Stance Detection for the Fake News Challenge with Attention and Conditional Encoding

Stephen Pfohl
Stanford University
spfohl@stanford.edu

Oskar Triebe
Stanford University
oskar2@stanford.edu

Ferdinand Legros
Stanford University
flegros@stanford.edu

Abstract

The in-progress Fake News Challenge is a public challenge tasking competitors to develop a stance detection tool that could ultimately be incorporated into a larger automatic fact-checking pipeline. 49,972 body-headline pairs are labeled with either "Unrelated", "Discusses", "Agrees", or "Disagrees", and it is the goal of the stance detection task to predict these labels. We applied the concepts of neural attention and conditional encoding to long short-term memory networks (LSTM) ultimately achieving a preliminary competition score of 0.808, improving over the competition baseline of 0.795 that relies on several hand-crafted linguistic features. Four models were evaluated: Bag of Words (BOW), basic LSTM, LSTM with attention, conditional encoding LSTM with attention (CEA LSTM). The attention models outperformed the simpler models on all performance metrics on the test set. In particular, the models with neural attention were able to achieve significantly higher F_1 scores predicting the infrequent stances "Agrees" and "Disagrees".

1 Introduction

This work is an in-progress submission to the public Fake News Challenge [1] that will conclude on June 1, 2017. The curators of the challenge succinctly describe the underlying problem of "Fake News" that the challenge addresses: "Fake news, defined by the New York Times as 'a made-up story with an intention to deceive' [2], often for a secondary gain, is arguably one of the most serious challenges facing the news industry today. In a December Pew Research poll, 64% of US adults said that 'made-up news' has caused a 'great deal of confusion' about the facts of current events" [3].

While it would seem natural to develop a system to directly classify new articles and headlines as fake or legitimate, the labeling of such a dataset is controversial and subject to the bias of the labelers. As a result, the focus of the challenge is to instead develop a stance detection tool that is able to automatically reason about the relationship between a new claim or headline and a set of articles. To be clear, stance detection refers to the task of classifying a pair of sources of text as agreeing, disagreeing, neutral, or unrelated. Such a tool may be used as a building block for a broader automated fact checking system that is able to rapidly assess the veracity of a new claim through the assessment of the level of agreement of the claim with the articles of several news organizations that have known levels of reliability and bias. If such an automated system is unrealizable, the stance detection tool may also be used to more aid human fact checkers in quickly assessing the reliability of a claim, as the tool would allow for efficient retrieval of articles that agree, disagree, or are neutral to the claim, allowing for the fact-checker to use their own judgment as to the veracity of the claim.

2 Competition Details

2.1 Description of Data

The administrators of the Fake News Challenge provide a training data set derived from the Emergent [4] dataset. Each of the articles in the Emergent data set is decomposed into the headline and the body text. The final dataset provided by the challenge administrators includes 1648 unique headlines and 1669 article bodies that have been paired to create 49,972 body-headline pairs, where the headline and the body used for training need not come from the same article. Each article-body pair has been labeled with a stance and stance labels include Unrelated, Disagrees, Agrees, and Discusses. It should be noted that the distribution of labels is highly unbalanced within the training set with 73.1% of examples (36529 examples) labeled as "Unrelated", 17.8% of examples (8896) labeled as "Discusses", 7.4% (3698) labeled as "Agrees", and 1.7% (850) labeled as "Disagrees".

2.2 Scoring

To address this issue of class imbalance, the competition will be scored on the basis of a weighted accuracy measure where 0.25 points are awarded correctly classifying a pair as "Unrelated", 0.25 points are awarded for an incorrect classification if the true label is one of "Agrees", "Disagrees", or "Discusses" and one of those labels was predicted, and 1.0 point is given if a correct classification is made the when the true label is "Agrees", "Disagrees", or "Discusses". The final competition score is reported as a percentage of the highest achievable score for a particular data set.

2.3 Official Baseline model

On March 1, 2017, the competition administrators released an official baseline implementation that uses a `GradientBoosting` classifier [5] with hand-crafted linguistic features including token overlap, polarity, and refutation. This model achieves a competition score of 79.53% over 10-fold cross-validation.

3 Relevant Literature

The work [4] that introduces the Emergent data set additionally presents an approach for stance detection where simple features are extracted from the headline and the associated claim are used as input to a regularized logistic regression classifier. This model attains performance that is comparable to the state of the art in stance detection. A related work [6] considers the usage of a conditional encoding scheme with two bidirectional Long Short Term Memory (LSTM) [7] models for stance detection for tweets towards some target. The tweet and the target are processed by separate LSTM models and the first hidden state of the LSTM that processes the target is initialized with the final hidden state of the LSTM that processes the tweet.

