Method Exploration: Relation Extraction Using BERT

PoChun Chang
Department of Linguistics and Philology
Uppsala University
po-chun.chang4319@student.uu.se

Abstract

Similar to many tasks in natural language processing, BERT has become the core element of relation extraction model in recent studies. Various ways of using BERT model for relation extraction have been proposed. In this paper, I revisit approaches utilized by recent papers and combine them into several experiments for relation extraction using TACRED dataset. My experiments are based on the change within three categories: input representation, model architecture and selecting of BERT outputs for prediction. In this paper, I demonstrate effectiveness of these approaches and the optimal task design for relation extraction in the presence of BERT model. The result indicates that by masking the entities within the input with their entity type and use their embedding output from a finetuned BERT model can achieve the best result among more complex setup such as BERT with bi-LSTM layer or input sequence with extra entity appending.

1 Introduction

Relation extraction is one of the fundamental tasks of natural language understanding. It provides relations between entities that can be used to build up knowledge graph for many downstream tasks including question answering, sentiment analysis or text summarizing. The goal of relation extraction is to predict the relation between two entities within a sentence. For instance, given a sentence "Obama was born in Honolulu", with its subject entity "Obama" and object entity "Honolulu", we will have to classify the relation of these two entities out of a fixed number of pre-defined relations, which is `per:city of birth`, in this case.

1.1 Studies Using TACRED Dataset

TACRED (Zhang et al., 2017) is the dataset used in this study. It is a popular dataset for training and evaluating of the relation extraction models. In recent years, several improvements of the TACRED F1 score have been made through the variation of language model, model architecture and the training method used. Zhang et al. (2017), apart from creating the TACRED dataset, incorporate the attention mechanism and position sequence encoding for better prediction of long sequences in comparison to Zeng et al. (2014)'s use of recurrent neural networks, Nguyen and Grishman (2015)'s covolutional neural networks and Zhou et al. (2016)'s combination of both. They use GloVe (Pennington et al., 2014) as the word embedding method and achieve around 65.1 F1 score. The later researches have soon shifted the focus of the word embedding method from GloVe to BERT (Devlin et al., 2019), receiving gradual improvement of the performance on TACRED dataset. In Shi and Lin (2019)'s work, the input sequence is formatted as the original sequence with entities appended in the end, with a bidirectional Long Short-Term Memory (bi-LSTM) layer on top of BERT before final output. While in the work of Baldini Soares et al. (2019), different variations of input and BERT embedding selection are experimented; higher F1 score is achieved without a bi-LSTM layer.

1.2 Research Aim

The improvement achieved by the change in input sequence and the reduction of model structure in the works of Shi and Lin (2019) and Baldini Soares et al. (2019) raises a few questions: Does the benefit of customized input of BERT outweigh the benefit of adding model complexity for relation extraction? And if it is true, what kind of input format and embedding selection for BERT can fully utilize its potential in relation extraction? In this work, I will explore the capability of BERT model in the task of relation extraction. My discussion will revolve around three aspects, the in-

put format of BERT, the selection of BERT output for prediction and the choice between using BERT as a fixed pre-trained word encoder or a model for finetuning.

2 Related Work

2.1 Relation Extraction using BERT

To formally define the task of relation extraction using BERT, let $\mathbf{x} = [x_0 \dots x_n]$ be a sequence of tokens tokenized from a relation statement for the input of BERT. Since BERT model requires special token [CLS] in the beginning of the sequence, and token [SEP] for the ending of sequence or separation of different sequence (Devlin et al., 2019), let x_0 = [CLS] and x_n = [SEP]. The subject entity is denoted as $\mathbf{e_s} = (i, j)$ and object entity is $\mathbf{e_o} = (k, l)$ where $0 < i < j < n$ and $0 < k < l < n$, so that $[x_i \dots x_j]$ is the subject in the sentence and $[x_k \dots x_l]$ is the object. The goal of relation extraction is to create a function $\mathbf{h_r} = f_\theta(\mathbf{x}, \mathbf{e_s}, \mathbf{e_o})$ which can learn the mapping of relation statement into a vector representing possibilities of all existing relations between $\mathbf{e_s}$ and $\mathbf{e_o}$.

