"The Pope Has a New Baby!" Fake News Detection Using Deep Learning

Samir Bajaj
Stanford University
CS 224N - Winter 2017
samirb@stanford.edu

Abstract

The objective of this project is to build a classifier that can predict whether a piece of news is fake based only its content, thereby approaching the problem from a purely NLP perspective. An important part of the goal is to compare and report the results from multiple different model implementations, and present an analysis of the findings. Several architectures are explored, and a novel design that incorporates an attention-like mechanism in a Convolutional Network is investigated.

1 Introduction

Fake news is increasingly becoming a menace to our society. It is typically generated for commercial interests—to attract viewers and collect advertising revenue. However, people and groups with potentially malicious agendas have been known to initiate fake news in order to influence events and policies around the world. It is also believed that circulation of fake news had material impact on the outcome of the 2016 US Presidential Election [1].

From an NLP perspective, this phenomenon offers an interesting and valuable opportunity to identify patterns that can be coded in a classifier. In our experiments, we will be ignoring all other signals (e.g., the source of the news, whether it was reported online or in print, etc.), and instead focus only the content matter being reported.

2 Data

The data used for this project was drawn from two different sources, both in public domain. Fake news articles were procured from an open Kaggle dataset [2]—this collection is made up of 13,000 articles that span multiple subjects. Authentic news articles (negative examples for the classifier) were extracted from the Signal Media News dataset [3]; reservoir sampling was used to select 50,000 entries uniformly at random from the corpus. The ratio of fake to non-fake examples was determined on the basis of empirical evidence in an attempt to reflect published news in the real-world. Binary $0/1$ labels were assigned to the real/fake news articles, respectively. The collection of 63,000 articles was subsequently shuffled, and split into 60% training, 20% dev/validation, and 20% test sets. The test set was stashed away for the final performance evaluation of the models, and the validation set was used for hyperparameter selection.

The hyperparameters used for the various models are listed in table 1.

3 Related Work

Text classification has a rich research history within the NLP community, and an equally impressive array of practical applications to showcase its importance. See [4] for an introduction and references.

While there exist tools and products to detect *sources* of fake news (e.g., whether a web site publishes misleading news), we will approach this problem as an instance of text classification, using only the content of the article as the source of features. Doing so affords us to focus on NLP-related algorithms, thereby allowing us to explore in depth the performance of a variety of models on a particular task.

4 Models

Several different models were implemented; the following sections describe each in detail. All models use pre-trained 300-dimensional GloVe [5] embeddings, which are not updated during training. Given that the training set is small compared to the magnitude of the corpora used to train GloVe vectors, it makes little sense to update the embeddings as part training the models.

4.1 Logistic Regression

This is a simple, linear model, that was implemented to serve as a baseline for comparison with other, more sophisticated models.

In this model, vectors corresponding to words in each news story are averaged to produce one embedding, which is first run through a linear layer and subsequently input to the sigmoid nonlinearity to predict the label:

$$\hat{y} = \boldsymbol{\sigma}(\boldsymbol{x}W + b) \tag{1}$$

All models used the logistic (sigmoid) cross-entropy loss, which is defined as:

$$J = -y \log \hat{y} - (1 - y) \log(1 - \hat{y}) \tag{2}$$

Figure 1 shows the loss as a function of training epochs.

4.2 Two-layer Feedforward Neural Network

This model also uses a fixed-size input $\boldsymbol{x} \in \mathbb{R}^{1\times 300}$, obtained as the average of all vectors corresponding to the words in the news story. It is inspired by the simple design explored in [6].

$$\boldsymbol{h} = \text{ReLU}(\boldsymbol{x}W + b_1) \tag{3}$$

$$\boldsymbol{h}_{\text{drop}} = \text{Dropout}(\boldsymbol{h},\ p_{\text{drop}}) \tag{4}$$

$$\hat{y} = \boldsymbol{\sigma}(\boldsymbol{h}_{\text{drop}}U + b_2) \tag{5}$$

Averaging the word vectors has intuitive appeal; doing so not only normalizes every example for length, but also takes every word in the content into account.

