Towards Automatic Identification of Fake News: Headline-Article Stance Detection with LSTM Attention Models

Sahil Chopra
Department of Computer Science
Stanford University
schopra8@stanford.edu

Saachi Jain
Department of Computer Science
Stanford University
saachi@stanford.edu

John Merriman Sholar
Department of Computer Science
Stanford University
jmsholar@stanford.edu

Abstract

As participants in Fake News Challenge 1 (FNC-1), we approach the problem of fake news via stance detection. Given an article as "ground truth", we attempt to classify whether a headline discusses, agrees, disagrees, or is unrelated to a given article. In this paper, we first leverage an SVM trained on TF-IDF cosine similarity features to discern whether a headline-article pairing is *related* or *unrelated*. If we classify the pairing as the former, we then employ various neural network architectures built on top of Long-Short-Term-Memory Models (LSTMs) to label the pairing as *agree*, *disagree*, or *discuss*. Ultimately, our best performing neural network architecture proved to be a pair of Bidirectional Conditionally Encoded LSTMs with Bidirectional Global Attention. Using our linear SVM for the *unrelated/related* subproblem and our best neural network for the *agree/disagree/discuss* subproblem, we scored .8658 according to the FNC-1's performance metric.

1 Overview

1.1 Motivation

In the wake of the 2016 Presidential Election, fake news has been a subject of increased discussion and debate. Accurate detection of fake news will allow for the elimination of deliberately deceptive news content, which will in turn promote a better-informed general public. As a result, there has been newfound interest in developing autonomous systems to identify fake news.

1.2 Task Definition and Dataset

In February, the Fake News Challenge 1 (FNC-1) was launched by a non-profit organization with the goal of developing tools to help fact checkers tag fake news. FNC-1 specifically focuses on stance detection. Given an article that acts as "ground truth", we are given a number of headlines that must be classified as *unrelated*, *agree*, *disagree*, or *discuss* in relation to the article. The organizers of FNC-1 hope that the winning solutions to this stance detection problem will be leveraged as filters to limit the number of articles the fact checkers will have to examine by hand.

Stance	Description	% of Provided Data
agree	article agrees with headline	7.36
disagree	article disagrees with headline	1.68
discuss	article discusses same topic as headline (no position)	17.83
unrelated	article unrelated to headline	73.13

Table 1: Distribution of Stances Among Headline-Article Pairs

In this paper, we propose a two-part solution to FNC-1. First, we suggest a linear classifier to classify headline-article pairs as *related* or *unrelated*. Second, we suggest several neural network architectures built upon Recurrent Neural Network Models (RNNs) to classify *related* pairings as *agree*, *disagree*, or *discuss*.

2 Background

As we developed our approach to FNC-1, we first explored existing research as it pertains to related NLP problems in entailment as well as stance detection. First, we examined papers regarding the Stanford Natural Language (SNLI) Dataset, which has been become popular in recent years when developing models to classify entailment and contradiction amongst hypothesis-premise pairs. From the original SNLI paper (Bowman et al. 2015) we derived two of our baseline models - a Bag of Words (BOW) Multilayer Perceptron (MLP) and a Long-Short-Term-Memory (LSTM) that receives concatenated hypothesis-premise pairs as inputs. Additionally, we drew heavily upon Tim Rocktaschel's *Reasoning About Entailment with Neural Attention* (Rockstachel et al. 2016). In the paper, Rocktaschel proposes an architecture of conditionally encoded LSTMs upon which attention is applied in order to classify entailment on the SNLI Dataset. We implemented and expanded upon these models in our proposed solution for FNC-1.

Secondly, we examined papers regarding stance detection itself. In our FNC-1 models, we leveraged ideas proposed in *Stance Detection with Bidirectional Conditional Encoding* (Augenstein and Rocktaschel 2016), where the authors used Bidirectional Recurrent Neural Networks (BiRNNs) to conditionally encode target phrases and tweets for the SemEval 2016 Stance Detection Challenge. Lastly, we implemented the Bilateral Multi-Perspective Matching Model (BiMpM) model (Wang et al. 2017) and applied it to FNC-1. As discussed later in our paper, the model takes word embeddings as inputs to a Bidirectional Siamese LSTM, applies four variants of attention on the output of the BiLSTM, feeds these attention-induced outputs through two separate BiLSTMs, concatenates the final hidden states, and uses a 2-Layer MLP for classification.

3 FNC-1 Dataset & Scoring Metrics

The FNC-1 Dataset consists of 1648 distinct headlines, 1683 distinct articles, and 49972 distinct headline-article pairings. The headlines had various lengths ranging from 10 to 220 words, while articles had lengths ranging from 25 to 5000 words (See Appendix A.1). Additionally, The FNC-1 Dataset was very heavily biased towards *unrelated* headline-article pairs (See Table 1). Recognizing this data bias and the simpler nature of the *related*/*unrelated* classification problems, the organizers of FNC-1 use the following weighted accuracy score as their performance metric. Along with more traditional F1 scores, we shall also use this metric to measure our performance on the task.

$$S_1 = Acc_{Related,Unrelated} \tag{1}$$

$$S_2 = Acc_{Agree,Disagree,Discuss} \tag{2}$$

$$S_{FNC} = .25S_1 + .75S_2 \tag{3}$$

FNC-1 will release an official test-set for final submissions in June. In the mean time they have released a 80-20 split on training articles that they themselves used when establishing a linear baseline. We used this 80-20 split as our training-test split, and then randomly sampled 20% of the articles in the train split to be used as our development set. This guaranteed that no articles that appeared in the train set, appeared in the development or test sets - and vice-versa.

4 Baseline Models

We implemented several baseline models to benchmark performance on the four-class classification problem. Please see Table 2 for baseline results.

