Learning Distributional Token Representations from Visual Features

Samuel Broscheit   **Rainer Gemulla**   **Margret Keuper**
University of Mannheim, Mannheim, Germany
{lastname}@informatik.uni-mannheim.de

Abstract

In this study, we compare token representations constructed from visual features (i.e., pixels) with standard lookup-based embeddings. Our goal is to gain insight about the challenges of encoding a text representation from low-level features, e.g. from characters or pixels. We focus on Chinese, which—as a logographic language—has properties that make a representation via visual features challenging and interesting. To train and evaluate different models for the token representation, we chose the task of character-based neural machine translation (NMT) from Chinese to English. We found that a token representation computed only from visual features can achieve competitive results to lookup embeddings. However, we also show different strengths and weaknesses in the models' performance in a part-of-speech tagging task and also a semantic similarity task. In summary, we show that it is possible to achieve a *text representation* only from pixels. We hope that this is a useful stepping stone for future studies that exclusively rely on visual input, or aim at exploiting visual features of written language.

1 Introduction

Language representation beyond the word level can be advantageous for words from the tail of the distribution, as has been shown in recent neural approaches for various tasks (Schütze, 2017; Kim et al., 2016; Lee et al., 2017; Wu et al., 2016; Sennrich et al., 2016). In these approaches, a neural model represents an input text based on its sequence of characters or character n-grams (instead of its words). This helps the model to handle out-of-vocabulary tokens and avoids the need of text segmentation or tokenization, which remains an unsolved problem for many languages. For some applications, e.g. language processing for social media, models should be able to capture the creative use of language. However, a disadvantage is that we loose the certainty of a known vocabulary. Also, regarding computational complexity, there is a trade-off between the memory that is required for large embedding lookup tables and the additional computational cost that is required to compose representations from low-level features.

In contrast to the characters of languages based on the Latin alphabet, the Chinese written language is defined over a set of ≈ 8000 characters that already carry meaning. The characters can either appear in a traditional or simplified form, and many share visual components that can indicate a related meaning. Thus, it is reasonable to hypothesize that encoding Chinese characters directly from their visual components might improve their token representation in a neural network model.

In recent studies, Liu et al. (2017) evaluated character encodings from visual features by classifying Wikipedia titles into 12 categories and Su and Lee (2017) evaluated such encodings by measuring correlation with human similarity judgments. Both studies found that using character encodings from visual input did not outperform or were equal to lookup-based embeddings. No support for the hypothesis above is thus provided.

In this study, we aim to explore the question of whether and when visual-feature representations are beneficial or detrimental. We make the following contributions: (i) Since the evaluation tasks from prior studies did not test the capabilities of the visual features for *text representation*, we propose to employ the task of neural machine trans-

Proceedings of the 3rd Workshop on Representation Learning for NLP, pages 1–8
Melbourne, Australia, July 20, 2018.

lation (NMT) for training and evaluation. We argue that NMT requires the token representation to serve as a reliable syntactic and semantic signal. (ii) Prior work reported evaluations for one architecture to encode the visual features. In this paper, we evaluate different settings and argue for architecture choices that performed well in our experiments. (iii) We provide evidence for the possibility to compute token representations from visual features that perform on-par with lookup-based embeddings in NMT. (iv) Finally, we revisit two of the tasks from prior work that use visual features: measuring correlation with semantic similarity judgments by humans as well as joint segmentation and part-of-speech tagging. We use the best models from the NMT evaluation in these two tasks. For semantic similarity, we find that token representation from pixels are clearly beneficial for unseen characters, while a lookup embedding performs better for seen characters. For joint segmentation and part-of-speech tagging we find no clear difference between lookup embeddings and character representation from pixels.

2 Related Work

We start by summarizing prior work, in which token representations were obtained from bitmaps of Chinese characters.

