MIT-MEDG at SemEval-2018 Task 7: Semantic Relation Classification via Convolution Neural Network

Di Jin
MIT CSAIL
Cambridge, MA
jindi15@mit.edu

Franck Dernoncourt
Adobe Research
San Jose, CA
dernonco@adobe.com

Elena Sergeeva
MIT CSAIL
Cambridge, MA
elenaser@mit.edu

Matthew B. A. McDermott
MIT CSAIL
Cambridge, MA
mmd@mit.edu

Geeticka Chauhan
MIT CSAIL
Cambridge, MA
geeticka@mit.edu

Abstract

SemEval 2018 Task 7 tasked participants to build a system to classify two entities within a sentence into one of the 6 possible relation types. We tested 3 classes of models: Linear classifiers, Long Short-Term Memory (LSTM) models, and Convolutional Neural Network (CNN) models. Ultimately, the CNN model class proved most performant, so we specialized to this model for our final submissions.

We improved performance beyond a vanilla CNN by including a variant of negative sampling, using custom word embeddings learned over a corpus of ACL articles, training over corpora of both tasks 1.1 and 1.2, using reversed feature, using part of context words beyond the entity pairs and using ensemble methods to improve our final predictions. We also tested attention based pooling, upsampling, and data augmentation, but none improved performance. Our model achieved rank 6 out of 28 (macro-averaged F1-score: **72.7**) in subtask 1.1, and rank 4 out of 20 (macro F1: **80.6**) in subtask 1.2.

1 Introduction

SemEval 2018 Task 7 (Gbor et al., 2018) focuses on relation classification and extraction on a corpus of 350 scientific paper abstracts consisting of 1228 and 1248 annotated sentences for subtasks 1.1 and 1.2, respectively. There are six possible relations: *USAGE*, *RESULT*, *MODEL-FEATURE*, *PART_WHOLE*, *TOPIC*, and *COMPARE*.

Given this data, our task is to take an example sentence, as well as the left and right entities within that sentence, and an indicator as to whether the relation is reversed, and predict the relation type for that sentence. In subtasks 1.1 and 1.2, all presented sentences have a relation.

We submitted predictions based on a self-ensembled convolutional neural network (CNN) model trained with a negative sampling augmented loss using ACL-specific embeddings as input features. We achieved rank 6 out of 28 (macro-averaged F1-score: 72.7) in subtask 1.1, and rank 4 out of 20 (macro F1 80.6) in subtask 1.2.

2 Related Work

Previous SemEval challenges have explored relation identification and extraction. The 2010 SemEval Task 8 (Hendrickx et al., 2010) explored classification of natural language relations, such as *CONTENT-CONTAINER* or *ENTITY-ORIGIN*. This challenge differs from ours in its generalizability; our relations are specific to ACL papers (e.g. *MODEL-FEATURE*) whereas the 2010 relations are more general, and may necessitate more common-sense knowledge than the 2018 relations. The 2010 data has been extensively studied and has offered significant opportunity for other researchers to test their model. Rink and Harabagiu (2010) produced a strong SVM/LR model to attack this challenge. Several deep architectures have also been proposed for this task, including the work of Cai et al. (2016), which demonstrated a novel approach merging ideas from recurrent networks and convolutional networks based on shortest dependency path (SDP). Xu et al. (2015a) and Santos et al. (2015) both used convolutional architectures along with negative sampling to pursue this task. More recently, Wang et al. (2016) used two levels of attention, one for input selection and the other for output pooling, to boost the performance of their model to state of the art.

The 2017 SemEval Task 10 (Augenstein et al., 2017) also featured relation extraction within

Model	Acc. (%)
SVM	64.0 ± 5.3
LR	65.3 ± 4.3
DEEP RF	63.1 ± 4.1
LSTM	61.4 ± 5.5
CNN	$\mathbf{66.3 \pm 4.4}$

Table 1: Comparison of best performance of different model types in our initial experimentation.

scientific publications. Here, however, there were only 2 relation types, *HYPONYM-OF* and *SYNONYM-OF*. One successful model on this task utilized a convolutional network operating on word, tag, position, and part-of-speech features (Lee et al., 2017), and found that restricting network focus to only the words between the requisite entities offered a notable performance improvement.

3 Methods

3.1 Pre-processing

Data was tokenized using the SpaCy tokenizer[1]. Part of speech (POS) tags were extracted using SpaCy, while lemmas and hypernyms were extracted via WordNet (Miller et al., 1990), inspired by Rink and Harabagiu (2010).

