-task learning (Collobert et al., 2011; Chen et al., 2017b; Hashimoto et al., 2016; FitzGerald et al., 2015).

3.4 Glyce-Char Embeddings

An overview of Glyce-Char embedding is shown in Figure 5. Images from different scripts of the same Chinese character are first stacked and then passed through CNNs to obtain image embedding h_{image}. h_{image} is directly output to the image classification model for the training of $\mathcal{L}(\text{cls})$. Glyph embeddings and charID embeddings are then combined to obtain Glyce-char embeddings. using concatenation, highway networks (Srivastava et al., 2015) or another fully connected layer.

3.5 Glyce-Word Embeddings

A Chinese word can be further broken down into characters. Recent work (Yin et al., 2016; Dong et al., 2016; Yu et al., 2017) combine Chinese character representations with word representations to capture more fined-grained semantics, and also to handle unknown words. The Glyce-word embedding is shown in Figure 5. We first obtain the glyph embedding for each character using CNNs, which is then concatenated with charID embedding to obtain Glyce-char embedding. Since Chinese words can consist of arbitrary number of characters, we use one layer of max pooling on all constituent Glyce-char embeddings to make the dimensionality invariant. The outputs from the pooling layer

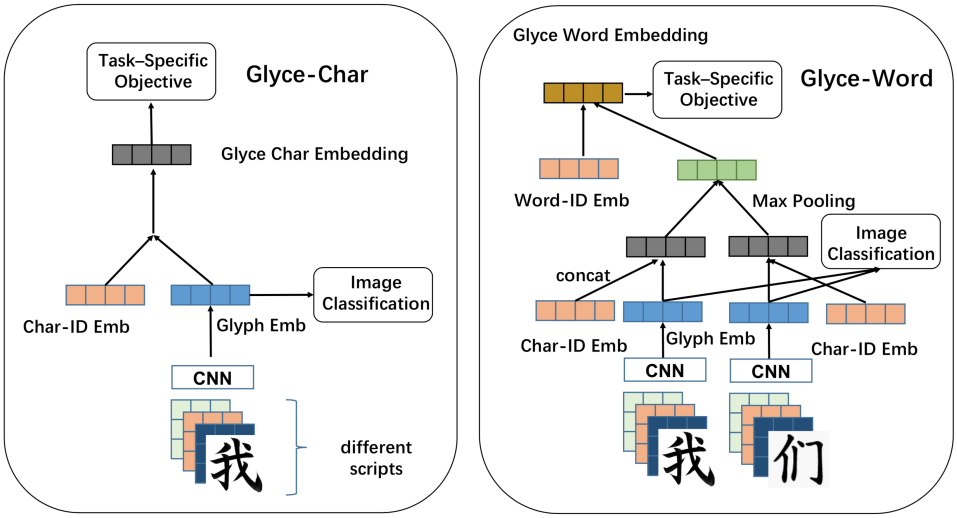

Figure 5: An overview of the Glyce character embedding and the Glyce word embedding.

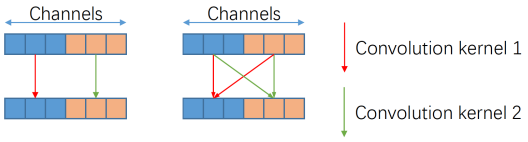

Figure 6: The mechanism of group convolution.

d=2048		
model	ppl	# parameter
charID	52.86	51.6M
d=1024		
model	ppl	# para
charID	53.87	17.4M
glyph (1 script, no img cls)	53.70	13.9M
glyph (8 script, no img cls)	172.56	19.2M
glyph (1 script, with img cls)	53.40	18.3M
glyph (8 script, with img cls)	53.10	23.6M
charID+glyph (1 script, no img cls)	51.64	24.6M
charID+glyph (8 script, no img cls)	391.29	30.1M
charID+glyph (1 script, img cls)	51.38	29.0M
charID+glyph (8 script, img cls)	**50.67**	34.5M

Table 3: Results for the character language modeling task, along with the number of parameters for each model.

which will further be combined with wordID embeddings to obtain Glyce-word embeddings using concatenation, highway networks (Srivastava et al., 2015) or another fully connected layer.

