Fake News, Real Consequences: Recruiting Neural Networks for the Fight Against Fake News

Richard Davis
Graduate School of Education
Stanford University
Stanford, CA 94305
rldavis@stanford.edu

Chris Proctor
Graduate School of Education
Stanford University
Stanford, CA 94305
cproctor@stanford.edu

Abstract

The Fake News Challenge (FNC-1) is a public competition that aims to find automatic methods for detecting fake news. The dataset for the challenge consists of headline-body pairs, with the objective being to classify the pairs as unrelated, agreeing, disagreeing, or discussing. We developed four neural network models for FNC-1, two using a feed-forward architecture and two using a recurrent architecture. After running over 100 experiments across different model architectures and hyperparameters, we determined that the best model was a bag-of-words followed by a three-layer multi-layer perceptron (BoW MLP). The BoW MLP model achieved categorical test-set accuracy of 93%, and on the competition-specific FNC-1 metric it achieved a test-set score of 89%—a full ten percentage points above the published baseline.

1 Introduction

Fake news, defined as a made-up story with an intention to deceive [1], has been widely cited as a contributing factor to the outcome of the 2016 United States presidential election. While Mark Zuckerberg, Facebook's CEO, made a public statement [15] denying that Facebook had an effect on the outcome of the election, Facebook and other online media outlets have begun to develop strategies for identifying fake news and mitigating its spread. Zuckerberg admitted identifying fake news is difficult, writing, "This is an area where I believe we must proceed very carefully though. Identifying the 'truth' is complicated."

The Fake News Challenge (FNC-1) [3] was organized in response to this controversy. FNC-1 is a public competition with the stated goal of using artificial intelligence to automatically identify fake news. FNC-1 is focused on an important part of identifying fake news: stance detection. Stance detection involves estimating the relative perspective (or stance) of two pieces of text relative to a topic, claim or issue. The goal of FNC-1 is to estimate the stance of an article relative to a headline. More specifically, the headline may agree with, disagree with, discuss, or be unrelated to the body text of the article.

2 Background

The primary task of FNC-1 is stance detection between headline-article pairs. Stance detection is the act of identifying the relationship between two pieces of text—in FNC-1, the possible relationships are agree, disagree, discuss, or unrelated. This version of stance detection extends the work of Ferreira & Vlachos [8].

Stance detection has been an active area of research for the past decade. There is prior work on detecting stance of participants in online debates [5, 9], student essays [2], and congressional debate

[13]. However, none of these models used neural networks, relying instead of hand-engineered features. Considering the success of neural networks in other, related areas (e.g., natural-language inference [14, 6, 10]), we hypothesized that neural networks should also perform well on stance detection.

3 Approach

3.1 Data

The data, derived from the Emergent Dataset [8], is supplied by the organizers of FNC-1. The data is composed of 49,972 pairs of headline and body text each with a corresponding class label of either unrelated, agree, disagree, or discuss. We split the data into 70% train, 15% validation, and 15% test. The test data was separated before training any models and only used for final model evaluation.

The data is composed of 1683 articles and 49972 headlines. Because the dataset is provided as a list of headlines, a list of articles, and a many-to-many mapping between them, nonstandard methods of splitting the data risk having information leaking between the train, validation, and test datasets. To avoid this, we used a dataset-splitting function provided by FNC-1.

3.2 Objective

Given an input of a headline and body text, the FNC-1 task is to classify the stance of the body text first in terms of whether or not the two are related. Then, if they are related, classify the body as agreeing with the article, disagreeing, or discussing the article without taking a stance. The FNC-1 competition metric is a weighted accuracy score: 25% for correctly classifying a headline/article pair as related or unrelated, and 75% for correctly classifying related headlines as agreeing, disagreeing, or discussing. As our goal was to perform well in FNC-1, we report results in terms of accuracy at classifying relatedness and stance rather than using a metric such as F1.