A related task that is conceptually similar to stance detection is the task of recognizing textual entailment. In this task, two sentences are processed and the goal is to determine if either the second sentence is a logical consequence of the first sentence, the sentences are contradictory, or they are not related. Recently, Rocktaschel et al. [8] demonstrated that the usage of conditional encoding of LSTM models, in the manner described previously, in conjunction with neural attention mechanisms [9] improves on the state of the art in recognizing textual entailment. Given that the task of stance detection and recognizing textual entailment are conceptually similar, it is a reasonable hypothesis that the usage of attention mechanisms in conjunction with conditional encoding will attain high performance on the stance detection task.

4 Approach and Methods

4.1 Preprocessing

The text of each headline and article body in the dataset was processed with the `word_tokenize` function available from the natural language toolkit (nltk) [10] and each unique token mapped to an integer identifier such that each document is represented as a sequence of positive integers. The

unique tokens in the text were mapped to 50 dimensional GloVe word embedding vectors [11] that were previously trained on a Twitter Corpus containing 2 billion tweets and a vocabulary of 1.2 million unique tokens. Any token in the text of the headlines and article bodies did not map to any token in the Twitter corpus was simply removed from the corresponding sequence. Since the text sequences considered were of various lengths, we zero padded all sequences to the maximum sequence length observed in the corpus. Proper masking techniques were used in each case so that the states produced on the zero-padded input did not factor into the loss calculations. Given that the majority of headline and article bodies are considerably shorter than the maximum length of either, sequences were truncated prior to being used as input to a model see Figure 5. The number of tokens to truncate at was considered to be a hyper-parameter.

4.2 Models

The four models that follow were considered for the stance detection task. The code for all models and experiments was developed in Python 2.7.10 and Tensorflow 0.12.1 [12].

4.2.1 Bag of Words

For a simple baseline, a bag of embedded words model was constructed where the embedding vectors of the headline and the body were averaged separately, concatenated, and used as the input to a feed to a feed-forward neural network with a softmax output layer.

4.2.2 Long Short Term Memory Network (LSTM)

In addition to the bag of words model, we consider an additional baseline in the form of a standard LSTM [7] model that processes a concatenation of tokens in the headline and the article body to produce a classification of the stance. No special token is used to delineate the transition between the header and the body in the concatenated input.

4.2.3 Attention

The attention mechanism presented in Rocktaschel et al. [8] was implemented in Tensorflow. A summary of the attention implementation is provided below.

Let all vectors be column vectors. Define the attention window the be the first L output states produced by the LSTM. Let k be the dimension of the hidden state, and N be the total sequence length. Let $Y \in \mathbb{R}^{k \times L} = [h_1, \ldots, h_N]$ be a matrix of the output states of the LSTM in the attention window. Let $e_L \in \mathbb{R}^L$ be a vector of 1s. Let $W^y, W^h, W^p, W^h \in \mathbb{R}^{k \times k}$ and $w \in \mathbb{R}^k$ be trainable matrices. A final state h^* is produced as follows:

$$M = \tanh(W^y Y + W^h h_N e_L^T) \tag{1}$$

$$\alpha = \mathrm{softmax}(w^T M) \tag{2}$$

$$r = Y \alpha^T \tag{3}$$

$$h^* = \tanh(W^p r + W^x h_N) \tag{4}$$

h^* is then used as the new final state of the network that may be passed to any downstream task.

4.2.4 Attention Model 1 - Modified Simple LSTM

As a first attempt at utilizing attention for the stance detection task, the previously described baseline LSTM model was augmented with the attention mechanism over the first L output states of the top layer of the LSTM.

4.2.5 Attention Model 2 - Conditional Encoding LSTM

Finally, an LSTM framework with conditional encoding [6] was considered and the attention mechanism applied. We will refer to this model as CEA (Conditional Encoding with Attention). In this setting, the headline and the body are processed separately in two LSTM models where the final

hidden state of the model used to process the headline is used as the first hidden state of the model used to process the bodies. The attention mechanism then operates over the the first L output states of the headline LSTM in conjunction with the final hidden state LSTM that processes the article bodies.

4.3 Evaluation

Prior to training, the dataset was split randomly into a training, development, and testing set. The split was defined such that 60% of the examples were in the training set and 20% of the examples were in each of the development and testing sets. All experiments were performed by minimizing the mean cross-entropy loss on the training set and monitoring the performance on the development set in terms of the class-level F_1 and the competition scoring function, as previously described. The performance on the development set was analyzed and the final parameter set selected for each class of model. For final evaluation at the selected parameter sets, the training and development sets were combined and the models trained once more with the selected parameters for subsequent evaluation on the testing set. No further parameter tuning was performed to mitigate sources of bias in the evaluation of the test performance.