Baseline Model	S_{FNC}
Lexicalized Classifier	**.7860**
BOW MLP	.7787
LSTM with Concatenated Input	.4005

Table 2: Baseline Results

4.1 Lexicalized Linear Classifier

Our first baseline model was a linear classifier that utilized the features described in the original SNLI paper (Bowman et al. 2015). This lexicalized classifier incorporates 3 feature types, calculated for each (headline, article) pairing, and uses an SVM classifier with a radial basis function (RBF) kernel. The three features utilized were: 1) Cosine distance between the TFIDF vectors of the headline and article, 2) Max BLEU Score between windows of the headline and the article, 3) Jaccard distance. The linear classifier performed well at distinguishing *related* versus *unrelated* labels but struggled over distinguishing the *related* subtypes (See Appendix A.2). Of the three baselines this model performed the best.

4.2 Bag of Words (BOW) Multi Layer Perceptron (MLP)

Our second baseline model was a BOW MLP that utilized 300 Dimensional GloVe Embeddings to represent the headline and article in vector space (See Appendix A.2).

4.3 LSTM with Concatenated Input

We performed softmax classification on the final hidden state of an LSTM, which received a concatenated headline-article as its input. We truncated this combined input at 1000 words, and used 300 Dimensional GloVe Embeddings to represent the headline and article in vector space (See Appendix A.2). Of the three baselines, this model performed the worst - simply choosing the majority class *unrelated* for nearly all inputs.

5 Methods

5.1 Split into Two Classification Problems

The above baselines attempted to immediately classify headline-article pairs into their final labels as *agree*, *disagree*, *discuss*, and *unrelated*. However, because unrelated samples comprised of over 73% of the data-set, these classifiers struggled to predict classes beyond the majority set, thus failing to capture the semantic differences between agree, disagree, and discuss. To address this issue, we split the four class problems into two more specific subproblems. In the first, we simply try to detect whether a headline and article are related, combining the agree, disagree, and discuss samples into a aggregate class *related*. In the second problem, given pairs that are already classified as *related*, we seek to label the pairs as *agree*, *disagree*, or *discuss*. We trained the two models separately on the train data, where the second problem is only trained on *related* samples from the training set. To produce the final predictions for the test set, we first feed the data to subproblem 1's model to filter out unrelated samples. We then send the remaining samples into the second model for further classification.

5.2 Subproblem 1: Related vs Unrelated via Linear Classifier

Subproblem 1 reduces down to a simple text similarity classification problem. For each article, headline pair we extracted features such as the cosine distance between TF-IDF vectors, max BLEU score, cross-grams, and Jaccard Distance (See Appendix A.3 for details). After some feature analysis, we chose the TF-IDF cosine distance because of its high correlation with related data. We then used a SVM classifier with a radial basis function (RBF) kernel.

5.3 Subproblem 2: Data Pre-Processing

As seen in Appendix A.1, the articles in the data set had a long tail of length distributions, with some articles totaling up to over 1300 words. However, because our models for subproblem 2 were built upon LSTMs, having over 1000 timesteps was both slow and counterproductive. Examining the length distributions described above, we truncated the articles to 800 tokens. In order to transform the inputs into vector space, we used 300 Dimensional GloVe vectors taken from the 6B token set

of Wikipedia and Common Crawl. We further created a randomly initialized UNK vector of zeros, for words that were not found in the GloVe set.

5.4 LSTM Attention Architectures

5.4.1 Conditionally Encoded (CE) LSTMs

LSTMs are gated RNNs that can store and forget memory from previous iterations. The model is centered around three types of gates: input, forget, and output. The equations for the LSTM are listed in Appendix A.4. Concatenation of the headline and article in the Basic LSTM baseline proved largely ineffective. Therefore, we moved to a conditionally encoded model as described in Rocktaschel et. al, 2016. The model involves two separate LSTMs, one for the headline and one for the article. The headline is fed through the first LSTM to extract the final hidden vector h_n. This hidden state is used to initialize the LSTM of the article, thus "conditioning" the article LSTM on the headline. For all LSTMs listed from this point onwards, we used Tensorflow's LSTMBlockCell implementation. After running the CE LSTMs, we passed the final hidden state of the article into a two layer MultiLayer Perceptron (MLP) with ReLU activation functions to project onto the three-class-stance space. Finally, we used softmax with cross entropy to evaluate loss.

5.4.2 Adding Global Attention

Building off of the conditional LSTM architecture above, we then added global attention of the headline onto the article as described in Rocktaschel et. al, 2016. Below, is the attention vector formulation for a single example (this was then extended for *batch size* samples at once). Let d be the size of the hidden layer, M the number of headline time steps, and N the number of article time steps. Let $Y \in \mathcal{R}^{M \times d}$ be the matrix of hidden vectors taken from the headline LSTM, while $h_N \in \mathcal{R}^{1 \times d}$ is the last hidden vector of the article LSTM. Moreover, let $e_M \in \mathcal{R}^{1 \times M}$ be a vector of M 1s; $v \otimes e_M$ involves replicating v M times. $W_y, W_h, W_x, W_p \in \mathcal{R}^{d \times d}$ and $w \in \mathcal{R}^{d \times 1}$ are trainable weight matrices.

$$M = \tanh(YW_y + h_N W_h \otimes e_M) \tag{4}$$
$$\alpha = \text{softmax}(Mw) \tag{5}$$
$$r = \alpha^T Y \tag{6}$$
$$h^* = \tanh(rW_p + h_n W_x) \tag{7}$$

The attended vector h^* is then passed into the 2 layer MLP for classification.

5.4.3 Adding Word-by-Word Attention

We also implemented Word-by-Word Attention as described by Rocktaschel et. al, 2016. Rather than only attending on the last hidden vector of the article, Word-by-Word Attention iterates through N time steps by attending on each hidden vector of the article, using the attention representation of the timesteps before. The Word-by-Word Attention formulation is listed below. To the above definitions from global attention we add the weight matrices $W_r, W_t \in \mathcal{R}^{d \times d}$. Furthermore, let h_t be the t'th hidden vector of the article LSTM.

$$M_t = \tanh(YW_y + (h_N W_h + r_{t-1} W^r) \otimes e_M) \tag{8}$$
$$\alpha_t = \text{softmax}(M_t w) \tag{9}$$
$$r_t = \alpha_t^T Y \tag{10}$$
$$h^* = \tanh(r_N W_p + h_n W_x) \tag{11}$$

See Figure 1 for a depiction of the Conditional LSTM with the different forms of attention.

