"Is it true?" – Deep Learning for Stance Detection in News

Neel Rakholia
ICME
Stanford University
neelvr@stanford.edu

Shruti Bhargava
ICME
Stanford University
shrutib@stanford.edu

Abstract

Stance detection is an important component of fake news detection. In this project we explore different neural net architectures for stance detection in news articles. In particular, we analyze the effectiveness of recurrent neural nets for this problem. We discover that a modified attentive reader model is well suited for the task. While our best deep learning model comfortably exceeds the baseline score set by Fake News Challenge, a simple feedforward network marginally outperforms it. As far as we are aware, our LSTM-based RNN model is the state of the art end-to-end deep learning model for this dataset.

1 Introduction

News is an important source of information for people. However, in a world increasingly shaped by social media, the proliferation of "fake" and often "hyper-partisan" news [1] has challenged traditional journalism sources and media infrastructure. Fake news is informally defined as "made-up stories with an intention to deceive." [1] By extension, fake news detection is the task of determining the probability of an article being fake. [2] There are several challenges to automatic detection of fake news: 1) determining if the facts present in the news article are correct; 2) analyzing the relationship between the article headline and article body; 3) quantifying the inherent bias of a written text etc. Our work attempts to tackle the second problem. In particular, we focus on developing a model for the Fake News Challenge.

1.1 Fake News Challenge and Problem Statement

FNC (Fake News Challenge) is a competition to encourage the exploration of AI (Artificial Intelligence) and NLP (Natural Language Processing) in combating the problem of fake news. As a stepping stone to the bigger goal, the challenge's first task is Stance Detection. This would then be part of a bigger pipeline, which would ideally be able to assist fact checkers. [3]

The Stance Detection task, as outlined by FNC, is to classify the text in a news article body with respect to the claim made in its headline. These pairs of headlines and bodies may or may not be from the same article. There are four categories into which the stance must be classified: agrees, disagrees, discusses and unrelated. The performance of a classifier is measured by a weighted accuracy metric (similar to the F1 measure). [1]

[1] It must be noted that FNC does not consider claims made by newsworthy individuals as fake news. It also explicitly excludes humorous or satirical content, which is meant to entertain and not deceive

1.2 Outline

Previous research in this area has focused on analyzing the relationship between two short sentences or phrases: natural language sentence matching (NLSM). For instance, NLI (Natural Language Inference) attempts to infer if a given hypothesis follows from the premise. [4]. SemEval 2016's task on stance detection in tweets was geared towards analyzing the relationship between tweets and their views on specific topics. [5] We consider FNC's stance detection problem as an extension of these two tasks. Because of the novelty of the task and dataset, we take an exploratory approach where we experiment with various simple and complex neural network architectures. We take the work of Augenstein et. al. on Conditional and Biconditional LSTMs (Long Short Term Memory) as an initial framework and add additional features to tackle specific issues. It must be noted that one major difference between the aforementioned tasks and our challenge is the length of the input. These tasks consider a pair of two sentences whereas our task involves a short sentence, that is a headline, and a lengthy news article body. The details of our models are outlined in the following sections.

2 Related Work

There are several existing approaches to NLSM. A majority of them involve constructing dense vector representations of the two phrases being compared and then computing the relationship between these two dense vectors using some metric. The dense vectors are constructed using neural network architectures with word embeddings as input. The following papers outline different ways to encode phrases and using these encodings to quantify the relationship between them.

2.1 Stance Detection with Bidirectional Conditional Encoding

Augenstein et. al. [6] attempt a more challenging version of the SemEval 2016 tweet stance detection task [5]: when the test target might not be mentioned in the training tweets explicitly. They experiment on encoding the tweet conditioned on the target. Such an encoding allows the context of a target to influence the encoding of a tweet. They further propose bidirectional conditional encoding in which they construct two encodings, one each for the target and the tweet: reading from right to left and left to right. They argue that this architecture ensures when a word is read by the BiLSTM (Bidirectional LSTM), both its left and right side contexts are taken into account. However, one drawback of this approach is that a tweet is conditioned on the target, but the target is not conditioned on the tweet. This could potentially improve the model for stance detection as highlighted in Wang et. al. [7]

2.2 Teaching Machines to Read and Comprehend

Hermann et. al. [8] address the lack of supervised natural language reading and comprehension models. They propose three LSTM-based neural networks to deal with long sequences of text. They first discuss Deep LSTMs, i.e. LSTMs with peepholes. They reason that a fixed width hidden vector is less capable of handling long sequences. Correspondingly, they propose the "Attentive Reader" model. This model encodes the comprehension (document) and question separately using BiLSTMs. It then encodes the document by weighing all output vectors of the document LSTMs in the "context of" question BiLSTM outputs. The weighted outputs are used to obtain the final dense vector representation for a document. The final model that they discuss is the "Impatient Reader" model. This a modified version of the "Attentive Reader" model in which BiLSTM for a query "pays attention" to the full document. The "Attentive Reader" and "Impatient Reader" models give comparable performances on different datasets. The success of these models on long texts is suggestive of their usefulness for the FNC challenge.

