Chemical-Gene Relation Extraction Using Recursive Neural Network

BioCreative Challenge VI track 5 submission

Sangrak Lim, Jaewoo Kang
Affiliation: Department of Computer Science and Engineering, Korea University, Seoul, Korea

***Abstract*— In this paper, we describe our system for the CHEMPROT task of BioCreative VI challenge. Extracting relations between chemical compounds and genes from biomedical literature is an important element to the biomedical application such as Precision Medicine (PM). The CHEMPROT task of BioCreative VI aims to promote the development of the text mining systems that can be used to find relationships between chemical compounds and genes automatically. We use a Recursive Neural Network to improve the performance of relation extraction. Our model uses a position feature, a subtree containment feature, and an ensemble method to improve the performance of CHEMPROT relation extraction. Our system scored 58.53% (F-score) at the CHEMPROT task of BioCreative VI challenge test set.**

Keywords—relation extraction; recursive neural network; text mining

I. INTRODUCTION

There is an increasing interest to find relationships between biological and chemical entities within natural language texts and store the relationship information in the form of structured databases. In the biomedical field, many new papers are published on the relationship between chemicals and genes. Since this relationship information is dispersed in each paper, it is necessary to extract and gather the relationship first that might benefit the biomedical application such as Precision Medicine (PM). PM refers to determining prevention and treatment strategies according to the individual variability to provide a more specific and effective treatment (1). Gonzalez et al. (2) mentioned that manual curation of relation extraction is still the current standard because the value of the text mining applications still have room for improvement. However, it is an laborious task to extract the relationships information by hand. To solve this problem, the CHEMPROT track in BioCreative VI is intended to facilitate the development of systems that automatically extract relationships from text in natural language and classifies them into several types that are important to biology.

The CHEMPROT track in BioCreative VI organizers manually annotated chemical-gene entity relations in abstracts and divide the relation types into 10 classes. Although all the classes are important in the biochemical and pharmacological perspective, only five groups are used for the evaluation. Table 1. shows the five groups and one example sentence for each group. The data given in the challenge consists of abstracts, entities, and relations between two entities. Each entity has position information in the abstract and an entity with the same name appears multiple times. For the example in the Table 1., we found that almost all the relations take place between the two entities in a same sentence. In the pre-processing step, we split the abstract into sentences and consider a sentence with two entities to have a candidate relationship. Details are covered in the section Ⅱ.

The CPR:3 is related with the upregulation and usually associated with words like “activate”, “promote”, and “increase activity of”. The CPR:4 is related with the downregulation and usually associated with words like “inhibitor”, “block”, and “decrease activity of”. The CPR:5 and CPR:6 is related with agonist and antagonist, respectively. These four groups all have distinctive features. However, when multiple entities co-occur in a sentence, it is difficult to determine if a relationship holds between the two target entities of interest. The CPR:9 is related with substrate. Unlike the above four groups, the CPR:9 do not have noticeable features, and thus the relation is difficult to extract.

We build a relation extraction system for the CHEMPROT track in BioCreative VI using the Recursive Neural Network (Recursive NN) based approach. Our Recursive NN model uses syntactical features of each node in a parse tree. Socher et al. (3) claimed that grammatical structure of natural language sentences is recursive. We believe that Recursive NN approach is effective for relation extraction task.

II. METHODS

Our system architecture is presented in the Fig 1.

A. Preprocessing

Preprocessing involves sentence splitting, and anonymizing target entities and chemical compounds. The abstract data consists of several sentences. We found that almost all the relations take place between the two target entities in a same sentence; therefore, we split the abstract into sentences and consider a sentence with two entities to have a candidate