4.4 Parameter Tuning Experiments

Given the high computational cost of a full grid search over all possible parameter settings, only a relatively small subset of permutations was considered for a rough sensitivity analysis. To gauge which parameters would likely have the largest effect on the model performance, initial tests were run on a smaller subset of the data (results not shown) and it was determined that the number of tokens in the truncation window and the number of hidden layers were the parameters with the largest effect. Thus, those two parameters were the ones considered for further tuning.

For all models, we fixed the number of hidden layers at 100 units, the dropout probability of 0.2 (LSTM models only). Additionally, the Adam [13] optimizer was used for all models and the word embedding vectors were considered to be trainable. The training process was run for 40 epochs and the batch size was set at 128 examples.

For experimentation, we additionally define a default parameter set for each model and then vary a single tuning parameter while all other parameters take on their default values. The development set performance is assessed for each experiment, as previously discussed. The default and tunable values for the parameters for each model are as follows:

- Bag of Words. Default truncation length 150 tokens of the article body, 2 layers. Learning rate of 0.005. Tuning truncation length at 75, 150, 300, and 600 tokens and 1, 2, and 4 layers.
- Basic LSTM. Default truncation 150 tokens in the concatenated representation and 2 layers. Learning rate of 0.005. Tuning truncation length at 75, 150, and 300 tokens and 1, 2, and 4 layers.
- LSTM with Attention. Default truncation 150 tokens in the concatenated representation and 1 layer. Learning rate of 0.001. Attention window of the first 15 tokens. Tuning truncation length at 75, 150, and 200 tokens and 1, 2, and 4 layers.
- LSTM with Conditional Encoding and Attention. Default truncation 150 tokens in the article body and 1 layer. Learning rate of 0.001. Attention window of the first 15 tokens. Tuning truncation length at 75, 150, and 300 tokens and 1, 2, and 4 layers.

5 Results

5.1 Length truncation and benefits of attention

Competition scores analysis

First, we observe in Figure 1 that models achieve a higher competition score for a truncation length of 75 tokens, with the exception of CEA, which attains comparable performance at 300 tokens.

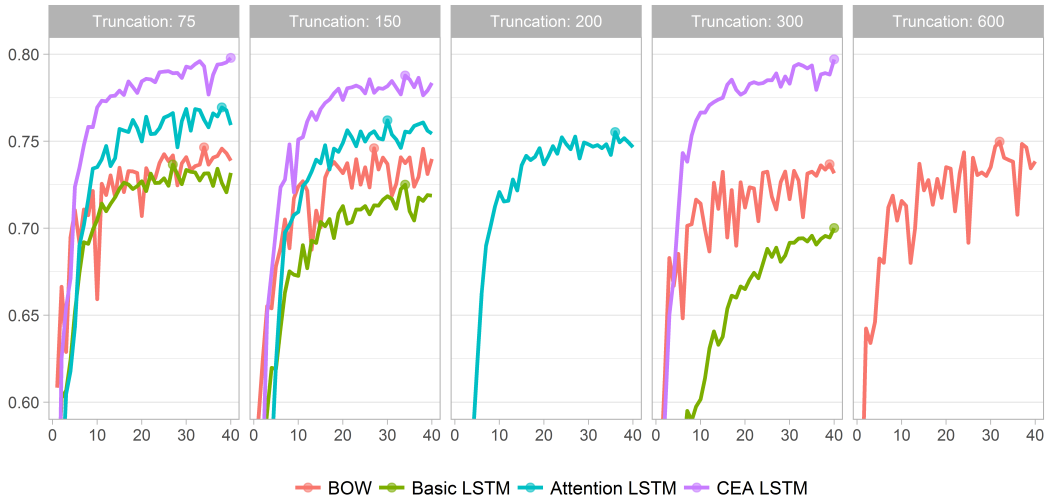

Figure 1: Competition score with varying truncation levels

The reason for this may be that the models have difficulty learning from long sequences. The performance of CEA on longer sequences demonstrates the potential benefits of the combination of conditional encoding and attention. Our interpretation of those results is that using longer input sequences adds noise to the input and only the CEA can extract meaningful information from this noisy input to improve the score. However, we observe a significant performance improvement from 150 tokens to 300 tokens, which indicates that the conclusion of an article may be more meaningful than the middle of it. The median article length is 315 tokens, so truncating at 300 tokens allows the model to process roughly half articles fully. Truncation at 150 tokens adds noise with respect to the truncation at 75 tokens without adding the benefits of processing articles' conclusions.