4 Experimental Results

In this section, we present results on a variety of Chinese NLP tasks,

4.1 Task1: Char-Level Language Modeling

The first task is character-level language modeling, in which we need to predict the forthcoming character given previous characters. We evaluate our model on the Chinese Tree-Bank 6.0 (CTB6). We followed the commonly adopted protocols, by which the dataset was dataset into 80%, 10%, 10% for training, dev and test. The dataset consists of 4,401 distinct Chinese characters. We use the LSTM structure, and tuned the number of layers, dropout rate, batch-size and learning rate using grid search for all settings.

Table 3 reports perplexity for each model, along with the number of parameters. *charID* means that we only charID embeddings. *glyph* means char embeddings are built only using glyph embeddings. *charID + glyph* means we use both. *1 script* means only using the simplified Chinese script, while *8 script* means using historical scripts and different writing styles, as discussed in Section 3.2.

We can see that the best glyph model outperforms the char model. For the d=1,024 setting, *glyph, 1 script, no img cls* outperforms the best charID setting, even though it has significantly fewer parameters (since the glyph model does not need to maintain a $D \times V$ look-up char embedding matrix). This validates that the glyph of Chinese character encodes significant levels of semantic information, and that as long as we use a proper CNN structure, glyph-based models outperform charID-based models. We observe progressive performance boost by adding the auxiliary image-classification objective and historical scripts. Interestingly, the image-classification objective is essential for the 8 script model, without which the value of ppl skyrockets. The explanation is intuitive: the image-classification objective pushes the CNN model to extract shared pictographic features

from different scripts, which is essential in understanding character semantics.

Combining charID embeddings with glyph embeddings gives additional performance boost, pushing the Ppl value down to 50.67 when combined with 8 historical scripts and the auxiliary image-classification objective. It is also worth noting that the best Glyce setting is also significantly better than the larger charID model with $d = 2,048$.

4.2 Task2: Word-Level Language Modeling

We use the same Chinese Tree-Bank 6.0 dataset for word-level language modeling evaluation. Train/dev/test sets are the same as the char-level seting. We use Jieba,[5] the most widely-used open-sourced Chinese word segmentation system to segment the dataset. The segmented dataset consists of about 50,000 words and we replaced words that appear less than 2 times as an UNK token. LSTM models are run on the word-level representations to predict following words. Different settings differ in how word embeddings are computed:

- wordID: Each word is associated with an embedding based on its unique wordID.
- charID: Word embeddings are computed based on the embeddings of its constituent characters, and no wordID embeddings are maintained.
- glyph: Similar to charID, but char embeddings are computed using glyph information rather than charID embeddings.
- wordID+charID: Word embeddings are computed based on its corresponding wordID embedding and constituent charID embeddings.
- wordID+glyph: Word embeddings are computed based on its corresponding wordID embedding and constituent char embeddings from glyph.
- wordID+charID+glyph: Word embeddings are computed based on its corresponding wordID embedding, constituent char embeddings from charIDs, and char embeddings from glyph.

Vector dimension is fixed to 512 for all models. Results are shown in Table 4. As can be seen, the glyph model, which only uses glyph information for characters, significantly outperforms the wordID model and the CharID model, and amazingly the wordID+charID model. Interestingly, wordID+glyph yields the best performance, better than wordID+charID+glyph. This show that the glyph information already provides significant (and enough) semantic information for the word-level language modeling task.

model	ppl	# parameter
wordID	199.9	28.6M
charID	193.0	18.9M
glyph	181.0	19.2M
wordID+charID	188.4	32.4M
wordID+glyph	175.1	38.5M
wordID+charID+glyph	176.0	36.0M

Table 4: Ppls for word-level language modeling.

4.3 Task3: Name Entity Recognition

For the task of Chinese NER, we used the widely-used OntoNotes, MSRA and resume datasets. Since most datasets don't have gold-standard segmentation, the task is normally treated as a char-level tagging task: outputting an NER tag for each character. The current SOTA tagging model is based on Lattice-LSTMs (Zhang and Yang, 2018), achieving better performances than CRF+LSTM (Ma and Hovy, 2016). We thus use the publicly available code for Lattice-LSTMs as a backbone,[6] replaced charID embeddings with Glyce-char embeddings, and followed the data splits, model setup and training criteria in Zhang and Yang (2018). Results are given as follows:

OntoNotes			
Model	P	R	F
CRF-LSTM	74.36	69.43	71.81
Lattice-LSTM	76.35	71.56	**73.88**
Lattice-LSTM+Glyce	82.06	68.74	**74.81** (+0.93)
MSRA			
Dong et al. (2016)	91.28	90.62	90.95
CRF-LSTM	92.97	90.80	91.87
Lattice-LSTM	93.57	92.79	**93.18**
Lattice-LSTM+Glyce	93.86	93.92	**93.89** (+0.71)
resume			
CRF-LSTM	94.53	94.29	94.41
Lattice-LSTM	94.81	94.11	**94.46**
Lattice-LSTM+Glyce	95.72	95.63	**95.67** (+1.21)

As can be seen, the Glyce-char model achieves new SOTA results for all of the three datasets: it outperforms currently SOTA results by 0.93, 0.71 and 1.21 respectively on the OntoNotes, MSRA and resume datasets.

[5] https://github.com/fxsjy/jieba

[6] thttps://github.com/jiesutd/LatticeLSTM.

4.4 Task4: Chinese Word Segmentation

The task of Chinese word segmentation (CWS) is normally treated as a char-level tagging problem. We used the widely-used CTB6, PKU and Weibo benchmarks for evaluation. Current SOTA results are also based on Lattice-LSTMs (Yang et al., 2018), and we used their publicly available code as a backbone,[7] replaced charID embeddings with Glyce-char embeddings, and exactly followed the data splits, model setup and training criteria.[8] Results are shown as follows:

Models	CTB6	PKU	Weibo
Zhou et al. (2017)	96.2	96.0	-
Yang et al. (2017)	96.2	**96.3**	**95.5**
Lattice+Word	**96.3**	95.9	95.1
Lattice+Subword	96.1	95.8	95.3
Lattice+Glyce-Char	**96.6** (+0.3)	**96.3** (+0.0)	**96.0** (+0.5)

As can be seen, the Glyce-char model sets new SOTA performances on the CTB6 and Weibo datasets, and matches the SOTA performance on the PKU dataset. From the statistics above, it is worth noting that Glyce-char is the only model that is able to achieve/match SOTA results across all datasets.

4.5 Task5: Part of Speech Tagging

Proposed by Ng and Low (2004), current POS models for Chinese predict the segmentation boundaries and POS tags simultaneously at the character level. Current SOTA models are based on Bidirectional RNN-CRF at the char-level (Shao et al., 2017). We thus use their publicly available code[9] as a backbone, replaced the char embeddings with Glyce-char embeddings, and followed data splits, model setup and training criteria in Shao et al. (2017). We use the CTB5, CTB6 and UD1 (Universal Dependencies) benchmarks to test our models. Results for different datasets are shown as follows:.

Models	CTB5	CTB6	UD1
Ensemble			
Shao et al. (2017)(ensemble)	94.38	-	89.75
Single			
Shao et al. (2017)(single)	**94.07**	90.81	89.41
Lattice+Glyce-Char	**95.61** (+1.54)	**91.34** (+0.53)	**90.77** (+1.36)

As can be seen, the single Glyce-word model output previous SOTA result from single model by 1.54 and 1.36 on the CTB5 and UD1 datasets. Additionally, the single Glyce-word model even outperforms the ensemble model in Shao et al. (2017), achieving new SOTA results fo 95.61 and 90.77 on the two datasets.

4.6 Task6: Dependency Parsing

For dependency parsing (Chen and Manning, 2014; Dyer et al., 2015), we used the widely-used Chinese Penn Treebank 5.1 dataset for evaluation. Our implementation uses the previous state-of-the-art Deep Biaffine model (Dozat and Manning, 2016) as a backbone. We replaced the word vectors from the biaffine model with Glyce-word embeddings, and exactly followed its model structure and training/dev/test split criteria. Results for unlabeled attachment score (UAS) and labeled attachment score (LAS) are given as follows:

Model	UAS	LAS
Ballesteros et al. (2016)	87.7	86.2
Kiperwasser and Goldberg (2016)	87.6	86.1
Cheng et al. (2016)	88.1	85.7
Biaffine	**89.3**	**88.2**
Biaffine+Glyce-Word	90.2 (+0.9)	89.0 (+0.8)

Results for previous models are copied from Dozat and Manning (2016); Ballesteros et al. (2016); Cheng et al. (2016). Glyce-word pushes SOTA performances up by +0.9 and +0.8.