2.3 Bilateral Multi-Perspective Matching for Natural Language Sentences

More recent work in this area has focused on designing new ways to incorporate attention into RNN-based neural net architectures. Wang et. al. [7] propose a BiLSTM architecture that consists of three main elements: 1) word representation layer, a dense vector representation of words; 2) context representation layer, a framework to encode sentences and phrases into vectors; and 3)

matching layer, to compute the similarity between contextual encodings at each time step to every other time step. An important difference between this work and Augenstein et. al.'s paper is that the encodings of the two sentences are independently constructed; one is not conditioned on the other. Additionally, the sentences are encoded in both directions: forward and backward.

It must be noted that while the first architecture has shown some success in identifying and classifying stance in tweets, it was not able to achieve the F1-score obtained when using a Support Vector Machine (SVM) classifier on n-grams. This highlights the effectiveness of using simple linear classifiers on a bag of words (BOW) for stance detection tasks. [9]

3 Approach

We apply several existing neural network architectures to the problem of stance detection in news articles. We further propose two novel architectural variations for the task and compare their performance to existing models. The following section outlines the models that we tested. They are ordered in an approximate order of the complexity of their architecture.

3.1 Notation

Let z be the number of headline-body pairs in the training set. Then, for any headline-body pair j, let $x_1, x_2, \ldots, x_{n_j}$ be the sequence of words or tokens (punctuation and other special characters) in the headline. Similarly, let $y_1, y_2, \ldots, y_{m_j}$ be the sequence of words or tokens in the corresponding body. n_j is the total number of words in the headline and m_j is the the number of words in the body. We represent each word or token with a $1 \times d$ dimensional pretrained dense vector or embedding. Let $\mathbf{d}_1, \ldots, \mathbf{d}_{n_j}$ be the embeddings of words in the headline and let $\mathbf{e}_1, \ldots, \mathbf{e}_{m_j}$ be the embeddings of words in the body. $\mathbf{d}_i, \mathbf{e}_i \in \mathbb{R}^d$.

3.2 Baseline: A modified feedforward neural net

A naive approach to encoding the headline and body as dense vectors is representing them as the mean of word embeddings. Quantitatively, if $\mathbf{a}_j$ is the vector representation of headline in pair j, then:

$$\mathbf{a}_j = \frac{1}{n_j} \sum_{i=1}^{n_j} \mathbf{d}_i \tag{1}$$

Correspondingly, let $\mathbf{b}_j$ be the encoding of body. Then:

$$\mathbf{b}_j = \frac{1}{m_j} \sum_{i=1}^{m_j} \mathbf{e}_i \tag{2}$$

Once these representations have been constructed for all pairs z, we concatenate $[\mathbf{a}_j\ \mathbf{b}_j]$ into a $1 \times 2d$ vector $\mathbf{w}_j \in \mathbb{R}^{2d}$ and feed it as input to a feedforward neural net.

$$\mathbf{h} = f(\mathbf{w}_j \mathbf{W_1} + \mathbf{b_1}) \tag{3}$$

$$\mathbf{h}_{drop} = dropout(h) \tag{4}$$

$$\mathbf{\hat{p}_j} = g(\mathbf{h}_{drop} \mathbf{W}_2 + \mathbf{b}_2) \tag{5}$$

Here, $\mathbf{W_1} \in \mathbb{R}^{2d \times H}$, $\mathbf{b_1} \in \mathbb{R}^H$, $\mathbf{W_2} \in \mathbb{R}^{H \times 4}$, and $\mathbf{b_2} \in \mathbb{R}^4$. H is the hidden layer size, $f(x)$ is the ReLU (Rectified Linear Unit) non-linearity and $g(x)$ is the softmax function. The dropout layer randomly drops neurons while training. It helps reduce overfitting. Finally, $\mathbf{\hat{p}_j}$ is a 1×4 vector denoting the probability of pair j belonging to each class (agree, disagree, discuss, and unrelated). We evaluate the loss using a simple cross entropy penalty. If $\mathbf{p_j}$ is a one-hot vector denoting the true class of pair j, then:

$$L_{CE}(\mathbf{p_j}, \mathbf{\hat{p}_j}) = \sum_{i=1}^{4} p_i \log(\hat{p}_i) \tag{6}$$

$$L_{CE} = \frac{1}{z} \sum_{j=1}^{z} L_{CE}(\mathbf{p_j}, \mathbf{\hat{p}_j}) \tag{7}$$

Most of the following models build on this architectural framework. In subsequent subsections, we abstract away from the dimensions of individual matrices and vectors to focus more on the overall structure.

