Stance Detection in News

Neha Iyer iyermohan@iisc.ac.in
Rupali Dakhane rupalid@iisc.ac.in
Abhishek Pathapati pathapatia@iisc.ac.in

Abstract

Stance Detection is a task that involves estimating the stance of a body text from a news article relative to a headline. There are four categories into which the stance must be classified: agrees, disagrees, discusses and unrelated. We explore different models for stance detection in news articles. We have applied the concepts of neural attention and conditional encoding to long short-term memory networks which is the state-of-the-art and combined it with XGBoost model. We achieved the best performance with this combined model that outperformed the baseline score on the test dataset.

1 Introduction

With the advent of social media such as Twitter, Facebook, news articles or stories get published at real time and are available to the global audience without undergoing any prior verification. News organizations often tend to pick up such articles bypassing the fact-checking process in order to gain readers attention. Also in a research poll, 64% of US adults said that made-up news has caused a great deal of confusion about the facts of current events. In order to combat the fake news problem, an on-line competition Fake News Challenge was organized to explore how machine learning and natural language processing techniques can be leveraged.

The fully automated Fake News detection is in itself a difficult task owning to the unavailability of labelled training data as fake or real, unavailability of dataset in the public domain, extremely diverse and unstructured training data format. However the process can be broken down into multiple stages. A first step is to understand what other news organizations are saying about a given claim or headline. Automation of this step is called stance detection. Given a headline as input the stance detector would all articles that may agree, disagree or discuss the headline. Using this information fact-checkers can use human judgment and reasoning to evaluate the validity of the headline. A solution of the stance detector can further be fed to a truth labelling system that can tentatively label a claim as true or false depending on the stances of other news organizations about the topic.

Stance detection in general means estimating the relative perspective (stance) of two pieces of text about a claim or topic. The version of stance detection we consider as our problem statement is to identify how a piece of news body is related to the news headline.

Our task is to analyze the effectiveness of different neural network models that given an input pair of headline and body text correctly classifies the stance into one of the four categories of agree, disagree, discuss or unrelated.

2 Related Works

The domain of stance detection and textual entailment are closely related. There has been lot of works on analyzing the relationship between two sentences. Example, natural language inference attempts to infer whether a given hypothesis follows from a premise. The SemEval-2016 Task 6 was directed towards automatically determining from the tweet whether the tweeter is in favor of the given target or against the target. For example given the target legalization of abortion, the tweet A foetus has rights too! Make your voice heard. is likely to be against the target. A majority of the work in natural language sentence matching involves constructing a dense vector representation for the sentences being compared and

computing the relationship between the representations. The dense vector representation are usually constructed by using neural network models with word embeddings as the input. (Augenstein et al., 2016) uses a conditional LSTM encoding which builds a representation of the tweet conditioned on the target. Such an encoding outperforms encoding the target and the tweet independently as it allows the context of target to influence the encoding of the tweet. The performance is further improved by using bidirectional encoding where two encodings are constructed for target and tweet thus both left and right context of the words are taken into account. (Hermann et al., 2015) proposed three LSTM based neural networks. First model is a Deep LSTM model that is able to embed long sequences into a vector representation which contains enough information for further processing. Given the embedded document and query the network predicts which token in the document answers the query. The problem of fixed width hidden vector of Deep LSTM is solved by incorporating attention mechanism to develop Attentive Reader model. The model encodes the 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 Impatient Reader model rereads the entire document as each query token is read. The result is an attention mechanism that allows the model to recurrently accumulate information from the document as it sees each query token, ultimately outputting a final joint document query representation for the answer prediction. (Vijayaraghavan et al., 2016) Apart from RNN based neural networks, convolutional based neural networks have also been explored in the field of stance detection. Detecting Stance in Tweets Using Character and Word-Level CNNs uses a combination of word-level and character-level CNNs for stance detection in tweets. It was observed that character-level model can outperform word-level model when sufficient train data is available. However to overcome lack of data, word2vec-based data augmentation technique is used to generate data.

3 Dataset

FNC provided train and test data sets which was derived from the Emergent Dataset. The input consists of a headline and a body text pair - either from the same news article or from two different articles. There are total 49,972 headline-body pairs in the train set. Each pair is labeled as either one of the four stances agree, disagree, discuss or unrelated. The test set consists of 25,413 headline-body pairs. The distribution of stance labels is highly skewed in both train and test set as shown in Table 1.

Table 1: Distribution of stance labels in Dataset

	Train (%)	Test (%)
Agree	1.6	7.4
Disagree	7.3	2.7
Discuss	17	20
Unrelated	73	72

4 Evaluation Metric

A weighted two-level scoring system is used to evaluate the models. On correctly classifying a headline-body pair as unrelated, a score of 0.25 will be added to the evaluation score. A score of 0.25 is awarded for an incorrect classification if the true label is either agree, disagree or discuss and the model predicted one of these labels. A score of 1.0 is awarded for correctly classifying the pair into one of the three categories of agree, disagree or discuss. The final score is the percentage of highest achievable score on the test set.