4.7 Task7: Semantic Role Labeling

For the task of semantic role labeling (SRL) (Roth and Lapata, 2016; Marcheggiani and Titov, 2017; He et al., 2018), we used the CoNLL-2009 shared-task. We used the current SOTA model, the k-order pruning algorithm (He et al., 2018) as a backbone,[10] and replaced word embeddings with Glyce-word embeddings. We exactly followed the data splits, model setup and training criteria. Results are given as follows:

[7] https://github.com/jiesutd/SubwordEncoding-CWS

[8] Using bi-directional stacking LSTMs – feeding outputs from forward LSTMs to backward LSTMs – Ma et al. (2018a) reported higher performances in CWS tasks than Lattice-LSTMs, but didn't open-source pertinent code to the moment when this paper is published. Based on email interactions, authors of the paper, suggested (1) obtaining pre-trained vectors using wang2vec with the following parameters: -window 5 -negative 10 -sample 1e-4 -size 64; and (2) using a _outside_ symbol to denote the bigram at the last character and UNK to denote unseens biagrams, the embedding of which are learned during training. We followed these suggestions and comprehensively tuned the parameters for neural net models. But the best performance we are able to achieve is still around 2-3 percent off the reported results. We conjecture that this might be due to the corpus used to pre-train the embeddings, and this corpus is not publicly released. We thus do not compare our results with Ma et al. (2018a), and consider results from Yang et al. (2018) as currently SOTA.

[9] https://github.com/yanshao9798/tagger

[10] Code opensourced at https://github.com/bcmi220/srl_syn_pruning

Model	P	R	F
Roth and Lapata (2016)	76.9	73.8	75.3
Marcheggiani and Titov (2017)	**84.6**	80.4	82.5
k-order pruning (He et al., 2018)	84.2	**81.5**	**82.8**
k-order pruning+Glyce-word	**85.4** (+0.8)	**82.1** (+0.6)	**83.7** (+0.9)

The results are copied from Roth and Lapata (2016); Marcheggiani and Titov (2017); He et al. (2018). The proposed Glyce model outperforms the previous SOTA performance by 0.9 F1 score, achieving a new SOTA score of 83.7.

4.8 Task8: Sentence Semantic Similarity

For the task of sentence semantic similarity, we used the BQ corpus (Chen et al., 2018). The BQ corpus contains 120,000 Chinese sentence pairs, and each pair is associated with a label indicating whether the two sentences are of equivalent semantic meanings. One can think the BQ corpus as a Chinese version of SNIL (Bowman et al., 2015), the construction of which is to capture whether one sentence paraphrases the other. The dataset is deliberately constructed in a way that for some pairs, its constituent two sentences have most of words overlapped but their meanings are completely different, while for other pairs, sentences share very few word overlaps but with equivalent meanings.

The task is transformed to a binary classification problem. The current SOTA model is based on the bilateral multi-perspective matching model (BiMPM) (Wang et al., 2017) which specifically tackles the matching problem between sentences. BiMPM significantly outperforms CNNs and LSTMs on sentence-matching tasks. Again, we exactly followed the setup from Chen et al. (2018), with the only difference being replacing the input embedding with Glyce embedding. The original SOTA result is based on characters rather than words. We thus use Glyce-char embeddings.[11] Results for different models are given as follows:

Models	P	R	F	A
BiLSTM	75.04	70.46	72.68	73.51
DIIN	81.58	81.14	81.36	81.41
BiMPM	82.28	81.18	**81.73**	**81.85**
BiMPM+Glyce	81.93 -	85.54 -	**83.70** (+1.97)	**83.34** (+1.49)

DIIN results are copied from Gong et al. (2017), BiLSTM results and BiMPM (SOTA) are copied from Chen et al. (2018). As can be seen, the proposed Glyce model outperforms the previous SOTA performance by +1.97, achieve a new SOTA score of 83.70 on the BQ corpus.

[11] Char embeddings actually outperform word embeddings on the BQ task.