Table 1. Five groups of CHEMPROT relations to be used for evaluation

Group	*CHEMPROT relations*	*Sentence Example*
CPR:3	UPREGULATOR|ACTIVATOR| INDIRECT_UPREGULATOR	<BC6ENT1>Amitriptyline</BC6ENT1>, but not any other tricyclic or selective serotonin reuptake inhibitor antidepressants, promotes <BC6ENT2>TrkA</BC6ENT2> autophosphorylation in primary neurons and induces neurite outgrowth in PC12 cells.
CPR:4	DOWNREGULATOR|INHIBITOR| INDIRECT_DOWNREGULATOR	Ginseng total saponins, <BC6ENT1>ginsenosides Rb2, Rg1 and Rd</BC6ENT1> administered intraperitoneally attenuated the immobilization stress-induced increase in plasma <BC6ENT2>IL-6</BC6ENT2> level.
CPR:5	AGONIST|AGONIST-ACTIVATOR| AGONIST-INHIBITOR	At 10(-6)M in transcription assays, none of these compounds showed progestin agonist activity, whereas <BC6ENT1>mifepristone</BC6ENT1> and its monodemethylated metabolite manifested slight <BC6ENT2>glucocorticoid</BC6ENT2> agonist activity.
CPR:6	ANTAGONIST	In another experiment, <BC6ENT1>cyanopindolol</BC6ENT1>, an antagonist of the <BC6ENT2>serotonin terminal autoreceptor</BC6ENT2>, also prolonged the clearance of 5-HT from the CA3 region.
CPR:9	SUBSTRATE|PRODUCT_OF| SUBSTRATE_PRODUCT_OF	Leukotriene A(4) hydrolase (<BC6ENT1>LTA(4)H</BC6ENT1>) is a cystolic enzyme that stereospecifically catalyzes the transformation of <BC6ENT2>LTA(4)</BC6ENT2> to LTB(4).

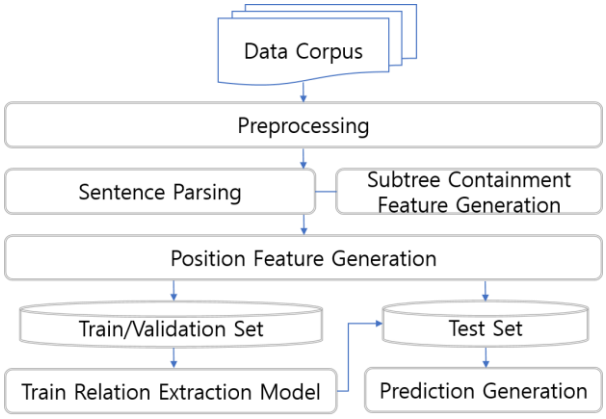

Figure 1. Overall System Architecture

relationship. For the training data, we assign group labels to the candidate relationship.

The biochemical entities usually have long and complex names. To reduce the complexity, we replace the target entities of interest with "BC6ENT1" and "BC6ENT2" in the order in which they appear. For the other entity mentions in a sentence, we apply the ChemDataExtractor [1] to find the chemical entities and replace them with "CHEM" (4).

B. Parsing Sentences

Our Recursive NN model in Natural Language Processing (NLP) uses syntactical features of each node in a parse tree. We use the Stanford NLP library to transform a sentence into a parse tree. After the parsing process, we use the "binarizer" provided by the Stanford Parser to convert the parse tree into a binary tree.

C. Subtree Containment Feature Generation

We calculate the subtree containment feature during the parsing stage. The subtree containment feature indicates that certain subtree contains an important entity. When one of the target entities exist in the leaves of the current node, the subtree containment feature is given a value of one; otherwise, it is given a value of zero. This feature is later converted into a vector with a size 10. If the value is one, every element of a vector is one; otherwise, every element in a vector is zero.

D. Position Feature Generation

Position feature embedding represents the relative distance of two target entities from each word position in a sentence (5). Every word in a sentence has two relative distances, $[d_1, d_2]$, where d_i is the relative distance to i_{th} target entity from the current word. In the training phase, each relative distance is converted into a vector with a size of 10. Since there are two distances, the total vector size of the position feature embedding is 20. Figure 2. shows the vector representation based on the relative distances. Note that when the distance difference is 5 or less, the vector is assigned to each difference value. If the distance is greater than 5, the same vector is given in units of 5. We skip the columns ranged from -2 to $-\infty$ of the relative distance due to space limitation.

E. Word Embedding

Word embedding is a set of low-dimensional vectors that are trained by an unsupervised language model. Word embedding combined with a neural network is a widely method to improve NLP performance (8,9). We used the PubMed-and-PMC-w2v word embedding, which is obtained from published materials [2] (10). The word embedding is initialized with Word2Vec using genism (11). The dimension size of the word embedding is 200.

F. Recursive Neural Network with TreeLSTM

The Long Short-Term Memory (LSTM) architecture is a popular variation of the recurrent neural network (6). The general LSTM is used for sequential data, such as sentences. We implemented tree-LSTM to apply the LSTM architecture for our tree-structured data (7). A node in a tree-LSTM receive input from multiple child nodes and update the hidden state of current node using the input.