4.3 Recurrent Neural Network

The RNN was initially implemented to process the entire length of every news article. However, with the longest training sample at 26,970 words, it quickly became computationally infeasible to train the model. Consequently, a limit of 200 time steps was imposed (a hyperparameter, determined via cross-validation), and shorter examples were padded with zero vectors at the beginning, so that the network gets to examine the words in the examples as it moves through the time steps to the end. Padding at the end of the examples produced higher loss.

$$\boldsymbol{h}_t = \boldsymbol{\sigma}(\boldsymbol{x_t}W_x + \boldsymbol{h}_{t-1}W_h + b) \tag{6}$$

The output at the final time step was recorded as the predicted label for the example. Figure 7 illustrates the basic architecture underlying the network used in this model, as well as in some of the others that follow.

4.4 Long Short-Term Memories

The Long Short-Term Memory (LSTM) unit was initially proposed by Hochreiter and Schmidhuber [7], and since then a number of modifications to the original unit have been made. Unlike the recurrent unit, which simply computes a weighted sum of the input signal and applies a nonlinear function, each LSTM unit maintains a memory c_t at time t, which is subsequently used to determine the output, or the activation, h_t, of the cell.

$$h_t = o_t \odot \tanh(c_t) \tag{7}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{8}$$
$$\tilde{c}_t = \tanh(x_t W_c + h_{t-1} U_c + b_c) \tag{9}$$
$$o_t = \boldsymbol{\sigma}(x_t W_o + h_{t-1} U_o + b_o) \tag{10}$$
$$i_t = \boldsymbol{\sigma}(x_t W_i + h_{t-1} U_i + b_i) \tag{11}$$
$$f_t = \boldsymbol{\sigma}(x_t W_f + h_{t-1} U_f + b_f) \tag{12}$$

See [8] for a good exposition of the motivation behind this mathematical formulation.

4.5 Gated Recurrent Units

Although RNNs can theoretically capture long-term dependencies, they are very hard to actually train to do this. Gated recurrent units are designed in a manner to have more persistent memory thereby making it easier for RNNs to capture long-term dependencies [8].

$$h_t = u_t \odot \tilde{h}_t + (1 - u_t) \odot h_{t-1} \tag{13}$$
$$\tilde{h}_t = \tanh(x_t W + (r_t \odot h_{t-1}) U + b) \tag{14}$$
$$u_t = \boldsymbol{\sigma}(x_t W_u + h_{t-1} U_u + b_u) \tag{15}$$
$$r_t = \boldsymbol{\sigma}(x_t W_r + h_{t-1} U_r + b_r) \tag{16}$$

The above equations can be thought of a GRU's four fundamental operational stages and they have intuitive interpretations that make this model much more intellectually satisfying.

4.6 Bidirectional RNN with LSTMs

As a final RNN model, a bidirectional recurrent network with LSTM cells was also implemented. The final states of the forward and backward layers are concatenated and passed through an affine transform before being input to the sigmoid to generate the prediction.

4.7 Convolutional Neural Network with Max Pooling

This model starts with the 300-dimensional GloVe embeddings of all the words in the example, and extracts bigrams, trigrams, 4-grams, as well as 5-grams from the text. Vectors corresponding to these n-grams are then concatenated and fed into a linear layer, where they are scaled using different weight matrices (but the same bias). A max pooling layer then selects the largest from the scaled inputs. The results are added and transformed via a ReLU unit, whose output is finally sent to a sigmoid nonlinearity to predict the label.

Figure 8 shows the CNN model architecture.

4.8 Attention-Augmented Convolutional Neural Network

As an exploratory exercise, a new architecture that augments a CNN with an attention-like mechanism was implemented. The intuition behind this approach is to determine if the features generated by the convolution process can be enhanced to capture influence from prior n-grams before they are max-pooled and sent through the nonlinearity. The idea is to learn an attention matrix for bigrams, and another for trigrams (only two n-grams were used in this specific model) that capture influence from a small set of n-grams preceding the convolution window.

