Protein-Protein Interaction Extraction using Attention-based Tree-Structured Neural Network Models

Sudipta Singha Roy and **Robert E. Mercer**
The University of Western Ontario
London, Ontario, Canada

Abstract

In order to comprehend underlying biological processes, it is necessary to identify interactions between proteins. It is typically quite difficult to extract a protein-protein interaction (PPI) from text data as text data is complex in nature. Unlike sequential models, tree-structured neural network models have the ability to consider syntactic and semantic dependencies between different portions of the text and can provide structural information at the phrase level. This paper investigates tree-structured neural network models for the PPI task and the results show their supremacy over sequential models and their effectiveness for this task.

Introduction

As the scientific literature grows at an exponential rate, the vast majority of biological information is currently available in text form residing in the scientific literature. MEDLINE database's size has grown by 4.2 percent annually during the last two decades and currently it contains approximately 26,000,000 records extracted from 5639 publications which is 23% more than what it contained in 2014 (Yadav et al. 2020). This huge amount of unstructured text from biomedical research articles is a valuable source of information for the biomedical natural language processing (NLP) domain.

As the volume of biomedical data continues to grow exponentially and due to the inherent complexity in the textual representations of these data, it is critical to pursue automatic information retrieval techniques to aid biologists in the detection and identification of useful information, and the arranging and maintaining of databases, as well as providing automatically generated decision support systems for medical professionals. Considering this issue, a lot of research has been conducted for inferring information concealed in these texts to assist health care and biomedical people, such as protein-protein interactions (PPIs), chemical-disease relation extraction, clinical relation extraction, drug-drug interactions, etc., as retrieving important information manually from this large volume of texts is both time consuming and expensive (Peng, Wei, and Lu 2016).

The majority of biological activities inside a cell, such as immune response, signal transduction, cellular organization, etc., are caused by different interactions between various proteins (Sledzieski et al. 2021). So, identifying the protein-protein interactions (PPIs) provides a better understanding of the functionalities, regulations, and communication between different proteins (Yao et al. 2019). Identifying PPIs entails figuring out how different proteins mentioned in a text are connected (Krallinger et al. 2008). This information may spread out through different parts of the whole document, however, the current work is restricted to identify PPIs present only inside single sentences (Pyysalo et al. 2008; Tikk et al. 2010). For instance, "LEC induced maximal migration of CCR1 and CCR8 transfected cells at 89.3 nmol/L and cell adhesion at 5.6 nmol/L." (Howard et al. 2000) reflects two PPI relations: LEC-CCR1 and LEC-CCR8 and no association between CCR1 and CCR8.

For such tasks, sequential deep learning-based models have been used in multiple research works (Hsieh et al. 2017; Yadav et al. 2019). However, if the data is structured rather than presented sequentially, these models are more likely to miss the underlying semantic compositionality (Ahmed, Samee, and Mercer 2019a) as they consider word order only but no linguistic structure (Li et al. 2015). By contrast, Recursive neural networks, commonly known as tree-structured neural network models, work over parsed tree representations of the sentences and thus preserve both the syntactic and semantics in a better way.

In this paper, we investigate six tree-structured neural network models for the PPI identification task. For working with dependencies between words in different portions of the sentence, we investigate dependency tree-structured neural nets, whereas to work with the phrase level information, constituency tree-structured neural nets are explored. Finally, two ensembles of these models are used for retrieving the PPIs present in the sentences. We provide an ample analysis of these models' performances over the benchmark PPI datasets which evince the supremacy of using tree-structured neural networks over sequential ones for this task.

Related Works

Numerous NLP methods have been developed for determining the associations between protein entities. In the initial stages, pattern-based techniques were widely used. In these approaches, on the basis of syntactic as well as lexical features, pattern-based rules were designed for extracting

the relationship (Blaschke et al. 1999; Leeuwenberg et al. 2015). However, these models were not capable of handling complex relationships specified in relational and coordinating clauses appropriately. In contrast to naïve pattern-based methods, dependency-based methods are more syntax attentive and offer a wider range of application coverage (Erkan, Özgür, and Radev 2007; Miyao et al. 2009).

