Augmenting Context Representation with Triggers Knowledge for Relation Extraction

En Li, Shumin Shi, Zhikun Yang, He Yan Huang

Augmenting context representation with triggers knowledge for Relation Extraction

En Li[1], Shumin Shi[1,2,*], Zhikun Yang[1], and HeYan Huang[1,2]

[1] School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
[2] Beijing Engineering Research Center of High Volume Language Information Processing and Cloud Computing Applications, Beijing, China
{3220190822,bjssm,3120191065,hhy63}@bit.edu.cn

Abstract. Relation Extraction (RE) requires the model to classify the correct relation from a set of relation candidates given the corresponding sentence and two entities. Recent work mainly studies how to utilize more data or incorporate extra context information especially with Pre-trained Language Models (PLMs). However, these models still face with the challenges of avoiding being affected by irrelevant or misleading words. In this paper, we propose a novel model to help alleviate such deficiency. Specifically, our model automatically mines the triggers of the sentence iteratively with the sentence itself from the previous iteration, and augment the semantics of the context representation from BERT with both entity pair and triggers skillfully. We conduct extensive experiments to evaluate the proposed model and effectively obtain empirical improvement in TACRED.

Keywords: triggers representation · knowledge augment · context aware · Relation Extraction.

1 Introduction

Relation Extraction (RE) aims to extract the organized relational knowledge in the shape of "knowledge graphs" from unstructured text [4]. For example, given the sentence "The *kitchen* is the last renovated part of the *house*" and a pair of nominals *kitchen* and *house*, the main goal of RE is to classify relation "*part_of*" from the context between these entity pair. It is the most powerful support for many downstream applications like graph completion, question answering, web search, information retrieval, path inference and logical rule reasoning [17].

Recently, self-attention such as Transformer [15] has also been explored for RE and has shown unexpectedly high performance. One popular paradigm of applying Transformer for RE is to leverage a single pre-trained language model which is pre-trained on large-scale unsupervised corpus, and fine-tune it on the specific task [7]. The other popular use case is to leverage the relational facts from knowledge graphs to guide relation selection.

Usually, models based on Transformer locate the target entity pair by replacing them with special tokens or inserting typed markers and incorporate

the corresponding feature transferred by Transformer to fit RE task. It seems like the researchers have formed a consensus that fusing the entity information is enough for identifying the correct relationship between the entity pair. However, these models fail to make correct extractions which are easy for human to understand, considerably hinder the performance of these fine-tuned models. In addition, utilizing existing relational facts is indeed a potential way towards more powerful RE models, but it has improved frustratingly slowly [16].

In this paper, we hope to alleviate this above problems by mining triggers of the sentence with the fine-tuned model from the previous iteration to make further use of training corpus. In other words, we introduce an auxiliary task that allows the model to score all tokens, augmenting context representation with real keywords which we define as triggers knowledge for RE and discard irrelevant tokens automatically.

The idea behind the auxiliary task is simple, we change the original sentence S into S_1 by randomly masking some tokens (t_{m1}, t_{m2}, ... , t_{mn}) and generate the labels y and y_1 with the same fine-tuned models. If the labels y and y_1 are different, we set the score of masked tokens (t_{m1}, t_{m2}, ... , t_{mn}) as 1 which means these tokens act as an import role in the current model. We repeat the same operation to predict each token an accurate score in the preprocessing step. We also locate the positions of the target entity pair to prevent them from being masked and finally concatenate these encodings including trigger, entity pair as well as the sentence encoding (embedding of the special first token in the setting of BERT [3]) as the input to a multi-layer neural network for classification.

We believe if the model can score the tokens correctly, its decision surface will be more robust about irrelevant or misleading words and encode more information about the keywords to obtain the better capability of classification. Hence, compared with previous models, our model can alleviate the wrong label problem by highlighting important tokens and do not need additional data.

In summary, the contributions can be summarized as follows: 1) We introduce a fresh perspective to mine the triggers of the sentence by exploring the fine-tuned model from the last iteration. 2) We instantiate the above model as an augment layer on the top of the pre-trained model. This allows the model to augment context representation with the knowledge of keywords and entity pair to combine their benefits. 3) Extensive experiments on TACRED [24] show that the proposed framework outperforms the previous methods, achieving the empirical results.

