TakeLab at SemEval-2018 Task 7: Combining Sparse and Dense Features for Relation Classification in Scientific Texts

Martin Gluhak, Maria Pia di Buono, Abbas Akkasi, Jan Šnajder
Text Analysis and Knowledge Engineering Lab
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3, 10000 Zagreb, Croatia
{name.surname}@fer.hr

Abstract

We describe two systems for semantic relation classification with which we participated in the SemEval 2018 Task 7, subtask 1 on semantic relation classification: an SVM model and a CNN model. Both models combine dense pretrained word2vec features and hancrafted sparse features. For training the models, we combine the two datasets provided for the subtasks in order to balance the under-represented classes. The SVM model performed better than CNN, achieving an F1-macro score of 69.98% on subtask 1.1 and 75.69% on subtask 1.2. The system ranked 7th among 28 submissions on subtask 1.1 and 7th among 20 submissions on subtask 1.2.

1 Introduction

Relation extraction is a traditional information extraction task which aims at detecting and classifying semantic relations between entities in text (Pawar et al., 2017). The task essentially induces structure from unstructured textual information, allowing us to obtain valuable information about the way in which entities interact, thus improving human capacity to analyze (often large) quantities of textual data. Relation extraction is typically framed as a classification task: pairs of entities from a document are inspected and the type of relation is predicted by means of local linguistic cues (Culotta et al., 2006).

Relation extraction has been extensively studied in the literature; see (Konstantinova, 2014) for a comprehensive overview. Etzioni et al. (2008) group the relation extraction approaches into three classes: (1) knowledge-based methods, (2) supervised methods, and (3) self-supervised methods. Traditional approaches mostly relied on shallow machine learning models with handcrafted features (GuoDong et al., 2005) and specific kernel methods (Zelenko et al., 2003). Some systems leverage unlabeled data to improve classification and use semi-supervised or unsupervised learning (Chen et al., 2006; Hasegawa et al., 2004). The current state of the art is a deep recurrent neural network model by Xu et al. (2016). Most research on relation extraction has leveraged standard benchmark datasets from the ACE (Doddington et al.) and SemEval-2010 Task 8 (Hendrickx et al., 2009).

This paper describes the systems with which we participated in the SemEval 2018 task 7 on Semantic relation extraction and classification in scientific papers. We focused on the subtask 1 (relation classification), which featured two scenarios: 1.1 Relation classification on clean (i.e., manually annotated) data and 1.2 Relation classification on noisy (i.e., automatically annotated) data. Both scenarios start out with pre-extracted entity pairs, which makes the task simpler than relation extraction proper. On the other hand, the task remains challenging because of the choice of the domain: scientific publications abound with complex syntactic structures and rely on specialist terminology, which makes it more difficult to predict the correct relation type. We framed the problem as a supervised classification task and devised two models: a support vector machine (SVM) model, which utilizes a rich set of features combining dense pretrained word2vec features and handcrafted sparse features, and a convolutional neural network (CNN) model. The best result was achieved with the SVM model, which ranked 7th in both scenarios of subtask 1.

2 Dataset

The organizers have provided different training datasets for scenarios 1.1 and 1.2 of subtask 1, each consisting of 350 abstracts of scientific papers from the Natural Language Processing (NLP) domain (Gábor et al., 2018) to be used for training the models. All entities representing domain concepts (e.g., *word sense disambiguation* and *translation*) are already annotated and listed in pairs according to one of the relations that holds between them. For example, in the sentence "High quality *translation* via *word sense disambiguation* and accurate word order generation of the target language", the entity *word sense disambiguation* is used for *translation*, hence it is annotated as an instance of the USAGE relation type. There are six distinct relation types: USAGE, TOPIC, COMPARE, MODEL-FEATURE, RESULT, PART-WHOLE. Except for COMPARE, all relations are asymmetric, which means that the direction of relation matters. For this reason, every asymmetrical relation instance has additionally been annotated with the direction of the relation using the "reverse" flag to indicate that the order of entities should be flipped.

The total number of training instances is 1228 and 1248 for subtask 1.1 and subtask 1.2 datasets, respectively. Each instance contains exactly two entities and both appear in the same sentence. The subtask 1.2 dataset uses the same annotation scheme as the first subtask, but the entities are automatically extracted rather than manually annotated, thus introducing noise.