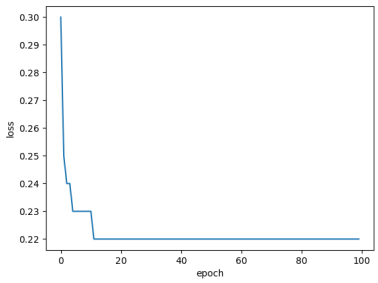

Figure 1: Loss for the Linear Model

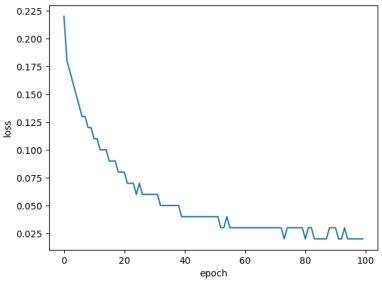

Figure 2: Loss for the Feed-Forward Neural Network Model

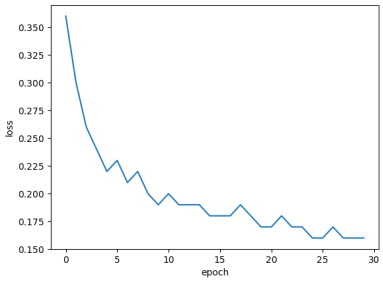

Figure 3: Loss for the RNN Model

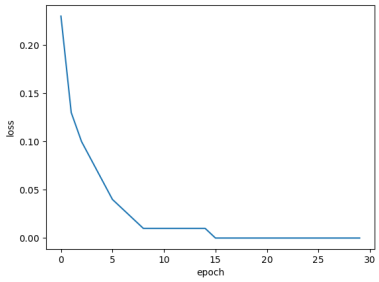

Figure 4: Loss for the GRU Model

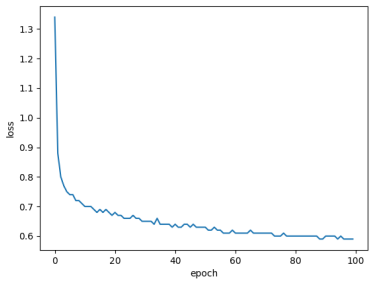

Figure 5: Loss for the CNN Model

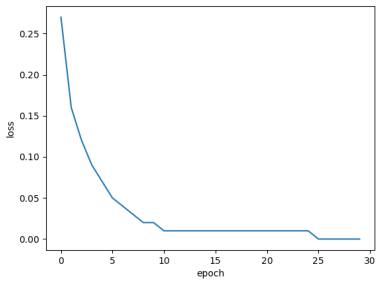

Figure 6: Loss for the LSTM Model

In other words, for words $x_i \in \mathbb{R}^{1\times k}$ in a window size h, we learn not only the convolution filter $W^{(h)}$, but also an attention matrix $W_a^{(h)}$, such that the feature map becomes

$$\boldsymbol{c} = \left[W^{(h)} \sum W_a^{(h)} x_{i-1:i-1+h}^T x_{i:i+h}\right] \tag{17}$$

Max pooling is subsequently applied to this augmented feature map.

5 Implementation

All code was written in Python 2.7, using TensorFlow r0.12.1 and NumPy. However, TensorFlow's implementation of LSTM and GRU cells was not used; instead, they were coded from scratch. Xavier initialization was used for all variables. Dropout was employed as a regularization mechanism for the feedforward and the recurrent neural networks.

Experiments were run on local CPU-only machines, as well as on GPU-enabled Azure clusters provided by Microsoft.

