Identifying *attack* and *support* argumentative relations using deep learning

Oana Cocarascu and **Francesca Toni**
Department of Computing
Imperial College London

Abstract

We propose a deep learning architecture to capture argumentative relations of *attack* and *support* from one piece of text to another, of the kind that naturally occur in a debate. The architecture uses two (unidirectional or bidirectional) Long Short-Term Memory networks and (trained or non-trained) word embeddings, and allows to considerably improve upon existing techniques that use syntactic features and supervised classifiers for the same form of (relation-based) argument mining.

1 Introduction

Argument Mining (AM) is a relatively new research area which involves, amongst others, the automatic detection in text of arguments, argument components, and relations between arguments (see (Lippi and Torroni, 2016) for an overview). We focus on a specific type of AM, referred to as *Relation-based AM* (Carstens and Toni, 2015), which has recently received attention by several researchers (e.g. see (Bosc et al., 2016; Carstens and Toni, 2017)). This type of AM aims at identifying argumentative relations of *attack* and *support* between natural language arguments in text, by classifying pairs of pieces of text as belonging to *attack*, *support* or *neither attack nor support* relations. For example, consider the three texts taken from Carstens and Toni (2015):

t_1: *'We should grant politicians immunity from prosecution'*

t_2: *'Giving politicians immunity allows them to focus on performing their duties'*

t_3: *'The ability to prosecute politicians is the ultimate protection against abuse of power'*

Here t_2 *supports* t_1, t_3 *attacks* t_1, and t_2 and t_3 *neither attack nor support* one another.

Relation-based AM is useful, for example, to pave the way towards identifying accepted opinions (Bosc et al., 2016) or divisive issues (Konat et al., 2016) within debates.

We propose a deep learning architecture for Relation-based AM based on Long-Short Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997). Within the architecture, each input text is fed, as a (trained or non-trained) 100-dimensional GloVe embedding (Pennington et al., 2014), into a (unidirectional or bidirectional) LSTM which produces a vector representation of the text independently of the other text being analysed. The two vectors are then merged (using element-wise sum or concatenation) and the resulting vector is fed to a softmax classifier which predicts whether the pair of input texts belongs to the *attack*, *support* or *neither* relations. The input texts may be at most 50 words long, but are not restricted to single sentences.

We experimented with several instances of the architecture and achieved 89.53% accuracy and 89.07% F_1 using unidirectional LSTMs and concatenation as the merge layer, considerably outperforming feature-based supervised classifiers used in the studies which presented the corpus we also use (Carstens and Toni, 2015, 2017).

The remainder of the paper is organised as follows. In Section 2 we discuss related work and the corpus we use. In Section 3 we describe our deep learning architecture and report experiments and results in Section 4. We conclude the paper and propose directions for future work in Section 5.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1374–1379
Copenhagen, Denmark, September 7–11, 2017.

2 Background

2.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) (Elman, 1990; Mikolov et al., 2010) are a type of neural networks in which the hidden layer is connected to itself so that the previous hidden state is used along with the input at the current step. RNNs tend to suffer from the vanishing gradients problem (Bengio et al., 1994) while trying to capture long-term dependencies.

LSTM models (Hochreiter and Schmidhuber, 1997), a type of RNNs, address this problem by introducing memory cells and gates into networks. LSTMs use memory cells to store contextual information and three types of gates (input, forget, and output gates) that determine which information needs to be added or removed to learn long-term dependencies within a sequence.

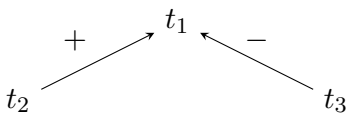
One problem with RNNs/LSTMs in natural language processing is that they do not make use of the information of future words. Bidirectional RNNs/LSTMs (BiRNNs/BiLSTMs) (Schuster and Paliwal, 1997) solve this problem by using both previous and future words while processing the input sequence with two RNNs: one in the forward and one in the backward direction, resulting in two vectors for each input.

2.2 Related work

Identifying relations between texts has recently received a great deal of attention, e.g. in Argument Mining (AM) (see (Lippi and Torroni, 2016) for a recent overview). In particular, *Relation-based AM* (Carstens and Toni, 2015) aims to automatically identify argumentative relations to create Bipolar Argumentation Frameworks (BAFs) (Cayrol and Lagasquie-Schiex, 2005).

BAFs are triples $\langle AR, attacks, supports \rangle$ consisting of a set of arguments AR and two binary relations $attacks$ and $supports$ between arguments.