Table 1 shows the breakdown of relation types for the two datasets. In general, the class distribution is rather imbalanced, especially the TOPIC relation, which is heavily under-represented in the dataset for subtask 1.1.

3 System Description

We devised two supervised machine learning models: an SVM with a rich set of features and a CNN model. We next describe these models in more detail.

3.1 SVM Model

Our best-performing model uses an SVM classifier with a combination of sparse and dense features.

Sparse feature encoding. To encode the sparse features, we adopt the method of Chen et al. (2006) who divide the sentence into contexts. In a nutshell, the method consists in describing the context in which the entities occur by dividing the sentence into five contexts around the entity mentions. Two of these five contexts are related to the entities involved in the relation, while the remaining three contexts represent the words before, between, and after the entity mention pair. Every feature describing a word contains the information about which context it is located in. It also additionally contains the position of the word relative to the start of that context. While this increases the total number of parameters, it also provides information about the word ordering in the context independently of the size of other contexts.

This encoding scheme is used to create features of word tokens, part-of-speech tags, and named entities. Similarly as Chen et al. (2006), who in turn followed Charniak (2000) and Zhang (2004), we experimented with additional constituency parse features describing grammatical functions and chunk tag information for all five contexts, and IOB-chains of the heads of the two entities. However, as preliminary experiments showed that these additional features do not provide any performance gains, we decided not to include them in our final models, intending to evaluate these results in future work.

Word windows. In the scientific domain sentence structures may be very complex, increasing the distance between the two entity mentions. This presents a considerable problem for sparse feature encoding of the context between two entities: as there the features are encoded for each word separately, the increase in distance yields a proportional increase in the number of parameters, which may cause overfitting. To mitigate this problem, instead of representing the contexts for the whole sentence, we use word windows that focus on the words around the entity mentions. Based on preliminary experiments, we decided to use different sizes of word windows for before, between, and after contexts: the window size for the context in between the entities is a maximum of eight words (at most four on the side of each entity; for longer distance the middle words get ignored), while the window size of the before and after contexts is at most two words to the left and right, respectively. The rationale for this is the assumption that words indicative of relation type are more likely to lie in between and close to one of the two entities.

Dependency features. Many systems from the literature make use of dependency features (Nguyen and Grishman, 2015a; Xu et al., 2016, 2015), which, intuitively, should be useful for re-

	Subtask 1.1			Subtask 1.2			
Relation type	Regular	Reversed	All	Regular	Reversed	All	Combined
USAGE	296	187	483	323	147	470	953
TOPIC	8	10	18	230	13	243	261
COMPARE	95	–	95	41	–	41	136
MODEL-FEATURE	226	100	326	123	52	175	501
RESULT	52	20	72	85	38	123	195
PART-WHOLE	158	76	234	117	79	196	430
Total	835	393	1228	919	329	1248	2476

Table 1: Class distribution in subtask 1 training datasets for scenario 1.1 (clean annotation) and scenario 1.2 (noisy annotation). Last column shows the combined counts of both datasets.

lation extraction and classification. Specifically for the relation classification task, we are interested in the connection between the two entities involved in a relation within an instance. Thus, after performing a dependency parse of the whole sentence in which the relation appears, we find the shortest dependency path between the two entities. As demonstrated by Xu et al. (2016), the shortest dependency path between two entities is advantageous over a raw word sequence or a whole parse tree, because it reduces irrelevant information and because the shortest path dependency relations focus on the action and agents in a sentence and are thus naturally suitable for relation classification. The dependency parsing was done using the SpaCy parser.[1]

Considering that the edges on the shortest dependency path also have directions, we split the path into two sub-paths based on the edge direction. Usually, the first path goes from the first entity node towards the lowest common ancestor in the dependency parse tree, while the other path goes downwards to the second entity from the ancestor node. Then, each sub-path is encoded into a series of one-hot sparse features that represent the edge dependency type and the distance of the edge from the starting point. Preliminary experiments showed that these features lead to substantial performance improvement.

Word embeddings and OOV words. To capture the semantic meaning of the words we used 300-dimensional pre-trained Google word2vec word embeddings.[2] In accordance with standard practice, which leverages the additive compositionality of word embeddings (Gittens et al., 2017), we represent each context as a sum of its word embeddings. As we divided the sentence into five context, the result is five vectors, which are subsequently concatenated into a single 1500-dimensional vector.