3.3 Independent Encoding (IE)

An unweighted mean of word embeddings fails to capture the order of words. To overcome this we use two independent LSTMs ($LSTM_{head}$ and $LSTM_{body}$) as encoders. [10] We first cap the maximum length of headline and body to $n_{\max}$ and m_{max} respectively. Shorter texts are padded with zeros. We then encode the headline for a pair j as follows. Let $\mathbf{d}_t$ be the word embedding, $\mathbf{c}_t$ the hidden state, and $\mathbf{h}_t$ the output. [6] (Refer to the appendix for equations) The output $\mathbf{h}_{j,head}$ corresponding to the last word of the headline[2] is chosen as the dense vector representation of the headline. A similar encoding is generated for the corresponding body $\mathbf{h}_{j,body}$[3] using an independent LSTM unit. These encodings are passed to a ReLU layer followed by a softmax:

$$\mathbf{h} = ReLU(\mathbf{h}_{j,head}\mathbf{W_{head}} + \mathbf{h}_{j,body}\mathbf{W_{body}} + \mathbf{b_1}) \tag{8}$$

$$\mathbf{h}_{drop} = dropout(h) \tag{9}$$

$$\mathbf{\hat{p}_j} = softmax(\mathbf{h}_{drop}\mathbf{W} + \mathbf{b}_2) \tag{10}$$

The loss computation is identical to the baseline.

3.4 Conditional Encoding (CE)

Conditional encoding was used by Rocktschel et. al. [11] for recognizing textual entailment. Instead of constructing two separate encodings for the headline and body, we condition the encoding of the body on the encoding of the headline. We do this by passing $\mathbf{h}_{j,head}$ and $\mathbf{c}_{j,head}$ (obtained from $LSTM_{head}$) as the initial state tuple $(\mathbf{h}_0, \mathbf{c}_0) = (\mathbf{h}_{j,head}, \mathbf{c}_{j,head})$ for $LSTM_{body}$. We then pass $\mathbf{h}_{j,head}$ and $\mathbf{h}_{j,body}$ to the layers outlined in equations 8, 9, and 10. Note that this is a slight deviation from the original implementation outlined by Augenstein et. al. They only pass $\mathbf{h}_{j,body}$ to the feedforward network instead of both $\mathbf{h}_{j,head}$ and $\mathbf{h}_{j,body}$.

3.5 Bidirectional Conditional Encoding (BCE)

Augenstein et. al. proposed an extension to Rocktaschel's work. Instead of using a unidirectional LSTM, they construct encodings using a BiLSTM. Let $(\overrightarrow{\mathbf{h}}_{j,head}, \overrightarrow{\mathbf{c}}_{j,head})$ and $(\overleftarrow{\mathbf{h}}_{j,head}, \overleftarrow{\mathbf{c}}_{j,head})$ be the final state tuples of headline and reversed headline obtained from $LSTM_{head}$. These encodings are passed as an initial state tuple to forward and backward initial states of $LSTM_{body}$. If $(\overrightarrow{\mathbf{h}}_0, \overrightarrow{\mathbf{c}}_0)$ and $(\overleftarrow{\mathbf{h}}_0, \overleftarrow{\mathbf{c}}_0)$ are the initial state tuples for the forward and backward pass of $LSTM_{body}$, then:

$$(\overrightarrow{\mathbf{h}}_0, \overrightarrow{\mathbf{c}}_0) = (\overrightarrow{\mathbf{h}}_{j,head}, \overrightarrow{\mathbf{c}}_{j,head}) \tag{11}$$

$$(\overleftarrow{\mathbf{h}}_0, \overleftarrow{\mathbf{c}}_0) = (\overleftarrow{\mathbf{h}}_{j,head}, \overleftarrow{\mathbf{c}}_{j,head}) \tag{12}$$

We obtain $\overrightarrow{\mathbf{h}}_{j,body}$ and $\overleftarrow{\mathbf{h}}_{j,body}$ as the final outputs from $LSTM_{body}$. These are passed to the feedforward network outlined in 8, 9, and 10.

3.6 Transfer Learning using SNLI (TL)

Zarrella et. al. [12] successfully used transfer learning to classify stance in tweets.[4] They pretrained their LSTM-based recurrent neural network on hashtag-tweet pairs. They subsequently trained it on tweet-target pairs to predict the stance between them. We take a similar approach. SNLI (Stanford Natural Language Inference) corpus consists of test-hypothesis pairs and the relationship between them. [13] While the length of bodies for the Fake News Challenge is larger than the length of SNLI texts, the classes for both tasks are similar. The relationship between texts and hypothesis

[2] x_{n_j} if $j < n_{max}$ else $x_{n_{max}}$

[3] There are also two corresponding hidden states $\mathbf{c}_{j,head}$ and $\mathbf{c}_{body,j}$ that we use in the next subsections

[4] Their model performed the best on SemEval Stance Detection in Tweets 2016.

in the SNLI corpus is classified as: contradiction, neutral, or entailment. The possible stances for the Fake News Challenge are: agree, disagree, discuss, unrelated. We attempt to exploit this similarity between tasks to first train our conditional encoding (CE) model on the SNLI corpus.[5] We subsequently use learned weights along with L2-regularization on hidden layer weights to train the model on fake news dataset.