3.2 Initial Experiments

We tested several machine learning methods on these data, including a logistic regression classifier over `tf-idf` features extracted from words, lemmas, hypernyms, and POS. Additionally, we tested deep random forests with multi-grain sequence scanning over word embeddings sequences (Zhou and Feng, 2017) and LSTM with attention (Zhou et al., 2016) over both word/lemma/hypernym embeddings, character sequence embeddings, and position indicators. Lastly, we tested a CNN model over these data, using word/lemma embeddings, position embeddings, and a variant of negative sampling. After optimizing all model configurations and doing preliminary hyperparameter optimization via automatic grid search, early comparisons between the differing model classes yielded the results in Table 1. These results were measured in accuracy over 15-fold cross validation on the 1.1 train set.

Given these initial results, we focused principally on the CNN model.

[1]https://github.com/explosion/spaCy

3.3 CNN Model Details

Figure 1 presents the architecture of the CNN model. The model first takes the tokenized sentence, as well as the targeted entities, and transforms it to a sequence of continuous embedding vectors (Subsection 3.3.1). Next, the model uses a convolution layer to transform the embedded sentence to a fixed-size representation of the whole sentence (Subsection 3.3.2). Finally, it computes the score for each relation class via a linear transformation (Subsection 3.3.3). The overall system is trained end-to-end via a cross entropy loss augmented with a variant of negative sampling (Subsection 3.3.4).

3.3.1 Feature Embeddings

Given a sentence $\boldsymbol{x} = [x_1, \ldots, x_n]$, the tokens x_i are featurized into continuous embedding vectors via concatenated word embeddings (e^{w_i}) and word position embeddings (e^{wp_i}): $e_i = [e^{w_i}, e^{wp_i}]$.

Word Embeddings Word representations are encoded by the column vector in the embedding matrix $W^{word} \in \mathbb{R}^{d^w \times |V|}$, where V is the vocabulary of the dataset. Each column $W_i^{word} \in \mathbb{R}^{d^w}$ is the word embedding vector for the i^{th} word in the vocabulary. This matrix is trainable during the optimization process and initialized by pre-trained embedding vectors described in Section 3.4.

Word Position Embeddings (WPEs) In general, the information needed to determine the sentence's relations mostly comes from the words close to the two entities. In addition, some information needs to be input into the model to indicate which words are entities. We use the word's relative position to either entity as a feature to fulfill the above-mentioned two functions. For instance, in the sentence "the **probabilistic model** used in the **alignment**" shown in Figure 1, the relative distance of all the words to the left entity "probabilistic model" is $-1, 0, 0, 1, 2, 3, 4$ and that to the right entity "alignment" is $-6, -5, -4, -3, -2, -1, 0$. Each relative distance is mapped into a vector of dimension d^{wp}, which is randomly initialized then updated during training. Each word w has two relative distances wp_1 and wp_2 with respect to two entities $entity_1$ and $entity_2$, and each distance is mapped to corresponding embedding vector and the position embedding e^{wp} of word w is the concatenation of these two vectors: $e^{wp} = [e^{wp_1}, e^{wp_2}]$.

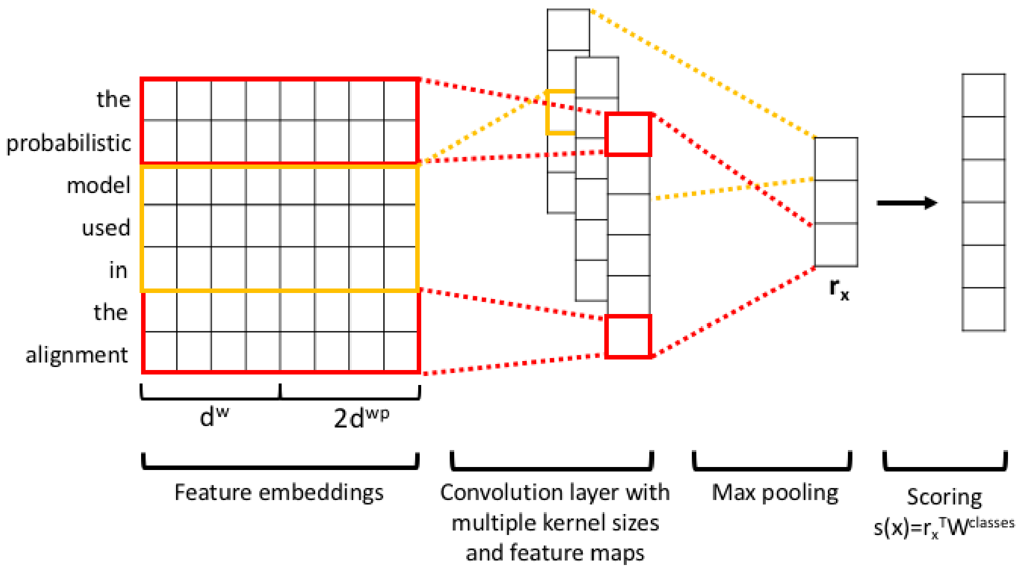
the
probabilistic
model
used
in
the
alignment
d^w
2d^wp
r_x
Feature embeddings
Convolution layer with multiple kernel sizes and feature maps
Max pooling
Scoring s(x)=r_x^T W^classes