[1] http://chemdataextractor.org/download

[2] http://evexdb.org/pmresources/vec-space-models/

relative distance	-1	0	1	2	3	4	5	6∼ 10	11∼ 15	16∼ 20	21∼ ∞
	0	0	1	1	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	0	0	1
	0	0	0	0	0	0	0	0	0	1	1
	0	0	0	0	0	0	0	0	1	1	1
	0	0	0	0	0	0	0	1	1	1	1
	0	0	0	0	0	0	1	1	1	1	1
	0	0	0	0	0	1	1	1	1	1	1
	0	0	0	0	1	1	1	1	1	1	1
	0	0	0	1	1	1	1	1	1	1	1
	1	0	1	1	1	1	1	1	1	1	1

Figure 2. Vector representation according to the distance between one of the target entities and a current word.

Table 2. Search process to find the best hyperparameter

Parameter	Test Range	Test Unit	Selected
Hidden Unit Size	64 - 256	64	256
Subtree Containment Size	1 - 10	2	10
Batch Size	64 - 256	64	256
Learning Rate	0.0005 – 0.01	0.0005	0.001
Keep Probability	0.5 – 0.9	0.1	0.5

After receiving a parse tree to train our model, we look up the pre-trained word embedding to assign real-valued vectors to each word. If a node is not a leaf, the word vector is randomly initialized. Our model is based on the recursive NN architecture of the child sum tree-LSTM model (7).

Let x_j denote the concatenation result of the vector representation of a word with feature vectors. For any node j, we have two forget gates for each child and write the sub-node expression of the forget gates for k-th child as f_{jk}. The $B(j)$ is the set of values (including h_k and c_k) from children of node j, since we use a binary tree, the size of $B(j)$ is 2. *i, f, o, c, h* are the input gate, forget gate, output gate, memory cell, and the hidden state respectively. u is a temporary value that could be added to the memory cell state. $drop(x)$ is a recurrent dropout function (12). The *mask* is a sampled vector from the Bernoulli distribution with success probability *keep_p*. Our tree-LSTM equations are described below.

$$\tilde{h_j} = \sum_{k \in B(j)} h_k, \tag{1}$$

$$i_j = \sigma(W^i[x_j, \tilde{h_j}] + b^i), \tag{2}$$

$$f_{jk} = \sigma(W^f[x_j, h_k] + b^f), \tag{3}$$

$$o_j = \sigma(W^o[x_j, \tilde{h_j}] + b^o), \tag{4}$$

$$u_j = tanh(W^u[x_j, \tilde{h_j}] + b^u), \tag{5}$$

$$c_j = i_j \odot drop(u_j) + \sum_{k \in B(j)} f_{jk} \odot c_k, \tag{6}$$

$$h_j = o_j \odot tanh(c_j) \tag{7}$$

$$drop(x) = \begin{cases} mask * x, & if \quad train \quad phase, \\ x, & otherwise \end{cases} \tag{8}$$

Equation (9,10) is a fully-connected layer we use as the output layer. The fully-connected layer output size is the same as the number of classes (six, one for false class, five for the group classification). At each node j, we choose the predicted label $\hat{y_j}$ for a given output. However, since the predicted value of the internal nodes in the tree is not important, we take only the predicted values extracted from the root node of the entire sentence when the final score is calculated. We use the softmax cross-entropy classifier to calculate the cost function. m is the total number of items in the training set.

$$\hat{p}(y|x_j) = W^{(fc)}h_j + b^{(fc)} \tag{9}$$

$$\hat{y} = \arg\max_y \hat{p}(y|x_j) \tag{10}$$

$$J(\theta) = -\frac{1}{m}\sum_k^m y^k log(softmax(\hat{p}(y^k|x^k))) \tag{11}$$

We use the Adam optimizer for gradient descent optimization. An input vector of a node in a tree uses the subtree containment feature vector, the position feature vector, and the vector representation of a word in a sentence. The size of the whole input vector x_j is 10 + 20 + 200.

G. Regularization

The original tree-LSTM model (7) used l2 regularization. The tree-LSTM model was implemented with the Tensorflow fold library (13) using recurrent dropout (12) instead of the l2 regularization. We found that recurrent dropout is effective.

H. Ensemble method

The random weight initialization changes the result of neural networks considerably (14). Since the CHEMPROT task is challenging, it is difficult to reproduce the exact same result for the single model, and we resolve this problem to some extent using the ensemble model. We sum the output probabilities (logits) of ensemble members, which are from the same repeated experiment for the evaluation. Our model using ensemble method utilizes four (for the first run) and six (for the second run) ensemble members.