3.7 Multi-pass Conditional Encoding (MPCE)

In addition to exploring the aforementioned models for this task, we also propose a variation to the CE model[6]. Conditioning the encoding of body on the encoding of headline is limiting: encoding of headline does not take into account the body for instance. Our modified CE model attempts to alleviate some of these issues. At every training step, we have two passes through $LSTM_{head}$ and $LSTM_{body}$. We label the outputs from this process as: $\mathbf{h}_{1,j,head}$, $\mathbf{h}_{1,j,body}$, $\mathbf{h}_{2,j,head}$ and $\mathbf{h}_{2,j,body}$. Here subscripts 1 and 2 refer to the pass number. $\mathbf{h}_{2,j,head}$ and $\mathbf{h}_{2,j,body}$ are subsequently passed to the $ReLU$ and $softmax$ layers to make a prediction. Note that this is different from a BiLSTM where forward and backward passes are executed independently on a given cell. Also inputs are reversed for BiLSTM. This is not the case for MPCE. The figure in appendix further elucidates the model's architecture.

3.8 Attentive Reader Model and its variations (ARM)

Attention provides another viable approach to constructing conditional encodings for headlines and bodies. We explore Hermann et. al's [8] work on LSTM-based neural network architectures for tackling machine comprehension tasks. They propose an attention mechanism for dealing with long texts that they call the Attentive Reader Model. We adapt their model for our task. Consider two independent single-layer BiLSTM cells: $LSTM_{head}$ and $LSTM_{body}$. Let the four final outputs obtained from these cells be: $\overrightarrow{\mathbf{h}}_{j,head}$, $\overleftarrow{\mathbf{h}}_{j,head}$, $\overrightarrow{\mathbf{h}}_{j,body}$, and $\overleftarrow{\mathbf{h}}_{j,body}$. Additionally, assume $\overrightarrow{\mathbf{h}}_{t,head}$, $\overleftarrow{\mathbf{h}}_{t,head}$, $\overrightarrow{\mathbf{h}}_{t,body}$, and $\overleftarrow{\mathbf{h}}_{t,body}$ are the outputs at any time step t. To construct a dense vector representation for the body, we perform the following computation.

$$\mathbf{h}_{j,head} = [\overrightarrow{\mathbf{h}}_{j,head} \overleftarrow{\mathbf{h}}_{j,head}] \tag{13}$$

$$\mathbf{h}_{t,body} = [\overrightarrow{\mathbf{h}}_{t,body} \overleftarrow{\mathbf{h}}_{t,body}] \tag{14}$$

$$\mathbf{m}_t = \tanh(w_{t,body} \odot \mathbf{h}_{t,body} + w_{head} \odot \mathbf{h}_{j,head}) \tag{15}$$

$$\mathbf{s}_t = softmax(w_t^T \mathbf{m}_t) \tag{16}$$

$$\mathbf{v}_{j,body} = \mathbf{h}_{body}\mathbf{s}_t \tag{17}$$

Here, $w_{t,body}$, w_{head}, w_t are scalar weights and $\mathbf{h}_{body}$ is a $m_j \times d$ matrix of outputs from $LSTM_{body}$ at every time step. Hermann et. al [8] only use such a construction for contexts. We go a step further and also use equations 19, 20, 21, 22, and 23 for constructing dense vector representations of headlines: $\mathbf{v}_{j,head}$ (we call this modified model **ARM2**). This addition is inspired from Wang et. al's paper: Bilateral Multi-Perspective Matching for Natural Language Sentences. [7] As a final step, we pass $\mathbf{v}_{j,head}$ and $\mathbf{v}_{j,body}$ to the layers outlined in equation 8, 9 and 10.

In addition to the models outlined above, we also experiment with CNNs as outlined in. [15] Please refer to the paper for implementation details.

4 Experiments

In this section we briefly outline the experiments we conducted on our models. We start by giving details about the dataset and our implementation.

[5] Our CE model achieved 77% accuracy on the validation set for SNLI

[6] Inspired by Ankit et. al. [14]

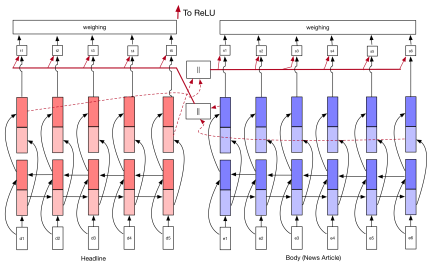


Figure 1: Attentive Reader Model (adapted from [8])

4.1 Dataset and Implementation

The FNC stance detection training dataset consists of two files, one with news article body IDs and content and the other with body IDs, stances and corresponding labels for the stance-body pairs. Only training data has been released at the time of this project. The test data will be released in June 2017. We utilize the generate_hold_out_split function provided by FNC to split training data into

Stance	Proportion
Agree	0.074
Disagree	0.017
Discuss	0.178
Unrelated	0.731

Properties	Train	Validation	Test
Total Data Points	40350	5041	4581
Unique Bodies	1346	168	169
Unique Headlines	1643	1445	1407

Table 1: Dataset properties

train, validation and test sets. The split ensures that these three sets have no overlapping bodies.[7] We then use R, to convert all text data to lower case and to obtain word count distributions for headlines and bodies in the train set. As a final preprocessing step, we tokenize the headlines and bodies using nltk word tokenize and encode the characters in ASCII.