Another prominent approach for extracting such relationships is to use kernel-based methods. These models learn profuse structural information by means of dependency structures and syntactic parse trees (Miwa et al. 2009; Kim et al. 2010). Some noteworthy methods use bag-of-words kernel (Sætre, Sagae, and Tsujii 2007), tree kernel (Zhang et al. 2006), convolution tree kernel (Chang et al. 2016), neighbourhood hash graph kernel (Zhang et al. 2011), and walk-weighted kernel (Kim et al. 2010).

With the recent blossoming of deep learning-based models, a lot of experiments have been conducted to extract PPI relations (Quan et al. 2016; Zhao et al. 2016). Peng and Lu (2017) deployed a double-channel convolutional neural network (CNN) for feature extraction. The first channel utilizes syntactic features like named entities, parts of speech, syntactic dependencies, chunk parsing information, distances from each word to the two interacting protein candidates, and the word itself. The second channel applies a convolution operation over each word's parent word information. Zhang et al. (2018) applied a three channel CNN for this task. The first, second, and third channel apply convolutional operations over original words in addition to the positional encoding, shortest dependency path information, and dependency relation encoding features, respectively. Zhao et al. (2016) trained an auto-encoder on the unlabelled training data for parameter initialization of a multi-layer perceptron (MLP) model which is then trained utilizing gradient descent for the PPI relation extraction task.

Following this, several research works have been conducted for this task using recurrent neural networks (RNNs) as these models perform better with sequential data. Hsieh et al. (2017) applied only a bi-directional long short term memory network (Bi-LSTM) on the sentences, and the vectors concatenated from the left and right-most LSTM output vectors are used as the feature vectors for the classification task. Yadav et al. (2019) utilized structured attention over the sequential Bi-LSTM which is fed with the shortest dependency path information between the unit pairs. In their following work, Yadav et al. (2020) utilized the self attention mechanism for multi-task learning incorporating both PPI and drug-drug interaction relation extraction. Ahmed et al. (2019) used a dependency tree-structured LSTM with structured attention for the same task and outperformed all of the above-stated sequential models.

The Model

This section describes our work in detail. Our investigation of the PPI relation extraction task examines four tree-structured neural network models: dependency and constituency tree-LSTMs with a self attention mechanism and tree-transformers. Their working principles are discussed first. Two ensembles of these models are then discussed.

Tree-LSTMs

A sentence can be represented by two tree-structured representations: constituency and dependency trees (Chen and Manning 2014). These representations provide syntactical information about the sentence by preserving word to word dependencies (dependency tree) and phrase level information (constituency tree). To utilize these structural syntactic information sources, Tai, Socher, and Manning (2015) introduced dependency (child sum tree-LSTM) and constituency (N-ary tree-LSTM) tree-LSTMs.

For the child sum tree, the internal gates of a component node are updated by the summed hidden state values of its child nodes. Then, using this updated hidden state value the other intermediate gates are updated as follows:

$$\tilde{h_j} = \sum_{\kappa \in C_j} h_{j\kappa} \tag{1}$$

$$i_j = \sigma(W_i x_j + U_i \tilde{h_j} + b_i) \tag{2}$$

$$o_j = \sigma(W_o x_j + U_o \tilde{h_j} + b_o) \tag{3}$$

$$\tilde{c_j} = \tanh(W_c x_j + U_c \tilde{h_j} + b_c) \tag{4}$$

Here, Ws and bs are weights and bias values, and C_j is the set of child nodes. In the child sum tree-LSTM, for each child node, there is a separate forget gate ($f_{j\kappa}$) which allows the model to selectively incorporate information for the parent node from the child nodes. For each child node, the corresponding cell state and forget gate values are then multiplied and finally all of these values are combined together to compute the forget gate value of the parent node. Then, the cell state (c_j) and hidden state (h_j) values of the parent node are computed using this forget gate value as follows:

$$f_{j\kappa} = \sigma(W_f x_j + U_f \tilde{h_j} + b_f) \tag{5}$$

$$\tilde{f_j} = \sum_{\kappa \in C_j} f_{j\kappa} \cdot c_\kappa \tag{6}$$

$$c_j = i_j \cdot \tilde{c_j} + \tilde{f_j} \tag{7}$$

$$h_j = o_j \cdot \tanh(c_j) \tag{8}$$

In the N-ary tree-LSTM, each parent node contains identical cell and hidden states for each of its children. The internal gate values and forget gates are computed as follows:

$$i_j = \sigma(W_i x_j + \sum_{\iota=1}^{N} U_{i\_\iota} \tilde{h_j} l + b_i) \tag{9}$$

$$o_j = \sigma(W_o x_j + \sum_{\iota=1}^{N} U_{o\_\iota} \tilde{h_j} l + b_o) \tag{10}$$

$$\tilde{c_j} = \tanh(W_c x_j + \sum_{\iota=1}^{N} U_{c\_\iota} \tilde{h_j} l + b_c) \tag{11}$$

$$f_{jk} = \sigma(W_f x_j + \sum_{\iota=1}^{N} U_{f\_j\iota} h_j l + b_f) \tag{12}$$

Just like in the child sum tree-LSTM, the final forget gate of the parent node is computed by multiplying the corresponding forget gate and cell state values and then summing them

(Eq. 13). The cell state (Eq. 7) and new hidden state (Eq. 8) values are computed as before.

$$\tilde{f}_j = \sum_{\iota=1}^{N} f_{j\iota} \cdot c_{j\iota} \tag{13}$$

Ahmed, Samee, and Mercer (2019a) introduced self attention for such tree structured recursive neural networks. It incorporates three matrices: *query, key,* and *value*. They are calculated as follows:

$$\textit{key} = \omega_k \mathcal{M}_k \quad \text{s.t.} \quad \omega_k \in \mathbb{R}^{d \times d} \tag{14}$$

$$\textit{value} = \omega_v \mathcal{M}_v \quad \text{s.t.} \quad \omega_v \in \mathbb{R}^{d \times d} \tag{15}$$

$$\textit{query} = \omega_q \mathcal{M}_q \quad \text{s.t.} \quad \omega_q \in \mathbb{R}^{d \times d} \tag{16}$$

For the child sum tree, the $\mathcal{M}$s are the concatenations of all of the child nodes' word vectors for a corresponding parent node, whereas in the N-ary tree-LSTM the word vectors under a constituent are concatenated. Then these *key, value,* and *query* matrices are aligned considering the representation's dimension (Eq. 17).

$$align \in \mathbb{R}^{n \times n} = (query)^T key \cdot (1/\sqrt{d}) \tag{17}$$

where *n* is the number of offspring nodes under any particular parent node and *d* is the normalizing factor. Then, `softmax` is applied over this *align* matrix to compute the attention probability matrix $\alpha \in \mathbb{R}^{n \times n}$. Finally, batch-wise matrix multiplication is applied between the attention matrix α and the matrix *value* to compute the attentive hidden states $\tilde{h} \in \mathbb{R}^{n \times d}$. Rows of this matrix are concatenated to produce the final hidden representation of the parent node for both the child sum and N-ary tree-LSTM.

Tree-Transformers

Ahmed, Samee, and Mercer (2019b) applied the concept of transformer (Vaswani et al. 2017) over the constituency and dependency trees, and introduced two tree-transformer models: constituency tree-transformer and dependency tree-transformer. Both of these models apply multi-branch attention over the child nodes' representations. Just like the self attention mechanism, this approach also uses *key, query,* and *value* matrices as follows:

$$\alpha = \texttt{softmax}(\frac{\textit{query key}^T}{\sqrt{d_k}})\textit{value} \tag{18}$$

where d_k is the dimension of the *key*. For the multi-branch attention (β_i), *n* copies of *key, query,* and *value* matrices are created with the appropriate weight matrices ω_i, where *n* is the number of branches, and finally a scaled dot product attention (Eq. 18) is applied over each branch (Eq. 19).

