Enhancing Relation Extraction by Using Shortest Dependency Paths Between Entities with Pre-trained Language Models

Haluk Alper Karaevli*†, Tunga Güngör*
*Department of Computer Engineering, Boğaziçi University, İstanbul, Turkey
† AI Enablement Department, Turkey R&D Center, Huawei Technologies

***Abstract*—Relation Extraction (RE) is the task of finding the relation between entities in a plain text. As the length of the sentences increases, finding the relation becomes more challenging. The shortest dependency path (SDP) between two entities, obtained by traversing the terms in the dependency tree of a sentence, provides a view focused on the entities by pruning noisy words. In the supervised form of the relation extraction task, Relation Classification, the state-of-the-art methods generally integrate a pre-trained language model (PLM) into their approaches. However, none of them incorporates the shortest dependency paths to the best of our knowledge.**

This paper investigates the effects of using shortest dependency paths with pre-trained language models by taking the R-BERT relation classification model as the baseline and building upon it. Our novel approach enhances the baseline model by adding the sequence representation of the shortest dependency path between entities, collected from PLMs, as an additional embedding. In the experiments, we evaluated the proposed model's performance for each combination of SDPs generated from Stanford, HPSG, and LAL dependency parsers with BERT and XLNet PLMs in two datasets, SemEval-2010 Task 8 and TACRED. We improved the baseline model by absolute 1.41% and 3.60% scores, increasing the rankings of the model from 8^{th} to 7^{th} and from 18^{th} to 7^{th} in SemEval-2010 Task 8 and TACRED, respectively.

***Index Terms*—Relation extraction, Dependency parse, Shortest dependency path, Pretrained language models.**

I. Introduction

THE internet is the most significant information source of our current world. However, most of the data it contains is in an unstructured text form which requires additional processing to be understandable by the machines. The tasks defined for such needs are gathered under the topic of Information Extraction. One of such tasks is Relation Extraction, in which the objective is to find the relations between entities given the plain text of a document.

The task of Relation Extraction is assessed in three general manners; supervised, distantly-supervised, and unsupervised. In supervised approaches, the aim is to predict the correct relation between entities in a sentence from a fixed number of relations. Distantly supervised methods have the same objective as their supervised peers, but a set of sentences is taken as input instead of one sentence. Not all sentences in a particular set represent relation between entity pairs. In unsupervised approaches, none of the entity tuples, the relations, or the corresponding sentences for the entities are known beforehand.

978-1-6654-9810-4/22/$31.00 ©2022 IEEE

Dependency parsers have been used in various aspects by the models from all approaches of Relation Extraction [1]–[7]. By providing a structured representation of the raw text using contextual connections, dependency parsers enable models to focus on the interactions between terms.

Shortest dependency path (SDP) is defined as the path with minimum contextual connections between two entities in the dependency tree. For example, for the sentence "The house at the end of the street is red." the SDP text constructed by combining the shortest dependency path between the words "house" and "red" would be "house is red".

In Relation Classification, the state-of-the-art methods generally integrate a pre-trained language model [8]–[10]. Pre-trained Language Models (PLMs) are language models trained on large corpora to learn contextual semantics of words without focusing on a specific task.

In this paper, the effects of using shortest dependency paths with pre-trained language models are investigated in the supervised relation extraction domain. The relation classification model R-BERT [11] is chosen as the baseline for this task. In R-BERT, the sentence and the entities are represented as separate vectors concatenated in the last layer. Because of its compartmentalized architecture, adding a new feature to the model is a considerably straightforward process.

We aim to improve R-BERT's performance by integrating PLM representation of the SDP between target entities to the input of R-BERT's last layer. Three dependency parsers named Stanford [12], HPSG [13], and LAL [14] are used to generate the SDPs. In addition to integrating the shortest dependency paths, XLNet [15] is applied as an alternative to BERT as the pretrained language model of R-BERT. In selection of XLNet, its improved performance over BERT in various nlp tasks and its autoregressive structure are considered. The experiments are conducted on SemEval-2010 Task 8 dataset [16], which is the most commonly used dataset in relation classification task, and TACRED [17] dataset constructed by Stanford University.