2.2 Related work using BERT

In the work of Shi and Lin (2019), the input for BERT model is customized. Besides the relation statement sequence $\mathbf{x}$, subject and object entities $\mathbf{e_s}$ and $\mathbf{e_o}$ are appended to $\mathbf{x}$ using with extra [SEP] tokens. Meanwhile, the original $\mathbf{e_s}$ and $\mathbf{e_o}$ within $\mathbf{x}$ are masked with their special entity type tokens to prevent the model from overfitting to specific entity strings. As the result, the input sentence "Obama was born in Honolulu" is modified as: "[CLS] [S-PER] was born in [O-LOC] [SEP] Obama [SEP] Honolulu [SEP]". The input then passes through BERT model and becomes a sequence of token embeddings, which are further concatenated with both subject and object's position embedding before moving on to a layer of bi-LSTM, a layer of perceptron and softmax for prediction. The use of subject and object position embedding stemmed from the work of Zhang et al. (2017). It is meant to provide bi-LSTM the positional information of $\mathbf{e_s}$ and $\mathbf{e_o}$, in addition to the semantic information coming from word embedding methods. Subject and object position embedding can be obtained by passing the position sequence into a position sequence encoder. In the case of subject position embedding, the subject position sequence $[p_0^s, \dots, p_{n+1}^s]$ is first created based on $\mathbf{e_s} = (i, j)$ with following definition:

$$p_m^s = \begin{cases} m - i, & m < i \\ 0, & i < m < j \\ m - j, & m > j \end{cases} \quad (1)$$

The subject position sequence will then go through a position sequence encoder whose parameters are learned through the training process and become a fixed length subject position embedding. The same applies to object position sequence $[p_0^o, \dots, p_{n+1}^o]$. In this paper, I will examine the same model architecture with and without the bi-LSTM and the input sequence with and without appended entities for analysis of their impacts on relation extraction model with BERT. In the work of Baldini Soares et al. (2019), various usage of BERT model are explored for relation extraction. For all their experiments, the output of BERT is fed straight to a layer of perceptron before a softmax layer, without the use of bi-LSTM. They introduce entity markers: the special token [e1] and [e2], which are inserted before and after $\mathbf{e_s}$ and $\mathbf{e_o}$ as indicators of entity location for BERT. In terms of BERT embedding selection for relation prediction, they explore using only the pooling output (the position of [CLS]), the average of outputs from entity sequences, and only the outputs of entity markers. They find the model predicting from the entity markers achieves the highest F1 score of 70.1. In my study, I will experiment the appending of entities and entity masking from Shi and Lin (2019), together with the selection of embeddings from entity masking tokens for prediction, as a way to mimic the implementing of entity markers; the experiment will allow me to look into each of these methods effectiveness on relation extraction and figure out the best combination.

3 Experiment Variation

For all the experiments I conducted in this paper, I used the $\text{BERT}_{\text{BASE}}$ model from Devlin et al. (2019) as the main component of my relation extraction model. I categorize my variations of the experiment based on three aspects: model configuration, input representation and choice of BERT output for relation extraction. Using the setting of these three aspects I aim to discuss their effect on the task and find the optimal system.

3.1 Model Configuration

From Section 2.2, we know that Shi and Lin (2019) utilizes an extra layer of bi-LSTM on top of BERT. However, in Baldini Soares et al. (2019), higher performance is achieved without the use of bi-LSTM. I aim to revisit the use of bi-LSTM and find out its importance to the relation extraction model. On the other hand, BERT model utilizes transformer architecture with fixed input format: [CLS] sentence 1 [SEP] sentence 2 [SEP] (Devlin et al., 2019). Considering that parts of my experiments have unorthodox formats of inputs based on the customization the experiments, finetuning BERT might be necessary for BERT to learn the new formats of inputs. Through the comparison between non-finetuned experiments and finetuned ones, I can verify the effect of finetuning in the presence of unorthodox input formats for BERT.