Finally, we observe two interesting features about the BoW model. First, it performs better than the simple LSTM. Second, its performance at truncation 600 is close to its best performance, it thus seems to make sense of the additional information provided by long sequences in spite of the added noise. This provides compelling evidence for attempting to use the CEA model for longer sequences in future iterations.

There are several avenues by which these analyses may be enhanced. First, we observe that some models, CEA in particular, would potentially benefit from training beyond 40 epochs as the development score is still increasing steadily at epoch 40 despite some noise. Second, the relatively poor performance of simple attention compared to CEA may be due to the fact that only a single hidden layer was used in these experiments. Contrary to CEA, the simple attention model is only composed of a single LSTM, so with a single layer the number of parameters trained is limited, possibly causing bias in the model.

F1 scores analysis

The F1 scores showed in Figure 2 support the previous analyses. The performance gap between CEA and the other models is striking as we inspect each class separately. The overall decrease in performance with increasing truncation level is confirmed for the LSTM and the simple attention model. The decrease in performance of CEA for truncation 150 is particularly visible on the 'Disagrees' class. Eventually, it seems that the decent performance of the BoW model is due to the good performance on the majority "Unrelated" class.

5.2 Hidden layers study

Making the number of hidden layers vary in $\{1, 2, 4\}$ showed that 2 was a good compromise given the limited amount of data we had.

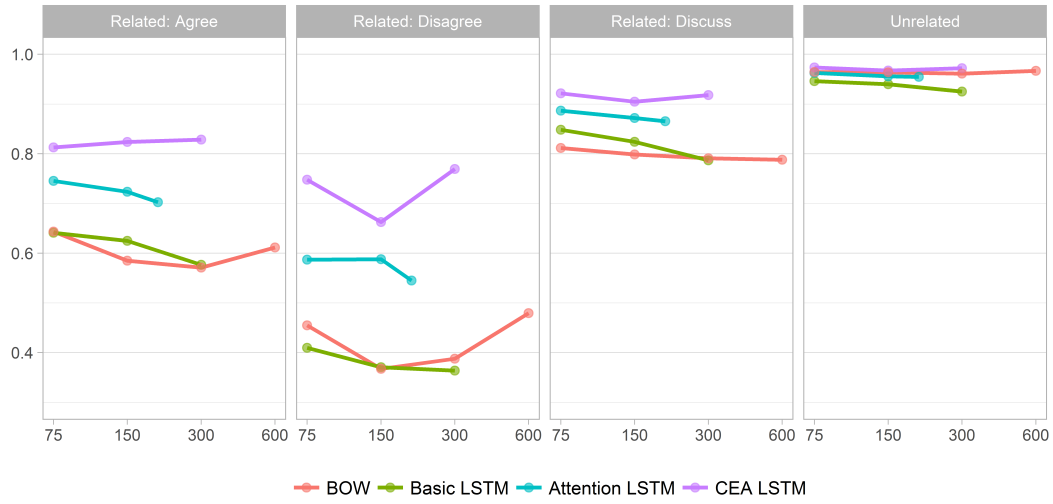

Figure 2: F1 scores with varying truncation levels

Competition scores analysis

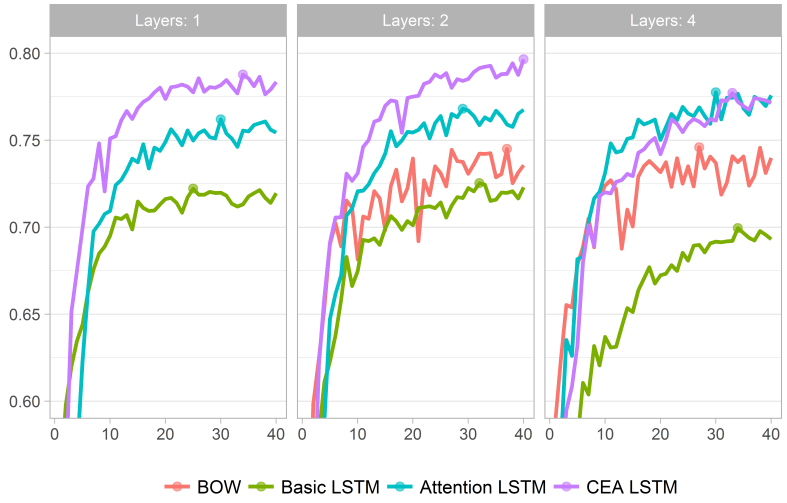

Figure 3: Competition scores with varying number of hidden layers

We see in Figure 3 that when hidden layers are set to 4, the learning curves of the simple attention model, CEA and LSTM are still quite steep when epoch 40 is reached. So those models may benefit from further training. However, the CEA with 2 layers outperforms them by a significant 2% margin, so this further training might not be relevant.