Su and Lee (2017) propose several embedding models for Chinese, including one from visual features. For the evaluation they create a dataset with Chinese word pairs in traditional Chinese. In their evaluation, they measure the correlation of the cosine similarity of embedded word pairs against human similarity judgments for those word pairs. In their experiments, they found that the representation from visual features was not competitive to an embedding lookup model, but a combination of lookup embeddings with visual features was. For their model, they train a 5-layer CNN auto-encoder on bitmaps of Chinese characters. The auto-encoder character representation is fed into two GRU layers and two fully connected layers with ELU activation to encode the characters into a word. The model was trained to predict the Skip-Gram objective (Mikolov et al., 2013) or the Glove objective (Pennington et al., 2014).

Liu et al. (2017) evaluate the classification of Chinese Wikipedia titles into 12 categories. They found that the representation from visual features representations was not competitive to embedding lookup models. They did find, however, that a combination of lookup-based embeddings and visual features can be beneficial. Their character encoder is a convolutional neural network (CNN). The CNN consists of 3 convolutional layers with max-pooling followed by a fully connected transformation layer with ReLU activation. The character encodings are fed as input to a recurrent neural network (RNN) that encodes the Wikipedia title. Both, character encoder and classifier were trained jointly.

Costa-jussà et al. (2017) investigate the neural machine translation from Chinese to Spanish. To investigate the helpfulness of visual features they augmented lookup embeddings by concatenating the corresponding bitmap features of the characters. They showed that this approach improved the performance of both, their character-based and their word-based NMT system. They did not study the exclusive use of visual features.

Shao et al. (2017) investigate joint part-of-speech tagging and segmentation for Chinese. In contrast to the studies mentioned above, they found that the inclusion of visual features did not help. However, in their study, the CNN encoder was not pre-trained and only a small amount of training data was available, which may explain this finding. In their approach, they augmented character and n-gram embeddings with visual features. The model is a 2-layer CNN with max-pooling and a consecutive fully-connected transformation layer. They concatenated the CNN-based character embeddings to the lookup embeddings and fed them to a BI-LSTM-CRF.

Other related approaches with regards to encoding symbols from visual features are: Deng et al. (2017) proposed to translate from images to a markup language; e.g.; images of mathematical expressions to LaTeX code. While their system performed remarkably well on those regular languages, they did not discuss their approach in the context of natural language. Remotely related is optical character recognition (OCR) and multi-model machine translation. While OCR could benefit from this study, its main goal is character recognition under noisy conditions. Multi-modal machine translation augments a main task with additional features or when the translation is from a non-textual source, e.g. images, to text (Elliott et al., 2017). In contrast, our goal is to study text representation from low-level features.

Model	transformation	activation	transformation	activation
CNN+FC+ReLU	CNN(484 → F),	BN+ReLU	FC(F → 512)	BN+ReLU
CNN+SM+FC	CNN(484 → F)	Softmax	FC(F → 512)	
CNN+ReLU	CNN(484 → 512)	BN+ReLU		
FC 1L	FC(484 → 512)			
FC 2L	FC(484 → 512)	BN+ReLU	FC(512 → 512)	

Table 1: Character encoder architectures. Abbreviations: FC = fully connected, SM = softmax, BN = Batchnorm (Ioffe and Szegedy, 2015), F = CNN output size (see Table 2).

	in	out	kernel	stride
(0): Conv2d	1	32	(3, 3)	(1, 1)
(1): ReLU				
(2): MaxPool2d			(2, 2)	(2, 2)
(3): Conv2d	32	32	(3, 3)	(1, 1)
(4): ReLU				
(5): MaxPool2d			(2, 2)	(2, 2)
(6): Conv2d	32	F	(4, 4)	(1, 1)

Table 2: Configuration of the CNN layers. F is the CNN output feature size and takes values in $[256, 512, 1024, 2048, 4096]$.

3 Character Encoding

In this section, we first describe how a tokenized text is mapped to vector representations of its tokens. Then, we describe the character encoders that we have evaluated.