4.9 Task9: Intention Identification

The Large-scale Chinese Question Matching Corpus (LCQMC) (Liu et al., 2018) aims at identifying whether two sentences have the same intention. This task similar to but not exactly the same as the paraphrase task in BQ: two sentences can have different meanings but share the same intent. For example, meanings for "*My phone is lost*" and "*I need a new phone*" are different, but their intentions are the same: buying a new phone. Each pair of sentences in the dataset is associated with a binary label indicating whether the two sentences share the same intention, and the task can be formalized as predicting this binary label. The setting of LCQMC is similar to that of BQ, and the model needs to capture the matching between two sequences. Again, current state-of-the-art results are from the bilateral multi-perspective matching model (BiMPM). We follow the setup in Liu et al. (2018) and train a BiMPM model using Glyce-char embeddings. Performances are given as follows:

Models	P	R	F	A
BiLSTM	70.6	89.3	78.9	76.1
BiMPM	77.6	93.9	**85.0**	**83.4**
BiMPM+Glyce	80.4 -	93.4 -	**86.4** (+1.4)	**85.3** (+1.9)

The BiLSTM results and the SOTA results for BiMPM are copied from Liu et al. (2018). As can be seen, the proposed Glyce model outperforms the previous SOTA performance by 1.4 for F1 and 1.9 for accuracy.

4.10 Task10: Chinese-English Machine Translation

Since Glyce helps sentence encoding, we believe that it will improve the encoding process of MT, which will consequently improve Ch-En translation performances. We use the standard Ch-En setting, and exactly followed the common setup adopted in Ma et al. (2018c); Chen et al. (2017a); Li et al. (2017); Zhang et al. (2018). The training set consists of 1.25M sentence pairs extracted from LDC corpora.[12] Dev set is from NIST 2002 and models are evaluated on NIST 2003, 2004, 2005, 2006 and 2008. The current state-of-the-art setting is from Ma et al. (2018c), which uses both the sentences (seq2seq) and the bag-of-words as targets in the training stage. We used their publicly available code as a backbone,[13] exactly followed its training

[12] LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06

[13] https://github.com/lancopku/bag-of-words

criteria and replaced word embeddings at the encoding time with Glyce-word vectors. We report BLEU scores (Papineni et al., 2002) for different models as follows:

TestSet	Seq2Seq +Attn	Seq2Seq +Attn+BOW	Glyce+Seq2Seq +Attn+BOW
MT-02	34.71	39.77	**40.56** (+0.79)
MT-03	33.15	38.91	**39.93** (+1.02)
MT-04	35.26	40.02	**41.54** (+1.52)
MT-05	32.36	36.82	**38.01** (+1.19)
MT-06	32.45	35.93	**37.45** (+1.52)
MT-08	23.96	27.61	**29.07** (+1.46)
Average	31.96	36.51	**37.76** (+1.25)

The seq2seq+attention results and the current SOTA results (Seq2Seq+BOW) are copied from Ma et al. (2018c). Glyce achieves a new SOTA record on the Ch-En translation benchmark, achieving a BLEU score of 37.76.

4.11 Task11: Sentiment Analysis

The task of sentiment analysis (Socher et al., 2013; Pang et al., 2002) is usually formalized as a binary or multi-class classification problem, assigning sentiment labels to text inputs. Currently widely used Chinese sentiment benchmarks include (1) Dianping: Chinese restaurant reviews crawled from the online review website dazhong dianping (similar to Yelp). Training set consists of 2,000,000 reviews and test set consists of 500,000 reviews. The data is processed in a way that 4 and 5 star reviews belong to the positive class while 1-3 star reviews belong to the negative class, making the task a binary-classification problem; (2) JD Full: shopping reviews in Chinese crawled from JD.com for predicting full five stars, making the task a five-class classification problem. Training set consists of 3M reviews and test set consists of 250,000 reviews; (3) JD binary: shopping reviews from JD.com with 4-and-5 star reviews belonging to the positive class and 1-and-2 to the negative class. 3-star reviews are ignored. Training set consists of 4M reviews and test set consists of 360,000 reviews. The datasets were first provided in Zhang and LeCun (2017). For our implementation, we trained a simple Bi-directional LSTM based on Glyce-char embeddings. We report the accuracy for the Glyce-char model with different baselines in the table below:

Model	Dianping	JD Full	JD Binary
char N-gram	76.41	51.82	91.08
word N-gram	76.97	51.70	91.18
char-EmbedNet	76.40	51.71	90.59
word-EmbedNet	75.45	49.95	89.63
char-fastText	**77.66**	**52.01**	**91.28**
word-fastText	77.38	51.89	90.89
Glyce-Char	**78.46** (+0.80)	**54.24** (+2.23)	**91.76** (+0.58)

The results for char N-gram (regression on char N-gram features), word N-gram (regression on word N-gram features), char-EmbedNet (CNN on chars), word-EmbedNet (CNN on words), char-fasttext (fasttext (Joulin et al., 2016) on chars) and word-fastText (fasttext on words) are directly copied from Zhang and LeCun (2017), and we picked the model with best reported performances. Using Glyce-char models, we are able to set new SOTA results on all of the three sentiment analysis datasets.

