OpenNRE: An Open and Extensible Toolkit for Neural Relation Extraction

Xu Han*, Tianyu Gao*, Yuan Yao, Demin Ye, Zhiyuan Liu†, Maosong Sun
Department of Computer Science and Technology, Tsinghua University, Beijing, China
Institute for Artificial Intelligence, Tsinghua University, Beijing, China
State Key Lab on Intelligent Technology and Systems, Tsinghua University, Beijing, China
{hanxu17,gty16,yy18,ydm18}@mails.tsinghua.edu.cn

Abstract

OpenNRE is an open-source and extensible toolkit that provides a unified framework to implement neural models for relation extraction (RE). Specifically, by implementing typical RE methods, OpenNRE not only allows developers to train custom models to extract structured relational facts from the plain text but also supports quick model validation for researchers. Besides, OpenNRE provides various functional RE modules based on both TensorFlow and PyTorch to maintain sufficient modularity and extensibility, making it becomes easy to incorporate new models into the framework. Besides the toolkit, we also release an online system to meet real-time extraction without any training and deploying. Meanwhile, the online system can extract facts in various scenarios as well as aligning the extracted facts to Wikidata, which may benefit various downstream knowledge-driven applications (e.g., information retrieval and question answering). More details of the toolkit and online system can be obtained from http://github.com/thunlp/OpenNRE.

1 Introduction

Relation extraction (RE) aims to predict relational facts from the plain text, e.g., extracting (*Newton*, the Member of, *the Royal Society*) from the sentence "*Newton* served as the president of *the Royal Society*". Because RE models can extract structured information for various downstream applications, many efforts have been devoted to researching RE. As the rapid development of deep learning in the recent years, neural relation extraction (NRE) models show the strong ability to extracting relations and achieve great performance, which makes more and more researchers and industry developers pay attention to this field.

Although the current NRE models are effective and have been applied for various scenarios, including supervised learning paradigm (Zeng et al., 2014a; Nguyen and Grishman, 2015; Zhang et al., 2015; Zhou et al., 2016), distantly supervised learning paradigm (Zeng et al., 2015; Lin et al., 2016; Han et al., 2018b), few-shot learning paradigm (Han et al., 2018c; Gao et al., 2019; Ye and Ling, 2019; Soares et al., 2019; Zhang et al., 2019), there still lack an effective and stable toolkit to support the implementation, deployment and evaluation of models. In fact, for other tasks related to RE, there have been already some effective and long-term maintained toolkits, such as Spacy[1] for named entity recognition (NER), TagMe (Ferragina and Scaiella, 2010) for entity linking (EL), OpenKE (Han et al., 2018a) for knowledge embedding, and Stanford OpenIE (Angeli et al., 2015) for open information extraction. Hence, it becomes necessary and significant to systematically develop an efficient and effective toolkit for RE.

To this end, we develop an open and extensible toolkit for designing and implementing RE models, especially for NRE models, which is named "OpenNRE". The toolkit prioritizes operational efficiency based on TensorFlow and PyTorch, which support quick model training and validation. Meanwhile, the toolkit maintains sufficient system encapsulation and model extensibility, which can meet some individual requirements of incorporating new models. To keep the ease of use, we implement many typical RE models and provide a unified framework for their data processing, model training, and experimental evaluation. For those developers aiming at training custom

* indicates equal contribution
† Corresponding author: Z.Liu(liuzy@tsinghua.edu.cn)