5.4.4 Bidirectional Global Attention

Building upon the Conditional LSTM with Global Attention of headlines onto articles, we added an additional layer of Global Attention that attended the article over the headline. The resulting two attention vectors were concatenated together and then fed into the MLP for classification.

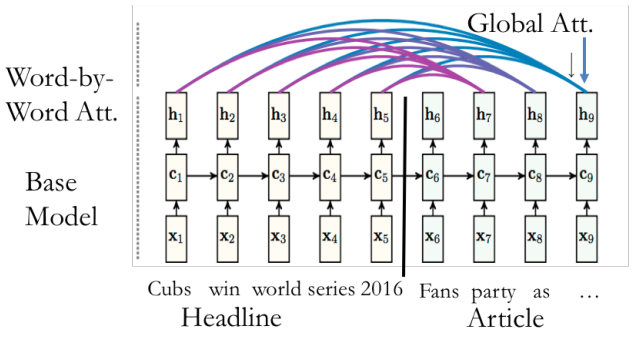

Figure 1: Conditional LSTM with depictions of Global and Word-by-Word Attention. Diagram modified from Rocktaschel et. al, 2016

5.4.5 Bidirectional Conditional LSTM with Bidirectional Global Attention

As the culminating model in this series of LSTM-based architectures, we implemented the Conditionally Encoded LSTM with Bidirectional Global Attention using Bidirectional RNNs (thus reading the text both forward and backward). This framework thus results in 4 attention vectors (attention in both directions each for the two directions of text encoding) which are concatenated together before being fed into the MLP layer. We additionally ran a separate version of the model that used two 5-layer-deep stacked Bidirectional LSTMs for the conditional encoding.

5.5 Bilateral Matching with Multiple Perspectives

Finally, we implemented a variation of the Bilateral Multi-Perspective Matching model described by Wang et. al, 2017 (Figure 2). The model proceeds in several layers:

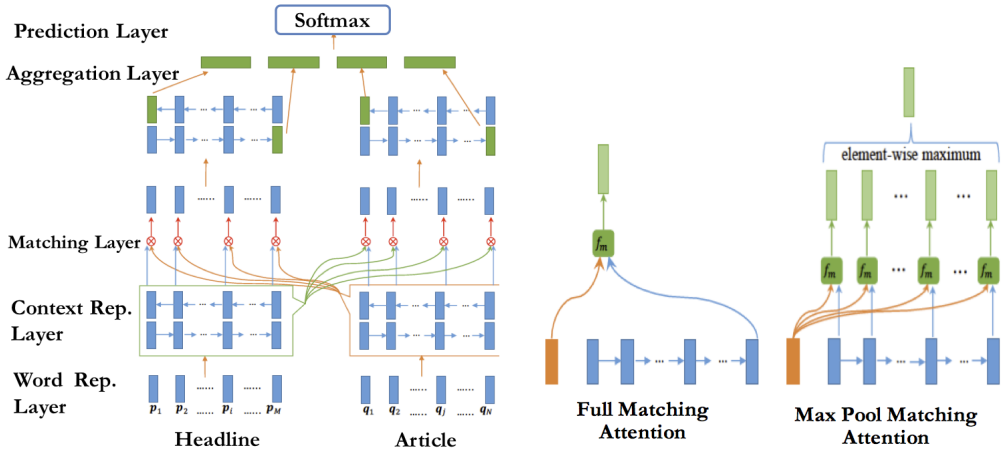

Figure 2: Bilateral Multi-Perspective Matching model (modified from Wang et. al)

(I) Word Representation: Like in the above LSTM models, we transform the inputs into vector space utilizing 300D GloVe embeddings.

(II) Context Layer: The headline and article are placed in a Siamese Net of bidirectional LSTMs. Unlike the above models, the article LSTM is not conditioned on the headline LSTM; instead, the LSTMs share weights. The outputs of this layer are two hidden vectors (one representing forward text encoding, one backward) per time step for both the article and headline.

(III) Attention Layer: The model described by Wang et. al describes four types of attention with perspectives: full matching, max pooling matching, attentive matching, and max-attentive matching. Each attention layer maps a hidden layer onto the perspective space. We implemented all four of these models; however, due to memory limitations, could only get results when using the full and max pooling matching models. These attention models are described in more detail in the next

section. Attention is applied in both directions (headline onto article and article onto headline) and for both encoding directions (forward to backward and backward to forward).

Unlike the attention models described by Rocktaschel et. al, which result in a singular attention weighted hidden vector per direction of attention, these layers of attention are applied per time step. More clearly, for attention of A onto B, each time step of A is matched against the entirety of the hidden states of B. Thus, the attention layer results in 4 perspective vectors per time step for each of article and headline (full matching/max pool for both forward and backward text encoding). These four vectors are concatenated together, resulting in an attention vector per time step of both articles and headlines.

(IV) Aggregation Layer: The attention vectors are placed into a set of two independent bidirectional LSTMs (one for the headline's attention vectors and one for the article's). Unlike in the context layer, these LSTMs do not form a Siamese Net, i.e. they do not share weights. From here the last hidden vector of each LSTM in each text encoding direction is extracted (resulting in four total vectors). These four are concatenated together and passed to the next layer.

(V) Class Projection: The concatenated input from the Aggregation Layer is then placed through a 2 layer MLP to project onto the stance-class-space.

(VI) Loss: Loss is again performed using softmax with cross entropy from the output of the class projection layer.

5.5.1 Attention Layers for Bilateral Multi-Perspective Matching Model

As mentioned above, the attention layers for the Bilateral Multi-Perspective Matching Model map hidden vectors into the perspective space. In short, if the model is applying attention $A \rightarrow B$, then we seek to find a perspective representation for each time step of A based on the entirety of B.