The example texts introduced in Section 1 form a BAF with $AR = \{t_1, t_2, t_3\}$ and $attacks$, $supports$ given graphically (as -, + respectively) as follows:



Carstens and Toni (2017) obtained 61.8% accuracy and 62.1% F_1 on a news articles using Support Vector Machines (SVMs) and features such as distance measures, word overlap, sentence metrics and occurences of sentiment words.

Bosc et al. (2016) used a corpus consisting of tweets to determine *attack* and *support* relations between tweets. Using an encoder-decoder architecture and two LSTMs (the second LSTM initialised with the last hidden state of the first LSTM), they obtained negative results (0.2 F_1 for *support* and 0.16 F_1 for *attack*).

Other works in AM use deep learning models to determine relations between arguments, but of a different kind than in our work. Notably, Habernal and Gurevych (2016) experimented with LSTM models extended with an attention mechanism and a convolution layer over the input pairs to determine whether an input argument is more convincing than the other input argument. Thus, their focus is on determining a "more convincing than" relation, rather than *attack* and *support* argumentative relations, between arguments.

Several authors used neural network models for tasks related to the form of AM we consider. Yin et al. (2016) proposed three attention mechanisms for Convolutional Neural Networks to model pairs of sentences in tasks such as textual entailment and answer selection, whereas dos Santos et al. (2016) proposed a two-way attention mechanism to jointly learn the representation of two inputs in an answer selection setting.

Bowman et al. (2015) used stacked LSTMs to determine entailment, neutral and contradiction relations amongst sentence pairs using the SNLI (Stanford Natural Language Inference) corpus, with the bottom layer taking as input the concatenation of the input sentences. Recognising textual entailment between two sentences was also addressed in (Rocktäschel et al., 2015) which used LSTMs and a word-by-word neural attention mechanism on the SNLI corpus.

Liu et al. (2016) proposed two models capturing the interdependencies between the two parallel LSTMs encoding two input sentences for the tasks of recognising textual entailment and matching questions and answers. Further, Koreeda et al. (2016) used a BiRNN with a word-embedding-based attention model to determine whether a piece of an evidence supports a claim that a phrase promotes or suppresses a value, using a dataset of 1000 pairs.

2.3 Dataset

Determining relations between any texts can be seen as a three-class problem, with labels $L = \{attack, support, neither\}$. We used a dataset covering various topics such as movies, technology and politics[1], where *attack* relations represent 31% of the dataset, *support* relations represent 32% of the dataset and *neither* relations represent 37% of the dataset.

We have also explored the use of other corpora (e.g. the SNLI corpus (Bowman et al., 2015) and Araucaria in AIFdb[2]) that we ultimately decided not to include due to their structure not being directly amenable to our analysis.

3 Architecture

Figure 1 summarises the deep learning architecture that we use for predicting which relation from $L = \{attack, support, neither\}$ holds between the first and the second texts in any input pair.

We do not limit input texts to be single sentences, but limit them to 50 words (as this is the average text length in our corpus): inputs whose size is smaller than this threshold are padded with zeros at the end to give sequences of exactly 50 words. The input texts are (separately) embedded as 100-dimensional GloVe vectors (Pennington et al., 2014), with the words that do not appear in the vectors being treated as unknown. As we will see in Section 4, we experimented with the pre-trained word representations (freezing the weights during learning) as well as learning the weights.

The architecture relies upon two parallel LSTMs to model the two texts separately. We experimented with both unidirectional and bidirectional LSTMs (see Section 4). In both cases, we set the LSTM dimension to 32, as this proved to be the best, amongst alternatives (64, 100, 128), for mitigating overfitting. In addition, our LSTMs use a Rectified Linear Unit (ReLU) activation, each returning a vector of dimension 32.

Each LSTM network produces a vector representation of the input text, independently of the other text being analysed. The two vectors are then merged and the resulting vector fed to a softmax classifier which predicts the label for the relation between the first and the second input texts. As we will see in Section 4, we experimented with two types of merge layer: *sum*, which performs element-wise sum, and *concat*, which performs tensor concatenation.