[1] https://spacy.io

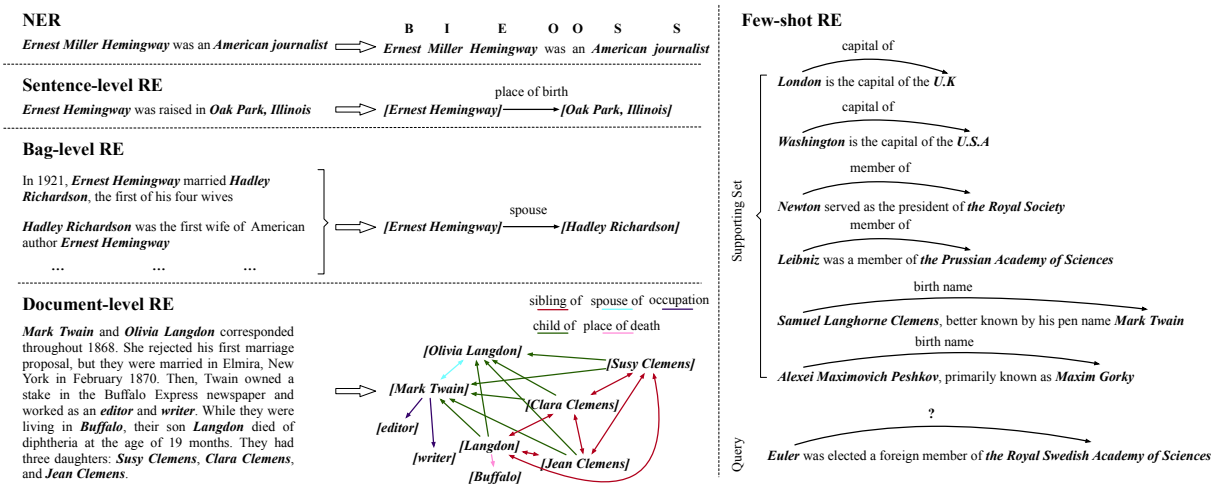


Figure 1: The examples of all application scenarios in OpenNRE.

models, they can quickly start up their RE system based on OpenNRE, without knowing too many technical details and writing tedious glue code. An online system is also available to extract structured relational facts from the text with friendly interactive interfaces and fast reaction speed. We will provide long-term maintenance to fix bugs and meet new requests for OpenNRE, and we think both researchers and industry developers can benefit from our toolkit.

2 Application Scenarios

OpenNRE is designed for various scenarios for RE, including sentence-level RE, bag-level RE, document-level RE, and few-shot RE. For completing a full pipeline of extracting structured information, we also enable OpenNRE to have the capacity of entity-oriented applications to a certain extent, e.g., NER and EL. The examples of these application scenarios are all shown in Figure 1.

2.1 Entity-Oriented Applications

For extracting structured information from plain text, it requires to extract entities from text and then predict relations between entities. In normal RE scenarios, all entity mentions have been already annotated and RE models are just required to classify relations for all annotated entity pairs. Although the entity-oriented applications are not the focus of our toolkit, we still implement specific modules for NER (Lample et al., 2016) and EL (Han et al., 2011). The NER modules can detect words or phrases (also named entity mentions) representing real-world objects. In OpenNRE, we provide two approaches for NER, one is based on spaCy, the other is based on fine-tuning BERT (Devlin et al., 2019). The EL modules can align those entity mentions to the entities in Wikidata (Vrandečić and Krötzsch, 2014) based on TagMe (Ferragina and Scaiella, 2010).

2.2 Sentence-Level Relation Extraction

The conventional methods often handle RE in the supervised learning paradigm and extract the relation between two entities mentioned within one sentence. As shown in Figure 1, each sentence is first manually annotated with two entity mentions. Then models are required to predict the relation between those annotated entity mentions. As there are many efforts to adopt models for this setting (Zeng et al., 2014a; Zhang et al., 2015; Zhou et al., 2016), OpenNRE is specially designed for the sentence-level RE scenario.

2.3 Bag-Level Relation Extraction

The supervised RE methods suffer from several problems, especially their requirements of adequate annotated data for training. As manually labeling large amounts of data is expensive and time-consuming, Mintz et al. (2009) introduce distant supervision to automatically label large amounts of data for RE by aligning knowledge graphs and text. Although distant supervision brings sufficient auto-labeled data, it also leads to the wrong labeling problem. Considering an entity pair may occur several times in different sentences, and there is a significant probability that some of these sentences can express the relation between the entity pair. Hence Riedel et al. (2010) and Hoffmann et al. (2011) introduce to aggregate the sentences mentioning the same entity pair into

a entity-pair bag. As shown in Figure 1, synthesizing the features of different sentences in a bag can provide more reliable information and result in more accurate predictions. The Bag-level setting is widely applied by various distantly supervised RE methods (Zeng et al., 2015; Lin et al., 2016; Han et al., 2018b), and thus it is also integrated into OpenNRE.

2.4 Document-Level Relation Extraction

Yao et al. (2019) have pointed out that multiple entities in documents often exhibit complex inter-sentence relations rather than intra-sentence relations. Besides, as shown in Figure 1, a large number of relational facts are expressed in multiple sentences, e.g., *Langdon* is the sibling of *Jean Clemens*. Hence, it is hard to extract these inter-sentence relations with both the sentence-level and bag-level settings. Although the document-level RE setting is not widely explored by the current work, we argue that this scenario remains an open problem for future research, and still integrate document-level RE into OpenNRE.