3.1 Preliminaries

Neural approaches that compute a function from text input commonly map each token t_i (e.g., a word) from the input text T to a dense vector representation $\mathbf{t}_i \in \mathbb{R}^d$ via some function $D(t_i) \rightarrow \mathbf{t}_i$. Lookup-based models associate each token of a fixed vocabulary V with its own dense representation (embedding). In particular, lookup-based models use (or learn) an embedding matrix $\mathbf{E} \in \mathbb{R}^{n \times d}$, where $n = |V|$. The j-th row of $\mathbf{E}$ holds the embedding of the j-th token in the vocabulary. Thus $D(t_i) = \mathbf{e}_j^T \mathbf{E}$, where j is the token number of token t_i in V and $\mathbf{e}_j$ denotes the j-th standard basis vector.

If we represent each character from its visual features—i.e., the pixels of its bitmap image—, we compute the embedding instead of performing a lookup in an embedding matrix. In particular, $D(t_i)$ computes a dense representation of token t_i from its pixels $p(t_i)$ via a character encoder C. Therefore, there is no fixed vocabulary of tokens any more. We have $D(t_i) = C(p(t_i))$.

3.2 Character Encoders

As discussed in Section 2, most of the prior related work—except Costa-jussà et al. (2017)—used CNNs to learn position-invariant features of the character image. In Table 2, we report the configuration of the convolutional layers that was used in our models. All studies report different numbers of layers and different numbers of CNN features, therefore, we vary the CNN feature size F as a hyper-parameter. In Table 1 we report the character encoder architectures, that roughly cover the architectures from previous work. After the CNN, Liu et al. (2017) and Shao et al. (2017) use fully connected layers (*CNN+FC+ReLU*), while Su and Lee (2017) directly feed the CNN features into a recurrent neural network encoder (*CNN+ReLU*). We also consider a similar setup as Costa-jussà et al. (2017) by including two settings with one or two fully connected layers, (*FC 1L* and *FC 2L*). In addition to previous work, we also evaluated a model architecture that computes a softmax activation over the image features (*CNN+SM+FC*). Our hypothesis is that the sparse activation from the softmax may act like a soft lookup (in an $F \times 512$ embedding matrix).

4 Experimental Study

In this section, we describe the results of our experimental study. First, we report how the character images were created, followed by the experiments for the neural machine translation task. To expand on these results, we describe experiments and results for joint segmentation and part-of-speech tagging, as well as for measuring correlation with semantic similarity judgments.

4.1 Character Images

We convert each character from the source vocabulary into a 22×22 binary representation of its glyph.[1] This was the lowest resolution that did not collapse nearby strokes into indistinguishable clusters.

4.2 Experiments for Machine Translation

For the machine translation experiments, we trained a NMT model to translate from Chinese to English. In the following sections, we describe the model, the data and the training settings, followed by the results.

NMT Model We used a standard sequence-to-sequence model with a recurrent encoder and attention-based decoder (Luong et al., 2015). This architecture does not represent the state of the art in neural machine translation. However, due to its wide adoption in empirical research, there is broad knowledge about suitable hyper-parameters, which makes it a preferred choice for our study.

The coarse architecture of this model can be described by an encoder enc and a decoder dec. The encoder enc is a function that takes a tokenized text T in a source vocabulary as input and computes a representation that is the input for the subsequent decoder. The decoder dec then creates a sequence of tokens in the target vocabulary. The final output is $\mathrm{dec}(\mathrm{enc}(T))$.

The input of the encoder enc is first transformed to a dense vector representation, i.e., each input token is transformed by function D of Section 3.1. In the following experiments, we evaluate different choices for D.

Data For training the translation model, we used a subset of the available data from the WMT 2017 Workshop on Machine Translation (Bojar et al., 2017),[2], namely the *News Commentary v12* corpus as well as *Casia2015* and *Neu2017* from the *CWMT Corpus*. Overall, these datasets yielded 3,277,330 sentences. For development and evaluation, we used WMT 2017 dev and test data, respectively.