3.1.1 Bi-LSTM

In my experiment, the output dimension from BERT for each token is as default: 768, representing the semantic information of the input. Together with 20-dimension position embedding that represents positional information for $\mathbf{e_s}$ and $\mathbf{e_o}$ (Zhang et al., 2017), the total dimension for each token embedding is 808. Since for each input batch, the max sentence length from padded sentences are different, some ways to summarize the BERT output are needed to achieve a fixed length vector representation for relation prediction. For my experiment, I follow the method from the work of Shi and Lin (2019), using bi-LSTM (Hochreiter and Schmidhuber, 1997) for such a purpose due to its ability in summarizing sequence information in both direction. The bi-LSTM layer on top of BERT takes in the 808-dimension embedding in each sequence time step, creating summary vector of the whole relation statement. The hidden states from last time steps in both directions will be concatenated for a perceptron layer of 300 dimension before a final softmax layer. The structure of bi-LSTM model is depicted in Figure 1.

3.1.2 Finetuning BERT

As mentioned in Section 3.1, customized BERT input might require BERT to be finetuned to be fully utilized. I will examine the performance of models trained with and without finetuning BERT to verify its effect. Wolf et al. (2020) offers huggingface package for the use of the BERT model. All outputs of the BERT model from my experiments are extracted from the last hidden layer, and the configuration follows the default setting in Huggingface.

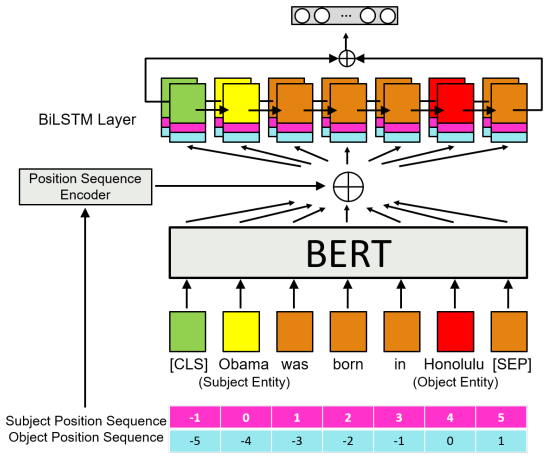


Figure 1: Model structure with a bi-LSTM layer. The output from BERT layer provides semantic information of each token in the relation statement, while the subject and object position sequences are fed to a positional sequence encoder, transforming into position embedding that add positional information to the BERT output through concatenation (Zhang et al., 2017). Note that in this figure, only the original input sequence without customization is depicted.

3.2 Input Representation

In Shi and Lin (2019), $\mathbf{x}$ is appended with $\mathbf{e_s}$ and $\mathbf{e_o}$ by inserting [SEP] token between them, while the entities in the actual sequence are masked with entity type tokens. Baldini Soares et al. (2019) has different approach. Their $\mathbf{x}$ is not appended, and their entity is surrounded by special [e1] and [e2] tokens. Whether it is better to put $\mathbf{e_s}$ and $\mathbf{e_o}$ in the end of the sequence, or leaving them in the original position surrounded by special tokens is the question I aim to answer using different input representation. Additionally, I will only use entity type token for entity masking instead of exploring the effect of [e1] and [e2] tokens because they are similar in the way they represent entities. Examples of different variations of input representation used in this study can be found in Table 1.

3.2.1 Entity Appending

Recall, in Section 2.2, the use of entity appending from Shi and Lin (2019) that place entities in the end of the sequence using [SEP] tokens, as a way of informing the BERT model the actual sequence of subject and object entities. In my experience I will follow the exact same approach, in juxta-

BERT Input Representation	Example
Original Sequence	[CLS] Obama was born in Honolulu [SEP]
Entity Appending	[CLS] Obama was born in Honolulu [SEP] Obama [SEP] Honolulu [SEP]
Entity Type Masking	[CLS] [S-PER] was born in [O-LOC] [SEP]
Entity Appending + Entity Type Masking	[CLS] [S-PER] was born in [O-LOC] [SEP] Obama [SEP] Honolulu [SEP]

Table 1: The variations of input representation for BERT used in this study.

position to the model using original sequence to discuss the effect of such approach.