2 Related Work

The pioneering works on supervised RE research employed a hand-built pattern approach, designing specific features [1] or kernel functions [2] to extract corresponding semantic relation between the entity pair in the text [6]. However, these methods are very time-consuming, human-intensive and quickly replaced.

Later inspired by the success of deep learning models in other NLP tasks, the deep learning-based RE has been extensively studied, which improves the

performance significantly and promote the follow-up research greatly. Studies in deep learning mainly focus on designing complex matching networks to model the relationship among text, entities and relations. [21] creatively come up with the concept of position embedding to specify the relative distances between words and entities and apply it to an end-to-end convolutional neural network (CNN), which shows promising results. To better handle long-distance dependency and time sequence between entity pair, [22] combine the concept of recurrent neural networks (RNN) for RE, which perform a max-pooling operation to effectively model feature extraction for prediction. [9] further use the weight of attention mechanism to aggregate global relational information. In order to consider the dependencies between entities, [23] adopt graph neural networks (GNN) over dependency trees to build entity graphs and identify the correct relations by inference models.

As compared to deep learning, RE is greatly enhanced by Pre-trained Language Models (PLMs), which benefits from bidirectional Transformer layers [15] and are pre-trained on large-scale unsupervised corpus [12]. Recent work on RE can be roughly divided into two categories. One focuses on fine-tuning pretrained language models on text with linked entities using relation-oriented objectives. [13] simply replace the entity mentions with special masks before feeding the text to BERT for fine-tuning, providing strong baseline for future research. [19] further incorporate entity-level information into the pretrained language model by insert special tokens before and after the target entities. [14] explore whether two relation instances share the same entities by proposing a matching-the-blanks objective and achieve new state-of-the-arts. The other line of work mainly studies injecting external context information into pre-trained language models. Methods of such, including Know-BERT [11] and ERNIE [25], align entities to their corresponding entities in KGs by encoding the graph structure and take the informative entity embeddings as input to the Transformer. Similarly, to improve the description accuracy of relation vectors, K-Adapter [18] injects factual and linguistic knowledge by introducing a plug-in neural adaptor. LUKE [20] further extends the pre-training objective of masked language modeling to entities and proposes an entity-aware self-attention mechanism.

3 Approach

In this section, we first formally introduce the problem of Relation Extraction and the input format. Then we present an overview of the proposed model and present each module in detail.

3.1 Problem Define

For supervised Relation Extraction, the input is a sentence S consisting of n tokens t_1, t_2,...,t_n, an entity e_1 with the span (i, j) and another entity e_2 with the span (p, q). The task is, for the target entity pair, to predict a correct relation from candidates. Usually let R denote a set of pre-defined relation labels

(including *no_relation*). Then the output of the task is a structured triples Y_r = $\{(e_1, e_2, \mathrm{r}) : e_1, e_2 \in S, r \in R\}$.

3.2 Input format

To make the model better capture the dependencies between the subject and object, we insert the special markers at the beginning and end of the entities. Specifically, we define special makers as $ and # and insert them before and after the subject and object, therefore modifying the input text to the format of "The $ kitchen $ is the last renovated part of the # house #".