4.12 Task12: Document Classification

The setup for document classification tasks is similar to sentiment analysis, in which a label indicating document genre is to be assigned to each input document. Prevalent datasets include (1) the Fudan corpus (Li, 2011). Following Xu et al. (2016); Cao et al. (2018), the dataset consists of 1218 environmental, 1022 agricultural, 1601 economical, 1025 political and 1254 sport documents; 70% of the total data is used for training and the rest for testing. (2) Ifeng: First paragraphs of Chinese news articles from 2006-2016. The dataset consists of 5 news categories. Training set is made up with 800,000 documents and test set contains 50,000 documents; (3) ChinaNews: Chinese news articles split into 7 news categories. Training set consists of 1.4M documents and test set consists of 50,000 documents. For our implementation, we trained a simple Bi-directional LSTM based on Glyce-char embeddings. We report accuracy for different models in the table below:

Model	Fudan	Ifeng	ChinaNews
char N-gram	-	78.48	86.63
word N-gram	-	81.70	89.24
char-EmbedNet	-	82.99	89.45
word-EmbedNet	-	79.18	85.25
char-fasttext	-	**83.69**	**90.90**
word-fasttext	-	83.35	90.76
cw2vec	95.3	-	-
Glyce-Char	**96.3** (+1.0)	**85.76** (+2.07)	**91.88** (+0.98)

The results for char N-gram, word N-gram, char-EmbedNet, word-EmbedNet, char-fasttext and word-fasttext are directly copied from Zhang and LeCun (2017), and we picked the model with best

performances. Results for cw2vec are copied from Cao et al. (2018). Again, the proposed Glyce-char model achieves the SOTA results across all the three datasets.

4.13 Task13: Discourse Parsing

Discourse parsing (Ji and Eisenstein, 2014; Li et al., 2014a; Jiang et al., 2018; Chuan-An et al., 2018) aims at identifying how discourse units are connected with each other to form the discourse hierarchy. It can be thought similar to syntactic parsing but basic units are text segments (referred to as elementary discourse unit, EDU for short) rather than words. The most commonly used benchmarkjis the Chinese Discourse Treebank (CDTB) dataset, which was constructed by Li et al. (2014b). Based on golden EDUs, the task consists of two stages: (1) discourse tree construction: hierarchically merging discourse units (EDUs) in the bottom-up fashion. This can further be transformed to a binary classification problem of deciding whether two neighboring EDUs should be merged; and (2) discourse relation labeling: identifying the relation between two EDUs that were joined in the tree construction process, which is formed as a multi-class classification problem. At the test time, a discourse tree is constructed using the CKY algorithm.

We followed the previous SOTA RvNN model (Chuan-An et al., 2018), which first maps each EDU to a vector representation using LSTMs. Representations for non-leaf nodes during tree construction process are constructed using treeLSTMs (Li et al., 2015a; Tai et al., 2015). We replaced the charID embeddings with Glyce-char embeddings, and followed the protocols defined in Chuan-An et al. (2018). For evaluation, we report results for parse tree construction (Structure), parse tree construction with sense labeling (+Sense), parse tree construction with center labeling (+Center), and parse tree construction with both sense and center labeling (Overall).

Result	RvNN	RvNN+Glyce
Structure	64.6	**67.9** (+3.3)
+Sense	42.7	**45.1** (+2.4)
+Center	38.5	**41.2** (+2.7)
Overall	35.0	**37.8** (+2.8)

The SOTA results are copied from Chuan-An et al. (2018), and using Glyce we are able to achieve new SOTA results for all of the four evaluation values.

5 Conclusion

In this paper, we propose the GLYCE: the Glyph-vectors for Chinese Character Representations. Glyce treats Chinese characters as images and uses Tianzige-CNN to extract character semantics and build representations. For better semantic modeling, we 1) use historical Chinese scripts; 2) design a Tianzige-CNN structure which tailored to Chinese character images; and 3) use image classification as an auxiliary task objective. With Glyce, we are able to set new SOTA results for almost all Chinese NLP tasks.