Suppose there are p perspectives. Firstly, we define a scoring function f_m to compare two d dimensional vectors u, v given a weight matrix $W \in \mathcal{R}^{p \times d}$ as follows:

$$m = f_m(u, v, W) \ni m_k = cosine\_sim(W_k \circ u, W_k \circ v) \tag{12}$$

The model as described in Wang et. al 2017 details four attention layers: full-matching, max-pooling matching, attentive matching, and max-attentive matching. Due to memory constraints, we only used the first two layers and will discuss them here; however, implementations of the latter two layers can be found in the source code. At this point attention is performed in one logical and text encoding direction: given a hidden vector h_i in the forward direction at the ith timestep of A, we wish to find the corresponding attentive representation m_i using the entirety of B. Computation of attention in the opposite logical and encoding directions proceed similarly.

Full-Matching This attention layer is closest to the global attention layer of the previous model. Given h_i, we return the score between h_i and the last hidden vector of B with respect to a weight parameter.

$$m_i^{\text{full}} = f_m(h_i, h_N, W) \tag{13}$$

Maxpooling-Matching Here, we take the score of h_i with respect to each hidden vector of B. We then take the element wise maximum for each dimension out of each of the scores computed.

$$m_{i_k}^{\max} = \max_{j \in 1 \ldots N} f_m(h_i, h_N, W)_k \tag{14}$$

6 Results

Our SVM with TF-IDF cosine similarity features performed very well on subproblem 1, receiving an F1-Score of .9712 (See Table 3 and Figure 3). For subproblem 2, we began with a CE LSTM that received an F1-Score of .730 and S_2, accuracy on sub problem 2, of .7859. From there we added global attention and saw an increase in performance. Building upon this model, we added bidirectional global attention and bidirectional conditional encoding to arrive at our best performing models (See Table 4). We then performed hyper parameter tuning on our Bidirectional CE LSTM with Bidirectional Global Attention (BiCE LSTM BiGA) (See Appendix A.6 for details on hyper

Model	F1 Score
SVM with TF-IDF Cosine Similarity	**.9712**

Table 3: Subproblem 1 Results

Model	F1 Score	S_2
Conditionally Encoded (CE) LSTM	.730	.7859
CE LSTM with Headline-to-Article Global Attention	.753	.8144
CE LSTM with Headline-to-Article Word-by-Word Attention	.768	.8263
CE LSTM with Bidirectional Global Attention	**.777**	.8324
Bidirectional CE LSTM with Bidirectional Global Attention (BiCE LSTM BiGA)	.761	**.8507**
5-Layer Bidirectional CE LSTM with Bidirectional Global Attention	.761	.8209
Bilateral Multi-Perspective Matching (Full, Maxpool Matching)	.760	.819

Table 4: Subproblem 2 Results

parameter tuning).

We also experimented with a CE LSTM with Word-By-Word Attention, and it performed reasonably well with an F1-Sore of .768 and S_2 of .8507 (See Table 4), but we did not continue to build upon this model because of prohibitive training time. Our implementation of the Bilateral Multi-Perspective Matching Model with Full and Maxpool Matching Layers performed reasonably well out of the box, using the parameters utilized in the original paper (Wang et al. 2017), scoring a F1-Score of .760 and S_2 of .819 (See Table 4 and Appendix A.5). We were unable to apply hyper parameter tuning to this model because of the extensive training time.

Ultimately, when we ran our entire pipeline, i.e. running the linear SVM classifier and feeding the headline-article pairs that had been classified as *related* into our neural networks, we performed well. Our BiCE LSTM BiGA scored S_{FNC} of.8658, while our Bilateral Multi-Perspective Matching Model scored S_{FNC} of.8501 (See Table 5). There is no leaderboard for the challenge, but those who have reported initial results on the FNC-1 Slack, claim to receive results in the .70 - .80 range as of March 22, 2017. Our models outperform these reported results.

7 Discussion

Examining the confusion matrix from our SVM (Figure 3), it seems like the model is performing well at classifying article-headline pairs for subproblem 1. In the future, we might tune this model to reduce the number of *related* False Negatives. Each of these mis-classifications leads to a decrease in the number of samples that might be correctly classified by our neural networks.
.

Interestingly, the optimal BiCE LSTM BiGA model does not classify any headline-article pairs as *disagree*, while a nearly performant BiCE LSTM BiGA with a slightly lower F1 Score (.001 lower) does correctly classify .2671 of the *disagree* (Figures 4 and 5). Neither our loss function nor the S_{FNC} metric provides greater weight to correctly classifying *disagree* rather than *agree* or *discuss*. Since *disagree* makes up the smallest contingent of the dataset at 1.7% of headline-article pairs, it is understandable that this is the hardest class to correctly label and that sacrificing performance on the other two more common classes will not yield the best scores according to given metric. If greater weight was assigned to correctly classifying *disagree* we could modify our cost function appropriately to up-weight the cost of mislabeling these headline-article pairs.

The Bilateral Multi-Perspective Matching Model with Full and Maxpool Attention layers performed similarly to the BiCE LSTM BiGA, and with further tuning might be able to out perform BiCE LSTM BiGA (Figure 6). Incorporation of the additional Attentive Matching and Max-Attentive Matching layers may also improve the results of model, given a powerful GPU.

As it stands, the BiCE LSTM BiGA seems to over optimize to the training data (See Appendix A.7), but when performing hyper parameter tuning, these models were still the most effective on the development set. We took care to ensure that there no were overlapping articles between the train, dev, and test sets to avoid polluting our results; but the fact that an over trained model still performs

Model	S_{FNC}
Bidirectional CE LSTM with Bidirectional Global Attention (BiCE LSTM BiGA)	**.8658**
Bilateral Multi-Perspective Matching (Full, Maxpool Matching)	.8501

Table 5: FNC-1 Results

the best on the dev set, seems to indicate that the dataset has some underlying similarities across articles.