III. Results

A. Experimental Setting

We use Tensorflow to implement our model (15). Most deep learning libraries such as Tensorflow assume machine learning models are static, which makes it difficult to use them with dynamic structured models (e.g., Recursive NN). We implement our Recursive NN model with Tensorflow Fold (13). The Tensorflow Fold is specifically designed to deal with dynamic structured problem.

B. Hyperparameter

We found the optimal parameters by moving one parameter within the specified test range by a specified test unit while other parameters were fixed. Table 2. illustrates the hyperparameter search process. Because the CHEMPROT task support the development set, we searched the hyperparameter on the development set, while the model was trained with the training set.

Table 3. The statistics of the BioCreative VI CHEMPROT corpus after preprocessing

	Abstracts	Positive	Negative	Total	Ratio
Train Set	1,020	4,157	12,807	16,964	1:3.08
Development Set	612	2,416	8,198	10,614	1:3.39
Test Set	3,399	-	-	58,523	-

Table 4. CHEMPROT results of our Recursive NN system on the test set.

Run	Precision	Recall	F-Score
First	0.6760	0.5159	0.5852
Second	0.6704	0.5194	0.5853

C. Data corpus

After we preprocess the given data, the number of negative instances is more than three times larger than the number of positives. Table 3. shows the statistics of the preprocessed corpus. We combined the training and development set as a training set for the final model.

D. Performance

The results of our system on the test dataset are shown on Table 4. We submitted two runs for different number of ensemble members. Our model using ensemble method utilizes four (for the first run) and six (for the second run) ensemble members. The best F1-score of our system is the 58.53% in the second run.

IV. CONCLUSION

We implemented the tree-LSTM architecture to understand the natural language sentences. Since we do not know the labels of the test data, we could not proceed the ablation study of additional features on the test set. Still, when we checked our model in validation data, using additional features showed better performance.

The source of our Recursive NN project is available at : https://github.com/arwhirang/recursive_chemprot

REFERENCES

1. Collins, F. S., & Varmus, H. (2015). A New Initiative on Precision Medicine. The New England Journal of Medicine, 372(9), 793–795.

2. Gonzalez, G. H., Tahsin, T., Goodale, B. C., Greene, A. C., & Greene, C. S. (2016). Recent Advances and Emerging Applications in Text and Data Mining for Biomedical Discovery. Briefings in Bioinformatics, 17(1), 33–42.

3. Socher R., Lin C. C., Manning C., and Ng A. Y. (2011). Parsing natural scenes and natural language with recursive neural networks. In Proceedings of the 28th international conference on machine learning (ICML-11);pages 129–136

4. Swain, M. C., & Cole, J. M. (2016). ChemDataExtractor: A Toolkit for Automated Extraction of Chemical Information from the Scientific Literature. Journal of chemical information and modeling, 56(10), 1894-1904.

5. Zeng, D., Liu, K., Lai, S., Zhou, G., & Zhao, J. (2014). Relation Classification via Convolutional Deep Neural Network. In International Conference on Computational Linguistics (COLING) (pp. 2335-2344).

6. Hochreiter S. and Schmidhuber J. (1997). Long short-term memory. Neural computation. 9(8),1735–1780.

7. 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

8. Bengio Y., Ducharme R., Vincent P., and Jauvin C. (2003). A neural probabilistic language model. journal of machine learning research, 1137–1155.

9. Mikolov T., Chen K., Corrado G., and Dean J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

10. Van Landeghem S., Björne J., Wei C. H., Hakala K., Pyysalo S., Ananiadou S.,Kao H. Y., Lu Z., Salakoski T., Van de Peer Y., et al. (2013). Large-scale event extraction from literature with multi-level gene normalization. PloS one, 8(4), e55814.

11. Řehůřek R. and Sojka P. (2010). Software Framework for Topic Modelling with Large Corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pages 45–50

12. Semeniuta S., Severyn A., and Barth E. (2016). Recurrent dropout without memory loss. arXiv preprint arXiv:1603.05118.

13. Looks M., Herreshoff M., Hutchins D., and Norvig P. (2017) Deep learning with dynamic computation graphs. arXiv preprint arXiv:1702.02181

14. Kadlec R., Schmid M., Bajgar O., and Kleindienst J. (2016). Text understanding with the attention sum reader network. arXiv preprint arXiv:1603.01547.

15. Abadi M., Agarwal A., Barham P., Brevdo E., Chen Z., Citro C., Corrado G. S.,Davis A., Dean J., Devin M, et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint. arXiv:1603.04467.