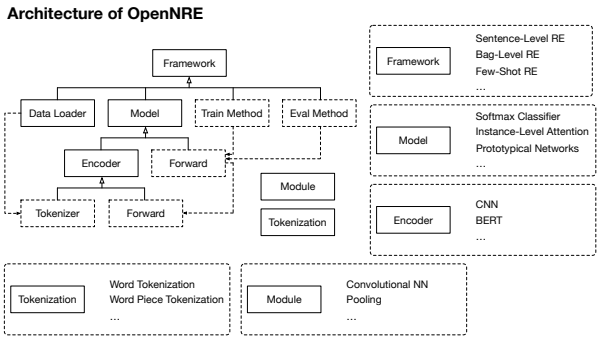
2.5 Few-Shot Relation Extraction

Though we can train a usable and stable RE system based on the above-mentioned scenarios, which can well predict those relations appearing frequently in data, some long-tail relations with few instances in data are still neglected. Recently, some methods have been proposed to provide a different view of this problem by formalizing RE as a few-shot learning problem (Han et al., 2018c; Gao et al., 2019; Ye and Ling, 2019; Soares et al., 2019; Zhang et al., 2019). As shown in Figure 1, each relation only have a handful of instances in the supporting set in a few-shot RE scenario, and models are required to be capable of accurately capturing relation patterns of these small amounts of training instances. Considering few-shot RE is important for handling long-tail relations, OpenNRE also provides a custom platform for further research in this direction.

3 Toolkit Design and Implementation

Our goal of designing OpenNRE is achieving the balance among system encapsulation, operational efficiency, model extensibility, and ease of use.

For system encapsulation, we build a unified underlying platform to encapsulate various data processing and training strategies, so that developers



Example Code

```
rel2id = 'relation->id dictionary'
# Define encoder
sentence_encoder = nrekit.encoder.BERTEncoder(
    max_length=80, pretrain_path='bert pretrain model path')
# Define model
model = nrekit.model.SoftmaxNN(sentence_encoder, len(rel2id), rel2id)
# Define framework
framework = nrekit.framework.SentenceRE(
    train_path='path of training data',
    val_path='path of validation data',
    test_path='path of test data',
    model=model,
    ckpt='path of checkpoint',
    batch_size=64,
    max_epoch=10,
    lr=3e-5,
    opt='bert_adam')
# Train
framework.train_model()
# Test
framework.load_state_dict(torch.load(ckpt)['state_dict'])
result = framework.eval_model(framework.test_loader)
```

Figure 2: The architecture and example code of OpenNRE. The structure shows the contents of each part of OpenNRE and how they are related. Based on OpenNRE, one can use only a few lines of code to define, train and evaluate RE models of different scenarios.

can maximize the reuse of code to avoid unnecessary redundant model implementations. For operational efficiency, OpenNRE is based on TensorFlow and PyTorch, which enables developers to train models on GPUs. For model extensibility, we systematically implement various neural modules and some special algorithms (e.g., adversarial training (Wu et al., 2017) and reinforcement learning (Feng et al., 2018)). Hence, it is easy to implement new RE models based on OpenNRE. We also implement some typical RE models so as to conveniently train custom models for specific application scenarios.

More specifically, OpenNRE attains the above four design objects through implementing the following five components.

3.1 Tokenization

The tokenization component is responsible for tokenizing input text into several input tokens. In OpenNRE, we implement both word-level tokenization and subword-level tokenization. These two operations satisfy most tokenization demands and help developers get rid of spending too much time writing glue code for data processing. For building a new tokenizer, extending the

`BasicTokenizer` class and implementing specific tokenization operations is convenient.

3.2 Module

The module component consists of various functional neural modules for model implementation, such as the essential network layers, some pooling operations, and activation functions. In order to adapt these modules for RE scenarios, we also implement some special RE neural modules (e.g., piece-wise pooling operation (Zeng et al., 2015)). Using these atomic modules to construct and deploy RE systems has a high degree of freedom.