Our contribution to the domain of relation classification is an improved R-BERT model that surpasses the performance of the baseline system in both SemEval and TACRED datasets. According to the results we received, in SemEval we improve the standing of the model from 8^{th} place to 7^{th} place. A

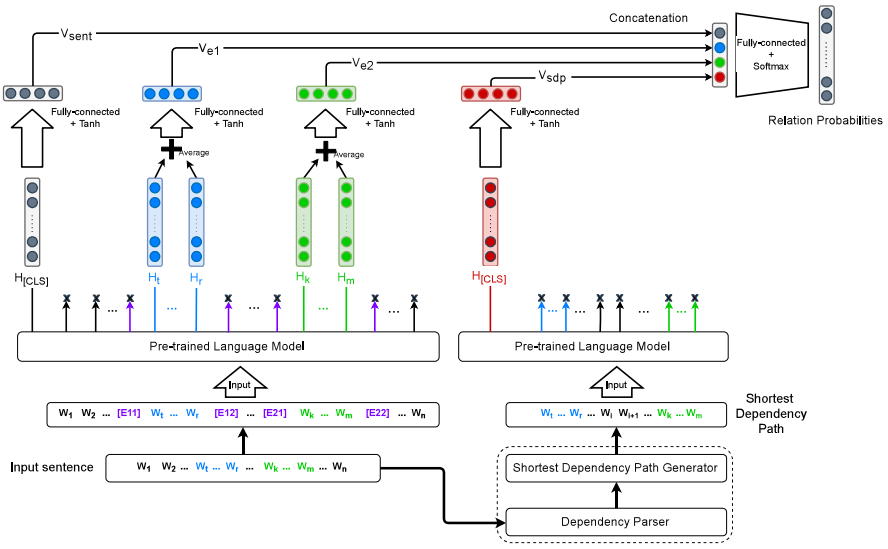

Fig. 1. Architecture of the proposed model. The dotted box shows the preprocessing steps.

greater change of standing is achieved in TACRED, from 18^{th} place (unofficial) to 7^{th}.

The source code of the proposed model can be found at https://github.com/HalukKaraevli/R-BERT-SDP

The paper is organized as follows: In Section 2 related works in the relation classification domain are given. In Section 3 the methodology of the model and the algorithm used to generate shortest dependency paths is given. In Section 4 the results of the experiments are presented. Section 5 concludes the paper.

II. Related Work

The depLCNN model of Xu et al. [18] aims to learn robust relation representations of the entities by feeding their shortest dependency paths through a convolutional neural network. In addition, they propose a novel negative sampling strategy addressing the relation directionality that finds the subject and object entity of the relation.

Liu et al. [6] focus on using information acquired from the dependency tree of the sentences in determining the relations between entities. A new representation of Augmented Dependency Path (ADP) and dependency-based neural network (DepNN) are proposed.

The contribution of Xu et al. [19] is the usage of "deep" recurrent neural networks in relation extraction. They argue that by using individual RNN's in each depth level for the shortest dependency path representation of the sentence, one can acquire the "whole" representation of the text which would contain information from different levels of abstraction.

In Lee et al. [20], entity-aware attention mechanism adds the attention weights of entity pairs to the representation of each word by using the relative positions of the words to the entities and the "latent types" of the entities acquired from Latent Entity Typing.

Similar to Lee et al. [20], Wu et al. [11] propose a way to incorporate entity-level information into the pre-trained language model to achieve an increase in performance of relation classification. Different from Lee et al. [20], the entity representations which are generated using the BERT pre-trained language model are explicitly fed to the classifier model.

III. Methodology

The general flow of the proposed model is shown in Fig. 1. We represent a sentence as $s = (w_1 w_2 \dots w_t \dots w_r \dots w_k \dots w_m \dots w_n)$, where w_x denotes the x^{th} word, t and r are the indexes of the first and last words that constitute the first entity, and k and m are the respective indexes of the second entity. To obtain the shortest dependency path, as shown on the right part of the figure, the sentence is given to a dependency parser and the outputs of the parser are fed to the SDP generator accompanied with the target entities. The SDP of the sentence is generated as $SDP_s = (w_t \dots w_r \dots w_k \dots w_m)$.

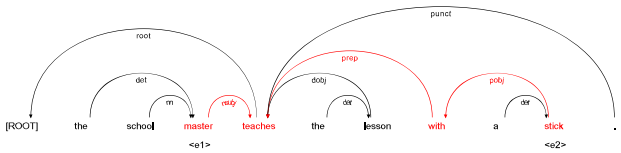

Fig. 2. Dependency tree representation of the sentence "The school master teaches the lesson with a stick." and the shortest dependency path.