Training For Chinese, we use characters as input. For English, we use byte pair encoding (Sennrich et al., 2016) with ≈32,000 symbols. We implemented the model in PyTorch (Version 0.3.1) (Paszke et al., 2017). The recurrent neural networks in the encoder and decoder are LSTMs (Hochreiter and Schmidhuber, 1997). Table 3 summarizes the hyper-parameters of the NMT model.

	Encoder	Decoder
Emb. size	512	512
Layers	bi-dir + uni-dir	uni-dir + uni-dir
Hid. size	512 + 1024	512 + 1024
Voc. size	8457	32413
Voc. type	character	byte pair enc.

Table 3: Hyperparameters of the sequence-to-sequence model for the NMT task.

For training, we used Adam (Kingma and Ba, 2014) with the standard parameters of PyTorch. We used a learning rate of 10^{-3} for 4 epochs, then we halved the learning rate every epoch until we reached 10^{-5}. For regularization, we used dropout with probability 0.2 in both encoder and decoder RNNs. We pretrained the decoder for 20 epochs to a perplexity of 108.21 on the validation data. This did result in: faster convergence, improved fluency of translations, and less variance of evaluation results. We trained all the models for 10 epochs. We selected the models by the best batch-based approximate BLEU score on the development set. This worked slightly better than selecting them by the best perplexity. We perform translation via beam search with a beam-size of 10 and length normalization of 0.9.

We trained the NMT models with each of the character encoders of Table 1 as well as with lookup-based embeddings (*EMB*) as a baseline.

Results Table 4 summarizes the results of the machine translation experiments. We include the result from a WMT 2017 system to give context for the expected BLEU score in this task. This result is from a baseline NMT system for Chinese to English (Wang et al., 2017), which is most comparable to our approach (i.e., no reranking, no ensembles, no special treatment of names and numbers). However, a crucial difference is that the WMT 2017 system uses pre-segmented text, which yields a vocabulary size of 300k on the

[1] We use ImageMagick's convert command together with the open source font NotoSansCJK-Regular.

[2] See http://www.statmt.org/wmt17/translation-task.html

[2] We use the BLEU implementation sacrebleu, https://github.com/awslabs/sockeye/tree/master/contrib/sacrebleu.

Model	F	BLEU
EMB	-	**16.63**
CNN+FC+ReLU	256	16.34
	512	16.33
	1024	**16.60**
	2048	16.40
	4096	16.20
CNN+SM+FC	512	15.85
	1024	15.49
	2048	15.36
	4096	**16.03**
CNN+ReLU	512	16.22
FC 1L	512	15.75
FC 2L	512	**16.32**
Word-segmented	-	19.4

Table 4: BLEU scores for translation zh-en on the WMT17 test data. Result for a word-segmented baseline model reported by Wang et al. (2017).

Model	full	low	mid	high
EMB	**16.63**	**15.60**	17.17	16.03
CNN+SM+FC	16.03	13.91	16.86	15.41
CNN+FC+R	**16.60**	14.47	17.11	**16.91**
FC 2L	16.32	15.10	**17.20**	16.37

Table 5: BLEU scores on the WMT17 test data for sentence buckets with low, medium and high frequency characters.

source side, while our system is character-based.

Our results indicate that using only one fully connected transformation layer *FC 1L* for character encoding is possible but does not yield comparable results to the best-performing convolutional architectures. However, the *FC 2L* comes close to the CNN models. Note that *CNN+SM+FC 4096* needs a larger number of CNN features to perform comparable to *CNN+FC+ReLU 1024*. In terms of BLEU score, the best character encoders perform equal to a standard lookup embedding.