Figure 1: Illustration of CNN model architecture.

3.3.2 Sentence Representation

After featurization, a sentence $\boldsymbol{x}$ of length N is represented as $\boldsymbol{e} = [e_1, e_2, ...e_N]$. We denote $\boldsymbol{e}_{i:i+j}$ as the concatenation of featurized tokens: $\boldsymbol{e}_{i:i+j} = [e_i, e_{i+1}, ..., e_{i+j}]$. A convolution operation involves a filter weight matrix $W \in \mathbb{R}^{(d^w+2d^{wp})\times k}$, which is applied to a window of k words to produce a new feature c_i, as represented by:

$$c_i = \tanh(W \cdot \boldsymbol{e}_{i:i+k-1} + b),$$

where $b \in \mathbb{R}$ is a bias term. This filter is applied to each possible window of words in the sentence $\boldsymbol{e}_{1:h}, \boldsymbol{e}_{2:h+1}, ..., \boldsymbol{e}_{N-h+1:N}$ to produce a feature map vector $\boldsymbol{c} = [c_1, c_2, ..., c_{N-k+1}]$. We then apply a max pooling operation to this feature map to obtain the maximum value $\hat{c} = max\{\mathbf{c}\}$ as the feature corresponding to this particular filter. This is how we extract one feature by one filter. And the model can use multiple filters with varying window sizes and filter parameters to produce multiple features. We concatenate all the obtained features to form the fixed size sentence representation $r_{\boldsymbol{x}}$.

3.3.3 Inference Scoring

Given the vector representation $r_{\boldsymbol{x}}$ of the sentence $\boldsymbol{x}$, class scores are computed via a linear transformation mediated by a trainable matrix $W^{classes}$:

$$s(\boldsymbol{x}) = W^{classes} r_{\boldsymbol{x}}.$$

At prediction time, the relation class is inferred by taking the index of maximum score.

3.3.4 Loss with Negative Sampling

After obtaining the score vector $s(\boldsymbol{x})$ for the sentence $\boldsymbol{x}$, we use a loss function motivated by ideas in negative sampling as follows. Let y be the correct label for sentence $\boldsymbol{x}$, and $I = \mathcal{Y} \setminus \{y\}$ be the set of all incorrect labels for $\boldsymbol{x}$. Then, we compute the loss:

$$L = \log\left(1 + e^{\gamma\left(m^+ - s(\boldsymbol{x})_y\right)}\right) + \log\left(1 + e^{\gamma\left(m^- + \max_{y' \in I}\left(s(\boldsymbol{x})_{y'}\right)\right)}\right),$$

where m^+ and m^- are margins, γ is the penalty scale factor. Minimizing this loss function will both increase the score of the correct label and decrease that of the wrong label. We used Adam optimizer (Kingma and Ba, 2014) to minimize the loss function.

3.4 ACL Corpus Embeddings

We pretrained 50-dimensional word embeddings on the ACL anthology corpus (Radev et al., 2013) using word2vec (Mikolov et al., 2013a,b) (we

Symbol	Name	Value
d^w	Word Embed. Size	50
d^{wp}	Pos. Embed. Size	42
d^c	Convolution Units	900
k	Convolution Kernel	2,3,4
m^+	Correct Label Margin	2.2
m^-	Incorrect Label Margin	0.7
γ	Penalty Scale Factor	3.1
λ	Learning Rate	0.0008
β	L2 Regularization	0.01
d	Dropout Ratio	0.5

Table 2: CNN model final hyperparameters.

have also tried 100-dimensional word embeddings pre-trained in the same way but it performed worse in this small dataset). The corpus was derived by downloading all ACL anthology articles in PDF format, then converting them into text via some imperfect optical character recognition (OCR). The corpus contains 25,938 papers totaling 136,772,370 tokens, with 49,600 unique tokens. We used the following parameters for word2vec: skip-gram model, maximum skip length between words of 10, negative sampling with 10 negative examples, discard words that appear less than 5 times, and 5 training iterations[2].