5 Official Baseline Model

FNC provided a baseline model that uses Gradient boosting coupled with hand-coded features such how many times an n-gram of headline appears in body text, presence of refuting words like fake, fraud , hoax" in the headline and body text. Using these hand-crafted features and gradient boosting classifier the baseline model achieved a weighted score of 79.53% on a held-out set with a 10-fold cross validation and 75.20% on the competition test set.

6 Approach

6.1 Data Preprocessing

As the headlines and the corresponding body text are read and concatenated accoding to the body-

id provided. The data is preprocessed by removal of punctuations and stop words. Along with tokenization we also perform stemming to obtain the root words. After tokenization, each unique token is mapped to an integer value so each headline-body pair is now represented by a list of integer values. The unique tokens in the text are then mapped to 50 dimensional GloVe (Pennington et al., 2014) word embeddings trained on 6 billion tokens of vocabulary size 400K on Wikipedia 2014 corpus. The unknown words if any, encountered in the test sequence are simply dropped. The headlines and body-texts were found to be of varying lengths. So the headline and body sequence length was set as a tuning parameter. The sequences shorter than the defined length were zero padded and the sequences larger than the maximum length were truncated.

6.2 1-D Word-Level CNN

This model applies a one-dimensional convolutional neural net (CNN) on the headline and body text. The output of this CNNs is then sent to a MLP with 4-class output.

Vector Embeddings: The news headlines and article bodies are represented at word level. We have represented them using Google News pretrained vectors to effectively represent the given dataset. The dimension of these embeddings used is 300.

Convolutional layers: Convolutional Neural Networks allow for efficient, effective parallel computation. The Headline and body vectors are seperately passed through convolution layers. There are 5 convolution layers coupled with the pooling layers. The number of hidden nodes in these layers are 256, 256, 512, 512, and 768. The output vectors obtained for both headline and body have dimension 768. These vectors are concatenated to give a vector of dimension 1536 given as input to the following dense layers.The model was regularized using dropout (p = 0.5) in all Convolutional layers.

Dense layers: The output of this CNN is then sent to an multi-layer perceptron with three dense layers having number of hidden nodes as 1024. Finally a softmax layer is added which gives a 4-class output – agree, disagree, discuss, and unrelated.

6.3 Xgboost Tree model

This model takes as input a few text-based features derived from the headline and body of an article. The features obtained for headline and body are then fed into Gradient Boosted Trees to predict the relation between the headline and the body. An efficient implementation XGBoost model is used to predict these relations. The labels are encoded into numeric targets and the texts of headline and body are obtained after preprocessing. Below are the features derived for this model.

1. Basic Count Features: This feature counts how many times a gram appears in the headline, how many unique grams there are in the headline, and the ratio between the two. The same statistics are computed for the body text, too. It then calculates how many grams in the headline also appear in the body text, and a normalized version of this overlapping count by the number of grams in the headline.

2. TF-IDF Features: This feature contains sparse vector representations of the headline and body by calculating tfi-df vector. Cosine similarity between the headline vector and the body vector is also computed.

3. SVD Features: Singular-Value Decomposition is applied to Tf-Idf vectors to obtain a compact, dense vector representation of the headline and body respectively and is included as a feature.

4. Word2Vec Features: Google News pretrained vectors of dimension 300 are used as feature to represent headline and body words.

5. Sentiment Features: We have used sentiment analyzer in the NLTK package to assign a sentiment polarity score to the headline and body separately.

6.4 Conditional Encoding with Attention (ACE)

Conditional Encoding: In order to compute relationship between headline and body we need to encode both of the sequences as dense vectors. LSTMs, due to their ability to store information for a long period of time, can readily be used to independently encode headline and body as dense vectors and taking their concatenation as input to an MLP classifier. Figure 1 shows a high-level overview of this model. The headline is read by an left LSTM and a second LSTM (right) with different parameters the body sequence (right), but

its first state is initialized with the last state of the previous LSTM (c5 in the example), i.e. it is conditioned on the representation that the first LSTM built for the headline.