To gain insight into the differences between the models, we split the test data into three buckets. Each sentence is scored with $1/n \sum_{i=1}^{n} \#t_i$, where $\#t_i$ is the training data frequency of character t_i. We partition the test data into a *low* and *high* bucket with the sentences scored in the lower and upper quartile, respectively, and a *mid* bucket with the remaining 2nd and 3rd quartile. Table 5 shows the results per bucket for *CNN+FC+ReLU 1024*, *CNN+SM+FC 4096*, *FC 2L* and *EMB*. Interestingly, the *CNN+FC+ReLU 1024* and *EMB* perform differently well in the frequency buckets. The *CNN+FC+ReLU 1024* model, surprisingly, performs better than *EMB* for high-frequency characters, while this is inversed for the low-frequency characters.

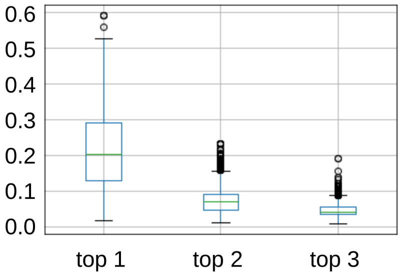

Figure 1: Distributions of the top-3 softmax activation magnitudes per character.

For the *CNN+SM+FC* encoders, our hypothesis was that the sparse activation of the softmax could act like a soft lookup function; i.e., it selects only few rows for the subsequent fully connected layer. We measured the top-3 softmax activation magnitudes per character from a random sample of 350 sentences from the training data (a total of 11234 words). As shown in Figure 1, the activations are indeed spiked, which supports our hypothesis. However, in our evaluation, we found no advantage over *CNN+FC+ReLU* encoders.

4.3 Experiments for Joint Part-of-Speech Tagging and Segmentation

In the translation experiment we evaluated the token representations for a syntactic *and* semantic signal. In this experiment, we want to investigate the *morpho-syntactic* information in the representations. To evaluate this, we employ the task of joint part-of-speech tagging and segmentation.

Model We used a Linear-Chain-CRF with a CNN encoder (Strubell et al., 2017) to compute the emissions. The encoder is a three-layer CNN with kernel size 3, iteratively growing dilations, residual connections, ReLU activations, and a hidden size of 512. We do not report numbers for a BI-RNN-CRF similar to Shao et al. (2017), because our implementation did not yield their results for unigrams which is most likely caused by the different embeddings.

Model	F1	
	adapt	fix
EMB	91.42	92.18
CNN+FC+ReLU 1024	91.99	90.75
FC2L 512	88.45	91.32
CNN+SM+FC 4096	71.54	87.63
1-gram BI-RNN-CRF*/**	*92.45*	
3-gram BI-RNN-CRF*	94.07	

Table 6: Results for joint segmentation and POS tagging on the test set of CTB-5.0. (*) 1-gram and 3-gram BI-RNN-CRF reported by Shao et al. (2017), (**) the 1-gram BI-RNN-CRF result was reported on the development set.

Data The Chinese Treebank 5.0 (Xue et al., 2005) has joint annotations for word segmentation and part-of-speech tags. The commonly used cross validation split was reported by Jiang et al. (2008). Instead of the BIES tagging scheme (begin, inside, end, single), we used the BI scheme, which worked better for our models.

Training We use the parameters from the *EMB*, *CNN+FC+ReLU 1024*, *FC2L 512* and *CNN+SM+FC 4096* of the NMT task. We either adjust these parameters during training (*adapt*) or keep them fixed (*fix*). For training, we used Adagrad (Duchi et al., 2010) with an initial learning rate of 0.1, and dropout for regularization with probability 0.1 for the encoder and 0.5 for the output classifier. We trained the models for 50 epochs and selected the model with the best accuracy on the development set.

Results Our results reported in Table 6 are averaged over two distinct runs each. For comparison, we show results reported for the 1-gram and 3-gram lookup model of Shao et al. (2017). Our models correspond to the 1-gram model, as they are character-based. We find no clear difference between lookup-based and the best character encoder models. Interestingly, *CNN+SM+FC 4096* (softmax) is the weakest model. The sparseness in the in the signal apparently removes syntactic information. Comparing the *adapt* and the *fix* setting, we see that adapting the character encoder during training is —in most cases— not helpful. This is most likely due to the small amount of training data.

