Talla at SemEval-2018 Task 7: Hybrid Loss Optimization for Relation Classification using Convolutional Neural Networks

Bhanu Pratap, Daniel Shank, Oladipo Ositelu, Byron V. Galbraith
Talla Inc.
Boston, MA, USA
{bhanu, daniel, ladi, byron}@talla.com

Abstract

This paper describes our approach to SemEval-2018 Task 7 – given an entity-tagged text from the ACL Anthology corpus, identify and classify pairs of entities that have one of six possible semantic relationships. Our model consists of a convolutional neural network leveraging pre-trained word embeddings, unlabeled ACL-abstracts, and multiple window sizes to automatically learn useful features from entity-tagged sentences. We also experiment with a hybrid loss function, a combination of cross-entropy loss and ranking loss, to boost the separation in classification scores. Lastly, we include WordNet-based features to further improve the performance of our model. Our best model achieves an F1(macro) score of 74.2 and 84.8 on subtasks 1.1 and 1.2, respectively.

1 Introduction

Classifying the relationship between entities is an important natural language processing (NLP) task that serves as a building block for a variety of NLP applications such as knowledge base construction and question-answering tasks. SemEval-2018 Task 7 (Gábor et al., 2018) provided entity-tagged texts from the ACL Anthology corpus and asked participants to identify and classify entity pairs into one of six semantic relationships.

Our approach to this problem consisted of selecting two architectures shown to be successful (Zeng et al., 2014; Nguyen and Grishman, 2015) in this problem domain and adapting them to this particular task. We found that pre-trained word embeddings were effective for this problem, as well as a combined loss function, and using WordNet features at a later stage of our model.

Sentence: High quality <*e1*>**translation**</*e1*> via <*e2*>**word sense disambiguation**</*e2*> and accurate word order generation of the target language .

Relation: USAGE

Figure 1: An example of sentence-level relation Instance. Sentences marked with entity positions were the input to our model.

2 System Description

Convolutional neural networks (CNNs) have been proven to significantly outperform other methods for Relation Classification (Zeng et al., 2014; dos Santos et al., 2015; Nguyen and Grishman, 2015). Our approach was inspired by Nguyen and Grishman (2015) and dos Santos et al. (2015), both systems being minimally dependent on explicit feature engineering. While Nguyen and Grishman (2015) relied solely on their model architecture to automatically extract useful features, we also included additional features based on part-of-speech tags and WordNet hypernyms. Following dos Santos et al. (2015), we trained our model on a hybrid objective function, a combination of cross entropy loss and ranking loss. Finally, we also trained our model in two stages to utilize large amounts of unlabeled ACL corpus abstracts (Bird et al., 2008). We describe these stages in detail in section 4.1.

Each abstract is first tokenized into sentences. For each sentence, we then formed training examples by taking all combinations of pairs of entities annotated in the sentence. If a pair is annotated with a relation label, we labeled the sentence as this relation. Otherwise, we labeled the sentence as an artificial class called *OTHER* . Figure 1 shows an example of a training instance in our sentence-level dataset.

For an input sentence, each word was mapped to a word-vector to form a sentence matrix. These sentence matrices were provided as inputs to the

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 863–867
New Orleans, Louisiana, June 5–6, 2018.

CNN. The output of this layer was then fed into a softmax layer to classify the relationship between two entities. Section 3 provides a detailed description of the underlying CNN.

3 Convolutional Neural Network for Relation Classification

Our model consists of a preprocessing feature generation step followed by a 2D convolutional layer with max pooling and then a fully connected layer with softmax output.

3.1 Preprocessing and Feature Generation

The input to our model was a raw sentence marked with entity positions. This raw sentence was first converted into a real-valued sentence matrix by tokenizing the sentence and then replacing each token with a corresponding word embedding. We used three different look-up tables: publicly available pre-trained word embeddings, randomly initialized word positions, and randomly initialized part-of-speech tags. Following Collobert et al. (2011), the final word embedding for each token in the sentence was a concatenation of these three embeddings.