3.3 Baseline

FNC-1 provides a baseline model consisting of a gradient-boosting classifier over n-gram co-occurrence between the headline and article, as well as several hand-tuned feature sets such as appearance of a provided set of highly-polarized words (e.g. ”fraud” and ”hoax”). TODO describe performance of baseline.

3.4 Model architectures

After some initial experimentation, four neural net architectures gave good early results so we decided to focus on improving these models. These models are described below.

The concatenated multilayer perceptron (Concat MLP) model (Figure 1) truncates and tokenizes the headline and article, keeping a fixed-size vocabulary of the most frequently-occurring tokens. Then the headline and article token sequences are each passed through separate embedding matrices. (We tried starting with the GlOVE pretrained embeddings [11], but found better results without using pretrained embeddings.) The output of each embedding matrix is concatenated and then passed through several dense hidden layers with dropout [12]. Finally, we use separate softmax layers to make two classifications: a binary decision of whether or not the article and headline are related, and then a classification of the headline and article as agreeing, disagreeing, discussing (no stance), or unrelated. For training, our loss function was a weighted combination of cross-entropy losses for each classification. We explored various weightings, but found 25% for the binary relatedness decision and 75% for the stance classification (which match the FNC-1 scoring weights) to work well.

The bag-of-words multilayer perceptron (BoW MLP) model (Figure 2) again tokenizes the headline and article using a fixed-size vocabulary of the most commonly-occurring tokens. These tokens are then reduced to a bag-of-words vector (whose dimension is the size of the vocabulary), indicating the distribution of tokens but not preserving any information about ordering or structure. We tried several different summation strategies, including a binary flag indicating that a token appears, token

count, token frequency (the percentage of tokens in the sample which are this token), and TF-IDF (which takes into account the relative frequency of the token in the corpus). The bag-of-words vector is then passed through several hidden layers with dropout. Finally, parallel softmax layers and loss functions are used in the same manner as the Concat MLP model.

The dual GRU (Dual GRU) model (Figure 3) truncates, zero-pads, and tokenizes the headline and article separately and passes each into an embedding layer. Each embedding is fed into a recursive neural network (RNN) based on gated recurrence units (GRU). In some configurations, both RNNs shared the same weights matrix. The outputs of the RNNs are concatenated, passed through a single dense layer with dropout, and then passed through a softmax layer to yield a stance classification. None of our efforts to tune this model yielded results comparable with the other models, so we abandoned it before implementing the parallel classifications present in the other models.

Finally, the bi-directional concatenated stacked LSTM (Bi-dir LSTM) model (Figure 4) truncates, zero-pads, and tokenizes the headline and article. Then the article is reversed and concatenated to the end of the headline, separated by an out-of-vocabulary token. This vector is then passed through a stack of bi-directional LSTMs. (Our intuition in this design was that the beginnings of the headline and of the article are likely the most salient; by putting them near each other we could enable the RNN to more easily learn relationships between the two.) As with the other models, we then used parallel softmax layers to perform both classifications.

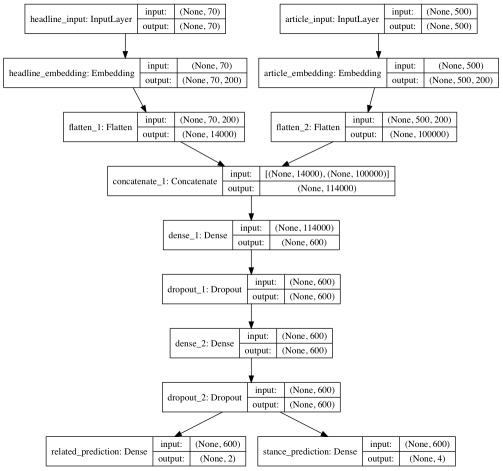

Figure 1: Concatenated multilayer perceptron (Concat MLP)