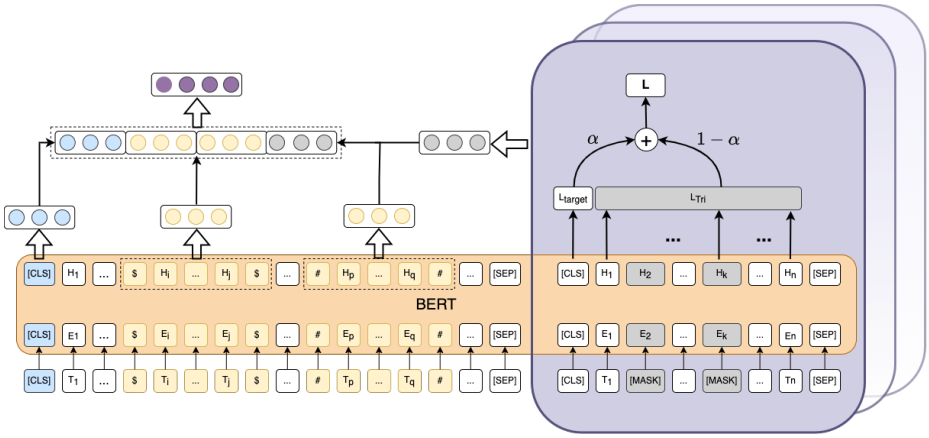

Fig. 1. Our Model Architecture

3.3 Model Architecture

As shown in Fig. 1, our approach consists of a triggers generation task and a relation classification task. The former first takes the input sentence and generate a batch of masked sentences by randomly masking certain tokens, trying to score all the tokens to distinguish the real keywords and irrelevant words by a loss function L. To be specific, if the predictions of the augmented sentence and original sentence are consistent, these masked tokens will be de-emphasized for achieving better result or they will be considered as keywords in inference step. The later task mainly captures both the semantics of the sentence and the triggers mined in the former task to better fit the classification task.

Definition of the triggers generation task The triggers generation step in training phase is described as follows. Given a sentence S with special markers, we construct a batch of masked sentences$(S'_1, S'_2, \ldots, S'_n)$ and predict the relation

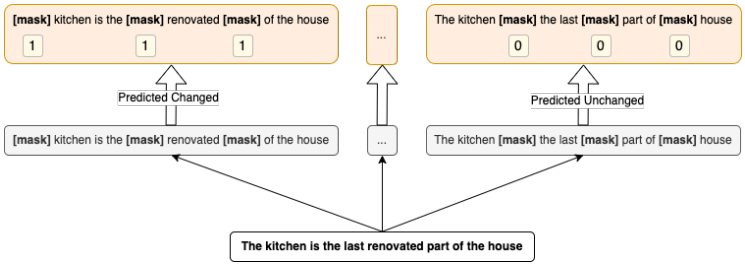

Fig. 2. An example of the weak token labels generation procedure

of all sentences with the same model. If the predicted labels are same, the masked tokens will be set to be 0. Otherwise, we will set the corresponding important tokens to be 1 as they have much impact on target task. Besides, to get rid of noises and make the label prediction more accurate, we abandon the training data that the predicted label of original sentence is wrong. To speed up the time of the preprocessing, we generate augmented sentences incrementally with a generation ratio β and larger means generate more sentences each epoch. Then we set the score for each token according to whether the label is same with the original sentence. The amount of masked tokens is controlled by a proportional parameter, which is related to the length of original sentence and set to 0.3 empirically.

Fig. 2 shows an example of the weak token labels generation procedure, given the sentence "The *kitchen* is the last renovted part of the *house*", we generate the masked sentence S_1' = "[mask] $ kitchen $ is the [mask] renovated [mask] of the # house #" or S_2' = "The $ kitchen $ [mask] the last [mask] part of [mask] # house #". S_1' does not change the prediction label, so the masked tokens are labeled by 0, that is, Y_1 = (_; 0; _; 0; _; 0; _), where "_" does not contribute to the loss function. On the other hand, S_2' flips the original prediction of S, so we have Y_2 = (1; _; 1; _; 1; _; 1). After getting the binary output vector, we fine-tune the triggers generation task with cross-entropy loss to pick out real triggers:

$$\mathcal{L}_{TRI} = -\sum_i l_{TRI}(y_i, y_i^t) \tag{1}$$

$$y_i^t = \sigma(w_{TRI}^i M(t_i)) \tag{2}$$

Where $M(t_i)$ denotes the model from the previous epoch, w_{TRI} is the fully connected layer of the i-th token for triggers generation task, σ is a softmax operation, y_i denotes the weak label of token t_i.