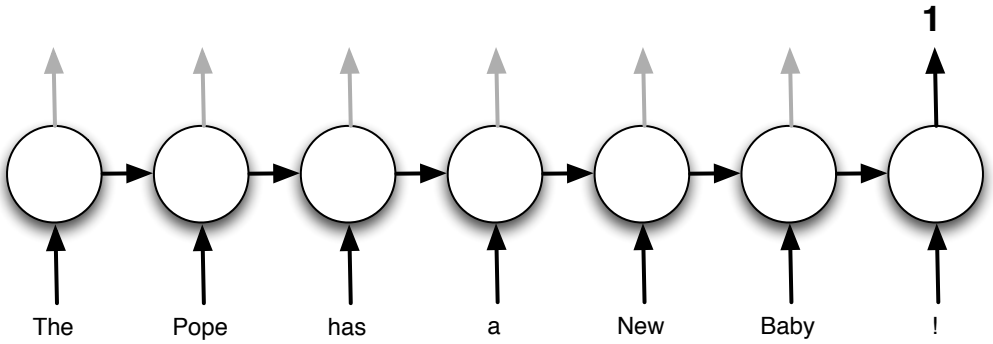

Figure 7: RNN Architecture for the Classification task

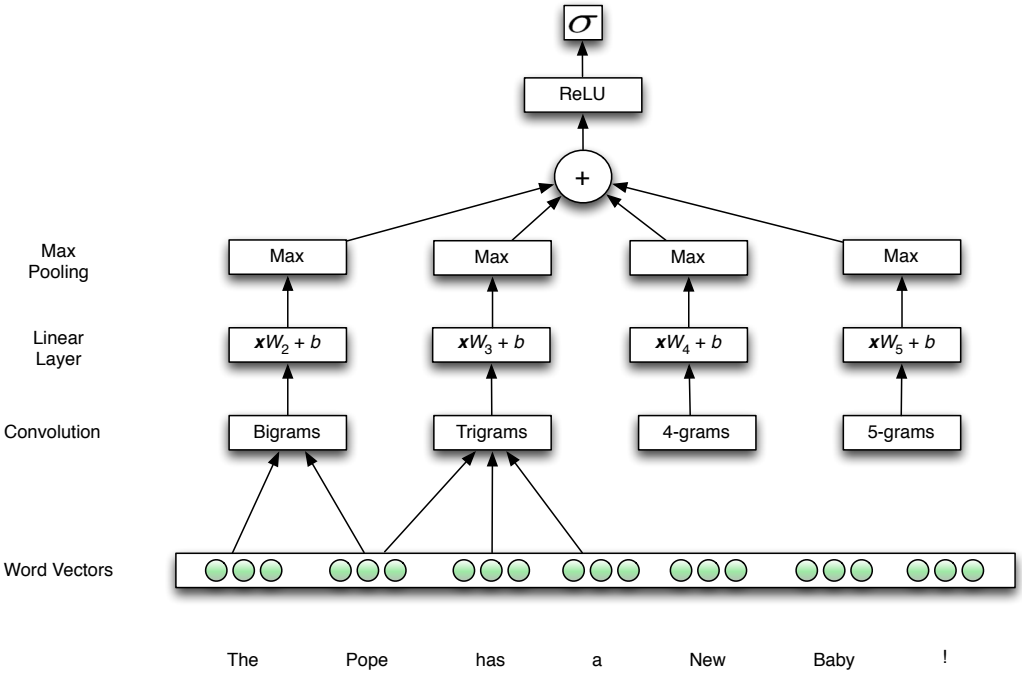

Figure 8: CNN Architecture for the Classification task

Table 1: Hyperparameters for the Various Models

Model	Embedding Size	Learning Rate	Optimizer	Hidden Size	Dropout	Time Steps	Epochs
Logistic Regression	300	10^{-1}	SGD	—	—	—	100
Feedforward Network	300	10^{-3}	Adam	200	0.5	—	100
RNN (Vanilla)	300	10^{-3}	Adam	100	0.5	200	30
GRUs	300	10^{-4}	Adam	100	0.5	200	30
LSTMs	300	10^{-4}	Adam	100	0.5	200	30
BiLSTMs	300	10^{-3}	Adam	100	0.5	200	20
CNN with Max Pooling	300	10^{-2}	Adam	100	—	—	100
CNN with Max Pooling and Attention	300	10^{-2}	Adam	100	—	—	10

Table 2: Model Performance on the Test Set

MODEL	PRECISION	RECALL	$\mathbf{F_1}$
Logistic Regression	0.96	0.49	0.65
Feedforward Network	0.89	0.74	0.80
RNN (Vanilla)	0.91	0.56	0.70
GRUs	0.89	**0.79**	**0.84**
LSTMs	0.93	0.72	0.81
BiLSTMs	0.88	0.75	0.81
CNN with Max Pooling	0.87	0.44	0.58
CNN with Max Pooling and Attention	**0.97**	0.03	0.06

6 Results and Discussion

The performance of each model employed in this project is summarized in table 2.