8 Conclusions & Future Work

In conclusion, our SVM with TF-IDF cosine similarity features performed very well on subproblem 1 with an F1-Score of .9712, and our Bidirectional CE LSTM with Bidirectional Global Attention (BiCE LSTM BiGA) with an F1-Score of .761 and and S_2 of .8507. Overall, we scored a $S_{FNC} = 0.8658$ which out performs the reported models on the FNC-1 Slack channel, which average .70 - .80.

Moving forward, we hope to submit results on the test set for FNC-1 that will be released in June. Additionally, we are planning to perform greater qualitative analysis to determine potential strategies for correctly classifying *disagree* headline-article pairs and look into other potentially relevant network architectures. We additionally hope to try tuning our Bilateral Multi-Perspective Matching Model and look for more powerful GPUs on which to run all four layers of attention.

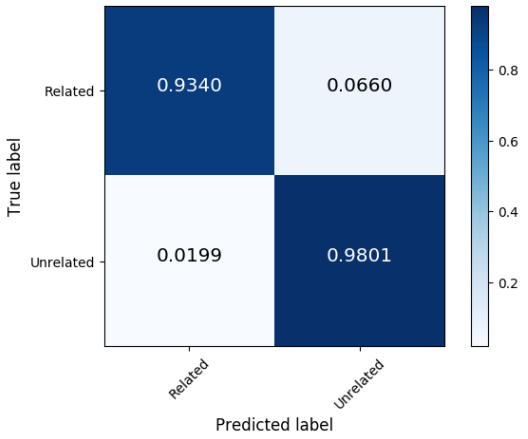

Figure 3: SVM with TF-IDF Cosine Similarity

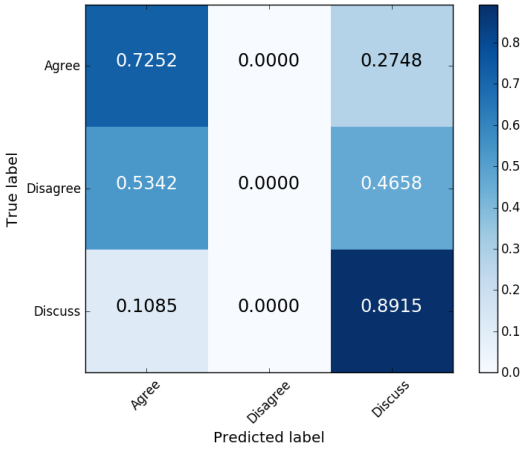

Figure 4: Optimal BiCE LSTM BiGA

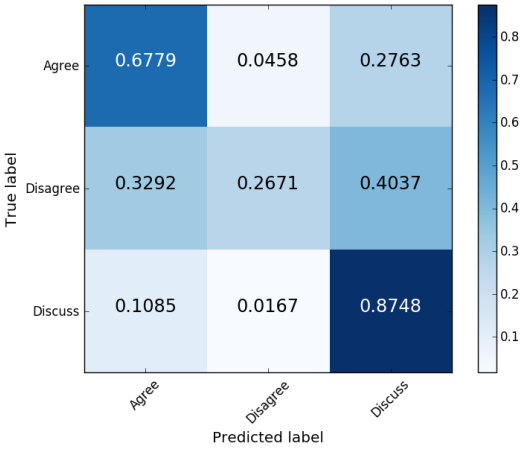

Figure 5: Suboptimal BiCE LSTM BiGA

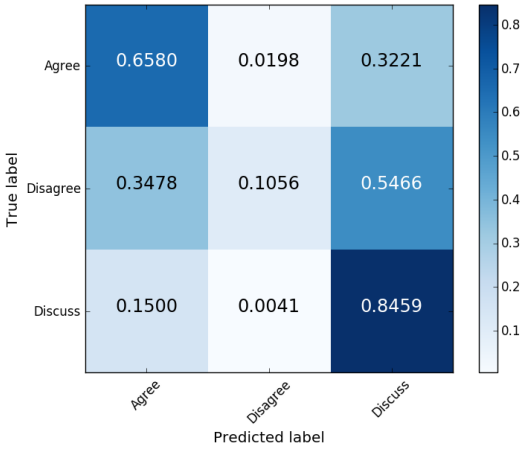

Figure 6: Bilateral Multi-Perspective Matching

Acknowledgments

We wish to thank Danqi Chen, who met frequently with us to provide insight and feedback for the challenges that we faced throughout our research. Danqi's knowledge of natural language processing enabled her to guide us to helpful academic resources, which were invaluable to our progress.

We also wish to thank Stanford's CS 224N course staff, who provided credits for Microsoft Azure instances, which enabled us to conduct computationally intensive research. In this same spirit, we wish to thank the Stanford Institute for Computational and Mathematical Engineering (ICME), who provided us with free access to their GPU cluster.

References

Fake News Challenge, http://www.fakenewschallenge.org/, 2017.

[Augenstein et al., 2016] Isabelle Augenstein, Tim Rocktaschel, Andreas Vlachos, and Kalina Bontcheva. Stance Detection with Bidirectional Conditional Encoding. 2016

[Bowman et al., 2015] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference, 2015.

[Pennington et al., 2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. 2014.