Using a pre-trained word2vec model combined with a scientific domain dataset led to many out-of-vocabulary (OOV) words. After some very basic word preprocessing, e.g., handling of hyphens and underscores, 1108 out of 9319 unique words from the datasets (combined datasets of subtasks 1.1. and 1.2) remained uncovered by the word2vec model. To tackle the OOV problem, we experimented with a fallback mechanism based on the context2vec (Melamud et al., 2016), an off-the-shelf lexical substitution system. Our idea was to retrieve the lexical substitutes for each OOV word[3] and retrieve their vectors instead – a technique akin to query expansion in information retrieval. We experimented with a number of variants of this method, however as none led to performance improvements, we decided not to include lexical substitution fallback in the final model.

Reversed relations. We use a boolean feature to indicate the reverse direction of a relation in an instance, as provided in the training datasets. With the addition of that feature the model can more easily differentiate between the regular and reversed instances of a class. We expand the number of classes from the initial six types of relations to include reversed relations as distinct classes, resulting in an 11-way classification model. The results obtained this way were better then using a 6-way classification model; while this appears counter-intuitive, we leave a more thorough investigation for future work. Once the predicted classes are obtained, we map regular and reversed relation types into one class, as in the original task.

Considering that COMPARE relation cannot be

[1]https://spacy.io/

[2]https://code.google.com/archive/p/word2vec/

[3]A lexical substitute is a meaning-preserving replacement of a word in its context.

reversed, we perform a post-prediction correction if a relation instance has an active reverse flag in the test set but the model predicted the instance to be of the COMPARE class. In this case, we change the prediction for that instance to be the second most probable class according to the predicted class probabilities.

3.2 CNN model

Motivated by the high performance that deep learning techniques have achieved in this area (Kumar, 2017), we experimented with a convolutional neural network (CNN) model. This model outperforms traditional feature-engineered approaches on relation extraction benchmarks, as shown by Nguyen and Grishman (2015b).

Architecture. The idea behind the use of a CNN model is to use the convolutional layer to capture the local content and meaning of a few consecutive words, depending on the size of the convolution kernel. Our CNN model comprises a single convolution layer connected to a max-pooling layer. As proposed by Nguyen and Grishman (2015b), pooling is followed by a dropout layer, which has been shown to work working well in fully-connected layers (Hinton et al., 2012; Wan et al., 2013; Krizhevsky et al., 2012). We use the softmax function at the output layer activation function. The model is implemented using TensorFlow.[4]

Features. The CNN model uses the same pretrained word2vec embeddings as the SVM model. Since the training data is limited, the word embeddings are kept static and not adjusted during training. In addition to word embeddings, the model is fed as input the position of words relative to the entities. In this way, the model is provided with the information on the positions of entities and the distance of context words, which could affect their relevance for predicting the relation type (Nguyen and Grishman, 2015b). Thus, each word has two position features: (1) a relative distance to the closest word for the first entity and (2) a relative distance to the closest word for the second entity. Position embeddings are randomly initialized to 50-dimensional vectors and are shared for the two entities. Unlike with the word embeddings, the position embeddings are adjusted during training. The assumption is that this relative distance from the entity mentions is inversely proportional to the importance of words. In line with this assumption, we capped the distance values at 15, as words at longer distances are likely not to have much effect on relation type.

Another feature that we added is the information about the shortest dependency path, as described by Nguyen and Grishman (2015a). Assuming that words from the shortest dependency path are more relevant than others, we add for each word a boolean value feature indicating whether that word belongs to the shortest dependency path. We also use an additional indicator feature to distinguish the relation instances with the REVERSE attribute, even though the CNN model has been trained as a 6-way classifier. All the features pertaining to a word are concatenated and considered as a single element when passed to the first layer of the network.

Lastly, we include word windows into the CNN model, in the same manner as we did for the SVM model, but this time increasing the window sizes of contexts around the entities, as this has less of an effect on the number of parameters than it was the case for the SVM model. Thus, we set the window size for between context to four words from each side, while the window size of before and after contexts has been set to the five words closest to the entities. For clarification, word windows simply decide the size of the sentence (not to be confused with position features, which embed previously discussed distances of each word inside the window).