For pre-trained word embeddings, we evaluated word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and Numberbatch (Speer and Chin, 2016), ultimately choosing word2vec as the best performer for this task.

The specific process used to generate the feature vector for a given token in a sentence is as follows. Let n be the number of tokens in a sentence $x = [x_1, x_2, ..., x_n]$ with x_{i_1} and x_{i_2} being the two head words of the two entities in relationship r. Relative positions between a token x_i and two entities are given by $(i - i_1)$ and $(i - i_2)$. These positions are also mapped into real-valued vectors using a position embeddings look-up table W_p. Also, W_e, and W_t embeddings look-up tables are used to map each word and its part-of-speech tag into a real valued vector. Finally, x_i is transformed into a vector $v_i = [e_i; u_{i_1}; u_{i_2}; t_i]^T$, where v_i is the concatenation of vectors $e_i, u_{i_1}, u_{i_2}, t_i$, e_i is the word-vector mapped using the W_e look-up table, u_{i_1} and u_{i_2} are the word-position vectors mapped using the W_p look-up table, and t_i is the word-part-of-speech vector mapped using the W_t look-up table. Dimension d of the final input vector is given by $d = (d_e + 2 * d_p + d_t)$, where d_e, d_p and d_t are the dimensions of pre-trained word-embeddings, word-position embeddings and word-part-of-speech tag embeddings. As a result of these look-up operations, the raw sentence $x = [x_1, x_2, ..., x_n]$ is transformed into a real-valued sentence matrix $\mathbf{x} = [\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}]$ of size $d \times n$.

3.2 2D Convolution with Max Pooling

We used multiple window sizes to extract features corresponding to various n-grams. Let w be a window size and n_w be the number of unique window sizes, a filter $\mathbf{f} = [\mathbf{f_1}, \mathbf{f_2}, .., \mathbf{f_w}]$ is a weight matrix where $\mathbf{f_i}$ is a column vector of size $d = (d_e + 2 * d_p + d_t)$. A convolutional operation is then performed using $\mathbf{x}$ and $\mathbf{f}$ to produce a feature map $s = [s_1, s_2, ..., s_{n-w+1}]$ as:

$$s_i = g(\sum_{j=0}^{w-1} f_{j+1}^T x_{j+i}^T + b)$$

where b and g are bias and ReLU (Nair and Hinton, 2010) activation function respectively. This convolutional operation was repeated for different filters and window sizes, and then a max pooling strategy Zhang and Wallace (2017) was applied to extract only 1 feature (the one with highest activation) from each feature map. That is, for each feature map s, a max function was applied to produce a single value: $p_f = max(s)$.

3.3 Classification Layer

We took all the individually selected features from the max pooling operation and concatenated them together, producing $z = [p_1, p_2, ..., p_m]$, where m is the number of feature maps and p_i is the pooled value for i^{th} feature map. A random proportion of input vector z was set to zero for regularization purposes to produce a drop-out version z_d of input vector z. The vector z_d was then fed into a dense layer followed by a softmax operation to produce the final classification probability for a relation class r as:

$$\mathbf{o} = \mathbf{C}\mathbf{z_d} + \mathbf{b}$$

$$p(r|\theta) = \frac{e^{o_i}}{\sum_{k=1}^{L} e^{o_k}}$$

where $\mathbf{o}$ is the output of the dense layer, $\mathbf{b}$ is a bias term, L is the number of relation categories, and $\mathbf{C}$ is a weight matrix of size $(n_w m \times L)$ with n_w being the number of unique window sizes and m being the number of filters.

3.4 Additional Features

In addition to the output of the CNN layer, we explored a variety of additional features derived from the input sentences.

Part-of-speech features, *pos* We randomly initialized embeddings for each part-of-speech tag and used these embeddings as additional input to our network. Part-of-speech tags for raw sentences were generated using *spaCy*[1].

WordNet hypernym features, *hyp* We incorporated WordNet hypernyms using the implementation[2] provided by Ciaramita and Altun (2006).