The learning curve of CEA with 2 hidden layers also looks like the model could be trained further. In particular, the potential gain for the CEA is clear in appendix Figure 8.

F1 scores analysis

The F1 scores displayed in Figure 4 show most models do not benefit from 4 layers. However as we discussed, that may be the consequence of a too low number of epochs. We observe a F1 boost for the 'Disagrees' class when CEA number of layers is set to 2, making the F1 of the minority class close to 0.8. Notably that the BoW model with a single layer gives a very poor performance, perhaps

because the number of parameters is too low with respect to all other models, yet the BoW model is comparable when additional layers are added.

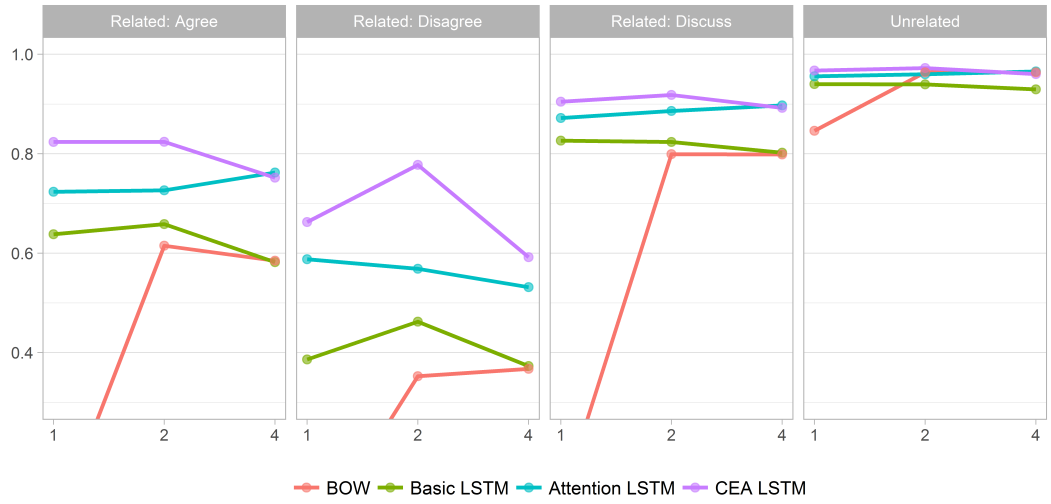

Figure 4: F1 scores with varying number of hidden layers

Sensitivity analysis conclusion

- The CEA model could benefit from extended training and truncation to longer sequences. Setting 2 hidden layers seems the best choice but should be compared to longer trained 4-layer model. Performance improvement is striking on minority classes.
- The simple attention model suffers from relatively high bias compared to CEA as it cannot benefit from the advantages of having a separate LSTM that processes the headlines and body. Further training with 4 layers may diminish the gap with CEA.
- The basic LSTM model yields a relatively poor performance on this task, which is probably due to the fact that processed sequences are too long.
- The BoW model yields a comparable competition score performance to other models when additional layers and considered, although it still performs poorly on the rare class.

5.3 Test performance overview

Following the parameter tuning process, a final set of parameters was selected and used for prediction on the final hold out set. The selected set of parameters can be found in the supplementary material in table 2 and the final performance results in table 1. In summary, a truncation length of 75 tokens was used in all cases except for the bag of words model, in which case 600 tokens was used. Additionally, a 2 layer network was selected in each case.

In general, the testing results reflect the results seen during the tuning process. In particular, each model performs comparably on the "Unrelated" class and the two models that incorporate attention have much greater class level F_1 for "Agrees" and "Disagrees". It shows the attention and conditional encoding mechanisms allow the model to better classify the rare labels. Let us note that the simple LSTM performs higher on this test set (0.789) than in the tuning process, which may indicate that the model improves with additional data. Finally, both the attention LSTM and the conditional encoding with attention LSTM outperform the competition baseline score of 0.795 by achieving a score of 0.804 and 0.808, respectively.

Table 1: Final results of the four models with selected hyperparameters are shown based on their evaluation on the held out test set (20% of data). The performance of the models increases with their complexity, with the Conditional Encoding LSTM with Attention (CEA LSTM) performing best on all metrics but the train loss.