The process for generating SDP from dependency parse output is explained below with an example sentence "The school

master teaches the lesson with a stick." whose dependency tree can be seen in Fig. 2:

1) Find the tokens that constitute each entity in the dependency tree. Ex: $e_1 = [master], e_2 = [stick]$
2) For each such token, generate the list of parents from the root to the token. Ex: $parent_{e_1} = [[root, teaches, master]]$, $parent_{e_2} = [[root, teaches, with, stick]]$
3) For token pairs (t_1, t_2) where t_1 and t_2 are the tokens of the first and second entities, compare items in the same index in the parents lists of the two tokens and discard previously compared items until a mismatch occurs. If no mismatch occurs, go to step 6. Ex: $t_1 = [root, teaches, master]$, $t_2 = [root, teaches, with, stick]$ Compare $root$. Continue. Compare $teaches$, Discard $root$. Compare $master$ and $with$. Mismatch occurs. Go to 4.
4) Put t_1's remaining parents list into a new list res in reversed order, without the last common item. Ex: $res = [master]$
5) Append t_2's remaining parents list (last common item included) to the output of the 4^{th} step. Go to step 7. Ex: $res = [master, teaches, with, stick]$
6) If the parents list of t_1 is emptied, return t_2's remaining terms. If t_2's parents list is emptied, then return the reverse of the remaining items in t_1's list. Include the last compared item in both cases.
7) Apply steps 3-6 for each token pair.
8) Take the shortest list as the shortest dependency path.
9) Replace the first and last items with the first and second entities.
10) Return the string of items joined with space.

Besides forming the shortest dependency path in preprocessing step, to identify the spans of the entities by the sentence encoder (i.e. the pre-trained model that receives the whole sentence), special tokens of [E11], [E12], and [E21], [E22] showing the starting and ending points of the first and second entities, respectively, are added to the sentence. With the addition of the special tokens, the sentence s becomes $s' = (w_1 w_2 \dots [\text{E11}] w_t \dots w_r [\text{E12}] \dots [\text{E21}] w_k \dots w_m [\text{E22}] \dots w_n)$, as shown on the left of the figure.

The sentence encoder (left part of the figure) tokenizes the tagged sentence, generates input mask and entity masks (where the tokens of a particular entity are 1, others are 0), and produces the embeddings H_i for all tokens and [CLS] which corresponds to the sentence embedding. The SDP encoder (right part of the figure), on the other hand, produces only the [CLS] embedding. Note that in the XLNet version, a module named sequence summary, which is a fully-connected layer with tanh activation, is applied to the <cls> (the XLNet equivalent of [CLS]) tokens' last hidden state.

The final embeddings for entities are produced by averaging the embeddings of the tokens of the entities. The averaged embedding is then fed to a fully-connected layer that uses tanh as the activation function. Thus, the equations for entity embeddings are as follows:

$$V_{e_1} = W_1 \left[\tanh \left(\frac{1}{r-t+1} \sum_{i=t}^{r} H_i \right) \right] + b_1$$

$$V_{e_2} = W_2 \left[\tanh \left(\frac{1}{m-k+1} \sum_{i=k}^{m} H_i \right) \right] + b_2$$

where W_1, W_2 and b_1, b_2 are weights and biases of the model.

The sentence and SDP embeddings also pass through their respective fully-connected layer with tanh activation:

$$V_{sent} = W_0 \left(\tanh \left(H_{[CLS]_{sent}} \right) \right) + b_0$$

$$V_{sdp} = W_3 \left(\tanh \left(H_{[CLS]_{sdp}} \right) \right) + b_3$$

All W_1, W_2, W_0, and W_3 have the same dimension. $W_i \in R^{dxd}$, where d is the dimension of the embeddings received from the pre-trained model.

Finally, the outputs from the fully-connected layers are concatenated and fed through a softmax layer:

$$h' = W_4 \left[\text{concat} \left(V_{sent}, V_{e_1}, V_{e_2}, V_{sdp} \right) \right] + b_4$$

$$p = \text{softmax} \left(h' \right)$$

The dimension of the resulting probability distribution p from the softmax layer is equal to the number of relations in the system. The loss function is chosen as the negative log-likelihood of the probability of the correct relation.

IV. Experiments

We evaluated the performance of the proposed approach on two datasets; SemEval-2010 Task 8 and TACRED. SemEval-2010 Task 8 dataset [16] consists of nine relation classes named as cause-effect, instrument-agency, product-producer, content-container, entity-origin, entity-destination, component-whole, member-collection, and message-topic and a no-relation class for negative samples. It has 10717 instances divided into train and test sets having 8000 and 2717 samples, respectively. TACRED [17] dataset has 41 relation classes consisting of 106,264 instances further stratified into train, dev and test sets with 68124, 22631, and 15509 instances. The classes in TACRED focus on organizations and people while in SemEval-2010 Task 8 we see broader, more general relations. As for the evaluation metrics that are used in the experiments, we follow the previous papers using the given datasets to be able to compare our performances.