$$\beta_i = \alpha_{i \in [1,n]}(\textit{query}_i\ \omega_i^{\textit{query}}, \textit{key}_i\ \omega_i^{\textit{key}}, \textit{value}_i\ \omega_i^{\textit{value}}) \tag{19}$$

A residual connection is then employed over these tensors followed by a layer-wise batch normalization layer. A scaling factor τ is applied in the end to produce the branch representation (Eq. 20). Following this, position-wise CNN (PCNN) is applied over every $\tilde{\beta}_i$ (Eq. 21). The attention-encoded representations of these semantic subspaces are computed by applying weighted summation where each $\gamma_i \in \mathcal{R}^n$ is a hyperparameter (Eq. 22). In the end, with `BranchAttn`, another residual connection is employed. This is then fed to a $\tanh$ layer and an element-wise summation (`EWS`) is performed to generate the parent node representation (Eq. 23). Here, χ and $\tilde{\chi}$ represent the input and the outcome of the attention module, respectively.

$$\tilde{\beta}_i = \texttt{LayerNorm}(\beta_i \omega_i^b + \beta_i) \times \tau_i \tag{20}$$

$$\texttt{PCNN}(x) = \texttt{Conv}(\textit{Relu}(\texttt{Conv}(x) + b_1)) + b_2 \tag{21}$$

$$\texttt{BranchAttn} = \sum_{i=1}^{n} \gamma_i \texttt{PCNN}(\tilde{\beta}_i) \tag{22}$$

$$\texttt{ParentNodeRep} = \texttt{EWS}(\tanh((\tilde{\chi} + \chi)\omega + b)) \tag{23}$$

Ensemble Architecture

After exploring the tree-structured LSTMs and transformer models, we investigated two ensemble models. In the first approach, we train all the models, and then, when testing, each sentence is fed to all of the models. All of the models predict the class label individually. Finally, a winner takes all method (Roy et al. 2018) is applied over these individual models' selections for the final class prediction. In our second approach, we utilize only the dependency and constituency tree-transformers. Each sentence is fed to both of the tree-transformers and then the sentence representations are concatenated and then fed to the following MLP for class label prediction. The intention behind investigating this model is to find out what happens if features containing both word-level dependencies and phrase-level information are used for the PPI relation extraction task. Figure 1 provides a sketch of this ensemble architecture.

Experiments and Performance Analysis

This section reports the results found for the tree-structured neural network models and the ensemble architectures with F1-score as the performance evaluation metric. It also provides a statistical description of the five standard benchmark PPI corpora for this task along with the pre-processing steps. The PPI problem has been formulated as a classification task. Finally, the performance of the tree-structured models are compared against the most prominent sequential and the previous tree-structured architectures used to solve this task.

For evaluating the investigated tree-structured neural networks, the models are tested on the five standard PPI corpora: AIMed (Bunescu et al. 2005), BioInfer (Pyysalo et al. 2007), IEPA (Ding et al. 2001), HPRD50 (Fundel, Küffner, and Zimmer 2007), and LLL (Nédellec 2005). For all the experiments, the converted version of the corpora are used as mentioned by Ahmed et al. (2019). All of the protein names in all five corpora are substituted with three special symbols: PROT0, PROT1 and PROT2. If any two proteins in a sentence are being considered as interacting with each other, they are replaced with PROT1 and PROT2. All the mentioned proteins in the sentence which are not being considered for interaction identification are replaced with PROT0. As an example, the sentence "LEC induced maximal migration of CCR1 and CCR8 transfected cells at 89.3 nmol/L and