Model	Competition Score	Train Loss	Stance F1 Score			
			Agrees	Disagrees	Discuss	Unrelated
BOW	0.752	0.156	0.635	0.478	0.819	0.968
Basic LSTM	0.789	0.019	0.805	0.581	0.910	0.973
Attention LSTM	0.804	0.019	0.845	0.746	0.925	0.974
CEA LSTM	0.808	0.026	0.866	0.793	0.936	0.978

6 Discussion

6.1 Limitations

We did not address the issue of *bleed-over* of the headlines and bodies between the training, development, and testing sets. Given that there are many more headline-body pairs than there are unique headlines and bodies in the dataset, there are likely many instances of bodies and headlines that appear in both the training set and the development and testing sets that were used for evaluation. As a result, it is possible that the results presented here are optimistic, particularly since the unreleased testing set that will be used for the final evaluation will contain an entirely new set of headlines and bodies. We will analyze this issue in further work.

6.2 Future Work

Parameters exploration. As pointed out in the sensitivity analyses, we could retrain our models with more epochs, 50 or 100 epochs. Longer sequence threshold could be set for the CEA model to understand the full benefits of attention. Different attention windows could be tested. Eventually, we could choose an embedding dimension greater than 50, and up to 300.
Data processing. To counter class imbalance, we could apply downsampling to our data. A custom loss function weighting heavily losses on minority class labels may be another solution. On the side of text processing, we could introduce a token for words not in the embedding dictionary.
Model extensions. We could explore bidirectional LSTMs, especially since end of articles may contain relevant conclusions. Implementing the word-level attention described in [8] is another possibility.

6.3 Conclusion

In this project, we managed to beat the 79.5% competition score baseline with a Conditional Encoding LSTM with Attention which yields a 80.8% score. Our work demonstrates the efficiency of attention techniques in extracting from a long sequence (the article bodies) information relevant to a small query (headline). Further work on modeling, processing and parameter tuning could enable us widening the gap with the competition baseline.

Acknowledgments

We would like to acknowledge and thank Danqi Chen for her mentorship on this project.

References

[1] Dean Pomerlau and Delip Rao. Post-facto Fake News Challenge.

[2] Sabrina Tavernise. As Fake News Spreads Lies, More Readers Shrug at the Truth - The New York Times, 2016.

[3] Michael Barthel, Amy Mitchell, and Jesse Holcomb. Many Americans Believe Fake News Is Sowing Confusion — Pew Research Center, 2016.

[4] William Ferreira and Andreas Vlachos. Emergent: a novel data-set for stance classification. In *HLT-NAACL*, 2016.

[5] scikit-learn developers. Choosing the right estimator.

[6] Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. Stance Detection with Bidirectional Conditional Encoding. *Empirical Methods in Natural Language Processing*, (2010):876–885, 2016.

[7] Sepp Hochreiter and Jrgen Schmidhuber. Long Short-term Memory. *Neural Comput.*, 9(9):1735–1780, 11 1997.

[8] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tom Kočiský, and Phil Blunsom. Reasoning about Entailment with Neural Attention. 9 2015.

[9] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective Approaches to Attention-based Neural Machine Translation. pages 1412–1421, 2015.

[10] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[11] Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global Vectors for Word Representation. 2014.

[12] Martn Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. 3 2016.

[13] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

7 Contribution of Team Members

- Stephen Pfohl Implemented the code for the LSTM models with attention and conditional encoding. Contributed towards code modularization. Contributed towards paper text and poster diagrams.
- Oskar Triebe Developed code for bag of words model and designed the object oriented models of project code, processed experiment results and made figures. Designed poster.
- Ferdinand Legros Developed for text processing, tokenization, results logging, and assisted with development of LSTM models. Contributed towards paper text and poster design.