The shortest dependency path texts are generated using the stanza library for the Stanford parser and the official GitHub repositories for the HPSG and LAL parsers. For the results acquired from the HPSG and LAL parsers, the weights of the best-performing models are taken from their respective repositories.

The experiments are conducted on two GPUs. For SemEval results, variants with base pre-trained models are trained with a GTX1060, others are trained with an RTX3090. For TACRED, all training processes are done with the RTX3090.

The source code of the model and the shortest dependency path generation process can be found in the link given in Section 1.

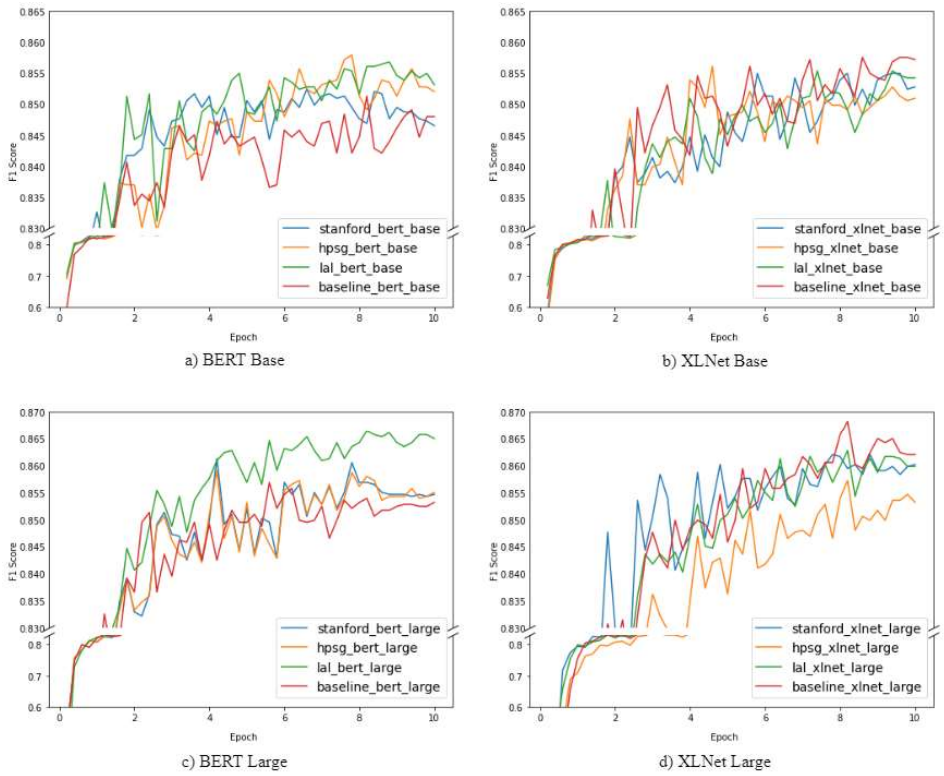

Fig. 3. Micro F1 scores of baseline and SDP enhanced models for each pre-trained language model on SemEval dataset.

A. *Experiments on SemEval-2010 Task 8*

In SemEval experiments, both the base version and the large version of the pre-trained language models BERT and XLNet are used with the baseline model and also with the three parsers. In this way, we build and compare 16 models. Each model is trained for ten epochs that are further divided into five checkpoints. The performance of the models are evaluated at each checkpoint. The learning rate of 2e-5 is applied to the models using the BERT pre-trained model, while this value is chosen as 1e-5 for the XLNet models. The dropout and batch size are, respectively, 0.1 and 16 in every model.

We provide the results in terms of both Micro F1 score and also Macro F1 score which was the official evaluation metric used in the shared task. In Micro F1 results the no-relation class is included. On the other hand, in the official Macro F1 results the no-relation class is omitted. In Micro F1 calculations, relation and its direction (e1 to e2 or e2 to e1) together are considered as a unique class. Thus, having two directions for each relation the no-relation class totals to 19 classes. In official Macro F1 calculations, only the relations are considered as unique classes (9 classes). But in order to correctly predict a sample's relation class, the model should also find the direction of the relation between entities, otherwise even though the relation is correct the sample is considered as a false prediction.

In Fig. 3, Micro F1 scores of the baseline and the three SDP enhanced models are compared for each pre-trained language model seperately. We observe that among all the models that use BERT$_{base}$ and BERT$_{large}$, the ones with the LAL parser achieve better Micro F1 scores than their peers.