3.3 Encoder

The encoder is applied to encode text into its corresponding embeddings to provide semantic features. In OpenNRE, we implement the `BaseEncoder` class based on the tokenization and module components, which can provide basic functions of text tokenization and embedding lookup. By extending the `BaseEncoder` class and implementing specific neural encoding architecture, we can implement various specific encoders. In OpenNRE, we have implemented the common convolutional and recurrent neural encoders, as well as the pre-trained encoder BERT.

3.4 Model

Some developers may not require to implement and verify their own RE models, and their main demand is to easily train and deploy custom models. To this end, we also replicate several typical RE models (Zeng et al., 2015; Zhang et al., 2015). Some special algorithms for enhancing RE models are also included in the toolkit, such as attention mechanism (Lin et al., 2016), adversarial training (Wu et al., 2017), and reinforcement learning (Feng et al., 2018). On the one hand, the model component enables us to train custom models without having to understand all technical details. On the other hand, the implemented models in this model component are all tutorial examples to show how to build models with OpenNRE.

3.5 Framework

The framework module is mainly responsible for integrating other four components and supporting various functions (including data processing, model training, model optimizing, and model evaluating). In OpenNRE, for all application scenarios mentioned in Section 2, we have implemented their corresponding framework. For other future potential application scenarios, we have also reserved interfaces for their implementation.

Model	Wiki80	SemEval
CNN	63.93	71.11
BERT	84.57	84.02
BERT-Entity	86.61	84.21

Table 1: Accuracies of various models on Wiki80 and SemEval 2010 Task-8 under the single sentence setting.

Model	F1	F1 (*)
BERT	0.880	-
BERT-Entity	0.883	0.892

Table 2: Micro F1 scores of various models on SemEval 2010 Task-8 under the sentence-level RE setting. "(*)" indicates the original results from Soares et al. (2019).

4 Experiment and Evaluation

In this section, we evaluate our toolkit on several benchmark datasets in different RE scenarios. The evaluation results show that our implementation of some state-of-the-art models with OpenNRE can achieve comparable or even better performance, as compared to the original papers.

4.1 Sentence-Level Relation Extraction

We experiment on two different encoders for sentence-level relation extraction: CNN (Zeng et al., 2014b) and BERT (Devlin et al., 2019). For CNN, we follow the setting of Nguyen and Grishman (2015), including using word and position embeddings. For BERT, we follow the setting of Soares et al. (2019). "BERT" in our paper refers to using entity markers in input and taking `[CLS]` as output. "BERT-Entity" refers to using entity markers in input and taking entity start as output like Soares et al. (2019).

We carry out experiments on two datasets of sentence-level RE: SemEval 2010 Task-8 (Hendrickx et al., 2009) and Wiki80. SemEval 2010 Task-8 contains 19 relations and 10,717 instances, 17.4% of which are with no relation. Wiki80 is derived from FewRel (Han et al., 2018c), a large scale few-shot dataset. It contains 80 relations and 56,000 instances from Wikipedia and Wikidata (Vrandečić and Krötzsch, 2014). Since Wiki80 is not an official benchmark, we directly report the results on the validation set. From Table 1 we can see that BERT-based models perform bet-

Model	5-Way 1-Shot	5-Way 5-Shot	5-Way 1-Shot (*)	5-Way 5-Shot (*)
Prototype-CNN	74.5	88.4	69.2	84.8
Prototype-BERT	80.7	89.6	-	-
BERT-PAIR	88.3	93.2	-	-

Table 3: Accuracies of various models on FewRel under the different few-shot settings. "(*)" indicates the original results taken from Han et al. (2018c).

Model	AUC	F1	AUC (*)	F1 (*)
CNN-ATT	0.333	0.397	0.318	0.380
CNN-ADV	0.337	0.406	-	-
CNN-RL	0.276	0.429	-	0.42

Table 4: AUC and F1 scores of various models on NYT10 under the bag-level RE setting. "(*)" indicates the original results taken from Lin et al. (2016) and Feng et al. (2018).

ter than the CNN model and achieve promising results on both datasets. Our implementation of BERT-Entity with OpenNRE achieves comparable results to the original work in Table 2.

4.2 Bag-Level Relation Extraction

We implement models with instance-level attention (Lin et al., 2016), adversarial training (Wu et al., 2017) and reinforcement learning (Feng et al., 2018) for bag-level relation extraction. Note that the latter two are based on the instance-level attention mechanism. We use the same CNN encoder as Section 4.1 and denote the three models as "CNN-ATT", "CNN-ADV" and "CNN-RL". We evaluate those three models on NYT10 (Riedel et al., 2010), a distantly supervised dataset based on New York Times corpus and FreeBase (Bollacker et al., 2008). Table 4 shows that our version of bag-level RE models achieves comparable or even better results than the original papers.