We need additional processing to transform text to numerical input that can be fed to neural networks. We encode words using GloVe 300d [16] pretrained word vectors. Any unknown word or token is initialized with a zero vector. Most RNN-based models require inputs to be of the same length for efficient computation. Consequently, we pad the headlines (or bodies) shorter than the maximum length (a hyperparameter) with zero tokens. We keep track of words in the original sentence by using boolean masks.

We implement the aforementioned models using Tensorflow 0.12.1 on Python 2.7. Weights are initialized with Xavier initialization and biases are initialized to zero. The models are trained in batches. Adam optimizer [17] is used to minimize the loss. We use dropout for LSTM hidden states and L2 regularization to prevent overfitting in $RELU$ and $softmax$ layers.

4.2 Evaluation Metrics

We analyze the performance of our models using the scoring function defined by FNC[8]. The function evaluates each prediction in a two step process: 1) Correctly classify headline and body text as related (agree, disagree, and discuss) or unrelated: 25% score weighting; 2) Correctly classify related pairs as agrees, disagrees, or discusses: 75% score weighting. [9] The "accuracy" of the model is then expressed as a percentage of the maximum possible score. Subsequently, the model with the highest score on the validation set is selected. In addition to this scoring metric, we also report the macro average F1-scores for each classification.

[7] https://github.com/FakeNewsChallenge/fnc-1-baseline
[8] https://github.com/FakeNewsChallenge/fnc-1-baseline
[9] http://www.fakenewschallenge.org/

Model	Train Score	Train F1	Valid Score	Valid F1	# Parameters
Baseline	89.78	0.731	83.67	0.604	309K
Baseline + WL	92.51	0.800	**85.61**	0.661	309K
IE	96.17	0.866	79.99	0.592	473K
CE	95.90	0.881	79.29	0.595	473K
BCE	98.75	0.949	78.70	0.574	912K
TL	93.04	0.540	81.76	0.660	473K
MPCE	97.30	0.840	81.20	0.589	473K
CNN	92.40	0.861	68.28	0.538	1803K
ARM	95.20	0.816	**84.96**	0.680	1011K
ARM2	98.85	0.961	**84.86**	0.683	1011K

Table 2: Performance of different models on FNC dataset

4.3 Preliminary Model Selection

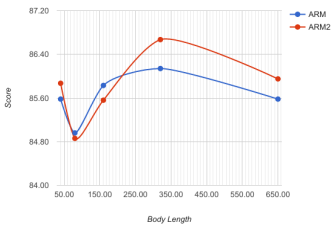
As a first step to identifying the best model for the task, we train the aforementioned models on FNC dataset with a fixed set of hyperparameters. These parameters were selected based on previous work in the field and coarse hyperparameter tuning.[10] For instance, Srivastava et. al. [18] suggest that keep probability values between 0.5 and 0.8 work well. Our hidden layer dimensions were chosen as a compromise between model complexity and GPU memory. One parameter that significantly affects both training time and model performance is the length that we truncate headlines and bodies to. We fix these values by looking at the histogram of text lengths (shown in appendix). We select maximum body length such that it is as small as possible without significantly compromising the performance. The list of hyperparameter values that we used is outlined in the appendix. Table 2 summarizes our initial findings. We observe that Baseline + WL[11], ARM, and ARM2 give the best validation scores. Moreover, ARM and ARM2 have the highest F1 score. Consequently, we further tune the hyperparameters of ARM and ARM2 to boost their performance.

4.4 Tuning Hyperparameters

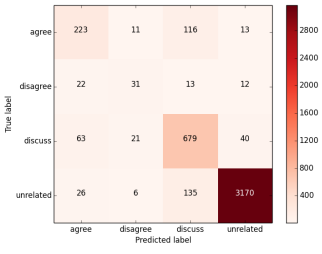
An exhaustive search over all hyperparameters is not feasible due to the long training times and limited computational resources. Therefore, we perform hyperparameter tuning in a greedy manner: we select the best value for a particular parameter and use that value for subsequent tuning.

4.4.1 Body Length

A significant parameter affecting both computation time and model performance is the length that we truncate a news article body to. We tune ARM and ARM2 over the following body lengths: 40, 80, 160, 320, 650. Our results are summarized in figure 3. We observe that models with short bodies



(a) Validation Score vs Body Length



(b) Confusion Matrix for ARM2

perform surprisingly well. While the score increases when we increase the length, it plateaus and

[10] We tried a few values in a large range and settled on values that worked well for all models.