Surprisingly, in the XLNet versions, the baseline models

TABLE I
OFFICIAL MACRO F1 SCORES OF THE MODELS ON SEMEVAL-2010 TASK 8 DATASET

Model	Epoch (Checkpoint)	Micro F1 w/ no-relation (19 Class)	Macro F1 w/o no-relation (Official)
BERT$_{base}$-baseline (original paper)	-	-	89.25 %
BERT$_{base}$-baseline	8 (1)	85.13 %	88.54 %
BERT$_{base}$-stanford	6 (3)	85.24 %	88.53 %
BERT$_{base}$-hpsg	7 (4)	85.79 %	89.09 %
BERT$_{base}$-lal	8 (4)	85.68 %	88.80 %
BERT$_{large}$-baseline	5 (3)	85.68 %	89.02 %
BERT$_{large}$-stanford	4 (1)	86.12 %	**89.90 %**
BERT$_{large}$-hpsg	4 (1)	85.90 %	88.75 %
BERT$_{large}$-lal	8 (1)	86.64 %	89.84 %
XLNet$_{base}$-baseline	8 (3)	85.76 %	88.80 %
XLNet$_{base}$-stanford	5 (4)	85.50 %	88.75 %
XLNet$_{base}$-hpsg	4 (3)	85.61 %	88.79 %
XLNet$_{base}$-lal	7 (2)	85.54 %	88.68 %
XLNet$_{large}$-baseline	8 (1)	86.82 %	89.83 %
XLNet$_{large}$-stanford	7 (4)	86.20 %	89.33 %
XLNet$_{large}$-hpsg	8 (1)	85.72 %	88.80 %
XLNet$_{large}$-lal	8 (1)	86.27 %	89.08 %

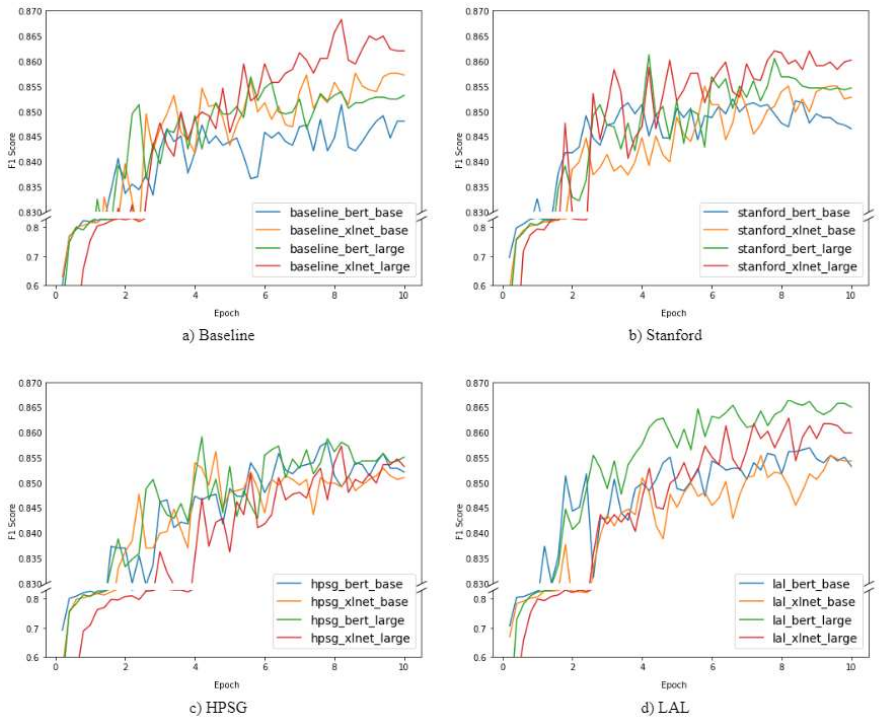

Fig. 4. Micro F1 scores of pre-trained language models for baseline and SDP enhanced models on SemEval dataset.

show the best scores, indicating that SDPs do not increase the performance of XLNet. The permutations of some of the terms that XLNet process (XLNet trains itself on every permutation of the given sequence) contain the words occurring in the shortest dependency path with the same order as in SDPs. We believe that the reason for the high performance of the baseline models is the incorporation of the SDP information into the XLNet embeddings by the XLNet's transformer heads that are using these orderings.

In Fig. 4, Micro F1 scores are shown separately for the baseline model and the models enhanced with the tree parsers. Both of Fig. 3 and Fig. 4 depict the same information but under different groupings of the results; grouped with respect to the pre-trained language models in the former and with respect to the parsers in the latter. In this way, each figure provides specific information about the distinct characteristics of the proposed models.