3.2.2 Special Entity Masking Token

Out of TACRED dataset, there are only two unique subject entity types. They are: `person` and `organization`. Yet out of the whole dataset, the actual subject entity sequences vary from data to data. For example, a subject of `person` type could be "Ali Akbar Salehi", "Mark Buse" or "Barry Goldwater". Not all of these entity sequences exist in the vocabulary of the $\text{BERT}_{\text{BASAE}}$ model. For example, "Barry Goldwater" is tokenized into “Barry”, “Gold”, “##water” from BERT’s wordpiece tokenizer. I argue in such case the tokens cannot be fully utilized by BERT’s pretrained information. Another issue is that these tokens are essentially human names, putting them into the model might result in overfitting during training. One solution mentioned in Shi and Lin (2019) is to mask the entity tokens with their entity types, so that the model can learn from the type of entities instead of actual entity sequence. To create the special entity type tokens for masking, I add them into the vocabulary of $\text{BERT}_{\text{BASE}}$ and allow BERT to initiate their embedding randomly and adjust through the process of finetuning. I aim to examine whether the masking using entity type token can improve the model’s generalizing ability.

3.3 Choice of BERT output

$\text{BERT}_{\text{BASE}}$ model contains 12 hidden layers. One is allowed to extract hidden states of certain layer as the embedding of input tokens. Meanwhile, Devlin et al. (2019) train BERT for next sentence prediction task using two sentences, extracting only the output of [CLS] token for prediction. Such architecture design forces BERT to utilize its attention mechanism to collect crucial information from the whole sequence and store them in the position of [CLS]. In the situation where BERT is finetuned for a specific task, the output of [CLS] position can be seen as the pooling output of the input sequence. As the result, one can develop many approaches in getting embeddings out of BERT model. In the approach of Wu and He (2019), the embedding of [CLS], together with the average of the embeddings across the entity spans are fed into respective perceptron layers before being concatenated together for final softmax. Which Bert layer’s embedding one uses and which tokens embedding one selects for prediction, might impact the performance of the relation extraction model. By using different choice of BERT output, including [CLS] pooling and entity type tokens, I aim to discuss the change they make in the performance. Notably, for all my experiments, only the embeddings from the last layer will be utilized so that the results can be compared together, and only the experiments with bi-LSTM model will utilize embeddings from every token in the input sequence. The example of model structure using different BERT output can be found in Figure 2.

3.3.1 Embedding from complete sequence

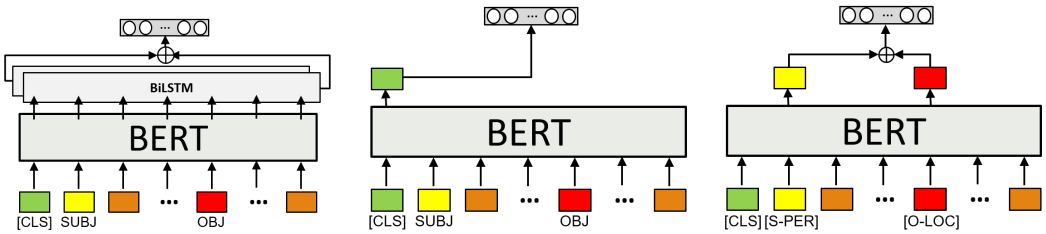
For my experiments with bi-LSTM models, all embeddings output from last layer of BERT will be used for the recurrent neural network. This serves as the direct comparison with experiments without bi-LSTM layer to see if the bi-LSTM is really needed.

3.3.2 [CLS] Pooling

I also experiment predicting using only the embedding of [CLS] token. This experiment is to see whether subject and object entity information can be learned as part of the [CLS] embedding. The [CLS] embedding is a 768 dimension vector that is supposed to represent the whole input sequence, which is then fed to a 300 dimension hidden layer (perceptron) for dimension reduction before softmax.

3.3.3 Entity Type Token

In Section 3.2.2, I discussed the input representation using special entity masking token (entity type token). I will also explore using the embeddings of them from BERT’s last layer. I will



(a) Using complete BERT output **(b)** [CLS] pooling **(c)** Using entity type tokens

Figure 2: Different choices of BERT output for relation extraction in this study. Note that in Figure 2a, the position sequence encoding is neglected from the illustration for the sake of simplicity. The detailed version of the model architecture is demonstrated in Figure 1.

concatenate both subject and object entity type tokens' embedding and feed them to a 300 dimension perceptron layer before softmax. This experiment is similar to Baldini Soares et al. (2019), where they use the embedding of entity markers that surround the entity sequence to achieve their best performing model.