4.3 Few-Shot Relation Extraction

We experiment on Prototypical Networks (Snell et al., 2017) for few-shot RE. For the encoder selection, we take CNN and BERT as described in Section 4.1. We also implement a model named "BERT-PAIR", which takes one supporting sentence and one query sentence as input, and directly outputs the probability that they share the same relation with the BERT sequence classification model. The experiments are carried out on FewRel described in Section 4.1. From Table 3 we can see that for both few-shot settings, our version achieves better results than the original results from Han et al. (2018c), proving that our implementation with OpenNRE is robust.

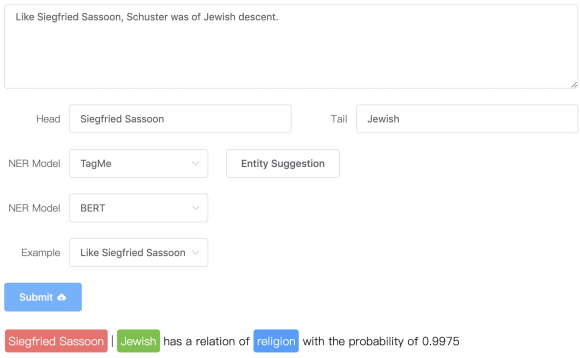


Figure 3: An example of the online system.

5 Online System

Besides the toolkit, we also release an online system. As shown in Figure 3, we train a model in the sentence-level RE scenario and deploy the model for online access. The online system can be directly applied for extracting structured facts from plain text. Meanwhile, all extracted entity mentions and relations can be aligned to Wikidata.

6 Conclusion

We propose OpenNRE, an open and extensible toolkit for relation extraction. OpenNRE achieves the balance among system encapsulation, operational efficiency, model extensibility, and ease of use. Based on OpenNRE, either training custom models or quick model validation becomes easy. Some experimental results also demonstrate that the models implemented by OpenNRE are efficient and effective, which can achieve comparable or even better performance as compared to the original papers. Furthermore, an online system is also available for meeting real-time extraction without training and deploying. In the future, we will provide long-term maintenance to fix bugs and meet new requests.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (No.

2018YFB1004503) and the National Natural Science Foundation of China (NSFC No. 61572273, 61772302). Han and Gao are supported by 2018 and 2019 Tencent Rhino-Bird Elite Training Program respectively. Gao is also supported by Tsinghua University Initiative Scientific Research Program.

References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of ACL*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In *Proceedings of AAAI*.

Paolo Ferragina and Ugo Scaiella. 2010. TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of CIKM*.

Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. 2019. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In *Proceedings of AAAI*.

Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in web text: a graph-based method. In *Proceedings of SIGIR*.

Xu Han, Shulin Cao, Xin Lv, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. 2018a. Openke: An open toolkit for knowledge embedding. In *Proceedings of EMNLP*.

Xu Han, Pengfei Yu, Zhiyuan Liu, Maosong Sun, and Peng Li. 2018b. Hierarchical relation extraction with coarse-to-fine grained attention. In *Proceedings of EMNLP*.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018c. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of EMNLP*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on SemEval*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of ACL*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of ACL*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL-IJCNLP*.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the NAACL-HLT*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of ECML-PKDD*.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Proceedings of NIPS*.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of ACL*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*.

Yi Wu, David Bamman, and Stuart Russell. 2017. Adversarial training for relation extraction. In *Proceedings of EMNLP*.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. Docred: A large-scale document-level relation extraction dataset. In *Proceedings of ACL*.

Zhi-Xiu Ye and Zhen-Hua Ling. 2019. Multi-level matching and aggregation network for few-shot relation classification. In *Proceedings of ACL*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of EMNLP*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014a. Relation classification via convolutional deep neural network. In *Proceedings of COLING*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014b. Relation classification via convolutional deep neural network. In *Proceedings of COLING*.

Shu Zhang, Dequan Zheng, Xinchen Hu, and Ming Yang. 2015. Bidirectional long short-term memory networks for relation classification. In *Proceedings of PACLIC*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. In *Proceedings of ACL*.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of ACL*.