8 Appendix

8.1 Data Characteristics

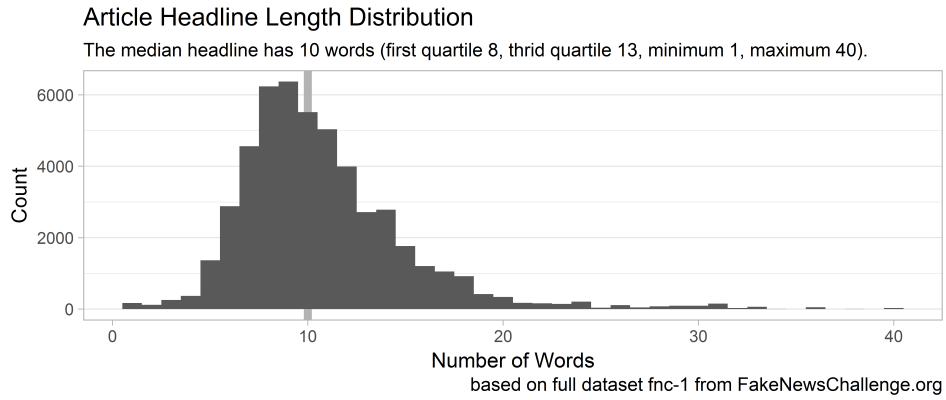

Figure 5: Distribution of article headline lengths

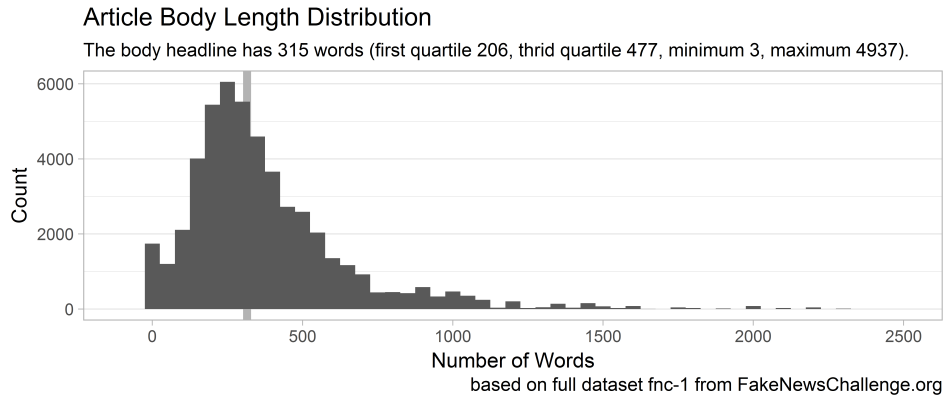

Figure 6: Distribution of article body lengths

8.2 Hyperparameter Selection for Models

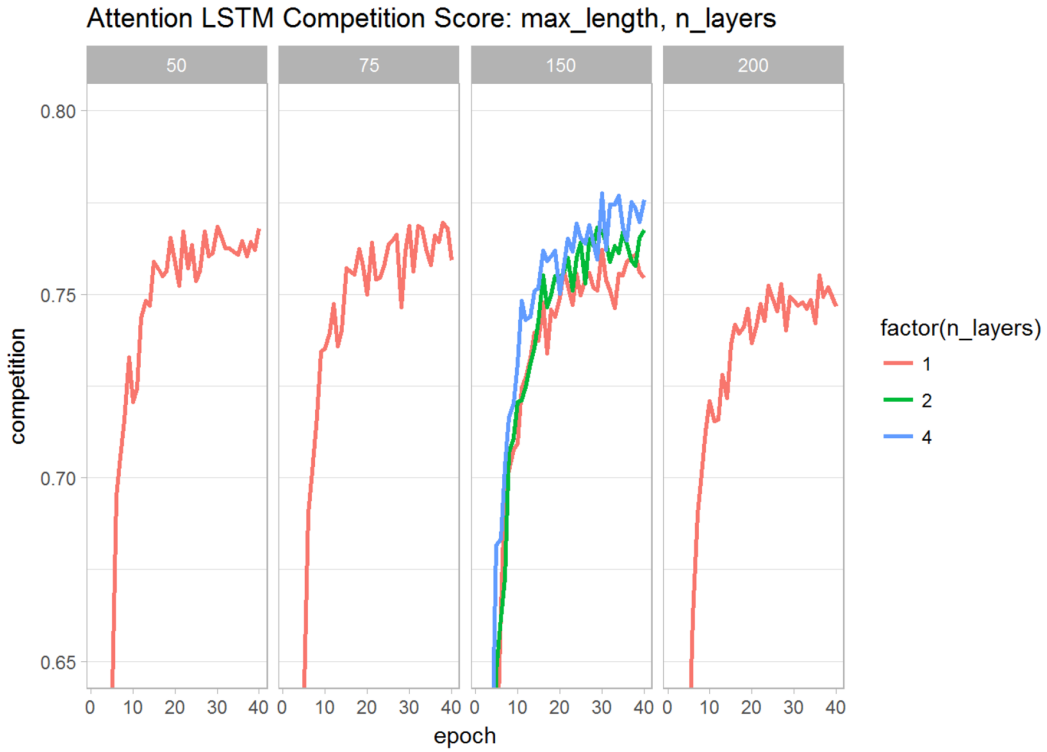

Figure 7: Competition score of simple attention model with varying number of hidden layers