4 Results

4.1 Model Tuning

We first optimized the CNN model hyperparameters via random search with 90 samples; final hyperparameters are shown in Table 2.

Beyond traditional hyperparameter optimization, a number of modifications with this model incurred performance gains during our final stages of experimentation, as determined by cross validation over either the 1.1 or 1.2 data. We detail the types of these changes below, then show the performance results obtained on the *test set* (not the cross validation results which motivated their use in our system) in Table 3.

Merged Training Sets Merging the 1.1 & 1.2 training datasets as a new training set had a large impact on the macro F1 score of our models. Both training datasets are relatively small, containing only approximately 1200 examples. Merging the 1.1 and 1.2 training sets helps equalize class imbalance and expand the dataset size, at the cost of introducing a biased distribution of relation types for either class alone.

Reversal Indicator Features Each entity pair was given the information whether the relation of it is reversed or not. We added this binary feature, which proved performant.

Custom ACL Embeddings Specializing our word vector embeddings pre-training source to an ACL-specific corpus (described in section 3.4) offered notable gains.

Context words We explored using a context window of varying sizes around the entity-enclosed text within the sentence. Our pre-submission cross validation experiments suggested a context window of ±50 words was optimal, but post-submission evaluation on the provided test set yielded better results with a ±20 word window. Empirically, the number of context words to be included needs to be optimized on the specific dataset.

Ensembling We trained 50 copies of our network, using different random initializations and dev sets (for early stopping), then averaged their scores for prediction. This reduced variance of our predictions and improved performance.

Besides the above successful strategies, we also tested some settings that, in reality had a negative effect. We experimented with attention based pooling as suggested by (Wang et al., 2016), but it hurt performance in our cross validation experiments so we discarded that mechanism. Additionally, we also tried replacing the original words with their corresponding lemmas—this increased performance in our cross validation results, however, (post-submission evaluation revealed) harmed our results on the SemEval test set, where it yielded average macro-F1 scores of 71.48% and 77.01% for subtasks 1.1 and 1.2, respectively, after 10 runs. The potential reason for this degradation could be that the word embeddings are all trained on original words instead of lemmas so the embeddings of lemmas cannot be as well initialized as original words. Additionally, we tried data augmentation (via synonym substitution), and up-sampling (via duplication to equalize

[2]-cbow 0 -window 10 -negative 10 -hs 0 -sample 1e-3 -threads 15 -binary 0 -iter 5 -min-count 5 -size 50

Condition	1.1 (%)	1.2 (%)
1.1 Train Set	49.0 ± 1.2	N/A
1.2 Train Set	N/A	66.5 ± 3.2
Merged Train Sets	68.5 ± 3.8	74.4 ± 3.2
Reversed Feature	69.0 ± 1.2	78.0 ± 3.6
ACL Embeddings	71.7 ± 0.7	80.5 ± 1.5
Context Words	71.3 ± 1.0	82.5 ± 1.6
Ensemble	**72.7**	**85.0**

Table 3: CNN Improvements over a series of modifications. Each row includes the modifications of the previous rows. All numbers are macro-F1 scores on test set after 10 runs in the form of {average}±{standard deviation} (the "Ensemble" row lacks deviation numbers as it, being a variance reduction technique, does not have the same sources of variation as the other models). We report ±20 context words here, which was found to be optimal in post-submission experimentation, but our submitted models used ±50 context words, which was preferred under initial cross validation.

data size of each label), but all proved ineffective at the cross validation level and were not included in our final submission.

4.2 Submission Results

We entered 6 submissions in total, 3 for subtask 1.1, and 3 for subtask 1.2. Their final performances are listed in Table 4. All the submissions were based on the ensemble model listed in Table 3 except that the lemmas were used instead of original words and we used ±50 context words. The major difference between submissions was the strategy for early stopping. Detailed settings for each submission of subtask 1.1 are:

Submission 1 We randomly extracted 10% training data as validation set and made test set predictions when the highest validation accuracy was reached.

Submission 2 We randomly extracted 10% training data as validation set with stratification (stratification is based on the proportion of labels in the train set) and made test set predictions when the highest validation accuracy was reached.

