CLEARumor at SemEval-2019 Task 7: ConvoLving ELMo Against Rumors

Ipek Baris[1,*] and **Lukas Schmelzeisen**[1,*] and **Steffen Staab**[1,2]
[1]Institute for Web Science and Technologies (WeST), University of Koblenz-Landau, Germany
[2]Web and Internet Science Group (WAIS), University of Southampton, United Kingdom
ibaris@uni-koblenz.de, lukas@uni-koblenz.de, staab@uni-koblenz.de

Abstract

This paper describes our submission to SemEval-2019 Task 7: RumourEval: Determining Rumor Veracity and Support for Rumors. We participated in both subtasks. The goal of subtask A is to classify the type of interaction between a rumorous social media post and a reply post as support, query, deny, or comment. The goal of subtask B is to predict the veracity of a given rumor. For subtask A, we implement a CNN-based neural architecture using ELMo embeddings of post text combined with auxiliary features and achieve a F_1-score of 44.6%. For subtask B, we employ a MLP neural network leveraging our estimates for subtask A and achieve a F_1-score of 30.1% (second place in the competition). We provide results and analysis of our system performance and present ablation experiments.

1 Introduction

Online social media has changed the way of communicating and disseminating media content and opinions, but also paved the way for spreading false or unverified rumors.

RumourEval 2019 (Gorrell et al., 2019) provides a dataset of labelled threads from Twitter and Reddit where each source post mentions a rumor. Subtask A (SDQC) consists of deciding for each post in a thread whether it is in a *support*, *deny*, *query*, or *comment* relation to the rumor. The goal of subtask B (Verification) it to classify the veracity of the rumor as *true*, *false*, or *unverified*. Figure 1 clarifies our terminology and the tasks.

Automated rumor classification is a challenging task as there is no definite evidence (e.g., authorized confirmation). In its absence, stance analysis is a useful approach. Systems that employ neural network architectures showed promising results in RumourEval 2017 (Derczynski et al., 2017), with the LSTM-based sequential model of Kochkina et al. (2017) performing best.

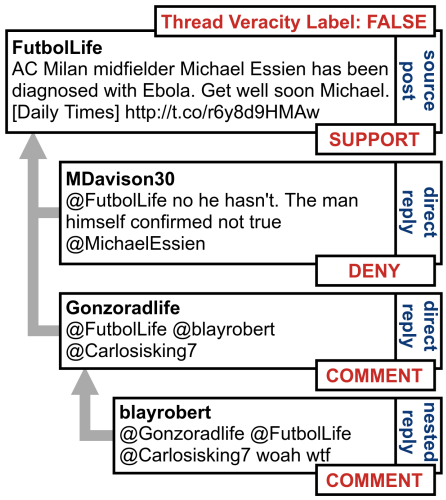

Figure 1: An example Twitter thread from the training dataset with SDQC labels for each post and a veracity label for the thread's source post. Any post that does not reply to another is a *source post*. Reply posts can be *direct replies* (replies to a source post) or *nested replies* (replies that reply to another reply post). A *thread* is the set containing a source post and all its reply posts.

In this paper, we describe our approach CLEARumor (ConvoLving ELMo Against Rumors) for solving both subtasks and provide empirical results and ablation experiments of our architecture. We make our PyTorch-based implementation and trained models publicly available[1].

2 System Description

After preprocessing the post text (Section 2.1) and embedding it with ELMo (Section 2.2), our architecture for subtask A (Section 2.3) passes the embedded text through a convolutional neural network (CNN) block, adds auxiliary features, and

*The first two authors contributed equally.

[1]https://github.com/Institute-Web-Science-and-Technologies/CLEARumor

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 1105–1109
Minneapolis, Minnesota, USA, June 6–7, 2019.

uses a multilayer perceptron (MLP) block for estimating class membership. These estimates are combined with further auxiliary features and fed into an MLP block for the classification for subtask B (Section 2.4).

2.1 Preprocessing

For preprocessing, we rely mostly on Erika Varis Doggett's tokenizer for Twitter and Reddit[2], with which we strip away all user handles (e.g., "@FutbolLife"), remove the number sign in front of hash tags (e.g., "#Ebola" becomes "Ebola"), remove URLs, and limit repetitions of the same character to at most three times (e.g., "heeeeey" becomes "heeey"). We further decided to lowercase all text, which resulted in improved performance over mixed case in initial experiments. Last, all posts are truncated after 32 tokens[3].