[Rocktaschel et al., 2016] Tim Rocktaschel, Edward Grefenstette, Karl Moritz Hermann, Tomas Kocisky, and Phil Blunsom. Reasoning about Entailment with Neural Attention. 2015

[Wang et. al, 2017] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral Multi-Perspective Matching for Natural Language Sentences. 2017

A. Appendix

A.1 Headline and Article Length Distributions

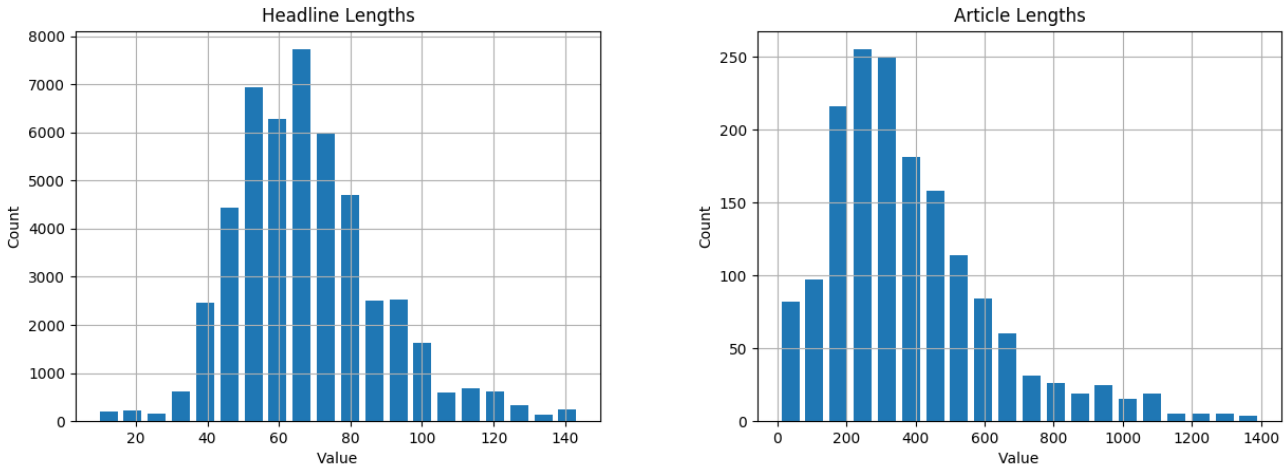

Figure 7: FNC-1 Headline and Article Lengths, 3 Standard Deviations

A.2 Baseline Models

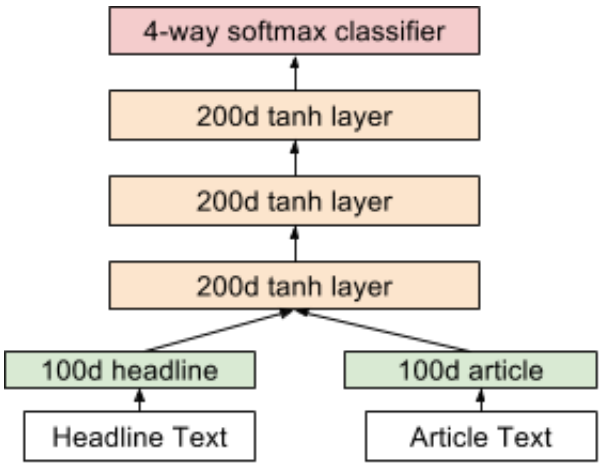

Figure 8: BOW MLP Model (Bowman et al. 2015)