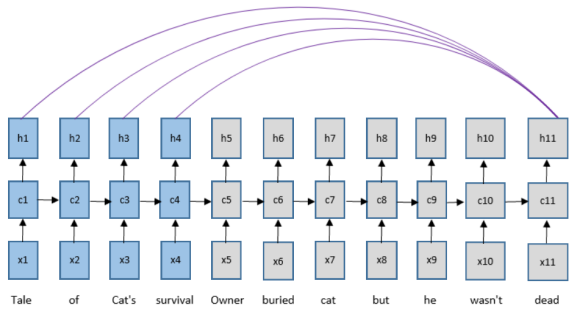

Figure 1: Conditional encoding with attention()

Attention: In order to avoid encoding the input headline into a fixed length representation, the intermediate vectors are output while reading the headline and train the second LSTM to learn to pay selective attention to these inputs and relate them to the out body text sequence. The attention mechanism implemented in (Rocktäschel et al., 2015) was adapted for stance detection and was implemented in Tensorflow 0.12.1. Let $Y \epsilon R^{k \times L}$ be a matrix consisting of output vectors $[h_1 ... h_L]$ that the first LSTM produced when reading the L words of the headline, where k is a hyper parameter denoting the size of embeddings and hidden layers. Furthermore, let $e_L \epsilon R^L$ be a vector of 1s and h_N be the last output vector after the headline and body were processed by the two LSTMs respectively. The attention mechanism will produce a vector α of attention weights and a weighted representation r of the headline via

$$M = tanh(W^Y Y + W^h h_N + e_L) \quad (1)$$

$$\alpha = softmax(W^T M) \quad (2)$$

$$r = Y\alpha^T \quad (3)$$

where W^y, $W^h \epsilon R^k$ are trained projection matrices, $w \epsilon R^k$ is a trained parameter vector and w^T denotes its transpose. The final sentence-pair representation is obtained from a non-linear combination of the attention weighted representation r of the headline and the last output vector h_N using

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

7 Experiments and Results

7.1 Data Split

Prior to model training, the train data was split into training and development sets randomly. The split was defined such that 80% of the examples were in the training set and 20% of the examples were in development set. The test set with labeled stances was provided by FNC. All experiments were performed by minimizing the mean cross-entropy loss on the training set and the performance was monitored on the development set in terms of competition score and category-level F1 measure. The final set of parameters were selected by analyzing the performance on development set. The final model was evaluated on the test set by capturing the competition score and F1 score for each output category.

7.2 Parameter tuning experiments

The parameter tuning for each model was performed on the validation data set. Due to the high computational cost involved with grid search, only a small subset of parameters were considered for tuning. In case of ACE model the body text truncation length was found to be a significant hyper parameter. The final parameters values used to train the models are as shown in Table 3.

7.3 Body length truncation and its benefits

As observed from Fig 5 the ACE model performs comparably well with body length values of 50 and 300. But the performance seems to degrade for length values 100 and 200. This demonstrates that attention with conditional encoding mechanism performs well for either short or long sequences. Our interpretation of these results is that the beginning and end of articles seem to hold meaningful information relevant to predict the stance label and attention mechanism is able to extract these information. Intermediate body lengths seems to add noise input which do not contribute to improving the training loss.

7.4 Overall Test Results

- *Individual model performance analysis:* Following the parameter tuning process, the final set of parameters as indicated in Table 3 is used for prediction on the test set provided by FNC. The performance of the three models described above was evaluated on the test data and the results obtained are as shown in

Table 2: Test performance of different models

			F1 score			
Model	**Competition score**	**Train Loss**	**Agree**	**Disagree**	**Discuss**	**Unrelated**
Baseline	0.752	NA	0.15	0.02	0.70	0.96
CNN	0.545	0.017	0.282	0.091	0.473	0.833
ACE	0.569	0.009	0.312	0.098	0.53	0.86
Xgboost	0.731	0.021	0.52	0.001	0.76	0.98
CNN + Xgboost	0.723	NA	0.54	0.04	0.76	0.98
ACE + Xgboost	**0.773**	**NA**	**0.59**	**0.061**	**0.79**	**0.987**

Table 3: Final model parameters

Parameters	**CNN**	**ACE**	**Xgboost**
learning rate	0.0002	0.001	0.3
hidden layer size	1024	100	NA
attention length	NA	15	NA
body max length	600	300	NA
batch size	NA	128	NA
drop out	0.5	0.5	NA
embedding size	300	50	300
epochs	65	55	500
hidden layers	3	2	NA
lambda	NA	NA	1
gamma	NA	NA	0
max_depth	NA	NA	6

Table 2. On the basis of individual model performances, XGBoost model seems to perform the best. The robustness of the model is due to its hand-picked features that allows the model to learn the complex mapping between the features and stance labels with relatively small training set. On the basis of individual class level performance, each model performs well for "unrelated" label. However both the neural models seem to outperform F1-score of XGBoost for "disagree" class. It shows are neural models in particular attention based models are good at classifying rare labels.

- *Combined model performance analysis:* Out of the combined models results as per Fig 2, the ACE + XGBoost combination outperforms the competition baseline of 75.20% by achieving a score of 77.3% on the test data. The robustness of XGBoost over smaller training set combined with attention mechanism applied over long article body sequences gives the best performance.