Table I summarizes the results given in Fig. 3 and Fig. 4. The table shows the best scores the models achieve with the related epoch information.

When we compare the pre-trained language model performance in each dependency parser approach given in Fig. 4, we see that the models that use XLNet PLMs give better results than the models with BERT in Micro F1 scores. However, the same deduction cannot be made from the official Macro F1 scores given in Table I.

Overall, we observe that for the SemEval-2010 Task 8 dataset, combining XLNet PLMs with dependency parsers does not increase the performance of the baseline XLNet model. However, integrating shortest dependency paths to the models with BERT PLMs increases both Macro F1 and Micro F1 scores of the baseline up to 1 absolute point.

B. Experiments on TACRED

In the TACRED dataset experiments, only the large versions of the pre-trained language models are used. All variations are trained with a learning rate of 1e-5 and a dropout rate of 0.1. Each model is trained for ten epochs, each divided into four checkpoints. At each checkpoint, the model's accuracy in the dev set is calculated.

The selection of the best checkpoint of each model to be used in the test set is made using two methods. The first method selects the checkpoint with the best accuracy score in the dev set among all model checkpoints. As the second method, an early stopping mechanism based on accuracy is applied. If the accuracy of the trained system does not surpass the current best score for five checkpoints, the training is stopped and the checkpoint with the best score until that moment is selected.

Fig 5 shows the performance of the baseline and the SDP enhanced models for each pre-trained language model and Fig 6 shows the performance of the pre-trained language models for each dependency parser. We see that the models that use XLNet learn more slowly than their BERT peers in all cases. Also, the models with BERT PLMs provide better results than their XLNet counterparts in the dev set.

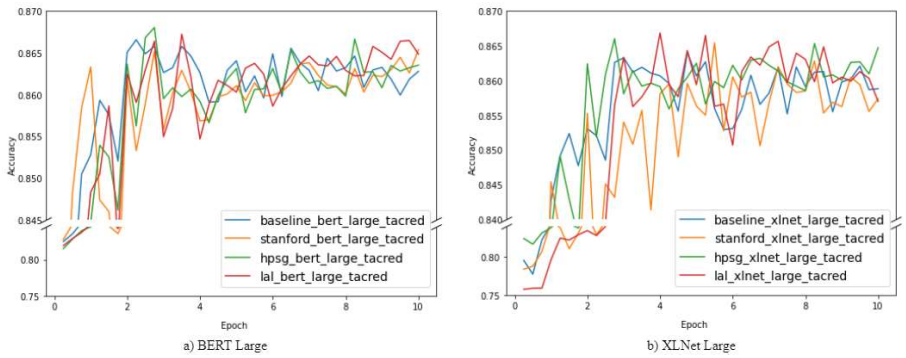

Fig. 5. Accuracy scores of baseline and SDP enhanced models for each pre-trained language model on TACRED dev dataset.