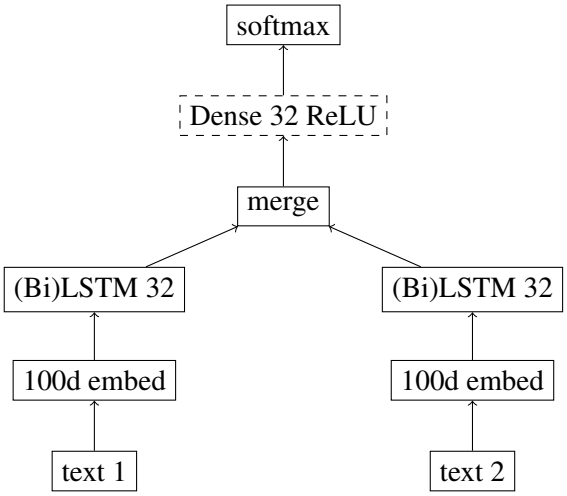


Figure 1: Our architecture: two (unidirectional or bidirectional) LSTMs are run with one text each. The dashed layer (Dense 32 ReLU) is optional.

Hyper-parameter	Value	Hyper-parameter	Value
Dropout	0.2	LSTM size	32
Embedding size	100	Dense size	32
Sequence length	50	Batch size	128

Table 1: Hyper-parameters for our (Bi)LSTMs.

After the merge layer, our architecture incorporates an optional dense feedforward layer. Our experiments (see Section 4) included testing whether the inclusion of this layer has an impact on the results. Again, we chose the dimension (32) as it proved better for mitigating overfitting than alternatives that we tried (64).

The values for the hyper-parameters used in our experiments (see Section 4) are summarised in Table 1. We used a mini-batch size of 128 and cross-entropy loss. To avoid overfitting, we applied dropout before the merge layer with probability 0.2, but not on the recurrent units. The hyper-parameters were optimised using the Adam method (Kingma and Ba, 2014) with learning rate 0.001, which turned out to give better performances than alternative optimisers we tried (Adagrad, Adadelta and RMSprop).

[1] https://www.doc.ic.ac.uk/~lc1310/
[2] https://corpora.aifdb.org

4 Results

We trained for 50 epochs or until the performance on the development set stopped improving, in order to avoid overfitting. The development set was 20% of the training dataset in the 10-fold cross-validation setup. In more detail, we run 10 stratified fold cross-validation for 5 times (so that each fold is a good representative of the whole). We report the average results of the 5x10 fold cross-validation in Table 2. As baseline, we used Logistic Regression (LR) and unigrams obtained from concatenating the two input texts.

We experimented with using BiLSTMs and unidirectional LSTMs with the two types of merge layers and using non-trained embeddings, namely pre-trained word representations (freezing the weights during learning), or trained embeddings, learning the weights during training.

We achieved 89.53% accuracy and 89.07% F_1 by concatenating the output of the two separate LSTMs. Unexpectedly, BiLSTMs performed worse than LSTMs (Table 2 only includes the best performing BiLSTM instance of the architecture, using concatenation and the feedforward layer). We believe this is because of the size of the dataset and that this effect could be diminished by acquiring more data. For the LSTM model with trained embeddings, the accuracy varied between 84.84% and 90.02%. Concatenating the LSTMs' output vectors yields better performance than performing element-wise sum of the vectors. We believe this is because this allows the system to encode more features, allowing the network to use more information.

Using the default, pre-trained word embeddings yields worse results compared to the baseline. We believe this is because the quality of word embeddings is dependent on the training corpora.[3] Training the word embeddings results in better performance compared to the baseline with improvements of up to 12% in accuracy and up to 11.5% in F_1.

In all cases, training the word embeddings results in dramatic improvements compared to freezing the embedding weights during learning, varying from 9.9% to 21.3% increase in accuracy and up to 25% in F_1. We also report the standard deviation of our models with trained embeddings. This shows that our best models (LSTMs with a concatenation layer) are stable and perform consistently on the task considered. Using One-Way ANOVA, the result is significant at $p < 0.05$ (the f-ratio value is 145.45159, the p-value is < 0.00001).

[3] Pennington et al. (2014) computed the 100-dimensional GloVe embeddings on a a dump of English Wikipedia pages from 2014 consisting of 400k words.

5 Conclusion

We proposed a deep learning architecture based on Long Short-Term Memory (LSTM) networks to capture the argumentative relation of *attack* and *support* between any two texts. Our architecture uses two (unidirectional or bidirectional) LSTMs to analyse separately two 100-dimensional (non-trained or trained) GloVe vectors representing the two input texts. The outputs of the two LSTMs are then concatenated and fed to a softmax classifier to predict the relation between the input texts.

Our unidirectional LSTM model with trained embeddings and a concatenation layer achieved 89.53% accuracy and 89.07% F_1. The results indicate that LSTMs may be better suited for Relation-based Argument Mining at least for non-micro texts (Bosc et al., 2016) than standard classifiers as used in e.g. Carstens and Toni (2017), as LSTMs are better at capturing long-term dependencies between words and they operate over sequences, as found in text.