Co-Training framework One straight-forward way to make full use of the triggers is training the target task with triggers generation task jointly. Intuitively, if some keywords highlight the essence of the sentence exists in training data,

we can readjust the weight of the tokens according to their relative importance to the target task and guide the model to capture more important information. Then we jointly optimize the two objectives in the training stage, the overall loss can be defined as a linear combination of two parts:

$$\mathcal{L} = \alpha \mathcal{L}_{target} + (1 - \alpha)\mathcal{L}_{\mathcal{TRI}} \tag{3}$$

$$\mathcal{L}_{target} = \sum_{i=1} l_{target}(y_i, y_i^s) \tag{4}$$

where l_{target} is the loss function of the target task; y_i and y_i^s denote the actual label of sentence s_i and the predicted label for the target task respectively; $\mathcal{L}_{target}$ denotes the loss function of the target task while $\mathcal{L}_{\mathcal{TRI}}$ represents the triggers generation task; α is a linear combination ratio which controls the relative importance of two losses.

After assigning the corresponding weight to each word, we will extract the words that help improve the target prediction and combine with the original model. Specifically, given a sentence S with entity e_1 and e_2, suppose its final hidden state output from BERT module is H. Then H_i to H_j are the final hidden state vectors from BERT for entity e_1, and H_p to H_q are the final hidden state vectors from BERT for entity e_2. We can get a vector representation for each of the two target entities by applying the average operation to corresponding vectors. Then each of the two vectors are fed into a feedforward network after an activation operation (i.e. $tanh$), and the outputs for e_1 and e_2 are H_1' and H_2' respectively:

$$H_1' = W_1 \left[tanh \left(\frac{1}{j - i + 1} \sum_{t=i}^{j} H_t \right) \right] + b_1 \tag{5}$$

$$H_2' = W_2 \left[tanh \left(\frac{1}{q - p + 1} \sum_{t=p}^{q} H_t \right) \right] + b_2 \tag{6}$$

To obtain a vector H_0' as the representation of the aggregate sequence, we do the same thing as before for the hidden state of first special token $[CLS]$:

$$H_0' = W_0 \left(tanh \left(H_0 \right) \right) + b_0 \tag{7}$$

To further leverage the information of the triggers, we apply a weighted sum of the reweighted tokens to get a single vector representation following with a $tanh$ activation operation and a fully connected layer.

$$H_t' = W_t \left[tanh \left(\frac{1}{k} \left(y_1^t * H_{t1} + ... + y_k^t * H_{tk} \right) \right) \right] + b_t \tag{8}$$

Where k means a total of k trigger tokens in this sentence mined and y_k^t represents the corresponding score reweighted for each token.

We concatenate H'_0, H'_1, H'_2, H'_t following a fully connected layer and a softmax layer, which can be expressed as following:

$$h = W_3 \left[concat \left(H'_0 . H'_1 . H'_2 . H'_t \right) \right] + b_3 \tag{9}$$

$$p = softmax \left(h \right) \tag{10}$$

Matrices W_0, W_1, W_2, W_t have the same dimensions, i.e. $W_0 \in R^{dd}, W_1 \in R^{dd}, W_2 \in R^{dd}$ and $W_3 \in R^{L4d}$, where d is the hidden state size from BERT and L is the number of relation types; p is the probability output; b_0, b_1, b_2, b_3, b_t are bias vector and we apply dropout before each fully connected layer during training.

4 Experiement

4.1 Dataset and Evaluation Metric

We evaluate the framework on TACRED [24] dataset in our experiments. TACRED was originally produced by human annotations with 106,264 examples built over English newswire and web text used in the TAC KBP English slot filling evaluations during the period 2009-2014. The dataset contains 41 semantic relation types and one artificial relation type $no\_relation$, which means that the relation does not belong to any of the 41 relation types. Besides, we evaluate Precision (P), Recall (R), and F1 scores following official suggestions in [24].

4.2 Implementation Details

In our experiments, we use the uncased basic model for the pre-train BERT model and tune all hyper-parameters based on F1 score on development set. We trained our model with 5 epochs and set learning rate as 2e-5. To accelerate the training speed, the maximum sequence length is set to 128 in our experiments and the extra length will be cut in each batch. Besides, we add dropout before each encoder layer and BertAdam optimizer is used. Further, we employ rigorous experiments to find the optimal hyper-parameters: loss combination ratio $\alpha \in \{0.7, 0.9\}$ and data generation ratio $\beta \in \{0.6, 1.0, 2.0\}$.