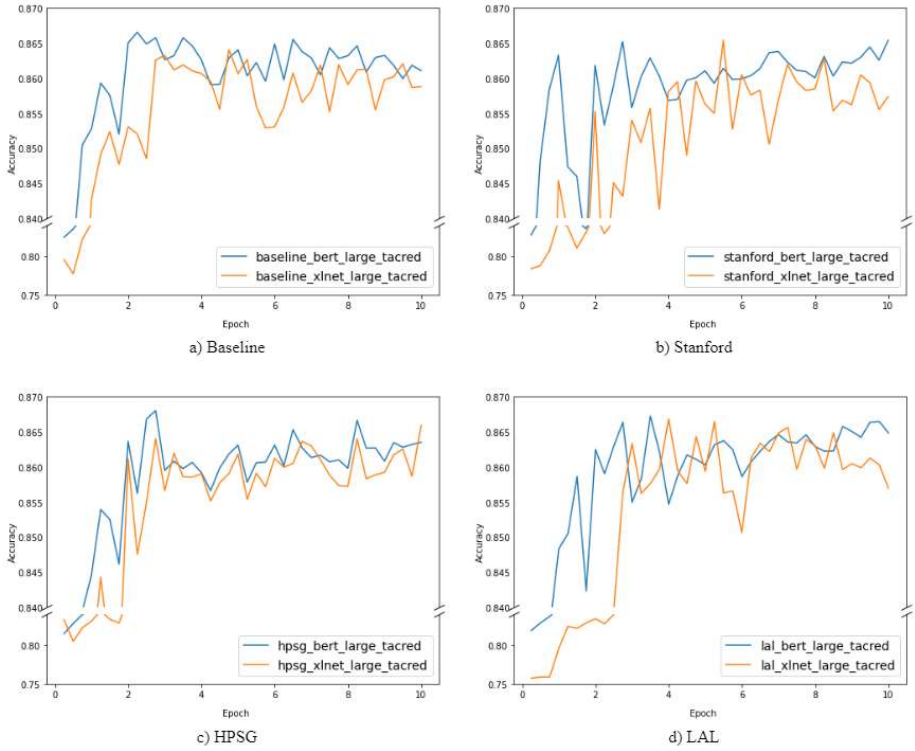

Fig. 6. Accuracy scores of pre-trained language models for baseline and SDP enhanced models on TACRED dev dataset.

TABLE II
MICRO F1 SCORES OF THE MODELS USING BEST OF 10 EPOCHS ON TACRED DATASET

Model	Epoch (Checkpoint)	Precision	Recall	F1
BERT$_{large}$-baseline	3 (0)	75.48%	68.09%	71.60%
BERT$_{large}$-stanford	2 (3)	78.00%	64.05%	70.34%
BERT$_{large}$-hpsg	2 (3)	79.45%	65.02%	71.52%
BERT$_{large}$-lal	3 (2)	76.28%	67.46%	71.60%
XLNet$_{large}$-baseline	4 (3)	75.21%	68.17%	71.52%
XLNet$_{large}$-stanford	5 (2)	77.88%	63.89%	70.19%
XLNet$_{large}$-hpsg	4 (0)	74.56%	70.59%	72.52%
XLNet$_{large}$-lal	4 (0)	74.87%	71.23%	73.01%

TABLE III
MICRO F1 SCORES OF THE MODELS USING EARLY STOPPING ON TACRED DATASET

Model	Epoch (Checkpoint)	Precision	Recall	F1
BERT$_{large}$-baseline	1 (3)	74.66%	68.10%	71.23%
BERT$_{large}$-stanford	1 (0)	77.80%	64.05%	70.14%
BERT$_{large}$-hpsg	2 (3)	79.45%	65.02%	71.52%
BERT$_{large}$-lal	3 (2)	76.28%	67.46%	71.60%
XLNet$_{large}$-baseline	3 (3)	73.32%	69.71%	71.47%
XLNet$_{large}$-stanford	2 (3)	75.65%	65.58%	70.26%
XLNet$_{large}$-hpsg	4 (0)	74.56%	70.59%	72.52%
XLNet$_{large}$-lal	4 (0)	74.87%	71.23%	73.01%

Table II and Table III show the Micro precision, recall, and F1 values, and also the best epoch and checkpoint for each model. The tables correspond to, respectively, the case where the best checkpoint is taken and the case with early stopping criterion. In both cases, the best two models are the ones that apply XLNet with LAL and HPSG parsers. These two XLNet models outperform the closest BERT version by 1.41% and 0.92%, respectively.

We observe in Table II that the best results are obtained from the checkpoints of the fifth epoch or earlier, suggesting that the system is not required to run for ten epochs. Additionally, comparing the results in Tables II and III indicate that the checkpoints and thus the results for half of the system variations are the same. For these systems, early stopping achieves the same score as running the system for 10 epochs and then selecting the best. For the other half, the decrease due to early stopping is negligible. Thus, we can significantly lower the time required to train a system by applying early stopping with five checkpoints without a notable decrease in the performance of the system.

V. Conclusions

In this paper, we proposed an improved version of the R-BERT model [11] by integrating shortest dependency path embeddings obtained from pre-trained language models to the relation representation of the R-BERT model.

We combined the base and large versions of the pre-trained language models BERT and XLNet with the dependency parsers Stanford, HPSG, and LAL. The experiments were conducted on two commonly used relation extraction datasets, SemEval-2010 Task 8 and TACRED. In the SemEval dataset, the proposed model achieved an F1 score of 89.90%, improving the baseline score of the original paper and the baseline score in this work by, respectively, 0.65% and 1.41%. In the TACRED dataset, the proposed model achieved 73.01% F1 score, surpassing the unofficial performance score of 69.40% of the baseline (obtained from paperswithcode website) by 3.6% and increasing the rank of the model from 18^{th} to 7^{th} in paperswithcode rankings.

In the experiments, we observed that all versions of the proposed model outperform the baseline model of BERT$_{base}$ in the SemEval-2010 Task 8 dataset except one case. In TACRED, LAL parser increases the performance of both PLMs, HPSG improves that of XLNet, and Stanford decreases their performance. In general, we can say that shortest dependency paths obtained from state-of-the-art dependency parsers improve the results in relation extraction.