In future work, we plan to test our model on corpora such as the Language of Opposition from AIFdb[4] (by converting the finer-grained relation types used in this corpus to argumentative relations of the kind we considered), on datasets proposed for different tasks (e.g. identifying textual entailment could be seen as identifying support) and thus possibly use the corpus proposed by Bowman et al. (2015), as well as the twitter dataset of Bosc et al. (2016) once it becomes publicly available. Also, attack and support relations of the kind we have considered in this paper may be seen as special types of discourse relations (Teufel et al., 1999; Lin et al., 2009). It would be interesting to see whether any corpora for identifying discourse relations could be useful for furthering our experimentation. Finally, we plan to incorporate an attention-based mechanism as well as additional features (e.g. extracted through sentiment analysis) to determine which parts of the texts are most relevant in identifying attack and support.

[4] https://corpora.aifdb.org

Baseline	**A%**	**P%**	**R%**	**F_1%**						
LR (unigrams)	77.87	78.02	77.87	77.89						
Model/	**Non-trained embeddings**				**Trained embeddings**					
Merge/Dense	**A%**	**P%**	**R%**	**F_1%**	**A%**	**P%**	**R%**	**F_1%**	**A std**	**F_1 std**
BiLSTM/c/T	60.72	64.36	52.64	57.36	70.66	73.18	62.96	66.93	2.06	4.60
LSTM/c/F	68.25	72.39	59.07	64.38	**89.53**	90.80	87.67	**89.07**	**0.47**	**0.73**
LSTM/c/T	68.68	72.77	58.21	63.49	90.02	90.89	88.26	89.41	2.09	2.92
LSTM/s/T	64.21	69.18	51.07	57.09	84.84	86.75	79.98	82.35	5.02	9.26

Table 2: 5x10 fold cross-validation results, using *c(oncat)* or *s(um)* for merging the output of the two (Bi)LSTMs, with (non-)trained embeddings; T (*True*)/F (*False*) represent inclusion/omission, respectively, of the Dense 32 ReLU layer. *std* represents standard deviation of 5x10 fold cross-validation.

References

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Transactions on Neural Networks*, 5(2):157–166.

Tom Bosc, Elena Cabrio, and Serena Villata. 2016. Tweeties squabbling: Positive and negative results in applying argument mining on social media. In *Computational Models of Argument COMMA*, pages 21–32.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Lucas Carstens and Francesca Toni. 2015. Towards relation based argumentation mining. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 29–34.

Lucas Carstens and Francesca Toni. 2017. Using argumentation to improve classification in natural language problems. *ACM Transactions on Embedded Computing Systems*. Forthcoming.

Claudette Cayrol and Marie-Christine Lagasquie-Schiex. 2005. On the acceptability of arguments in bipolar argumentation frameworks. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty: 8th European Conference*, pages 378–389.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Ivan Habernal and Iryna Gurevych. 2016. What makes a convincing argument? Empirical analysis and detecting attributes of convincingness in web argumentation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1214–1223.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Barbara Konat, John Lawrence, Joonsuk Park, Katarzyna Budzynska, and Chris Reed. 2016. A corpus of argument networks: Using graph properties to analyse divisive issues. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)*.

Yuta Koreeda, Toshihiko Yanase, Kohsuke Yanai, Misa Sato, and Yoshiki Niwa. 2016. Neural attention model for classification of sentences that support promoting/suppressing relationship. In *Proceedings of the Third Workshop on Argument Mining*.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 343–351.

Marco Lippi and Paolo Torroni. 2016. Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology*, 16(2):10.

Pengfei Liu, Xipeng Qiu, Yaqian Zhou, Jifan Chen, and Xuanjing Huang. 2016. Modelling interaction of sentence pair with coupled-lstms. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1703–1712.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *CoRR*, abs/1509.06664.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR*, abs/1602.03609.

Mike Schuster and K. Kuldip Paliwal. 1997. Bidirectional recurrent neural networks. *Transactions on Signal Processing*, 45(11):2673–2681.

Simone Teufel, Jean Carletta, and Marc Moens. 1999. An annotation scheme for discourse-level argumentation in research articles. In *EACL 9th Conference of the European Chapter of the Association for Computational Linguistics*, pages 110–117.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: attention-based convolutional neural network for modeling sentence pairs. *TACL*, 4:259–272.