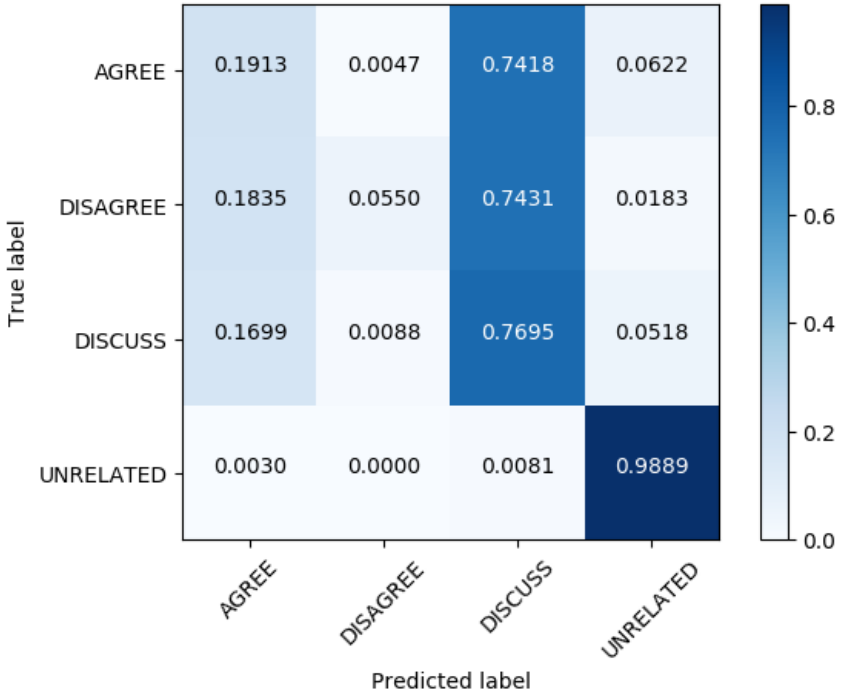

Figure 9: Confusion Matrix for Lexicalized Linear Classifier

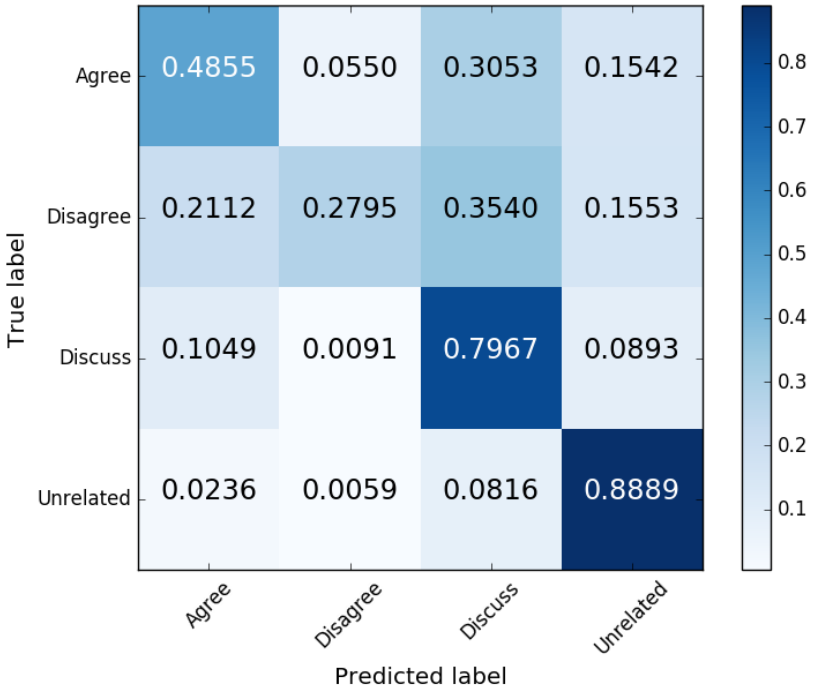

Figure 10: Confusion Matrix for BOW MLP

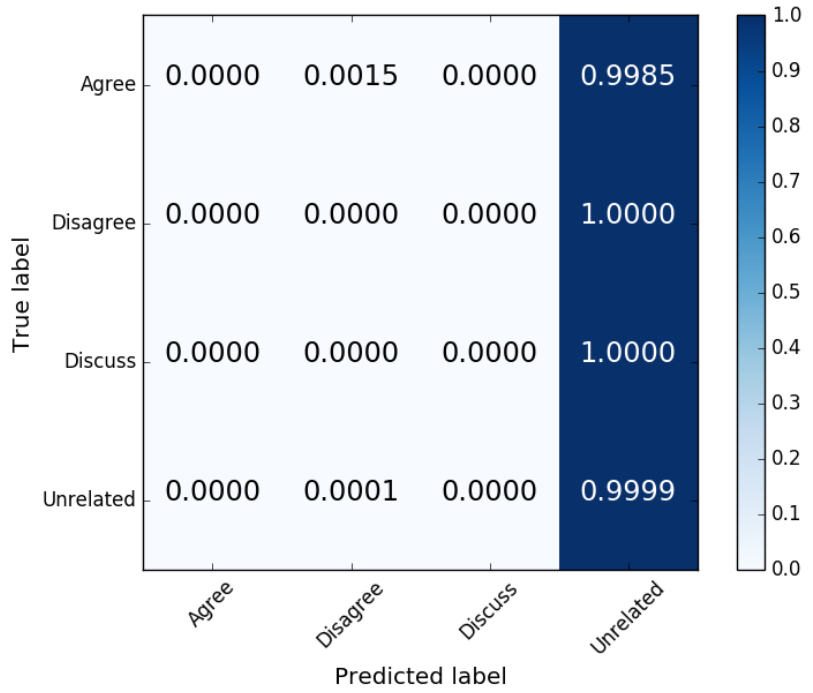

Figure 11: Confusion Matrix for LSTM with Concatenated Input

A.3 Subproblem I Linear Classifier Features

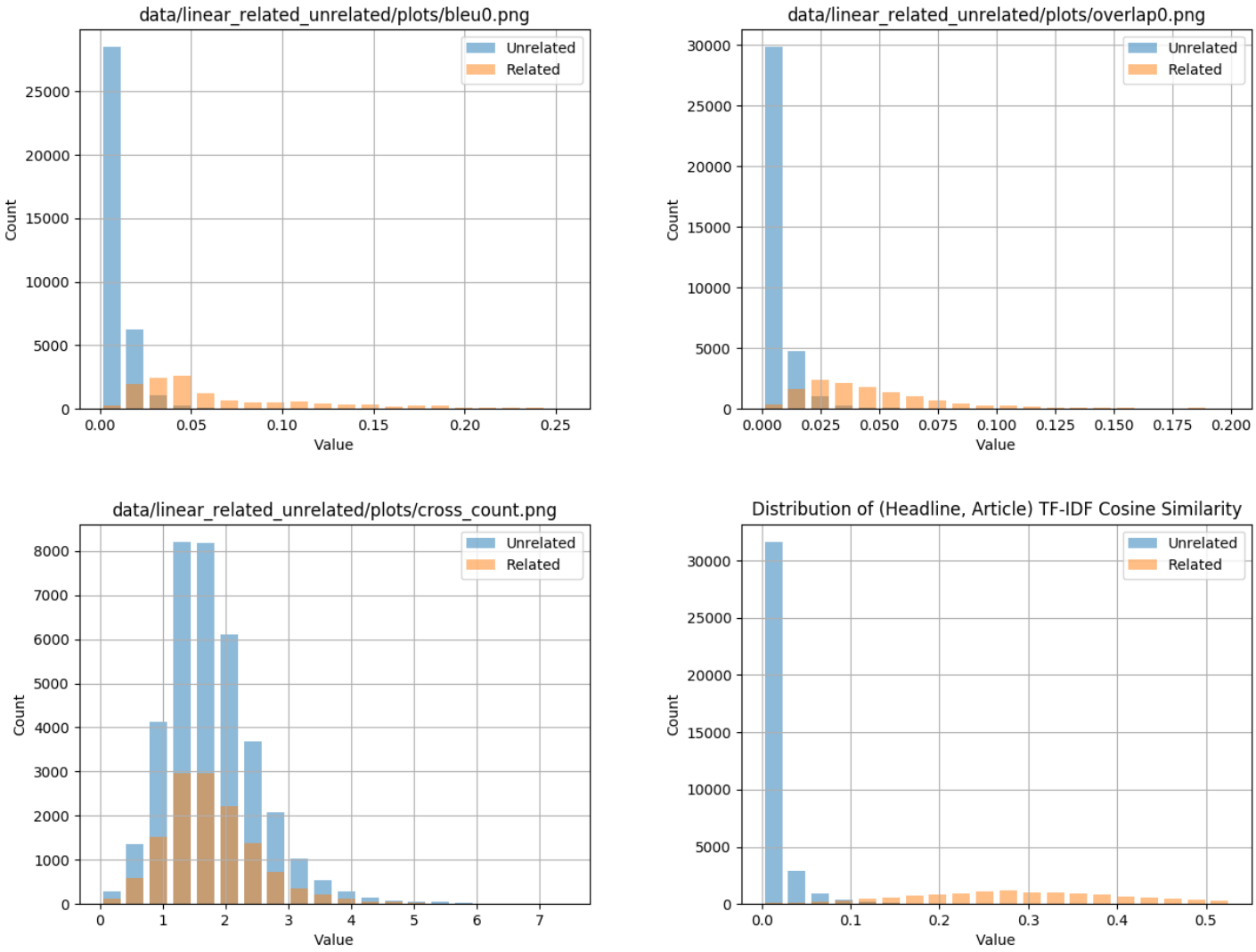

Figure 12: Feature Analysis Plots for the Lexicalized Linear Model

For the Linear Classifier used for Subproblem I, we initially extracted the following features for each headline, article pair:

1. Cosine distance between tf-idf vectors of the headline and article

2. Maximum BLEU score of the heaadline with respect to the segmented article. We segment the article into windows of length equal to the length of the headline, with a stride equal to one-half the length of the headline, and take the maximum BLEU score of the headline with respect to any article segment.

3. Overlap between headline and article, measured by normalized Jaccard distance over the set of words in the headline and the set of words in the article.

4. Cross-grams between the article and headline as specified by Bowman et. al.

After performing feature analysis (Figure), we only used cosine distance between tf-idf because of the high correlation between the tf-idf scores and the classification of the pair as related.

A.4 LSTM Equations

The LSTM is based heavily on three types of gates: input, output, and forget.

$$\text{Input Gate: } i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1}) \tag{15}$$

$$\text{Forget Gate: } f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1}) \tag{16}$$

$$\text{Output Gate: } o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1}) \tag{17}$$

$$\text{Memory Generation: } \hat{c_t} = \tanh(W^{(c)}x_t + U^{(c)}h_{t-1}) \tag{18}$$

$$\text{Final Cell: } c_t = f_t \cdot c_{t-1} + i_t \cdot \hat{c_t} \tag{19}$$

$$\text{Hidden Vector: } h_t = o_t \cdot \tanh(c_t) \tag{20}$$

A.5 Parameters For Major Models

Parameter	Value
Drop Out Rate	0.9
Train Size	0.8
GloVe Embedding Size	300
Max Article Length	800
Batch Size	50
Number of Epochs	5
Beta (L2 Regularization Constant for 2 Layer MLP)	.01
Learning Rate	.001
Hidden Size (LSTM)	300
Hidden Size (MLP)	150

Table 6: Hyper Parameters Utilized for All Models Except BiCE LSTM BiGA and Bilateral Multi-Perspective Matching

Parameter	Value
Drop Out Rate	1
Train Size	0.8
GloVe Embedding Size	300
Max Article Length	800
Batch Size	50
Number of Epochs	15
Beta (L2 Regularization Constant for 2 Layer MLP)	.001
Learning Rate	.0001
Hidden Size (LSTM)	300
Hidden Size (MLP)	150

Table 7: Hyper Parameters Utilized for Final BiCE LSTM BiGA Model

Parameter	Value
Drop Out Rate	0.9
Train Size	0.8
GloVe Embedding Size	300
Max Article Length	800
Batch Size	10
Number of Epochs	5
Beta (L2 Regularization Constant for 2 Layer MLP)	.01
Learning Rate	.0001
Hidden Size (LSTM)	100
Hidden Size (MLP)	150
Num Perspectives	20
Context Hidden Size	100

Table 8: Hyper Parameters Utilized for Bilateral Multi-Perspective Matching

A.6 Hyper Parameter Tuning

We performed hyper parameter tuning for the parameters in A.5 for the Conditional LSTM and the Bidirectional Conditional LSTM with Bidirectional Global Attention. Specifically we tuned learning rate, dropout, and regularization parameters. A visualization of the learning rate vs regularization parameters can be seen in Figure 14.

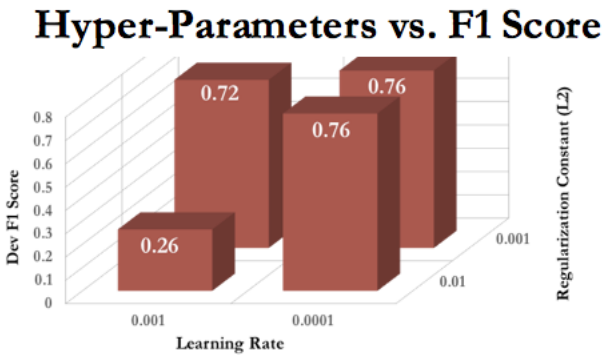

Figure 13: Learning Rate and L2 Regularization Hyper-Parameters vs. F1 Score for Bidirectional C.E. LSTM w. Bidirectional Global Attention. We Also Tuned Drop Out Rates.

A.7 Train vs. Test Accuracy

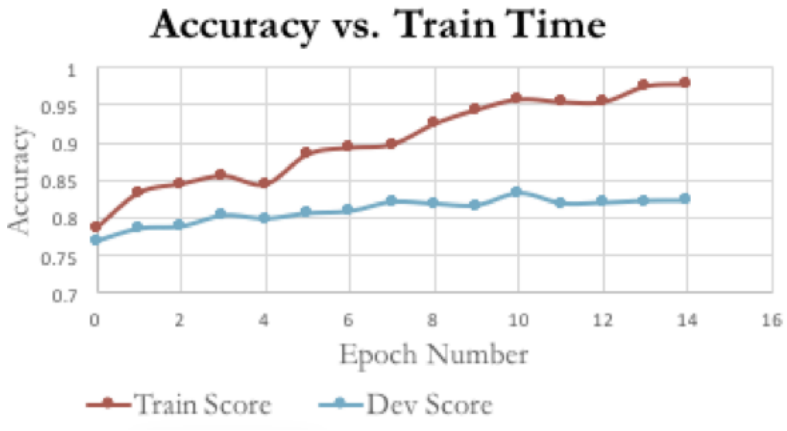

Figure 14: Accuracy vs training time for Train/Dev for Bidirectional C.E. LSTM w. Bidirectional Global Attention