Semantic Similarity between two entities, *sim* We computed the cosine similarity between the word-embeddings of the head-words of the two entities in a relation instance.

REVERSE flag feature, *rev* We applied an indicator function on the REVERSE flag of the relationship instance.

We fed *hyp*, *sim* and *rev* features as additional inputs to the classification layer. While *pos* features were provided as input to the convolution layer.

4 Training Methods

We evaluated three different loss functions for training our model: cross-entropy loss, ranking loss (dos Santos et al., 2015), and a weighted combination of the two, where

$$L_{combined} = \alpha L_{ranking} + (1 - \alpha) L_{cross\ entropy}$$

with α as a weighting parameter. The combined loss function was determined to be the most effective.

4.1 Two-Staged Training

To make use of unlabeled data for fine-tuning word-position and word-part-of-speech embeddings, we trained our model in two-stages following Severyn and Moschitti (2015): a *distant* training stage and a *supervised* training stage.

Distant Training We first created a distantly supervised training dataset using unlabeled ACL corpus abstracts (Bird et al., 2008) based on the naive assumption that *two entities have the same relationship across all aligned sentences*. By aligned sentences, we mean all sentences which have exactly two entities. In order to create distantly supervised training data based on the above assumption, we performed the following operations:

i) All the sentences in the ACL-corpus were indexed in an IR system. Here we used *Whoosh*[3].

ii) For each relation instance in the labeled training data, the top 40 sentences which contained both the entity texts in the relation were returned from the IR system.

iii) Result sentences in which the distance (number of characters) between the two entity texts was greater than 170 were removed. This number was derived from distance statistics from given labeled datasets.

iv) The remaining sentences were labeled with the same relationship as the relation instance in (ii).

Following above steps, we created distant-datasets of around 1600 and 11000 training instances for subtasks 1.1 and 1.2, respectively. We also verified that there is no overlap between these generated distant-datasets and the provided test datasets.

Once the distantly supervised training data is created, we train our model using these datasets to fine-tune only word-position and part-of-speech tag embeddings, while keeping word-embeddings fixed.

Supervised Training In the second stage, we initialized our model with the fine-tuned embeddings trained in the distantly supervised training stage and then train our model using the provided labeled training data. In this stage we also train word-embeddings but freeze them for first 10 epochs to prevent any large updates.

5 Experiments and Results

The class labels for subtasks 1.1 and 1.2 are highly imbalanced (Table 1). To compensate for this imbalance, we trained our models for subtasks 1.1 and 1.2 jointly on a combined dataset and used class-weights to weight our loss function.

5.1 Resources and Hyperparameters

We chose all the hyperparameters based on the model performance on our validation set. All experiments below use the hyperparameters as shown in Table 2.

[1] spacy.io

[2] sourceforge.net/projects/supersensetag/

[3] http://Whoosh.readthedocs.io/en/latest/index.html

Class	Train 1.1	Test 1.1	Train 1.2	Test 1.2
USAGE	0.39	0.49	0.38	0.35
PART-WHOLE	0.19	0.20	0.16	0.16
MODEL-FEATURE	0.27	0.19	0.14	0.21
COMPARE	0.08	0.06	0.03	0.01
RESULT	0.06	0.06	0.10	0.08
TOPIC	0.01	0.01	0.19	0.19
n	1228	355	1249	355

Table 1: Distribution of classes within datasets. Here, n is the total number of instances in a dataset.

Parameter	Value
Window Sizes	2,3,4,5
Number of Filters	25
Word-Embeddings Size (d_e)	300
Word-Position Embeddings Size (d_p)	25
Positive Class Margin (m^+)	2.5
Negative Class Margin (m^-)	0.5
λ	1.0
α	0.1
learning rate	0.01

Table 2: Hyperparameters used in all experiments.

Our final model is a soft-voting ensemble of the best models obtained using 10-fold stratified cross-validation. All the models were implemented using *TensorFlow*[4]. We trained our models using a stochastic gradient descent optimizer with momentum (Sutskever et al., 2013). Lastly, based on our experiments, we chose the 300-dimensional word2vec pre-trained word embeddings trained on the Google News corpus.