2.2 ELMo Embeddings

The task of word embedding is to represent each word in a given sentence by a vector, which among other things allows for encoding words at the input layer in a neural network architecture. Traditional embedding methods such as word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014) work independently of context and always map the same word to the same vector.

In contrast, ELMo (Peters et al., 2018) is a recent embedding approach based on bidirectional LSTM networks that considers the context a word occurs in and is thereby able to address certain linguistic peculiarities, e.g., that the same word can have different meanings depending on its context. Further, ELMo incorporates subword units and is thereby able to represent words not seen during training successfully, an important benefit for the social media domain, where users frequently misspell existing words or introduce new ones. Formally, given a sequence of words $w_1 w_2 \dots w_n$ ELMo represents the k-th word w_k as

$$\mathbf{ELMo}_k^{\text{task}} = \gamma^{\text{task}} \sum_{j=0}^{L} s_j^{\text{task}} \mathbf{h}_{k,j}, \qquad (1)$$

where L gives the number of internal layers that were used to train ELMo, $\mathbf{h}_{k,j}$ is the contextual vector representation of layer j for word k, and γ^{task} and the s_j^{task} are scalars that can be tuned specifically for the task at hand.

We report results for the pretrained model `elmo_2x4096_512_2048cnn_2xhighway_5.5B`[4] for which $L = 2$ and which outputs 1024-dimensional embedding vectors (but didn't notice drastic improvements over the much smaller models). ELMo allows us to fine-tune γ^{task}, s_j^{task}, and even the $\mathbf{h}_{k,j}$ by backpropagating gradients to them, but we decided against this, because the RumourEval dataset is very small (cf. Table 1) adjusting these weights can quickly lead to overfitting, and keeping the weights constants allows us to precompute and store all ELMo embeddings once before the training process which results in a major boost in performance.

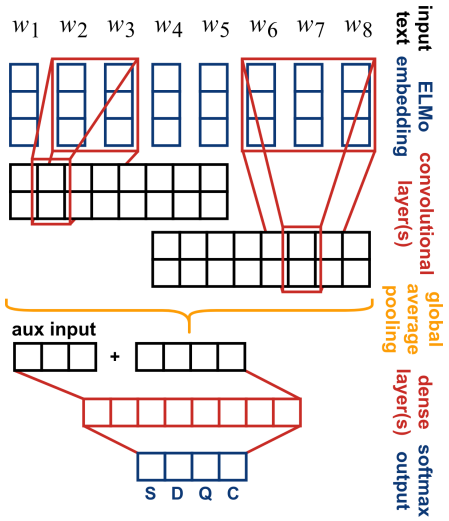

Figure 2: CLEARumor architecture for subtask A.

2.3 Subtask A

Our architecture for subtask A is visualized in Fig. 2. First, the tokenized text of the post that is to be classified as either support, deny, query, or comment is represented with an ELMo embedding. Next, the embedded text is fed into L_{conv}-many convolutional layers. Here, a single convolutional layers consists of multiple 1D-convolution operations with a set of different kernel sizes S, each mapping onto C convolutional channels, which are then concatenated along the channel axis. Each convolution operation is batch normalized (Ioffe and Szegedy, 2015) after a ReLU activation. To maintain an equal sequence length, sequences are padded with zero vectors. The result-

[2] https://github.com/erikavaris/tokenizer

[3] Only 10 out of the total 6634 Twitter posts are longer than this, while a few Reddit posts are up to 1,000 tokens long which would result in very impractical batch sizes.

[4] https://allennlp.org/elmo

ing sequence representation is transformed into a single $|S| \cdot C$-dimensional vector via global average pooling. This sequence vector is concatenated with a vector of auxiliary features that encodes meta information about the post under classification (detailed in the next paragraph). Following is a stack of L_{dense}-many dense layers, for which dropout-regularization (Srivastava et al., 2014) is performed after ReLU activation. Finally, a single linear layer that is softmax-activated yields the four estimates of class membership. Parameters are optimized using Adam (Kingma and Ba, 2015) and a cross-entropy loss.

We use the following auxiliary features: (1) a two-dimensional Boolean vector encoding whether the post is from Twitter or Reddit; (2) a five-dimensional real-valued vector encoding meta-information for the post author: whether the user is verified or not, the number of followers they have, the number of accounts they follow themselves, and a ratio of the latter two numbers[5]; (3) the cosine similarity of the averaged ELMo embeddings of the post under classification to those of the thread's source post (defined to be 1 for source posts); and (4) a three-dimensional Boolean vector encoding whether the post is a source post, a direct reply, or a nested reply.