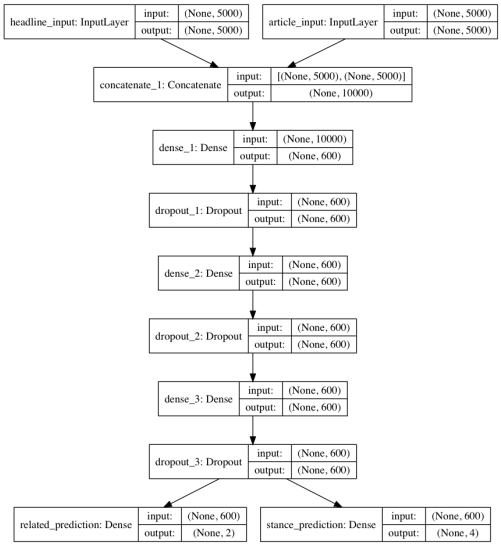

Figure 2: Bag-of-words multi-layer perceptron (BoW MLP)

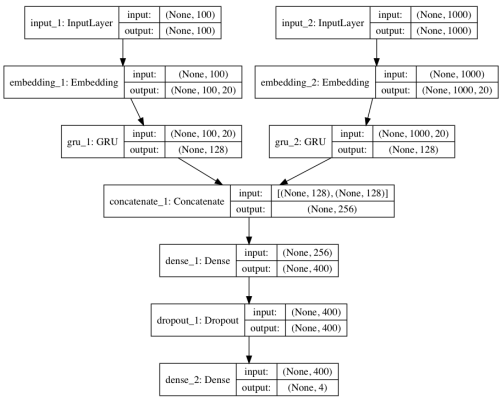

Figure 3: Dual GRU

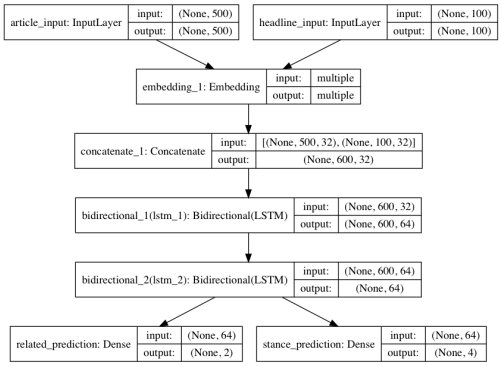

Figure 4: Bi-directional, concatenated, stacked LSTM (Bi-dir MSTL)

3.5 Model evaluation

The these model architectures have many possible variations arising from changing hyperparameters such as vocabulary size, hidden layer dimensionality, and dropout rate, as well as parameters controlling the structure of the model such as the number of hidden layers to stack together. In order to methodically explore the space of possible models, we developed a framework allowing us to specify a model configuration as a JSON object 5. The `ModelConfig` class handles reading and writing model configurations, as well as initiating training and evaluation and logging test results. We then developed a `ModelPipeline` class which accepts a configuration and then has methods to preprocess raw input data, build, fit, train, and evaluate the model. We implemented a subclass of `ModelPipeline` for each of our four model architectures. We used Keras [7] to build the models, dropping down to TensorFlow [4] for several extensions.

This framework allowed us to engage in a highly productive exploration of possible models. Our procedure was to meet and form hypotheses about how the models could be improved, and then design a series of experiments to test these hypotheses. For example, we might try a range of different vocabulary sizes. When we suspected important interactions between factors (such as dropout rate and number of hidden layers), we tried a grid of possible values. We used a script to specify configurations for these parameters, sequentially changing values in a base configuration. We checked these configurations into version control and then were able to distribute the tests to four different machines. After gathering the test results, we used a command-line tool to filter, sort, and display test results that informed our next set of hypotheses. The end result of this cycle was a sort of stochastic gradient descent through the space of possible models, optimizing one or two dimensions at a time.

```
{
    'model_class': 'BowMLP',
    'model_module': 'bow_mlp',
    'hidden_layers': [
        [600, 'relu', 0.35],
        [600, 'relu', 0.35]
    ],
    'matrix_mode': 'freq',
    'vocabulary_dim': 3000
    'compile': {
        'loss': {
            'related_prediction': 'binary_crossentropy',
            'stance_prediction': 'categorical_crossentropy'
        },
        'loss_weights': {
            'related_prediction': 0.25,
            'stance_prediction': 0.75
        },
        'metrics': ['accuracy'],
        'optimizer': 'nadam'
    },
    'fit': {
        'epochs': 15
    },
}
```