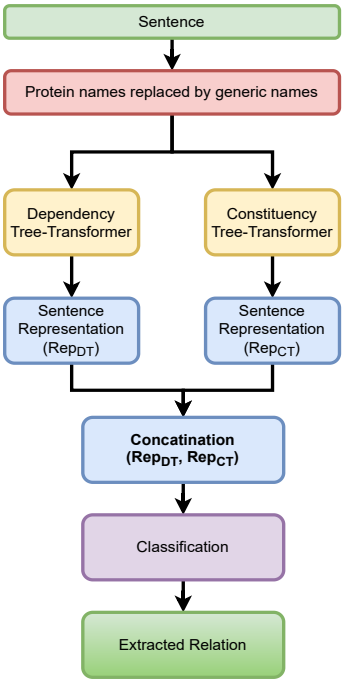

Figure 1: Working procedure of the ensemble architecture combining features from the dependency and constituency tree-transformers.

cell adhesion at 5.6 nmol/L.", the protein names LEC, CCR1 and CCR8 are replaced by PROT1, PROT2 and PROT0 accordingly as this time the intention is to retrieve the relation between LEC and CCR1. When the target proteins are LEC and CCR8, then these two protein names are replaced by PROT1 and PROT2 accordingly, and CCR1 would be replaced with PROT0. The nature of an interaction between two proteins can be positive or negative. For the above mentioned two examples, the interactions are positive whereas the interaction between CCR1 and CCR8 is negative as there is no interaction between them. There are three possible interactions present in this example sentence. So, the modified corpora contains three variants of this sentence with two positive and one negative interaction. In a similar way, for every sentence in the corpora with η proteins present in it, there are ${}^{\eta}C_2$ variants in the modified corpora. The demographics of these five modified corpora are presented in Table 1. In addition, representing the protein names by a few generic names enhances the data further by having multiple samples for these generic names rather than a few samples for each real protein name. For the evaluation of these models, we used 10-fold cross validation using StratifiedK-Fold from the scikit-learn package.

Both of the tree-LSTM models are initialized with learning rate 0.1. For each iteration, if the validation accuracy drops compared to the previous iteration, the learning rate is reduced by 80%. The batch size is 10. The memory and attention dimension is set to 150. The MLP hidden dimension is 300. Training dropout is used with value 0.1. For the training of the tree-LSTM models the 'SGD' optimizer is used.

Table 1: Overall demographics of the modified corpora

Corpus	Original Sentences	Positive Interactions	Negative Interactions
AIMED	1,995	1,000	4,834
BioInfer	1,100	2,534	7,132
IEPA	486	335	482
HPRD50	145	163	270
LLL	77	164	166

For the tree-transformer models, the initial learning rate is 0.1 and the same learning rate decay approach is used. Six PCNN layers are used in the multi-branch attention block. For the experiments, six branches of attention layer are used. Each PCNN layer is composed of 2 CNNs. The first CNN layer employs 341-dimension kernels without any dropout and 300-dimension kernels are used in the second layer of the CNN. In the second layer, dropout 0.1 is used in all cases just like Ahmed, Samee, and Mercer (2019b). The hyperparameters of the tree-transformers are updated using the 'Adagrad' optimizer. All of the models (both tree-LSTMs and tree-transformers) are fed with Bio-RoBERTa (Gururangan et al. 2020) word embeddings which are not updated during training. We also tried fasttext (Bojanowski et al. 2017) and Bio-WordVec (Zhang et al. 2019) embeddings. However, the best results are with the Bio-RoBERTa embeddings.

Table 2 shows the performance of the tree-structured LSTMs, transformers, and the ensemble architectures over the five benchmark PPI corpora and some prominent sequential and tree-structured models for comparability. Among these five, AIMed contains many erroneous annotations. In addition, having nested named entities makes it more difficult to work with (Ahmed et al. 2019).

From this table it is clearly visible that all of the tree-structured models outperform the sequential models for this task. Among the two investigated tree-LSTM models (child sum and N-ary treeLSTM), the child sum tree-LSTM with self-attention performs slightly better than the N-ary tree-LSTM with self attention (average F1-scores are 84.67% and 84.28% accordingly). Overall, both of the investigated tree-transformer models perform better than the tree-LSTM models. However, among the tree-transformer architectures, the dependency tree-transformer (DT-transformer) performs better than the constituency tree-transformer (CT-transformer) and it happens for all five corpora. So, these two observations suggest that neural networks based on dependency trees perform slightly better than the models built on constituency trees. The reason behind this may be that because the sentences here are quite complex in nature, word-level dependency provides more useful information.