[11] Here we modify the baseline so that misclassification of agree, disagree, and discuss stance is penalized higher than other misclassifications

drops for very long bodies. This highlights the importance of the first few 100 words in determining the stance of headline-body pairs. ARM2 with a body length of 320 gives us the best validation score. We select this model for further tuning.

4.4.2 L2 Regularization and Peepholes

The large difference between train score and validation in table 2 indicates overfitting. To remedy this we regularize the weight matrices in ReLU and softmax layers of ARM2 by imposing a L2 penalty. We sweep over a large range of L2 regularization constant values (10, 1, 0.1, 0.01, 0.001, and 1e-4) and plot the results in figure 5 in appendix.

We also experiment with peep holes, a parameter that allows TensorFlow LSTM gates to look at cell states[12]. Running ARM2 with peepholes gave us a validation score of 86.06. Neither parameter changes yield a gain in performance.

4.5 Evaluation

4.5.1 Comparison with Baseline

As a final step, we run our best model, ARM2, for 20 epochs on the training set. We subsequently report the best validation and test scores and compare them to the best scores obtained from the baseline (also run for 20 epochs). While ARM2 outperforms Baseline + WL on the validation set, it

Model	Train Score	Train F1	Valid Score	Valid F1	Test Score	Test F1
Baseline + WL	96.24	0.906	88.11	0.749	**87.74**	0.732
ARM2	99.50	0.979	**88.38**	0.756	85.99	0.701

Table 3: Performance of our best models on FNC dataset

scores lower on the test set. A couple of reasons for this could have been:

1. **Overfitting**: From table 2 and table 3, it is evident that ARM2 is overfitting despite having dropout. We perhaps need to tune dropout in conjunction with L2 regularization constant to get good results.
2. **Dataset size**: We only have 40,000 training examples. It is well documented that deep learning models outperform MLP (multi-layer perceptrons) as the dataset becomes larger.

4.5.2 Analysis of Errors

ARM2 does not do so well on the true disagrees. This can be attributed to the fact that disagree labels make less than 2% of the training data. ARM2 also frequently mis-classifies headline-body pairs that agree as discusses. This is shown in figure (b). Consider the following example:
Headline: sugarhill gang's big bank hank dead at 57
Body: ...a member of the sugarhill gang, whose pioneering hit "rapper's delight" brought hip hop to mainstream audiences 35 years ago, died tuesday of complications from cancer...
... a beefy, boisterous presence onstage, hank handled vocals in the early to middle portion of...

While the first few sentences of the article agree with the headline, a majority of the text is devoted to discussing it. There are other cases where the headline has distinct clues about the stance, but the model is not able to capture these features. For example:
Headline:hoax alert: father o'neal, who has met god and thinks she's a woman, is made up...
Body: a catholic priest from masschussetts, who was reported dead for close to an hour before medics were able to revive him, has made a shocking revelation upon his return to life...father john micheal o'neal, who was rushed to massachusetts general hospital on thursday, january 29, 2015, was revived through the aid of a high-tech machine called lucas 2, which kept the blood flowing to his brain as doctors managed to unblock vital arteries and return his heart to a normal rhythm, after a major heart attack...

[12]http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Explicitly specifying key words, and incorporating them in our model would help reduce errors like this. More examples are listed in the appendix.

It must be noted that both our baseline and ARM2 models not only **exceed the gradient boosting baseline set by FNC by more than 6.5 percent**, but they also do a much better job on classifying the "disagree" labels. See the appendix for the confusion matrix of the FNC baseline.

5 Conclusion and Future Work

In this paper, we explored several different neural network architectures for stance detection in news articles. We discovered that attention-based models, in particular, a variation of the Attentive Reader Model (ARM2) works particularly well for this challenge. While deep learning models show success in beating the baseline score set by FNC,[13] they are marginally outperformed by a simple feedforward neural network. This reflects the need to more carefully tune our deep learning model and incorporate more regularization. Some potential extensions to this work include: 1) Experimenting with a wider range for L2 regularization constant and dropout values, 2) Using GloVe 840B cased tokens for better representation of words, 3) Splitting the dataset into splits with disjoint headlines as well as bodies at the expense of some training data, 4) Implementing a two-part model: one that first predicts the stance as related or unrelated and further classifies related inputs, and 5) Including additional hand-crafted features like TF-IDF (Term Frequency-Inverse Document Frequency) to the model.

References

[1] S. Tavernise, "As fake news spreads lies, more readers shrug at the truth," Dec 2016. [Online]. Available: https://www.nytimes.com/2016/12/06/us/fake-news-partisan-republican-democrat.html

[2] N. J. Conroy, V. L. Rubin, and Y. Chen, "Automatic deception detection: Methods for finding fake news," *Proceedings of the Association for Information Science and Technology*, vol. 52, no. 1, p. 14, 2015.