Figure 5: An example model configuration file

4 Experiments

4.1 Model iteration

In the course of developing our models, we ran over 100 experiments. We were able to improve all of our models, but none so successfully as BoW MLP. Through the process of hyperparameter tuning on the BoW MLP, we were able to raise our score on the FNC-1 metric from 80% to 89%!

In this section we describe the process of developing and tuning the BoW MLP model. In each of the following experiments, we generated a new split of training and validation data (holding out the test set). While cross-validation or bootstrap would have provided more reliable results, this was a compromise to allow us to conduct more tests.

We began by exploring different operations for summing word tokens into a single bag-of-words vector, and found that word frequency was substantially more effective than the other possibilities we tried 6. Next, we considered the size of vocabulary to use. The graph 7 shows results of using a vocabulary of 1000, 3000, and 5000 words. We also tried higher values, but saw little improvement. The next set of experiments considered different numbers of hidden layers and dropout values. We expected an interaction between these factors, so we conducted a two-dimensional search of one to eight layers with dropout set to 0.5 8, 0.35 9, and 10. We found several configurations yielding similar results, and prioritized those having fewer training parameters. Finally, we explored changing the dimensionality of each hidden layer 11.

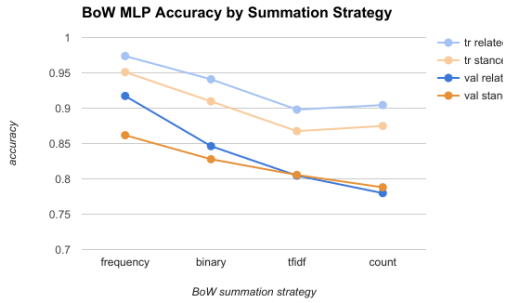

Figure 6: Tuning BoW summation strategy.

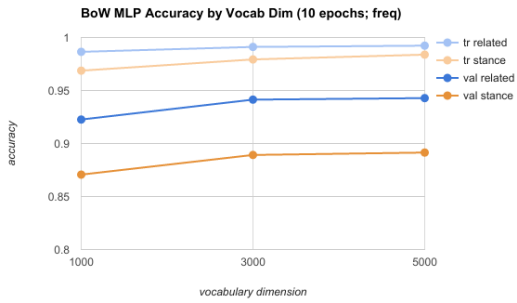

Figure 7: Tuning vocabulary size.

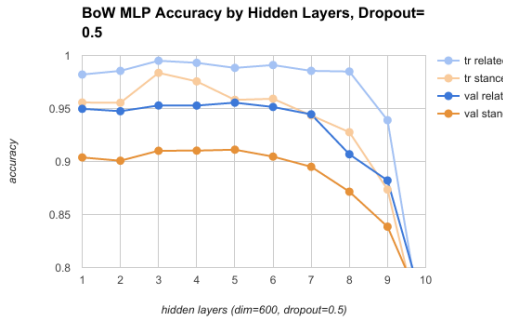

Figure 8: Tuning hidden layers and dropout.

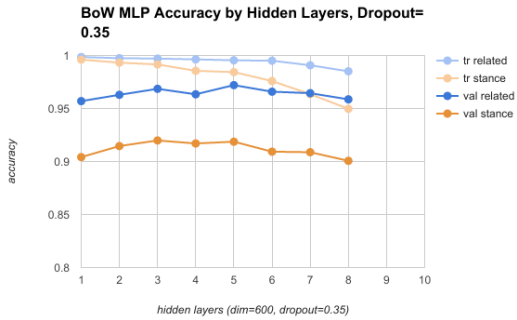

Figure 9: Tuning hidden layers and dropout.