4 Experiments

Regardless of the model variation, all my experiments share the same following model configuration:

- **BERT Architecture:** 12 layers, 768 hidden dimension
- **Weight Initialization:** $\text{BERT}_{\text{Base}}$
- **Post BERT Layer:** Perceptron layer 300
- **Training Epochs:** 30
- **Learning Rate:** 6e-3 with SGD
- **Batch Size:** 20
- **Dropout rate:** 0.5
- **Bi-LSTM dimension (if exist):** 768
- **Position Embedding (With bi-LSTM):** 20

The models for the evaluation of TACRED test set are selected from the ones with the highest dev set F1 score in each experiment. The results of the test set are shown in Table 2. Experiment 4 which is based on the work of Shi and Lin (2019) achieves F1 score of 68.5 which is similar to 67.8 from the original paper. And experiment 6 that predicts relation using entity type tokens is similar to the work of Baldini Soares et al. (2019), achieving F1 score of 68.8, which is slightly lower than the F1 score of 70.1 from the original paper. Since the original paper uses $\text{BERT}_{\text{LARGE}}$, a significant larger pre-trained model than the $\text{BERT}_{\text{BASE}}$ model, the difference in F1 score might be negligible.

5 Analysis

5.1 The baseline

The conclusion of the paper will be discussed using the experiment number in Table 2. To begin, experiment 0 is performed without finetuning of BERT. The average of the entire sequence of embeddings output from BERT is used for the prediction of relation. It achieves significantly lower F1 score than experiment 1 where bi-LSTM layer is utilized, suggesting that bi-lstm is important for summarizing the BERT output in the situation where BERT is not finetuned. Experiments 1 and 2 are conducted without finetuning the BERT model as well. Their only difference is the input sequence; experiment 1 uses the complete sentence as input, while experiment 2 has entities appended to the original sequence. The results from both experiments are similar (around 57 F1 score), it could be that entity appending is not helpful to the model, or that BERT model is not able to adapt to the input format different from how it is pretrained in (Devlin et al., 2019)'s work without finetuning.

5.2 The effectiveness of finetuning BERT

Experiment 2 and 3 demonstrate the effectiveness of allowing BERT to be finetuned with performance improvement being around 6 F1 score in experiment 3. Note that, however, we still cannot be sure whether finetuning BERT allows the use of the entity appending input format in both experiments to be beneficial, as finetuning BERT might simply allow BERT's inner attention mechanism to work in favor of the task, regardless of the appending of entity.

Experiment ID	Model Configuration		Input Representation	Choice of BERT output	TACRED F1 Score
	BERT Finetuned	**Bi-LSTM Layer**			
0	No	No	Original Sequence	Complete Sequence (Average)	2.2
1	No	Yes	Original Sequence	Complete Sequence	57.1
2	No	Yes	Entity Appending	Complete Sequence	56.2
3	Yes	Yes	Entity Appending	Complete Sequence	62.4
4	Yes	No	Entity Appending	Complete Sequence (Average)	65.7
5	Yes	Yes	Entity Appending + Entity Type Masking	Complete Sequence	68.5
6	Yes	No	Entity Appending + Entity Type Masking	[CLS] Pooling	67.1
7	Yes	No	Entity Appending + Entity Type Masking	Entity Type Token	**68.8**
8	Yes	No	Entity Appending + Entity Type Masking	[CLS] Pooling Concatenates Entity Type Token	68.6
9	Yes	No	Entity Type Masking	Entity Type Token	68.7
Shi and Lin (2019)					67.8
Baldini Soares et al. (2019) ($\text{BERT}_{\text{LARGE}}$)					70.1

Table 2: The results from all the experiments conducted in this study. Each experiment has different variation in model configuration, input representation and choice of BERT output. The experiment ID will be used as reference in the discussion.

5.3 Bi-LSTM is not needed

Experiment 3 and 4 showcase the effectiveness of Bi-LSTM when BERT is finetuned. In both experiments, the entire BERT output is used for prediction of relations. In the case of experiment 3, the entire BERT output is fed into a layer of bi-LSTM for summarizing of the sequence, resulting in worse score when compared to experiment 4, where the average of the BERT output embeddings are used directly for prediction. Though from Section 5.1, we know that bi-LSTM is effective when BERT is not finetuned, the comparison between experiment 3 and 4 has shown the reversed result in the situation where BERT is finetuned. Such a finding suggests that, when finetuned, the attention mechanism in BERT might be robust enough for BERT's output to be directly used for prediction, and that the bi-LSTM layer only adds complexity to the model and cannot handle embeddings output from BERT properly.