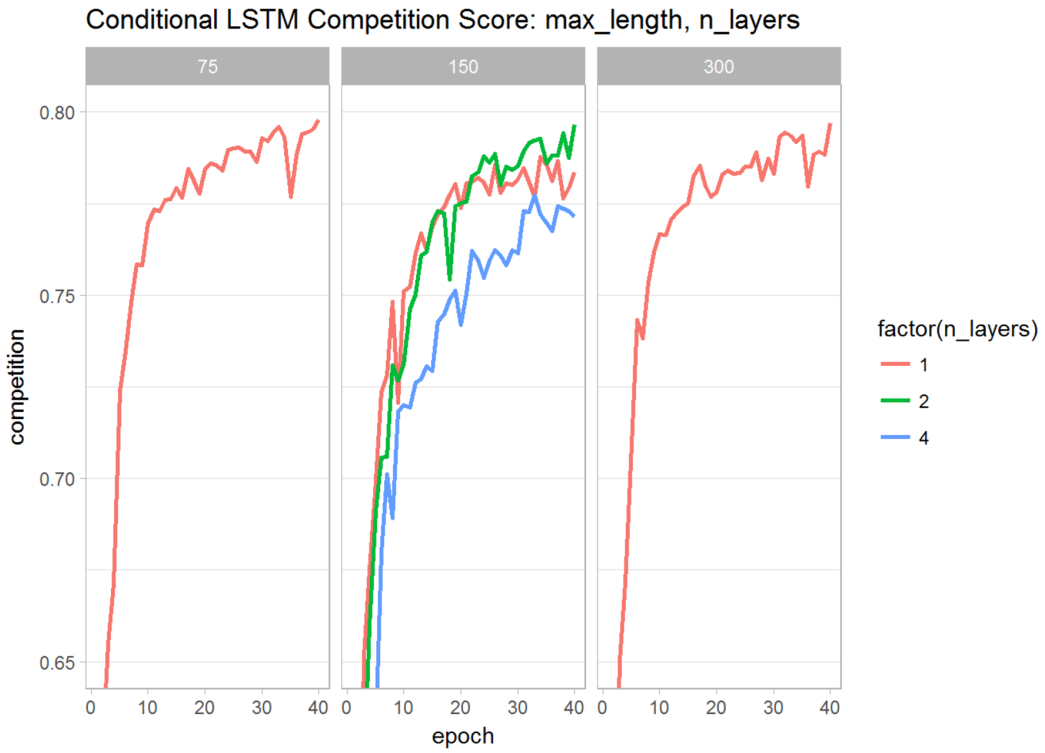

Figure 8: CEA Competition scores with varying number of hidden layers

8.3 Final Model Performance

Table 2: Hyperparameters of the final models are shown as used in for the test set evaluation.

Hyperparameter	BOW	Basic LSTM	Attention LSTM	CEA LSTM
max_length	NA	75	75	NA
lr	0.005	0.001	0.001	0.001
hidden_size	100	100	100	100
n_classes	4	4	4	4
attention_length	15	15	15	15
b_max_len	600	NA	NA	75
batch_size	128	128	128	128
downsample	FALSE	FALSE	FALSE	FALSE
dropout	0.8	0.8	0.8	0.8
embed_size	50	50	50	50
h_max_len	42	NA	NA	42
hidden_next	0.6	0.6	0.6	0.6
n_epochs	40	40	40	40
n_layers	1	2	2	2
num_samples	39977	39977	39977	39977
trainable_embeddings	Variable	Variable	Variable	Variable
vocab_size	1193515	1193515	1193515	1193515
epoch	40	40	40	40

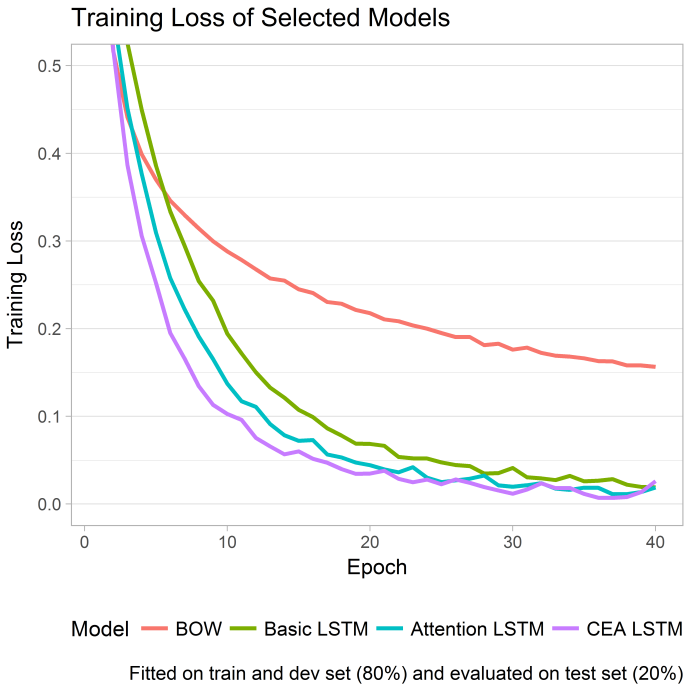

Figure 9: Training Loss rapidly decreases for most models. All models but BOW achieve close to 0 traingin loss.