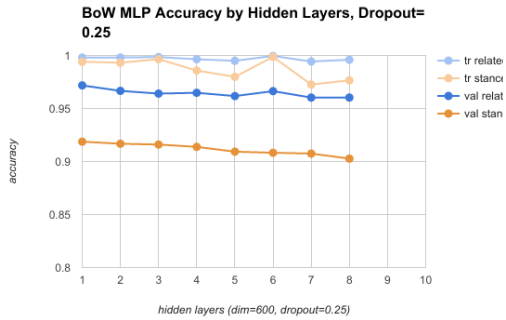

Figure 10: Tuning hidden layers and dropout.

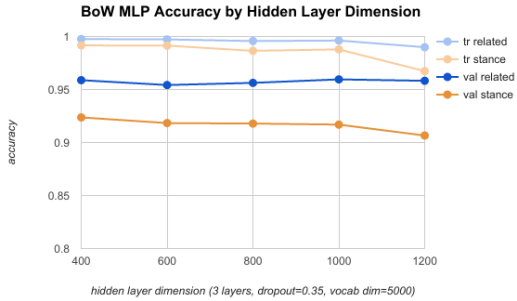

Figure 11: Tuning hidden layer size.

4.2 Results

After tuning the hyperparameters, we evaluated our three best models on the test set. The test set was separated at the start of the project, before training any models, and only used at the conclusion to evaluate final performance. Per model performance is as follows:

	Validation set			Test set		
model	related	stance	FNC-1	related	stance	FNC-1
Baseline	0.95	0.87	0.79	0.95	0.87	0.79
BoW MLP	**0.97**	**0.93**	**0.89**	**0.97**	**0.92**	**0.89**
Bi-Dir LSTM	0.79	0.73	0.68	0.78	0.72	0.69
Concat MLP	0.74	0.75	0.73	0.72	0.50	0.50

Table 1: Results for best models using each architecture

The best-performing model was the bag-of-words multi-layer perceptron (BoW MLP) 2. On the validation set, this model was able to correctly classify headline-body pairs as related or unrelated with 97% accuracy, and when the headline and body were related, it was able to correctly identify the stance (agrees, disagrees, discusses) with 93% accuracy. On the FNC-1 competition metric, it beat the baseline on by a full 10%. The set of hyperparameters for this model included:

- Vocabulary size of 5000
- Three 600-dimensional hidden layers, each with 35% dropout
- Stance loss function weighted 75%, relatedness loss function weighted
- Frequency BoW summation strategy

5 Conclusion

Using a tuned BoW MLP, we were able to beat the baseline on the FNC-1 score by 10%. Considering that the baseline was already achieving an FNC-1 score of 79%, adding an additional 10% is a significant improvement.

None of the recurrent neural networks were able to beat the baseline. During hyperparameter tuning, we tried multiple versions of both recurrent models, but to no avail. This is a surprising result given the success that recurrent neural networks have had in achieving state-of-the-art results in other, similar domains (e.g., natural-language inference).

There are a number of reasons that the recurrent models might be under-performing. First, there may not be enough data for a recurrent neural network to be able to generalize. In fact, we did see that the RNNs would quickly learn to fit the training data with near-perfect accuracy, while unfortunately never achieving high scores on the validation data. Second, even though GRU's and LSTM's remember information for longer than vanilla RNN's, it is possible that they were forgetting important information that the BoW model was capturing. Finally, a more complicated architecture that employed attention or conditional encoding may have helped improve the accuracy. Our next steps include implementing these models.

We also plan to explore new ways of improving on the performance of our BoW MLP model. First, we plan to implement a summation strategy which takes into account a dependency parse of the headline and sentences in the article. We hypothesize that weighting word token frequencies by their inverse depth in the dependency tree would help the model to distinguish between when words are central to the meaning and when they are more peripheral. We conducted some initial experiments preprocessing text with Stanford CoreNLP, but have not yet tested a bag-of-words summation strategy using dependency parsing. Second, we plan to include hand-tuned features along with the words as input to the model.