As hyperparameters, we employ a learning rate of 10^{-3}, a batch size of 512, and train for 100 epochs. In our loss function, we weigh the estimates of support, deny, and query equally but that of comment at only a fifth of the strength because of the imbalance of the dataset. We add L2-regularization with a weight of 10^{-2}. In our reported results we use $L_{\text{conv}} = 1$ convolutional layer, with kernel sizes $S = \{2, 3\}$ each mapping into $C = 64$ channels, after which follow $L_{\text{dense}} = 3$ dense layers with 128 hidden units each and a dropout of 0.5.

2.4 Subtask B

For subtask B we build a single feature vector that we feed into a MLP classifier. We reuse all the auxiliary features from subtask A except the last two, because all posts under classification in subtask B are source posts. We further add the following features: (1) a two-dimensional Boolean vector encoding whether media (an image or a URL) is attached to the post, (2) the upvote-to-downvote ratio of the post for Reddit (manually set to 0.5 for Twitter), (3) a two-dimensional real-valued vector encoding which fraction of the thread's posts are direct replies and which fraction are nested replies, (4) the averaged support, deny, and query probability estimates from subtask A averaged over all posts in the thread. Similarly to subtask A, this feature vector is fed into a stack of L_{dense}-many dense layers with dropout-regularization (Srivastava et al., 2014) after ReLU activation, after which a single softmax-activated linear layer yields estimates for the three classes true, false, or unverified.

Our model was trained with a learning rate of 10^{-3} and a batch size of 128 for 5000 epochs. In our loss calculation, we weigh the unverified class at 0.3 of the strength of the other two, and add a L2-regularization weight of 10^{-2}. We used $L_{\text{dense}} = 2$ dense layers with 512 hidden units each and a dropout of 0.25.

3 Evaluation

The dataset of RumourEval 2019 is summarized in Table 1. Our results for subtask A and B are

Subtask A		**S**	**D**	**Q**	**C**	**Σ**
Train	Twitter	910	344	358	2907	**5217**
	Reddit	15	34	37	612	
Dev	Twitter	94	71	106	778	**1485**
	Reddit	8	11	14	403	
Test	Twitter	141	92	62	771	**1827**
	Reddit	16	9	31	705	
Σ		**1184**	**561**	**608**	**6176**	**8529**

Subtask B		**T**	**F**	**U**	**Σ**
Train	Twitter	137	62	98	**327**
	Reddit	7	17	6	
Dev	Twitter	8	12	8	**38**
	Reddit	2	7	1	
Test	Twitter	22	30	4	**81**
	Reddit	9	10	6	
Σ		**185**	**138**	**133**	**456**

Table 1: Number of labelled instances for both subtasks of the RumourEval 2019 dataset broken down into (1) class frequencies, per (2) social media platform, and (3) training, development, and test dataset.

[5]We use min-max scaling based on the training data for these features. For Reddit the respective concepts don't exist and a vector of zeros is used instead.

Subtask A	Dev	Test					CV
	Macro-F_1	**Macro-F_1**	**S-F_1**	**D-F_1**	**Q-F_1**	**C-F_1**	**Macro-F_1**
Always Comment	22.1	22.3	0.0	0.0	0.0	89.4	—
Submitted	41.3	37.4	**46.7**	0.0	11.7	**91.2**	—
$\text{CLEAR}^{\text{aux}}$	**44.8**±0.6	42.7±0.6	29.6±0.6	**17.8**±2.4	43.9±1.0	79.5±1.3	47.1±4.5
$\text{CLEAR}^{\text{aux}}_{\text{MLP}}$	42.2±1.2	40.7±1.6	30.7±2.7	0.0±0.0	**51.6**±3.2	80.5±2.7	44.7±4.2
$\text{CLEAR}_{\text{CNN+MLP}}$	39.7±2.0	39.0±2.2	16.2±2.3	14.8±3.4	41.0±6.7	84.0±2.6	43.3±4.5
$\text{CLEAR}^{\text{aux}}_{\text{CNN+MLP}}$	42.9±2.2	**44.6**±2.6	34.6±3.7	15.4±3.1	42.2±8.3	86.1±1.1	**47.2**±3.8

Table 2: Evaluation results for subtask A. All reported scores are multiplied by 100. We provide the macro-averaged F_1-score for the development (Dev), the test (Test) datasets and for 10-fold cross validation (CV). For the test dataset, we further provide the individual F_1-scores per class. "Always Comment" is a baseline predicting always the most common class. "Submitted" are the results we officially submitted to RumourEval 2019. For our CLEARumor architecture we provide multiple ablation experiments. $\text{CLEAR}^{\text{aux}}_{\text{CNN+MLP}}$ is our full system, $\text{CLEAR}_{\text{CNN+MLP}}$ the same but without the auxiliary features, $\text{CLEAR}^{\text{aux}}_{\text{MLP}}$ instead uses no convolutional layers, and $\text{CLEAR}^{\text{aux}}$ just concatenates averages ELMo embeddings with auxiliary features uses a single linear layer.