For each dataset among these four standalone models, DT-Transformer has the highest F1-scores (87.88%, 95.37%, 82.56%, 88.01%, and 91.46% for the AIMed, BioInfer, IEPA, HPRD50, LLL datasets, accordingly). For the Ensemble - Winner Takes All model, a little performance boost is achieved for four datasets. For HPRD50, the F1-score is a bit less than the two transformer models. It achieves a better average F1-score compared to all of the standalone mod-

Table 2: Performance evaluation of the models by means of F1-score (in %). The sequential models are marked with †. Here, NT-LSTM: 2-ary tree-LSTM over constituency tree, CT-LSTM: Child sum tree-LSTM over dependency tree, CT-Transformer: Constituency tree-transformer and DT-Transformer: Dependency tree-transformer, DT+CT-Transformer: Combination of dependency and constituency tree-transformers.

Methods	AIMed	BioInfer	IEPA	HPRD50	LLL	Avg.
Chang et al. (2016) †	60.6	69.4	71.4	71.5	80.6	70.7
Hsieh et al. (2017) †	76.9	87.2	76.31	80.51	78.3	79.84
Zhang et al. (2018) †	56.4	61.3	75.1	63.4	76.5	66.54
Yadav et al. (2020) †	77.33	76.33	-	-	-	76.83
Tai, Socher, and Manning (2015)	80.6	88.1	76.4	82.0	84.8	82.38
Ahmed et al. (2019)	81.6	89.1	78.5	81.3	84.2	82.94
NT-LSTM + Self Attn	82.99	90.87	78.2	83.22	86.14	84.28
CT-LSTM + Self Attn	83.06	91.01	78.9	83.59	86.78	84.67
CT-Transformer	87.51	94.95	82.5	87.73	91.32	88.80
DT-Transformer	87.88	95.37	82.56	88.01	91.46	89.06
Ensemble - Winner Takes All	87.94	95.48	82.63	87.95	91.49	89.09
DT + CT-Transformer	**88.15**	**96.01**	**83.24**	**88.94**	**92.18**	**89.70**

els. The DT+CT-Transformer model combines features from both of the constituency and dependency tree transformers. The reasons behind choosing only the tree-transformer-based models are that both of the transformer-based models perform better than the LSTM-based models and both the word dependency-level and phrase-level information are already being provided by the transformer-based models. The DT+CT-Transformer outperforms all other models for every dataset with an average F1-score 89.70%. Furthermore, this approach is computationally less expensive compared to the previously mentioned ensemble model as that method requires four models to be trained whereas for the DT+CT-Transformer only two models and an additional MLP are required to be trained. Additionally, the results can be explained by means of the attention value on each node as presented by Ahmed, Samee, and Mercer (2019b).

Conclusions

In this work, we have explored various tree-structured neural network models for the PPI relation extraction task. The experimental results show that the tree-structured models, because of having additional syntactical information at word dependency and phrase-level, perform better than the sequential models. Among all of the explored models, the combined model with both the dependency and constituency tree-transformers performs the best as it utilizes both the word dependency and constituency information. However, opportunities for improvement in this field remain. In the future we want to explore graph-based neural network models with attention mechanisms, and to leverage additional features for this task. Further analysis of results based on AUC and ROC curves can be performed.

Acknowledgments

We thank the reviewers for their constructive comments. This research is partially funded by The Natural Sciences and Engineering Research Council of Canada (NSERC) through a Discovery Grant to R. E. Mercer.

References

Ahmed, M.; Islam, J.; Samee, M. R.; and Mercer, R. E. 2019. Identifying protein-protein interaction using tree LSTM and structured attention. In *2019 IEEE 13th Int. Conf. on Semantic Computing (ICSC)*, 224–231.