4.4 Experiments for Word Similarity

In our final experiment, we evaluated the representation of semantic information. The task is to compute a similarity score for pairs of words. The evaluation is the Spearman's correlation between the scores and numerical similarity judgments by humans. The words in this data set are translated into traditional Chinese. The training data from the NMT task is mostly from news and web sources, which is why our vocabulary contains many but not all Chinese characters. The characters in the modern simplified Chinese sometimes can be visually similar to their predecessors in traditional Chinese, i.e. they share visual components with a related meaning. Therefore, we can also evaluate the capability of the models to generalize to unseen characters.

Data and Experimental Setup For the experiments we use the WordSim-240, WordSim-296 and SimLex-999 datasets provided and described by Su and Lee (2017). Due to the use of traditional Chinese we are missing at least one character in 430 out of 1536 test examples. On average, we are missing 1.1 characters per word, where each word has in average 2.13 characters. We split the data into word pairs in which all characters are completely covered (*seen words*), and into word pairs where at least one character is not seen (*unseen words*). We evaluate *unseen words* in two settings: either we remove the unseen characters (*seen chars*) or not (*all chars*).

We did not train any model for this experiments. Most of the words in the dataset are composed of multiple characters, therefore, we average the output of the character encoder into a single word-vector. Subsequently, we compute the cosine similarity between these vectors of word pairs.

Results The results in Table 7 show that, for words with *seen* characters, the lookup embeddings correlate better with human judgments of similarity than the embeddings based on character encoders. However, especially the results for *unseen* words in WordSim-240 show that the character encoders can generalize to *all characters*. Surprisingly, the *FC2L* model, the model without CNNs, yields the best results. In the results reported by Su and Lee (2017), the lookup embedding model *SG-EMB* performs much better than *EMB*. However, they learn a RNN-based character composition, while we average the embeddings.

Model	WordSim-240			WordSim-297			SimLex-999		
Words	seen	unseen		seen	unseen		seen	unseen	
Characters		all	seen		all	seen		all	seen
EMB	**0.33**	n/a	0.06	**0.47**	n/a	0.15	**0.32**	n/a	**0.32**
CNN+SM+FC 4096	0.09	0.11	0.07	0.22	0.23	0.16	0.25	0.29	0.26
CNN+FC+R 1024	0.24	0.17	0.07	0.35	0.28	0.25	0.26	**0.30**	0.25
FC 2L 512	0.21	**0.26**	**0.15**	0.36	**0.29**	**0.30**	**0.33**	0.28	0.19
SG-EMB	0.59	n/a	n/a	0.59	n/a	n/a	0.36	n/a	n/a
SG-CNN	0.34	n/a	n/a	0.36	n/a	n/a	0.24	n/a	n/a

Table 7: Results for semantic similarity on the WordSim-240, WordSim-296 and SimLex-999 data. Lookup SkipGram (*SG-EMB*) and SkipGram char. encoder (*SG-CNN*) reported by Su and Lee (2017).

5 Conclusion

We have shown that it is possible to compute useful text representations from visual features, i.e., pixels of Chinese characters. We used neural machine translation as a framework for training and evaluating token representations. In the NMT experiment, we found that representations from visual features are competitive to lookup embeddings in a standard NMT model. In contrast to our expectations, the visual features outperform lookup embeddings on high-frequency characters, but are weaker in low-frequency characters. We conjecture that one of the reasons is that shared visual features between characters *can* have related meanings or related syntactic functions, but also the opposite can be true. For example, 沐 is the reflexive verb of ”washing” and it contains the phonetic component 木, which can also mean ”tree”. Therefore, the visual information for rare characters introduces probably as many difficulties as it can be helpful.

We performed additional experiments akin to prior studies and could show that the advantage of a representation by visual features is the ability to generalize to unseen Chinese characters. Also, we could show that joint part-of-speech tagging and segmentation achieved similar results with representation from visual features.