[3] "Fake news challenge stage 1 (fnc-i): Stance detection." [Online]. Available: http://www.fakenewschallenge.org/

[4] B. MacCartney, "Natural language inference." [Online]. Available: https://nlp.stanford.edu/~wcmac/papers/nli-diss.pdf

[5] "Semeval-2016 task 6." [Online]. Available: http://alt.qcri.org/semeval2016/task6/

[6] I. Augenstein, T. Rocktäschel, A. Vlachos, and K. Bontcheva, "Stance detection with bidirectional conditional encoding," *CoRR*, vol. abs/1606.05464, 2016. [Online]. Available: http://arxiv.org/abs/1606.05464

[7] Z. Wang, W. Hamza, and R. Florian, "Bilateral multi-perspective matching for natural language sentences," *arXiv preprint arXiv:1702.03814*, 2017.

[8] K. M. Hermann, T. Kociský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," *CoRR*, vol. abs/1506.03340, 2015. [Online]. Available: http://arxiv.org/abs/1506.03340

[9] S. Mohammad, S. Kiritchenko, P. Sobhani, X. Zhu, and C. Cherry, "Semeval-2016 task 6: Detecting stance in tweets," in *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 31–41. [Online]. Available: http://www.aclweb.org/anthology/S16-1003

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[11] T. Rocktäschel, E. Grefenstette, K. M. Hermann, T. Kociský, and P. Blunsom, "Reasoning about entailment with neural attention," *CoRR*, vol. abs/1509.06664, 2015. [Online]. Available: http://arxiv.org/abs/1509.06664

[13] FNC has released a baseline model that achieves 79.53 % score on the validation set

[12] G. Zarrella and A. Marsh, "MITRE at semeval-2016 task 6: Transfer learning for stance detection," *CoRR*, vol. abs/1606.03784, 2016. [Online]. Available: http://arxiv.org/abs/1606.03784

[13] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "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, 2015.

[14] A. Kumar, O. Irsoy, J. Su, J. Bradbury, R. English, B. Pierce, P. Ondruska, I. Gulrajani, and R. Socher, "Ask me anything: Dynamic memory networks for natural language processing," *CoRR*, vol. abs/1506.07285, 2015. [Online]. Available: http://arxiv.org/abs/1506.07285

[15] P. Vijayaraghavan, I. Sysoev, S. Vosoughi, and D. Roy, "Deepstance at semeval-2016 task 6: Detecting stance in tweets using character and word-level cnns," *CoRR*, vol. abs/1606.05694, 2016. [Online]. Available: http://arxiv.org/abs/1606.05694

[16] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

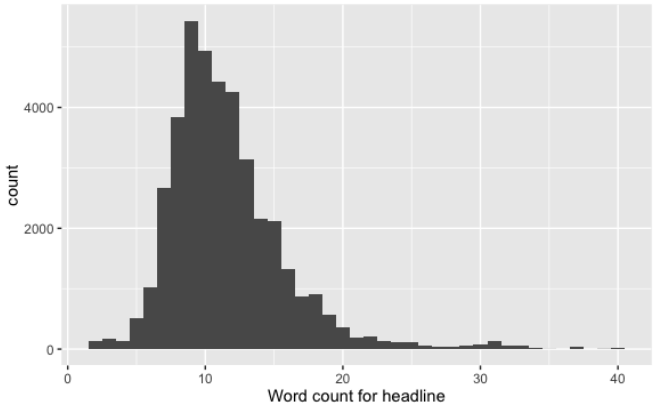
[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[18] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

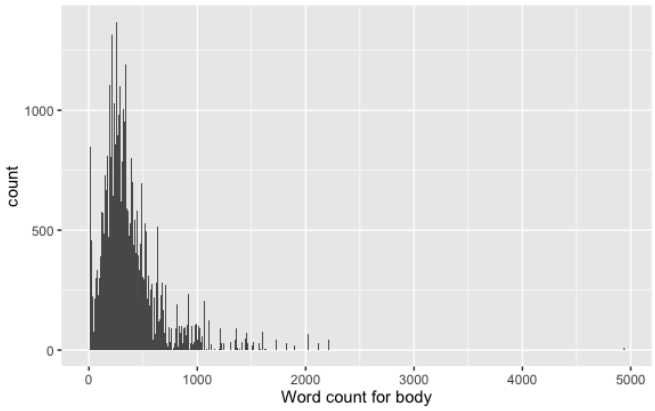
6 Contributions

1. Neel: Implemented 50% of the code, ran 50% of the hyperparameter sweeps, did 50% preprocessing, and wrote the approach section of the report.
2. Shruti: Did a majority of illustrations and plots of the paper. Ran 50% of hyperparameter sweeps, did 50% preprocessing, and helped with 50% of the code.