5.2 Evaluation

Model	F1-Macro Subtask 1.1	F1-Macro Subtask 1.2
CNN	72.8	84.1
CNN+*pos*	72.1	82.8
CNN+*pos*+*hyp*	**74.4**	82.4
CNN+ *pos*+*hyp*+*sim*	73.7	**84.8**[a]
CNN+*pos*+*hyp*+*sim*+*rev*	74.2[b]	84.7
—"— *(2-staged)*	73.9	84.7
Overall Best(DS3Lab)	81.7	90.4

[a] *our official submission ranked second*
[b] *our official submission ranked fifth*

Table 3: Performance of our final model on Subtasks 1.1 and 1.2. Additional features are incrementally added to our plain CNN model. Here *(2-staged)* refers to the results of our experiments using 2-staged training method.

.

Table 3 shows the results of our ablation studies using different feature sets on subtasks 1.1 and 1.2. It shows that the simple similarity (*sim*) feature helps in case of subtask 1.2, while it degrades the performance in case of subtask 1.1. Similarly, WordNet features with part-of-speech features boosted the performance only of subtask 1.1. Fine tuning using the two-staged training approach did not yield any performance gain in either the subtasks.

We also evaluated the effect of loss function choice. Table 4 shows the results of our final model trained on different loss functions. A combination of ranking loss and cross entropy loss does yield a performance boost.

Loss Function	F1-Macro Subtask 1.1	F1-Macro Subtask 1.2
Cross Entropy Loss	72.7	83.9
Ranking Loss	70.6	81.5
Combined Loss	**74.2**	**84.7**

Table 4: Effect of Loss Functions

While we did not provide a formal submission for subtask 2, we evaluated our approach on it given the labeled test data. Table 5 shows the results of our experiments on subtask 2 (relation extraction). While this method did not outperform the top submissions, it still demonstrated competitive results.

Model	F1-Macro Subtask 2(ANY)
CNN	35.0
CNN+*pos*	**36.6**
CNN+*pos*+*hyp*	34.7
Overall Best(UWNLP)	50.0

Table 5: Performance of our final model on Subtask 2(Relation Extraction).

6 Conclusion

Our experiments indicate that pre-training on an unlabeled corpus did not noticeably impact performance on our evaluation set. Our plain CNN model (without any external features) has comparable performance to the competition's best submission. We also observed improved performance of our model on Subtask 1.1 when using the WordNet features as additional input to the final layer. Finally, when we combine the cross-entropy and ranking loss functions, performance of our model improved on both Subtasks 1.1 and 1.2.

[4] https://www.tensorflow.org/

References

Steven Bird, Robert Dale, Bonnie Dorr, Bryan Gibson, Mark Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir Radev, and Yee Fan Tan. 2008. The acl anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC-08)*, Marrakech, Morocco. European Language Resources Association (ELRA). ACL Anthology Identifier: L08-1005.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 594–602, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

Kata Gábor, Davide Buscaldi, Anne-Kathrin Schumann, Behrang QasemiZadeh, Haïfa Zargayouna, and Thierry Charnois. 2018. Semeval-2018 Task 7: Semantic relation extraction and classification in scientific papers. In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, USA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814. Omnipress.

Thien Huu Nguyen and Ralph Grishman. 2015. *Relation extraction: Perspective from convolutional neural networks*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

CÃcero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *ACL (1)*, pages 626–634. The Association for Computer Linguistics.

Aliaksei Severyn and Alessandro Moschitti. 2015. Unitn: Training deep convolutional neural network for twitter sentiment classification. In *SemEval@NAACL-HLT*, pages 464–469. The Association for Computer Linguistics.

Robert Speer and Joshua Chin. 2016. An ensemble method to produce high-quality word embeddings. *CoRR*, abs/1604.01692.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1139–III–1147. JMLR.org.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 2335–2344.

Ye Zhang and Byron C. Wallace. 2017. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In *IJCNLP(1)*, pages 253–263. Asian Federation of Natural Language Processing.