4.3 Compared Methods

We compare out method against results by multiple classic methods recently published:

– **PA-LSTM** [24] creatively combines the bi-directional LSTM [5] with position-aware attention to encode the text into an embedding, which is then fed into a softmax layer to predict the relation.

- **C-GCN** [23] proposes an extension of graph convolutional network which pools information over arbitrary dependency structures and apply a novel pruning strategy to the input trees by keeping words immediately around the shortest path[10] between the two entities to obtain the representation of entities.
- **BERT-BASE** [13] is the first to successfully apply BERT in relation extraction. They concatenate the embedding of the BERT with position embedding and the final prediction is based on the concatenation of the final hidden state in each direction from the BiLSTM, fed through an MLP.
- **BERT-EM** [14] explores variants of architectures for extracting representations from deep Transformers network and present a pre-trained training objective of matching the blanks. In our experiment, we reimplement it without the pre-trained task.
- **R-BERT** [19] is a model that locate the target entities and transfer the information through the pre-trained architecture following incorporate the corresponding encoding of the two entities into the pretrained language model for relation classification.
- **SpanBERT** [8] extends BERT by introducing a training objective of span prediction and replacing the entity pair by their NER tags, achieving improved performance on RE.

4.4 Result and Analyse

Table 1. Experimental results on TACRED

Model	**TACRED**		
	Prec.	*Rec.*	$F1$
sequence-based Models			
PA-LSTM	67.7	63.2	65.4
C-GCN	69.9	63.3	66.4
$Transformer$-*based* Models			
BERT_{BASE}	73.3	63.1	67.8
BERT_{EM}	69.4	66.8	67.9
R-BERT	71.9	62.5	67.3
SpanBERT	70.2	66.3	68.2
Our Model	**71.3**	**65.4**	**68.7**

Table 1 shows the experimental results of different models on TACRED. All experiments are based on the publicly available implementation of base version of $BERT_{BASE}$ as the encoder, and we rerun their officially released code using the recommended hyper-parameters in their papers. There is no doubt that Transformer-based models surpasses all sequence-based models, so we mainly compare our model with some classical Transformer-based models. Besides, we want to demonstrate the specific contributions by the components besides the

pre-trained BERT component. For this purpose, we compare our model with $BERT_{BASE}$, $BERT_{EM}$ and *R-BERT* respectively. As we can see, our model gets much improvement compared to $BERT_{BASE}$ and $BERT_{EM}$, which demonstrates the strong empirical results based on the proposed approach cause these two other models merely use the context representation enhanced by BERT for relational classification. We infer that it is because the representation of the $[CLS]$ is just a general sentence representation rather than a maximum adaptation to relation extraction. It also can be observed that our model achieves 1.4 F1 absolute points better than *R-BERT*, indicating that score generation can promote the accuracy of the model not rely solely on the augment of the entity pair. Besides, we compare our own model with the latest pre-trained language model *SpanBERT* and achieve comparable results, which will be the direction of our future research.

Table 2. Results with different settings on TACRED

Settings	**TACRED**		
	Prec.	*Rec.*	*F*1
DEFAULT	**71.3**	**65.4**	**68.7**
w/o triggers knowledge	70.0	65.0	67.4
w/o separate tokens	72.3	60.7	66.0
w/o entities	67.1	61.8	64.3

4.5 Triggers Knowledge study

In this part, we first analyze the method of incorporating triggers knowledge and then we detect the effectiveness of different label generation ratio.