Logographic languages like Chinese or Japanese are very different from alphabetic languages like English or French. Unfortunately, most previous efforts for processing Chinese are borrowed from methods experimented on English. Glyce suggests some avenues for fruitful research in NLP tasks with logographic languages such as Chinese.

Acknowledgement

We want to gratefully thank Xinlei Chen for significantly helpful discussions in developing the original idea for Glyce in 2014. We also want to thank Prof. Dan Jurafsky for providing helpful suggestions and comments.

For many tasks, Glyce embeddings are incorporated into previous SOTA models. We thus want to thank those who open-sourced their SOTA model code, which dramatically reduced our work to reproduce previous SOTA performances. Specifically, we want to thank Yang Jie and Yue Zhang for open-sourcing their Lattice LSTM code for NER (Zhang and Yang, 2018) and CWS (Yang et al., 2018); Yan Shao (Shao et al., 2017) for open-sourcing the bidirectional RNN-CRF model for POS tagging; Shuming Ma (Ma et al., 2018c) for open-sourcing the code for the seq2seq+bag-of-words model for Ch-En MT; Shexia He (He et al., 2018) for releasing the code for the k-order pruning algorithm for semantic role labeling; Zhiguo Wang (Wang et al., 2017) for releasing the code for BiPMP.

Last but by means the least, special thanks are owed to the infrastructure team at Shannon.AI, including Shibo Tao, Honglei Yan and many others, for spending countless days and nights providing infra support for Glyce, without whom this work is absolutely impossible.

References

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A Smith. 2016. Training with exploration improves a greedy stack-lstm parser. *arXiv preprint arXiv:1603.03793*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for chinese. *arXiv preprint arXiv:1606.04300*.

Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Yongjian Wu, and Feiyue Huang. 2017. Fast and accurate neural word segmentation for chinese. *arXiv preprint arXiv:1704.07047*.

Shaosheng Cao, Wei Lu, Jun Zhou, and Xiaolong Li. 2018. cw2vec: Learning chinese word embeddings with stroke n-gram information.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017a. Improved neural machine translation with a syntax-aware encoder and decoder. *arXiv preprint arXiv:1707.05436*.

Jing Chen, Qingcai Chen, Xin Liu, Haijun Yang, Daohe Lu, and Buzhou Tang. 2018. The bq corpus: A large-scale domain-specific chinese corpus for sentence semantic equivalence identification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4946–4951.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015a. Long short-term memory neural networks for chinese word segmentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1197–1206.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017b. Adversarial multi-criteria learning for chinese word segmentation. *arXiv preprint arXiv:1704.07556*.

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huan-Bo Luan. 2015b. Joint learning of character and word embeddings. In *IJCAI*, pages 1236–1242.

Hao Cheng, Hao Fang, Xiaodong He, Jianfeng Gao, and Li Deng. 2016. Bi-directional attention with agreement for dependency parsing. *arXiv preprint arXiv:1608.02076*.

Lin Chuan-An, Hen-Hsen Huang, Zi-Yuan Chen, and Hsin-Hsi Chen. 2018. A unified rvnn framework for end-to-end chinese discourse parsing. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 73–77.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147*.

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

Falcon Z Dai and Zheng Cai. 2017. Glyph-aware embedding of chinese characters. *arXiv preprint arXiv:1709.00028*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee.

Chuanhai Dong, Jiajun Zhang, Chengqing Zong, Masanori Hattori, and Hui Di. 2016. Character-based lstm-crf with radical-level features for chinese named entity recognition. In *Natural Language Understanding and Intelligent Applications*, pages 239–250. Springer.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 960–970.

Yichen Gong, Heng Luo, and Jian Zhang. 2017. Natural language inference over interaction space. *arXiv preprint arXiv:1709.04348*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2016. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2061–2071.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 13–24.

Feng Jiang, Sheng Xu, Xiaomin Chu, Peifeng Li, Qiaoming Zhu, and Guodong Zhou. 2018. Mcdtb: A macro-level chinese discourse treebank. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3493–3504.

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

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*, pages 2741–2749.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *arXiv preprint arXiv:1603.04351*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Jiwei Li, Rumeng Li, and Eduard Hovy. 2014a. Recursive deep models for discourse parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2061–2069.

Jiwei Li, Minh-Thang Luong, Dan Jurafsky, and Eudard Hovy. 2015a. When are tree structures necessary for deep learning of representations? *arXiv preprint arXiv:1503.00185*.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. *arXiv preprint arXiv:1705.01020*.