7 Appendix



(a) Headline Histogram - number of words in train dataset



(b) Body Histogram - number of words histogram in train dataset

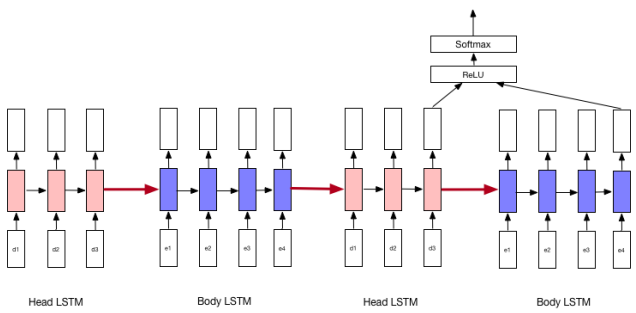


Figure 4: Multi-pass Conditional Encoding Model

LSTM equations

$$\mathbf{H} = [\mathbf{d}_t \quad \mathbf{h}_{t-1}] \tag{18}$$
$$\mathbf{i}_t = \sigma(\mathbf{W}^i\mathbf{H} + \mathbf{b}^i) \tag{19}$$
$$\mathbf{f}_t = \sigma(\mathbf{W}^f\mathbf{H} + \mathbf{b}^f) \tag{20}$$
$$\mathbf{o}_t = \sigma(\mathbf{W}^o\mathbf{H} + \mathbf{b}^o) \tag{21}$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}^t \tanh(\mathbf{W}^c\mathbf{H} + \mathbf{b}^c) \tag{22}$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{23}$$

List of hyperparameters

1. Maximum headline length: 20
2. Maximum body length: 80
3. Size of each batch: 32
4. Size of word embeddings: 300
5. Size of the hidden layer (we use the same size for LSTM cell state): 128
6. Initial learning rate: 0.001
7. Keep probability: 0.8 (Dropout: 0.2)
8. Number of epochs: 10
9. L2 regularization constant: 0

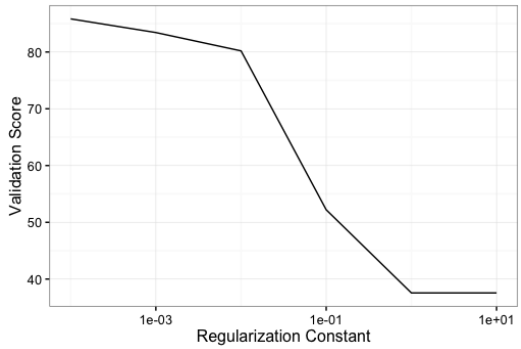


Figure 5: Validation Score vs Body Length for ARM and ARM2 models

Headline	Body	Stance	Prediction
sugarhill gang's big bank hank dead at 57	”(cnn) – a member of the sugarhill gang, whose pioneering hit ””rapper's delight”” brought hip hop to mainstream audiences 35 years ago, died tuesday of complications from cancer... ””big bank hank,”” whose real name was henry jackson, died early tuesday in englewood, new jersey, according to david mallie, who manages the two surviving sugarhill gang members. the new york native was 57. a beefy, boisterous presence onstage, hank handled vocals in the early to middle portion of ””rapper's delight,”” which despite its extended length – one version was more than 14 minutes long – became the first rap song to reach the top 40 on the u.s. billboard charts. jackson traded rhymes with bandmates ””wonder mike”” wright and guy ””master gee”” o'brien and spoke some of the song's catchiest lines, including ””ho-tel, mo-tel, holiday inn/if your girl starts acting up, then you take her friend.””...	Agree	Discuss
hoax alert: father o'neal, who has met god and thinks she's a woman, is made up (just like god herself)	a catholic priest from masschussetts, who was reported dead for close to an hour before medics were able to revive him, has made a shocking revelation upon his return to life...father john micheal o'neal, who was rushed to massachusetts general hospital on thursday, january 29, 2015, was revived through the aid of a high-tech machine called lucas 2, which kept the blood flowing to his brain as doctors managed to unblock vital arteries and return his heart to a normal rhythm, after a major heart attack...	Disagree	Discuss
christian bale to play steve jobs in upcoming biopic	”welsh actor christian bale has withdrawn from the role of steve jobs in a forthcoming biopic of the apple entrepreneur. according to the hollywood reporter, bale decided he ””was not right for the part after much deliberation and conflicting feelings.”” the as-yet-untitled biopic, which is coming out of sony's studios, has been written by the social network writer aaron sorkin and will be directed by danny boyle. in mid-october, sorkin announced bale's involvement in a tv interview, explaining that the batman actor didn't even need to audition. sorkin explained he had high hopes for bale: ””there isn't a scene or a frame that he's not in. so it's an extremely difficult part and he is gonna crush it.”” the sony film has been in progress since 2012 and was due to start filming this winter, after former director david fincher dropped out of the project in april over a pay dispute. boyle is in la this week to meet with the potential cast. seth rogen is said to be in discussions to play jobs' colleague and apple co-founder steve wozniak, but no official announcments have been made.”	Disagree	Agree

Table 4: Error analysis examples

	agree	disagree	discuss	unrelated
agree	118	3	556	85
disagree	14	3	130	15
discuss	58	5	1527	210
unrelated	5	1	98	6794
Score: 3538.0 out of 4448.5 (79.53%)				

Figure 6: Confusion matrix for FNC baseline