For triggers knowledge, we create three more different settings as Table 2. "w/o triggers knowledge" means we don't perform treatment on data features in preprocessing step. The second configuration is to discard the special separate tokens (i.e. '$' and '#') but keep the average pooling of entities representation. "w/o entities" just takes the hidden vector output of the "[CLS]" for classification. From the ablation study, we get the observation that when one component is discarded, the performance will decline with varying degrees. Without triggers knowledge, the performance drops sharply which demonstrates the triggers knowledge incorporated is useful for this task. Special separate tokens are also important, we infer that an early fusion of separate tokens can further improve performance cause they transfer the location information of entities into the model. Of the methods, "w/o entities" has the worst performs, with its almost 4.4 F1 points worse than our model, which means entity information makes important contributions to our approach.

For label generation ratio, we conduct experiments under different generation ratio β as Fig. 3. The larger β means each epoch more masked sentences are generated as training samples. $BERT_{MASK}$ means just carry out the token

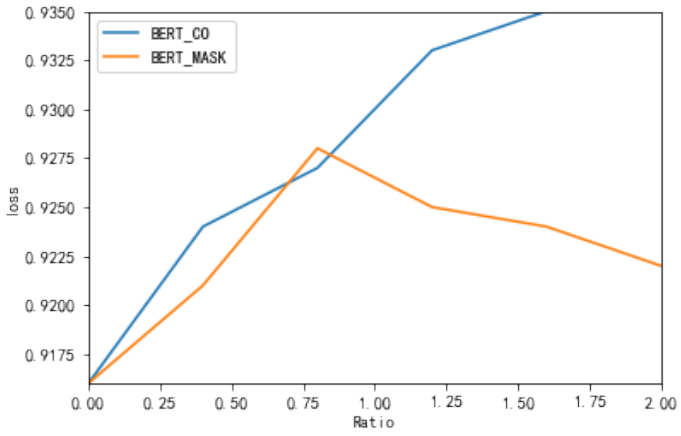

Fig. 3. Experiments on different generation ratio

score generation study for BERT model but don't augment the sentence vector with keywords. In the beginning, the two models continue to improve with more masked sentences generated. However, the performance of $BERT_{MASK}$ degrades dramatically after β is greater than 0.8. One possible reason is that identifying relevant words and merging the reweighted tokens directly will influence the robustness of our model, resulting in sufficient learning on pre-trained knowledge.

4.6 Training cost comparison

One possible questioned shortcoming of our method is the extra training cost of the token score generation. To further evaluate the extra cost, we conduct extended experiments on loss value of specific time scaled. The detailed results on three different models are presented in Table 3. Note that our method contains more parameters, so it converges slower at the beginning. However, our model starts to achieve a lower loss than other models in later iterations since it can provide complementary information for BERT. Besides, as shown in Fig. 3, compared with the total training time, this cost is completely acceptable since the final performance gain justifies the extra training cost.

Table 3. Experiments on loss value of specific time scaled

Time/min	20	40	60	80
BERT_{BASE}	0.588	0.113	0.016	0.015
R-BERT	0.633	0.105	0.014	0.011
Our Model	**0.701**	**0.128**	**0.012**	**0.008**

5 Conclusion

In this paper, we present a simple but effective approach to incorporate triggers knowledge for Relation Extraction. Triggers generation task automatically produces token-level attention labels and picks out real keywords by probing the fine-tuned model from the previous iteration. We further integrate the semantics of entity pair and triggers knowledge to augment the sentence representation. We conduct experiments on the TACRED benchmark dataset and achieve competitive results. In future work, we will extend to span-level keywords augmentation.

Acknowledgement

The authors wish to thank the reviewers for their helpful comments and suggestions.This work was also supported by the National Key Research & Development Program (Grant No. 2018YFC0831700) and National Natural Science Foundation of China (Grant No. 61671064, No. 61732005)