Ronglu Li. 2011. Fudan corpus for text classification.

Yancui Li, Fang Kong, Guodong Zhou, et al. 2014b. Building chinese discourse corpus with connective-driven dependency tree structure. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2105–2114.

Yanran Li, Wenjie Li, Fei Sun, and Sujian Li. 2015b. Component-enhanced chinese character embeddings. *arXiv preprint arXiv:1508.06669*.

Frederick Liu, Han Lu, Chieh Lo, and Graham Neubig. 2017. Learning character-level compositionality with visual features. *arXiv preprint arXiv:1704.04859*.

Xin Liu, Qingcai Chen, Chong Deng, Huajun Zeng, Jing Chen, Dongfang Li, and Buzhou Tang. 2018. Lcqmc: A large-scale chinese question matching corpus. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1952–1962.

Minh-Thang Luong and Christopher D Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. *arXiv preprint arXiv:1604.00788*.

Ji Ma, Kuzman Ganchev, and David Weiss. 2018a. State-of-the-art chinese word segmentation with bi-lstms. *arXiv preprint arXiv:1808.06511*.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. 2018b. Shufflenet v2: Practical guidelines for efficient cnn architecture design. *arXiv preprint arXiv:1807.11164*, 5.

Shuming Ma, Xu Sun, Yizhong Wang, and Junyang Lin. 2018c. Bag-of-words as target for neural machine translation. *arXiv preprint arXiv:1805.04871*.

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

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. *arXiv preprint arXiv:1703.04826*.

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

Hwee Tou Ng and Jin Kiat Low. 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*.

Minh Nguyen, Gia H Ngo, and Nancy F Chen. 2018. Multimodal neural pronunciation modeling for spoken languages with logographic origin. *arXiv preprint arXiv:1809.04203*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. *arXiv preprint arXiv:1605.07515*.

Yan Shao, Christian Hardmeier, Jörg Tiedemann, and Joakim Nivre. 2017. Character-based joint segmentation and pos tagging for chinese using bidirectional rnn-crf. *arXiv preprint arXiv:1704.01314*.

Xinlei Shi, Junjie Zhai, Xudong Yang, Zehua Xie, and Chao Liu. 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, pages 594–598.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Yaming Sun, Lei Lin, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2014. Radical-enhanced chinese character embedding. In *International Conference on Neural Information Processing*, pages 279–286. Springer.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Mi Xue Tan, Yuhuang Hu, Nikola I Nikolov, and Richard HR Hahnloser. 2018. wubi2en: Character-level chinese-english translation through ascii encoding. *arXiv preprint arXiv:1805.03330*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*.

Jian Xu, Jiawei Liu, Liangang Zhang, Zhengyu Li, and Huanhuan Chen. 2016. Improve chinese word embeddings by exploiting internal structure. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1041–1050.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural word segmentation with rich pretraining. *arXiv preprint arXiv:1704.08960*.

Jie Yang, Yue Zhang, and Shuailong Liang. 2018. Subword encoding in lattice lstm for chinese word segmentation. *arXiv preprint arXiv:1810.12594*.

Rongchao Yin, Quan Wang, Peng Li, Rui Li, and Bin Wang. 2016. Multi-granularity chinese word embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 981–986.

Jinxing Yu, Xun Jian, Hao Xin, and Yangqiu Song. 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*, pages 286–291.

Jiacheng Zhang, Yang Liu, Huanbo Luan, Jingfang Xu, and Maosong Sun. 2018. Prior knowledge integration for neural machine translation using posterior regularization. *arXiv preprint arXiv:1811.01100*.

Ting Zhang, Guo-Jun Qi, Bin Xiao, and Jingdong Wang. 2017. Interleaved group convolutions. In *Computer Vision and Pattern Recognition*.

Xiang Zhang and Yann LeCun. 2017. Which encoding is the best for text classification in chinese, english, japanese and korean? *arXiv preprint arXiv:1708.02657*.

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

Yue Zhang and Jie Yang. 2018. Chinese ner using lattice lstm. *arXiv preprint arXiv:1805.02023*.

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.

Hao Zhou, Zhenting Yu, Yue Zhang, Shujian Huang, XIN-YU DAI, and Jiajun Chen. 2017. Word-context character embeddings for chinese word segmentation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 760–766.