A promising future work would be the investigation of different dependency representations, such as the augmented shortest dependency path in place of the vanilla version. In this work, we did not take into account the labels of the dependencies between the words in the extraction of the shortest dependency paths. Embedding this information into the shortest dependency paths can further increase the performance of the proposed model. Finally, the current system works only on English texts. Investigation of the model's performance in different languages could be an important extension.

References

[1] K. Fundel, R. Küffner, and R. Zimmer, "Relex—relation extraction using dependency parse trees," *Bioinformatics*, vol. 23, no. 3, pp. 365–371, 2007.

[2] M. Wang, "A re-examination of dependency path kernels for relation extraction," in *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*, 2008.

[3] C. Liu, W. Sun, W. Chao, and W. Che, "Convolution neural network for relation extraction," in *International Conference on Advanced Data Mining and Applications*. Springer, 2013, pp. 231–242.

[4] M. Miwa and M. Bansal, "End-to-end relation extraction using lstms on sequences and tree structures," *arXiv preprint arXiv:1601.00770*, 2016.

[5] Y. Zhang, P. Qi, and C. D. Manning, "Graph convolution over pruned dependency trees improves relation extraction," *arXiv preprint arXiv:1809.10185*, 2018.

[6] Y. Liu, F. Wei, S. Li, H. Ji, M. Zhou, and H. Wang, "A dependency-based neural network for relation classification," *arXiv preprint arXiv:1507.04646*, 2015.

[7] Z. Li, Z. Yang, C. Shen, J. Xu, Y. Zhang, and H. Xu, "Integrating shortest dependency path and sentence sequence into a deep learning framework for relation extraction in clinical text," *BMC medical informatics and decision making*, vol. 19, no. 1, pp. 1–8, 2019.

[8] Q. Tao, X. Luo, H. Wang, and R. Xu, "Enhancing relation extraction using syntactic indicators and sentential contexts," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2019, pp. 1574–1580.

[9] H. Wang, M. Tan, M. Yu, S. Chang, D. Wang, K. Xu, X. Guo, and S. Potdar, "Extracting multiple-relations in one-pass with pre-trained transformers," *arXiv preprint arXiv:1902.01030*, 2019.

[10] C. Li and Y. Tian, "Downstream model design of pre-trained language model for relation extraction task," *arXiv preprint arXiv:2004.03786*, 2020.

[11] S. Wu and Y. He, "Enriching pre-trained language model with entity information for relation classification," *arXiv preprint arXiv:1905.08284*, 2019.

[12] D. Chen and C. D. Manning, "A fast and accurate dependency parser using neural networks," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 740–750.

[13] J. Zhou and H. Zhao, "Head-driven phrase structure grammar parsing on penn treebank," *arXiv preprint arXiv:1907.02684*, 2019.

[14] K. Mrini, F. Dernoncourt, Q. Tran, T. Bui, W. Chang, and N. Nakashole, "Rethinking self-attention: Towards interpretability in neural parsing," *arXiv preprint arXiv:1911.03875*, 2019.

[15] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.

[16] I. Hendrickx, S. N. Kim, Z. Kozareva, P. Nakov, D. Ó Séaghdha, S. Padó, M. Pennacchiotti, L. Romano, and S. Szpakowicz, "SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals," in *Proceedings of the 5th International Workshop on Semantic Evaluation*. Uppsala, Sweden: Association for Computational Linguistics, Jul. 2010, pp. 33–38. [Online]. Available: https://aclanthology.org/S10-1006

[17] Y. Zhang, V. Zhong, D. Chen, G. Angeli, and C. D. Manning, "Position-aware attention and supervised data improve slot filling," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, 2017, pp. 35–45. [Online]. Available: https://nlp.stanford.edu/pubs/zhang2017tacred.pdf

[18] K. Xu, Y. Feng, S. Huang, and D. Zhao, "Semantic relation classification via convolutional neural networks with simple negative sampling," *arXiv preprint arXiv:1506.07650*, 2015.

[19] Y. Xu, R. Jia, L. Mou, G. Li, Y. Chen, Y. Lu, and Z. Jin, "Improved relation classification by deep recurrent neural networks with data augmentation," *arXiv preprint arXiv:1601.03651*, 2016.

[20] J. Lee, S. Seo, and Y. S. Choi, "Semantic relation classification via bidirectional lstm networks with entity-aware attention using latent entity typing," *Symmetry*, vol. 11, no. 6, p. 785, 2019.