References

1. Alicante, A., Corazza, A.: Barrier features for classification of semantic relations. In: Proceedings of the International Conference Recent Advances in Natural Language Processing 2011. pp. 509–514 (2011)
2. Bunescu, R., Mooney, R.: A shortest path dependency kernel for relation extraction. In: Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing. pp. 724–731 (2005)
3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
4. Han, X., Gao, T., Lin, Y., Peng, H., Yang, Y., Xiao, C., Liu, Z., Li, P., Sun, M., Zhou, J.: More data, more relations, more context and more openness: A review and outlook for relation extraction. arXiv preprint arXiv:2004.03186 (2020)
5. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
6. Huffman, S.B.: Learning information extraction patterns from examples. In: International Joint Conference on Artificial Intelligence. pp. 246–260. Springer (1995)
7. Jiang, H., Cui, L., Xu, Z., Yang, D., Chen, J., Li, C., Liu, J., Liang, J., Wang, C., Xiao, Y., et al.: Relation extraction using supervision from topic knowledge of relation labels. In: IJCAI. pp. 5024–5030 (2019)
8. Joshi, M., Chen, D., Liu, Y., Weld, D.S., Zettlemoyer, L., Levy, O.: Spanbert: Improving pre-training by representing and predicting spans. Transactions of the Association for Computational Linguistics **8**, 64–77 (2020)
9. Lin, Y., Shen, S., Liu, Z., Luan, H., Sun, M.: Neural relation extraction with selective attention over instances. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 2124–2133 (2016)
10. Liu, Y., Wei, F., Li, S., Ji, H., Zhou, M., Wang, H.: A dependency-based neural network for relation classification. arXiv preprint arXiv:1507.04646 (2015)

11. Peters, M.E., Neumann, M., Logan IV, R.L., Schwartz, R., Joshi, V., Singh, S., Smith, N.A.: Knowledge enhanced contextual word representations. arXiv preprint arXiv:1909.04164 (2019)
12. Sarzynska-Wawer, J., Wawer, A., Pawlak, A., Szymanowska, J., Stefaniak, I., Jarkiewicz, M., Okruszek, L.: Detecting formal thought disorder by deep contextualized word representations. Psychiatry Research **304**, 114135 (2021)
13. Shi, P., Lin, J.: Simple bert models for relation extraction and semantic role labeling. arXiv preprint arXiv:1904.05255 (2019)
14. Soares, L.B., FitzGerald, N., Ling, J., Kwiatkowski, T.: Matching the blanks: Distributional similarity for relation learning. arXiv preprint arXiv:1906.03158 (2019)
15. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
16. Verga, P., Belanger, D., Strubell, E., Roth, B., McCallum, A.: Multilingual relation extraction using compositional universal schema. arXiv preprint arXiv:1511.06396 (2015)
17. Wang, H., Lu, G., Yin, J., Qin, K.: Relation extraction: A brief survey on deep neural network based methods. In: 2021 The 4th International Conference on Software Engineering and Information Management. pp. 220–228 (2021)
18. Wang, R., Tang, D., Duan, N., Wei, Z., Huang, X., Cao, G., Jiang, D., Zhou, M., et al.: K-adapter: Infusing knowledge into pre-trained models with adapters. arXiv preprint arXiv:2002.01808 (2020)
19. Wu, S., He, Y.: Enriching pre-trained language model with entity information for relation classification. In: Proceedings of the 28th ACM international conference on information and knowledge management. pp. 2361–2364 (2019)
20. Yamada, I., Asai, A., Shindo, H., Takeda, H., Matsumoto, Y.: Luke: deep contextualized entity representations with entity-aware self-attention. arXiv preprint arXiv:2010.01057 (2020)
21. Zeng, D., Liu, K., Lai, S., Zhou, G., Zhao, J.: Relation classification via convolutional deep neural network. In: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers. pp. 2335–2344 (2014)
22. Zhang, D., Wang, D.: Relation classification via recurrent neural network. arXiv preprint arXiv:1508.01006 (2015)
23. Zhang, Y., Qi, P., Manning, C.D.: Graph convolution over pruned dependency trees improves relation extraction. arXiv preprint arXiv:1809.10185 (2018)
24. Zhang, Y., Zhong, V., Chen, D., Angeli, G., Manning, C.D.: Position-aware attention and supervised data improve slot filling. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. pp. 35–45 (2017)
25. Zhang, Z., Han, X., Liu, Z., Jiang, X., Sun, M., Liu, Q.: Ernie: Enhanced language representation with informative entities. arXiv preprint arXiv:1905.07129 (2019)