Submission 3 No validation set was used. Test set predictions were made at fixed number of training epochs, chosen by cross validation using training data.

Detailed settings for each submission of subtask 1.2 are:

Subtask	Submission	Macro-F1 (%)
1.1	1	71.5
	2	72.3
	3	**72.7**
1.2	1	**80.6**
	2	76.4
	3	79.8

Table 4: Final submission performance.

Submission 1 We randomly extracted 10% training data as validation set and made test set predictions when the highest validation accuracy was reached.

Submission 2 No validation set was used. Test set predictions were made at fixed number of training epochs, chosen by cross validation using training data.

Submission 3 Settings are the same as submission 2 except that we added the embeddings of the two entities (max pooled) to the sentence representation vector extracted by CNN model as additional features prior to scoring and inference.

In summary, early stopping made based on the validation set accuracy does not guarantee better test set performance than that using a fixed number of training epochs. Stratifying the label ratio of validation set according to the training set does help improve the test set performance. The benefit of adding embeddings of entities as extra features is not clear.

5 Future Work

Our custom ACL embeddings offered significant performance boosts over embeddings pre-trained on the corpus in the general domain such as wikipedia, but the corpus we obtained for pretraining is noisy due to the imperfect OCR. Cleaning up the input ACL dataset may result in better embeddings, offering even larger performance gains. Additionally, we could try restricting the ACL dataset to only those papers published more recently, in hopes to further specify the embedding space to a relevant subset—of course, this would also have reduced the embedding dataset size, so it may have cost more than it gained and experimentation would be warranted.

We also never explored any dependency tree-based featurizations of our data, though those were found to be helpful in prior works (Cai et al., 2016; Xu et al., 2015b).

The class imbalance of the training dataset is one of the greatest obstacles, where performance of common classes is a lot better than the rare classes. To tackle this problem, adversarial generative network (GAN) models (Goodfellow et al., 2014) could be used for data augmentation so that the data size of all labels can be equalized.

6 Conclusion

We tested linear classifiers, sequential random forests, LSTM models, and CNN models on these data. Within each model, we explored many variations, including two models of attention, negative sampling, entity embedding or sentence-only embeddings, among others. The most performant combination found was a CNN model trained over ACL-specific embeddings and a negative sampling augmented loss, without attention. We submitted self-ensembled predictions from this model both with and without early stop–ultimately early stop proved efficacious only on task 2. This model achieved rank 6 out of 28 (macro F1 72.7) in subtask 1.1, and rank 4 out of 20 (macro F1 80.6) in subtask 1.2.

Acknowledgements

This work was supported (in part) by funding grants U54-HG007963 from National Human Genome Research Institute (NHGRI) and P50-MH106933 from National Institute of Mental Health (NIMH).

References

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. Semeval 2017 task 10: Scienceie - extracting keyphrases and relations from scientific publications. *CoRR*, abs/1704.02853.

Rui Cai, Xiaodong Zhang, and Houfeng Wang. 2016. Bidirectional recurrent convolutional neural network for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 756–765.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Kata Gbor, Davide Buscaldi, Anne-Kathrin Schumann, Behrang QasemiZadeh, Hafa Zargayouna, and Thierry Charnois. 2018. Semeval-2018 task 7: Semantic relation extraction and classification in scientific papers. In *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, USA, June 2018.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. SemEval-2010 Task 8 : Multi-Way Classification of Semantic Relations Between Pairs of Nominals. *Computational Linguistics*, (June 2009):94–99.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ji Young Lee, Franck Dernoncourt, and Peter Szolovits. 2017. MIT at SemEval-2017 Task 10: Relation Extraction with Convolutional Neural Networks.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244.

DragomirR. Radev, Pradeep Muthukrishnan, Vahed Qazvinian, and Amjad Abu-Jbara. 2013. The ACL anthology network corpus. *Language Resources and Evaluation*, pages 1–26.

Bryan Rink and Sanda Harabagiu. 2010. UTD: Classifying Semantic Relations by Combining Lexical and Semantic Resources. *Proceedings of the 5th International Workshop on Semantic Evaluation*, (July):256–259.

Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. *arXiv preprint arXiv:1504.06580*.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation Classification via Multi-Level Attention CNNs. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. *CoRR*, abs/1506.07650.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1785–1794.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212.

Zhi-Hua Zhou and Ji Feng. 2017. Deep forest: Towards an alternative to deep neural networks. *arXiv preprint arXiv:1702.08835*.