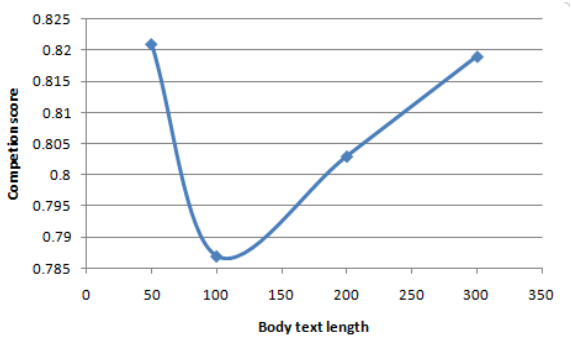

Figure 2: Body length parameter tuning

8 Conclusion

In this project, we explored three models for stance detection in news articles and we managed to achieve a score of 77.3% beating the score achieved by baseline model. This score was achieved by our best model Conditional Encoding LSTM with Attention. The gap between the scores can be further increased by carefully tuning the model and further processing.

9 Future Work

Following directions can be explored to improve upon the model performances.

1. Parameters exploration: The models can be retrained with different number of epochs and in ACE model the attention window can be increased from current attention window size(15) to understand the full benefits of attention. We could also choose higher dimension word embeddings.

2. Model Extensions: We could explore bidirectional LSTMs, especially since end of articles may contain relevant conclusions. We could also explore word level attention in the model.

3. Other extensions: To handle class imbalance, loss function with larger weights to rare classes can be used to achieve better score. A two part model can also be implemented which first predicts the stance as related or unrelated and further classifies related inputs into agree, disagree or discuss.

References

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. *arXiv preprint arXiv:1606.05464* .

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. MIT Press, Cambridge, MA, USA, NIPS'15, pages 1693–1701. http://dl.acm.org/citation.cfm?id=2969239.2969428.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664* .

Prashanth Vijayaraghavan, Ivan Sysoev, Soroush Vosoughi, and Deb Roy. 2016. Deepstance at semeval-2016 task 6: detecting stance in tweets using character and word-level cnns. *arXiv preprint arXiv:1606.05694* .

A Training loss for ACE model

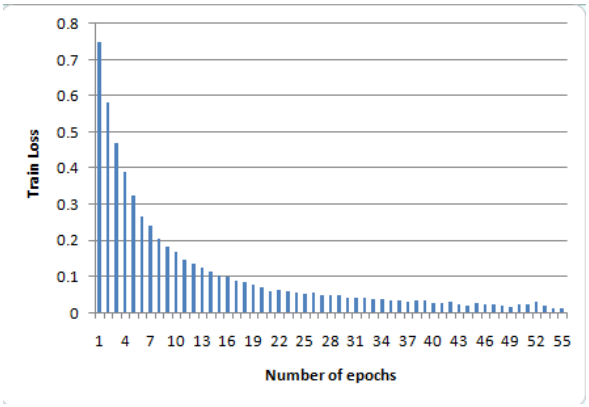

Figure 3: Train loss versus epoch

B Validation score for ACE model

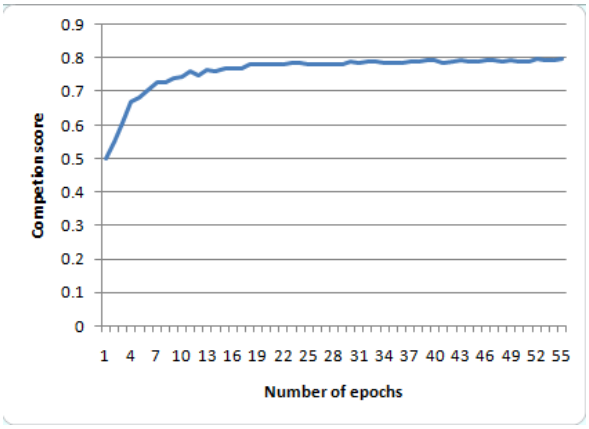

Figure 4: Validation score versus epoch

C Training loss for XGBoost model

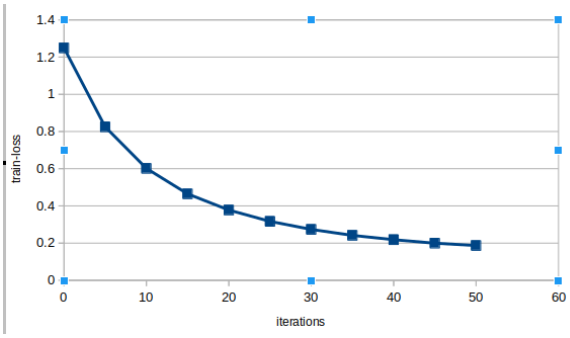

Figure 5: Train loss versus epoch