With regard to the character encoder architecture, we find only a slight advantage of using CNNs to simply using two fully connected layers in the NMT experiment, while 2 fully connected layers outperform CNNs in the similarity experiment. Whether or not there is an advantage could only be answered by an exhaustive hyperparameter search.

Future work is to explore how we can create a model that can distinguish helpful from unhelpful visual information for rare characters.

References

Ondrej Bojar, Christian Buck, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno-Yepes, Philipp Koehn, and Julia Kreutzer, editors. 2017. *Proceedings of the Second Conference on Machine Translation, Copenhagen, Denmark, September 7-8, 2017*, WMT 2017.

Marta R. Costa-jussà, David Aldón, and José A. R. Fonollosa. 2017. Chinese–Spanish neural machine translation enhanced with character and word bitmap fonts. *Machine Translation*, 31(1):35–47.

Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M. Rush. 2017. Image-to-Markup Generation with Coarse-to-Fine Attention. In *Proceedings of the 34th International Conference on Machine Learning, Sydney, NSW, Australia, 6-11 August 2017*, ICML 2017, pages 980–989.

John Duchi, Elad Hazan, and Yoram Singer. 2010. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. Technical Report UCB/EECS-2010-24, EECS Department, University of California, Berkeley.

Desmond Elliott, Stella Frank, Loïc Barrault, Fethi Bougares, and Lucia Specia. 2017. Findings of the Second Shared Task on Multimodal Machine Translation and Multilingual Image Description. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers, Copenhagen, Denmark, September 7-8, 2017*, WMT 2017, pages 215–233, Copenhagen, Denmark.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6-11 July 2015*, ICML 2015, pages 448–456.

Wenbin Jiang, Liang Huang, Qun Liu, and Yajuan Lü. 2008. A Cascaded Linear Model for Joint Chinese Word Segmentation and Part-of-Speech Tagging. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, ACL 2008, pages 897–904.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware Neural Language Models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI 2016, pages 2741–2749, Phoenix, Arizona. AAAI Press.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully Character-Level Neural Machine Translation without Explicit Segmentation. *TACL*, 5:365–378.

Frederick Liu, Han Lu, Chieh Lo, and Graham Neubig. 2017. Learning Character-level Compositionality with Visual Features. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, ACL 2017, Vancouver, Canada.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, September 17-21, 2015*, EMNLP 2015, pages 1412–1421.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, NIPS 2013, pages 3111–3119.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, October 25-29, 2014, Doha, Qatar*, EMNLP 2014, pages 1532–1543.

Hinrich Schütze. 2017. Nonsymbolic Text Representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, EACL 2017, pages 785–796.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, ACL 2016.

Yan Shao, Christian Hardmeier, Jörg Tiedemann, and Joakim Nivre. 2017. Character-based Joint Segmentation and POS Tagging for Chinese using Bidirectional RNN-CRF. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, Taipei, Taiwan, November 27 - December 1, 2017 - Volume 1: Long Papers*, IJCNLP 2017, pages 173–183.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and Accurate Entity Recognition with Iterated Dilated Convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, September 9-11, 2017*, EMNLP 2017, pages 2670–2680.

Tzu-ray Su and Hung-yi Lee. 2017. Learning Chinese Word Representations From Glyphs Of Characters. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, September 9-11, 2017*, EMNLP 2017, pages 264–273, Copenhagen, Denmark.

Yuguang Wang, Shanbo Cheng, Liyang Jiang, Jiajun Yang, Wei Chen, Muze Li, Lin Shi, Yanfeng Wang, and Hongtao Yang. 2017. Sogou Neural Machine Translation Systems for WMT17. In *Proceedings of the Second Conference on Machine Translation*, WMT 2017, pages 410–415, Copenhagen, Denmark.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*, abs/1609.08144.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The Penn Chinese TreeBank: Phrase Structure Annotation of a Large Corpus. *Natural Language Engineering*, 11(2):207–238.