Ahmed, M.; Samee, M. R.; and Mercer, R. E. 2019a. Improving tree-LSTM with tree attention. In *2019 IEEE 13th Int. Conf. on Semantic Computing (ICSC)*, 247–254.

Ahmed, M.; Samee, M. R.; and Mercer, R. E. 2019b. You only need attention to traverse trees. In *Proc. 57th Ann. Meet. of the Assoc. for Computational Linguistics*, 316–322.

Blaschke, C.; Andrade, M. A.; Ouzounis, C. A.; and Valencia, A. 1999. Automatic extraction of biological information from scientific text: protein-protein interactions. In *Seventh Int. Conf. on Intell. Systems for Molecular Biology*, 60–67.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2017. Enriching word vectors with subword information. *Trans. of the Association for Computational Linguistics* 5:135–146.

Bunescu, R.; Ge, R.; Kate, R.; Marcotte, E.; Mooney, R.; Ramani, A.; and Wong, Y. W. 2005. Comparative experiments on learning information extractors for proteins and their interactions. *Art. Intell. in Medicine* 33(2):139–155.

Chang, Y.-C.; Chu, C.-H.; Su, Y.-C.; Chen, C. C.; and Hsu, W.-L. 2016. Pipe: a protein–protein interaction passage extraction module for biocreative challenge. *Database* 2016.

Chen, D., and Manning, C. D. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural-Language Processing (EMNLP)*, 740–750.

Ding, J.; Berleant, D.; Nettleton, D.; and Wurtele, E. 2001. Mining medline: abstracts, sentences, or phrases? In *Biocomputing 2002*. World Scientific. 326–337.

Erkan, G.; Özgür, A.; and Radev, D. 2007. Semi-supervised classification for extracting protein interaction sentences using dependency parsing. In *Proc. of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 228–237.

Fundel, K.; Küffner, R.; and Zimmer, R. 2007. Relex—relation extraction using dependency parse trees. *Bioinformatics* 23(3):365–371.

Gururangan, S.; Marasović, A.; Swayamdipta, S.; Lo, K.; Beltagy, I.; Downey, D.; and Smith, N. A. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 8342–8360.

Howard, O. Z.; Dong, H. F.; Shirakawa, A.-K.; and Oppenheim, J. J. 2000. LEC induces chemotaxis and adhesion by interacting with CCR1 and CCR8. *Blood, The Journal of the American Society of Hematology* 96(3):840–845.

Hsieh, Y.-L.; Chang, Y.-C.; Chang, N.-W.; and Hsu, W.-L. 2017. Identifying protein-protein interactions in biomedical literature using recurrent neural networks with long short-term memory. In *Proc. of the Eighth Int. Joint Conf. on Natural Language Proc. (Vol. 2: Short Papers)*, 240–245.

Kim, S.; Yoon, J.; Yang, J.; and Park, S. 2010. Walk-weighted subsequence kernels for protein-protein interaction extraction. *BMC Bioinformatics* 11(1):1–21.

Krallinger, M.; Leitner, F.; Rodriguez-Penagos, C.; and Valencia, A. 2008. Overview of the protein-protein interaction annotation extraction task of BioCreative II. *Genome Biology* 9(2):1–19.

Leeuwenberg, A.; Buzmakov, A.; Toussaint, Y.; and Napoli, A. 2015. Exploring pattern structures of syntactic trees for relation extraction. In *International Conference on Formal Concept Analysis*, 153–168.

Li, J.; Luong, M.-T.; Jurafsky, D.; and Hovy, E. 2015. When are tree structures necessary for deep learning of representations? *arXiv preprint arXiv:1503.00185*.

Miwa, M.; Sætre, R.; Miyao, Y.; and Tsujii, J. 2009. A rich feature vector for protein-protein interaction extraction from multiple corpora. In *Proc. of the 2009 Conf. on Empirical Methods in Natural Language Processing*, 121–130.

Miyao, Y.; Sagae, K.; Sætre, R.; Matsuzaki, T.; and Tsujii, J. 2009. Evaluating contributions of natural language parsers to protein–protein interaction extraction. *Bioinformatics* 25(3):394–400.