Subtask B	Dev		Test		CV	
	Macro-F_1	**RMSE**	**Macro-F_1**	**RMSE**	**Macro-F_1**	**RMSE**
Submitted	41.7	0.743	28.6	0.764	—	—
$\text{CLEAR}_{\text{Subtask-B}}$	35.4±0.5	**0.676**±0.005	**30.1**±0.8	**0.754**±0.005	**26.7**±13.4	**0.733**±0.113
$\text{CLEAR}_{\text{NileTMRG}}$	**53.5**	0.761	18.6	0.846	—	—

Table 3: Evaluation results for subtask B. We report F_1 (multiplied by 100) and RMSE (root mean squared error) scores for the development (Dev), the test (Test) datasets and for 10-fold cross validation (CV). $\text{CLEAR}_{\text{Subtask-B}}$ is our subtask B architecture using the subtask A estimates from $\text{CLEAR}^{\text{aux}}_{\text{CNN+MLP}}$. $\text{CLEAR}_{\text{NileTMRG}}$ uses the same estimates but computes task B results using the NileTMRG system (Enayet and El-Beltagy, 2017).

detailed in Table 2 and Table 3, respectively.

The reported results differ from our official submission, because we continued to tune hyperparameters afterwards. We report results as trained on the training dataset and then evaluated on the development and test datasets, as provided by the RumourEval organizers. Because neural network experiments are naturally nondeterministic (Reimers and Gurevych, 2018) and we did indeed notice huge variances when retraining models, we report the mean and standard deviation over 10 runs for each experiment. Additionally, we report scores from a 10-fold cross validation over the whole dataset. Simple cross-validation would be inappropriate in our setting, because for example a split could result in the case where the same rumors occur in both the training and the test dataset which would allow a model to just memorize which posts are rumorous. We ensure that this does not happen in our case, by keeping all posts belonging to the same rumor[6] in the same cross validation fold. Note that scores on the organizer split and the cross validation are not directly comparable as different fractions of the whole dataset are used for training (~60-70% for the organizer split and ~90% for 10-fold cross validation).

4 Conclusion

We have presented CLEARumor, our architecture for the RumourEval 2019 shared tasks. In future we aim to generalize our approach, e.g., we currently use domain-specific features for characterizing the post author popularity, such as number of followers for Twitter, which are not available for all social media platforms. Besides investigating how well our approach translates to other languages, we are interested in studying the results for other pretrained word representation approaches, e.g., BERT (Devlin et al., 2018).

[6]For Twitter posts, the dataset contains rumor-topic labels for each thread, so we ensure that each topic only occurs in one fold. For Reddit posts, no labelling is available, so we can only ensure that all posts of a thread occur in the same fold.

Acknowledgments

Ipek Baris works for the EU-Project Co-Inform which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 770302. Lukas Schmelzeisen is supported by the German Research Foundation (DFG) as part of the Open Argument Mining project (grant STA 572/18-1).

References

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. SemEval-2017 Task 8: RumourEval: Determining rumour veracity and support for rumours. In *SemEval@ACL*, pages 69–76. ACL.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, abs/1810.04805.

Omar Enayet and Samhaa R. El-Beltagy. 2017. NileTMRG at SemEval-2017 Task 8: Determining Rumour and Veracity Support for Rumours on Twitter. In *SemEval@ACL*, pages 470–474. ACL.

Genevieve Gorrell, Kalina Bontcheva, Leon Derczynski, Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2019. SemEval-2019 Task 7: RumourEval: Determining Rumour Veracity and Support for Rumours. In *SemEval@NAACL-HLT*. ACL. To appear.

Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Elena Kochkina, Maria Liakata, and Isabelle Augenstein. 2017. Turing at SemEval-2017 Task 8: Sequential Approach to Rumour Stance Classification with Branch-LSTM. In *SemEval@ACL*, pages 475–480. ACL.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*, pages 1532–1543. ACL.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *NAACL-HLT*, pages 2227–2237. ACL.

Nils Reimers and Iryna Gurevych. 2018. Why Comparing Single Performance Scores Does Not Allow to Draw Conclusions About Machine Learning Approaches. *CoRR*, abs/1803.09578.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958.