Our strategy of parameterizing the entire model pipeline proved very effective at allowing us to run experiments optimizing the model. We plan to generalize our framework for future natural language processing research. One area which is currently not well-supported is capturing training state generated during preprocessing. Specifically, Keras's tokenizer maintains an internal representation of the corpus, which it uses to select words to be included in the vocabulary and for word frequency statistics. Therefore when saving model weights after training, it is necessary to persist the tokenizer's state as well. There is currently no mechanism in Keras for storing the tokenizer's state.

Although it was disappointing not to achieve high accuracy with the more exotic neural architectures, we achieved our primary goal of significantly improving on the baseline FNC-1 score. We accomplished this by identifying a promising, simple model (BoW MLP) and leveraging our experimental framework to tune hyperparameters in roughly 100 different experiments. By tuning hyperparameters, we were able to raise the FNC-1 score of our BoW MLP model from 80% to 89%, outperforming the baseline by a full 10%. These results are a promising step in the direction of a fully-automated system for detecting fake news.

Acknowledgments

Thanks to our project mentor, Rob Voight, for his thoughts and guidance, and to the tireless teachers and staff of cs224n, for putting together such an ambitious and rewarding class.

References

[1] As fake news spreads lies, more readers shrug at the truth. In *https://www.nytimes.com/2016/12/06/us/ fake-news-partisan-republican-democrat.html*.

[2] Automated classification of stance in student essays: An approach using stance target information and the wikipedia link-based measure - semantic scholar.

[3] http://www.fakenewschallenge.org/.

[4] ABADI, M., AGARWAL, A., BARHAM, P., BREVDO, E., CHEN, Z., CITRO, C., CORRADO, G. S., DAVIS, A., DEAN, J., DEVIN, M., GHEMAWAT, S., GOODFELLOW, I., HARP, A., IRVING, G., ISARD, M., JIA, Y., JOZEFOWICZ, R., KAISER, L., KUDLUR, M., LEVENBERG, J., MANÉ, D., MONGA, R., MOORE, S., MURRAY, D., OLAH, C., SCHUSTER, M., SHLENS, J., STEINER, B., SUTSKEVER, I., TALWAR, K., TUCKER, P., VANHOUCKE, V., VASUDEVAN, V., VIÉGAS, F., VINYALS, O., WARDEN, P., WATTENBERG, M., WICKE, M., YU, Y., AND ZHENG, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[5] ANAND, P., WALKER, M., ABBOTT, R., TREE, J. E. F., BOWMANI, R., AND MINOR, M. Cats rule and dogs drool!: Classifying stance in online debate. In *Proceedings of the 2nd workshop on computational approaches to subjectivity and sentiment analysis*, Association for Computational Linguistics, pp. 1–9.

[6] BOWMAN, S. R., ANGELI, G., POTTS, C., AND MANNING, C. D. A large annotated corpus for learning natural language inference.

[7] CHOLLET, F. Keras. `https://github.com/fchollet/keras`, 2015.

[8] FERREIRA, W., AND VLACHOS, A. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, ACL.

[9] HASAN, K. S., AND NG, V. Stance classification of ideological debates: Data, models, features, and constraints. In *IJCNLP*, pp. 1348–1356.

[10] LIU, Y., SUN, C., LIN, L., AND WANG, X. Learning natural language inference using bidirectional LSTM model and inner-attention.

[11] PENNINGTON, J., SOCHER, R., AND MANNING, C. D. Glove: Global vectors for word representation. In *EMNLP*, vol. 14, pp. 1532–1543.

[12] SRIVASTAVA, N., HINTON, G. E., KRIZHEVSKY, A., SUTSKEVER, I., AND SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting. 1929–1958.

[13] THOMAS, M., PANG, B., AND LEE, L. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, Association for Computational Linguistics, pp. 327–335.

[14] WANG, S., AND JIANG, J. Learning natural language inference with LSTM.

[15] ZUCKERBERG, M. Facebook post. In *https://www.facebook.com/zuck/posts/10103253901916271*.