4 Evaluation and Results

As subtask 1.1 and 1.2 are very similar and differ only in how the labeling was carried out (manual or automatic), we decided to combine the training sets of the two tasks in one training set. This allowed us to increase the number of instances for the TOPIC relation type, which was severely under-represented in subtask 1.1 training set (cf. Table 1).

The hyperparameters for the SVM model were selected using cross-validated grid search. The CNN model was trained using early stopping with batch size of 64. Kernel sizes of the convolution layers are 3, 4, and 5 words, each size having 32 kernels. Adam algorithm was used for the model optimization. A dropout rate of 0.5 was used during the training.

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

Test set	Model	P	R	F1
Subtask 1.1	SVM	64.64	75.57	69.68
	CNN	63.35	72.42	67.58
Subtask 1.2.	SVM	71.64	80.24	75.69
	CNN	14.78	15.81	15.28

Table 2: Relation classification results

The evaluation was performed on the test sets provided by the task organizers. Each test set is comprised of 150 scientific paper abstracts with 350 relation instances. Table 2 shows macro-averaged precision, recall, and F1 score for the two models on the two test sets.

Differently from previous findings (Nguyen and Grishman, 2015b), in our case the SVM outperformed the CNN model. This might be explained by the relatively small size size of the training set. On subtask 1.1, the CNN performs slightly worse than the SVM model, while it fails completely on the second subtask. We presume this might be due to an implementation error, but we were unable to identify the problem. In the official SemEval competition, the SVM model ranked 7th among 28 submissions on subtask 1.1 and 7th among 20 submissions on subtask 1.2.

5 Conclusion

We described two models for relation classification with which participated in the SemEval-2018 Task 7, subtasks 1.1 and 1.2 on relation classification: an SVM model and a CNN model. Our models combine sparse, handcrafted features and dense features based on word embeddings. Although deep learning models currently excels at relation extraction tasks, in our case, probably due to the small relatively small training set available, SVM outperformed the CNN model, ranking 7th in both evaluation runs. Overall, the tasks proved to be very challenging, mainly due to the peculiarities of the domain.

For future work, we intend to retrain our models on a larger dataset using distant supervision based on publicly available scientific corpora, and also experiment with training domain-specific word2vec embeddings.

Acknowledgments

This research has been partly supported by the European Regional Development Fund under the grant KK.01.1.1.01.0009 (DATACROSS).

References

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139. Association for Computational Linguistics.

Jinxiu Chen, Donghong Ji, Chew Lim Tan, and Zhengyu Niu. 2006. Relation extraction using label propagation based semi-supervised learning. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 129–136. Association for Computational Linguistics.

Aron Culotta, Andrew McCallum, and Jonathan Betz. 2006. Integrating probabilistic extraction models and data mining to discover relations and patterns in text. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 296–303. Association for Computational Linguistics.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. The automatic content extraction (ACE) program-tasks, data, and evaluation.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. 2008. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74.

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.

Alex Gittens, Dimitris Achlioptas, and Michael W. Mahoney. 2017. Skip-gram-zipf+ uniform= vector additivity. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 69–76.

Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 427–434. Association for Computational Linguistics.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 415. Association for Computational Linguistics.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian

Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. SemEval-2010 Task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99. Association for Computational Linguistics.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Natalia Konstantinova. 2014. Review of relation extraction methods: What is new out there? In *International Conference on Analysis of Images, Social Networks and Texts_x000D_*, pages 15–28. Springer.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Shantanu Kumar. 2017. A survey of deep learning methods for relation extraction. *arXiv preprint arXiv:1705.03645*.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61.

Thien Huu Nguyen and Ralph Grishman. 2015a. Combining neural networks and log-linear models to improve relation extraction. *arXiv preprint arXiv:1511.05926*.

Thien Huu Nguyen and Ralph Grishman. 2015b. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 39–48.

Sachin Pawar, Girish K Palshikar, and Pushpak Bhattacharyya. 2017. Relation extraction: A survey. *arXiv preprint arXiv:1712.05191*.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066.

Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2016. Improved relation classification by deep recurrent neural networks with data augmentation. *arXiv preprint arXiv:1601.03651*.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. 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.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research*, 3(Feb):1083–1106.

Zhu Zhang. 2004. Weakly-supervised relation classification for information extraction. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 581–588. ACM.