5.4 The effectiveness of entity masking

Another way of boosting the performance for TACRED relation extraction can be found from comparing experiment 3 and 5. When the entities are masked with their entity type tokens, the result has 6 F1 score improvement. The discovery justifies the argument in Section 3.2.2, that it is better to mask variety of entity sequences into their entity type, so that the model does not only learn from specific sequences and overfit to them.

5.5 The choice of BERT output

Experiments 6, 7 and 8 are designed to explore the effectiveness of using different BERT output as prediction of relation extraction task. These three experiments are conducted without bi-LSTM; the selected embedding outputs of BERT model are fed directly to a perceptron layer before softmax. In experiment 6, only the embedding of [CLS] token is used for prediction, as it is supposed to represent the whole input sequence and can utilize the attention mechanism in BERT to collect important information between the entities. The result, however, is worse than experiment 7 by 1.3 F1 score. The reason can be that it is too taxing for [CLS] alone to collect important sequence and entity information from the input.

The experiment 7 uses concatenation of subject and object entity type token for prediction of their relation. It reaches highest result of 68.8 F1 score in all my experiments, demonstrating the effectiveness of directly using entities' embedding for prediction. Such a finding is intuitive as the entities in the sequence are the core elements for relation extraction. Meanwhile, its performance is on par with experiment 5 without needing the extra bi-LSTM layer, providing us a smaller yet equally robust solution to relation extraction.

Knowing that by simply choosing entity type tokens' BERT output can achieve impressive result, I want to explore whether by adding the information of the whole sequence will further improve the performance. Similar to the usage of concatenating output of [CLS] with the average of

entity token sequences for prediction in (Wu and He, 2019), experiment 8 uses the concatenation of [CLS] (representing the information of the whole sequence) and entity type tokens embeddings from BERT output for prediction. The result is still similar to experiment 7, meaning the concatenation of [CLS] is not necessary. And together with the finding from experiment 6, it seems that [CLS] token does not contribute much in the task of relation extraction.

5.6 Entity Appending or not?

Recall Section 5.2, the improvement achieved from experiment 2 to 3 can be attributed to either BERT model's inner mechanism help improve the performance under finetuning setup, or that finetuned BERT model is able to utilize entity appending in the input sequence for better prediction. The experiment 7 and 9 answer this question as they only differ in the input format; experiment 7 has entity appending together with entity type masking while experiment 9 only uses entity type masking. In fact, the input in experiment 9 do not content any texts of entities. The performance of these two models are similar, suggesting that BERT do not need to see the actual texts of the entities for relation prediction, thus making the importance of entity appending in input sequence negligible.

6 Conclusion

This paper aim to discuss whether the architectural addition of LSTM model in BERT relation extraction outweighs the configuration of BERT input format and output selection, and analyzes the results from different input format and output selection of BERT for further understanding their effects in relation extraction and inner work of BERT model. From the results, I conclude that bi-LSTM is not needed for the relation extraction model particularly when BERT is finetuned, as it merely increases the model complexity and computing resource need without providing any significant improvement of the model. Meanwhile, the subject and object entities in the input sequence should be mask with special entity type tokens for the generalizing ability of the model. Entity appending in the input is not needed, as the model can perform similarly without knowing the actual sequence of the entities. Lastly, using entity type tokens' embedding output is as good as using the whole sequence output or in concatenation with the [CLS] pooling output, meaning that the entity tokens embedding hold crucial information for relation extraction.

References

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2895–2905, Florence, Italy. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 39–48, Denver, Colorado. 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, Doha, Qatar. Association for Computational Linguistics.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer,

Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Shanchan Wu and Yifan He. 2019. Enriching pre-trained language model with entity information for relation classification. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 2361–2364, New York, NY, USA. Association for Computing Machinery.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45, Copenhagen, Denmark. Association for Computational Linguistics.

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 the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212, Berlin, Germany. Association for Computational Linguistics.