Nédellec, C. 2005. Learning language in logic-genic interaction extraction challenge. In *Proceedings of the Learning Language in Logic 2005 Workshop (LLL05)*, 31–37.

Peng, Y., and Lu, Z. 2017. Deep learning for extracting protein-protein interactions from biomedical literature. *arXiv preprint arXiv:1706.01556*.

Peng, Y.; Wei, C.-H.; and Lu, Z. 2016. Improving chemical disease relation extraction with rich features and weakly labeled data. *Journal of Cheminformatics* 8(1):1–12.

Pyysalo, S.; Ginter, F.; Heimonen, J.; Björne, J.; Boberg, J.; Järvinen, J.; and Salakoski, T. 2007. Bioinfer: a corpus for information extraction in the biomedical domain. *BMC Bioinformatics* 8(1):1–24.

Pyysalo, S.; Airola, A.; Heimonen, J.; Björne, J.; Ginter, F.; and Salakoski, T. 2008. Comparative analysis of five protein-protein interaction corpora. *BMC Bioinformatics* 9(3):1–11.

Quan, C.; Hua, L.; Sun, X.; and Bai, W. 2016. Multichannel convolutional neural network for biological relation extraction. *BioMed Research International* 2016.

Roy, S. S.; Hossain, S. I.; Akhand, M.; and Murase, K. 2018. A robust system for noisy image classification combining denoising autoencoder and convolutional neural network. *International Journal of Advanced Computer Science and Applications* 9(1):224–235.

Sætre, R.; Sagae, K.; and Tsujii, J. 2007. Syntactic features for protein-protein interaction extraction. In *2nd Int. Symposium on Languages in Biology and Medicine (Short Papers)*, volume 319 of *CEUR Workshop Proceedings*.

Sledzieski, S.; Singh, R.; Cowen, L.; and Berger, B. 2021. Sequence-based prediction of protein-protein interactions: a structure-aware interpretable deep learning model. *bioRxiv*.

Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Tikk, D.; Thomas, P.; Palaga, P.; Hakenberg, J.; and Leser, U. 2010. A comprehensive benchmark of kernel methods to extract protein–protein interactions from literature. *PLoS Computational Biology* 6(7):e1000837.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* 30.

Yadav, S.; Ekbal, A.; Saha, S.; Kumar, A.; and Bhattacharyya, P. 2019. Feature assisted stacked attentive shortest dependency path based Bi-LSTM model for protein–protein interaction. *Knowledge-Based Systems* 166:18–29.

Yadav, S.; Ramesh, S.; Saha, S.; and Ekbal, A. 2020. Relation extraction from biomedical and clinical text: Unified multitask learning framework. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*.

Yao, Y.; Du, X.; Diao, Y.; and Zhu, H. 2019. An integration of deep learning with feature embedding for protein–protein interaction prediction. *PeerJ* 7:e7126.

Zhang, M.; Zhang, J.; Su, J.; and Zhou, G. 2006. A composite kernel to extract relations between entities with both flat and structured features. In *Proc. of the 21st Int. Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, 825–832.

Zhang, Y.; Lin, H.; Yang, Z.; and Li, Y. 2011. Neighborhood hash graph kernel for protein–protein interaction extraction. *Journal of Biomedical Informatics* 44(6):1086–1092.

Zhang, Y.; Lin, H.; Yang, Z.; Wang, J.; Zhang, S.; Sun, Y.; and Yang, L. 2018. A hybrid model based on neural networks for biomedical relation extraction. *Journal of Biomedical Informatics* 81:83–92.

Zhang, Y.; Chen, Q.; Yang, Z.; Lin, H.; and Lu, Z. 2019. Biowordvec, improving biomedical word embeddings with subword information and mesh. *Scientific data* 6(1):1–9.

Zhao, Z.; Yang, Z.; Lin, H.; Wang, J.; and Gao, S. 2016. A protein-protein interaction extraction approach based on deep neural network. *International Journal of Data Mining and Bioinformatics* 15(2):145–164.