There were two major takeaways from these experiments. First, in accordance with the observations reported in [9] and [10], the RNN architecture with GRUs outperformed one with LSTM cells in the task at hand. This result holds despite the fact that a positive bias was added to the LSTM's forget gate. Not only did the RNN with GRUs achieve the better F_1 score (and the best overall), it converged faster than the one using LSTM units. While the faster convergence time can be attributed to the fact that GRUs have fewer gates, and hence fewer computations to generate the output, the fact that the GRU exposes its full content without any control likely helped it perform better overall, given that examples in the data set were long news stories, averaging 515 words across both classes. Further, the bidirectional RNN model equipped with LSTMs didn't do significantly better than the unidirectional network, reporting almost the same F_1 score.

The fact that an RNN architecture won out was a very interesting result in itself. It is worth noting that some of the news articles in the dataset were thousands of words long, and RNNs, even those equipped with LSTMs or GRUs, cannot retain memory for more than a hundred or so time steps. The secret to the success of RNNs in this task may have to do with the fact that the network employed a large internal state of size 100, often posing challenges in training and resource usage. A smaller internal state didn't perform as well.

Second, it was unsurprising to see the feedforward network perform well as a classifier. Its success can be attributed to the fact that its input comprises the mean across the embeddings of all the words in the example. As a consequence, the model takes into account contributions from potentially salient words that identify the news story.

It was disappointing to see that the Convolutional network didn't perform as well, given that multiple n-grams were used as inputs to the model, and that a max pooling layer was incorporated. The assumption was that the max pooled inputs would be a strong indicator to the classifier; but apparently that signal isn't adequate. Perhaps additional features would have helped. Further, enhancing the CNN with an attention-like device didn't add much value (although this model achieved the highest precision), but that's also because only ten epochs of training could be completed within the given time and resource constraints. It is also worth noting that the introduction of attention vectors increases the number of parameters to learn, necessitating a larger training set, which was fixed for all experiments conducted in the project to allow timely completion and fair comparison of model performance.

The precision numbers are high for all models because the dataset is (deliberately) skewed, with authentic news examples (i.e., negative examples) outnumbering the fake news examples by 4 to 1. Consequently, any model can deliver high accuracy simply by guessing every test example as negative. However, only the feedforward network and the RNNs with complex activation units achieved a recall greater than 0.7, indicating that the models were able to extract some subset of relevant features to classify more of the positive examples correctly.

In a very high dimensional space, linear classifiers are generally able to take advantage of the sparsity and find a hyperplane that separates the classes reasonably well. However, with dense vectors like the GloVe embeddings used in all experiments for this project, it was apparent that a nonlinear kernel is necessary for good results. The logistic regression model that serves as the baseline is devoid of any nonlinearity, and therefore reports one of the lowest F_1 scores among the bunch.

7 Future Directions

A complete, production-quality classifier will incorporate many different features beyond the vectors corresponding to the words in the text. For fake news detection, we can add as features the source of the news, including any associated URLs, the topic (e.g., science, politics, sports, etc.), publishing medium (blog, print, social media), country or geographic region of origin, publication year, as well as linguistic features not exploited in this exercise—use of capitalization, fraction of words that are proper nouns (using gazetteers), and others.

I also believe that the idea of attention-enabled CNNs is a promising avenue worth exploring further. Finally, more sophisticated models—for example, pointer and highway networks—definitely merit investigation.

I plan to continue this effort after the class ends.

Acknowledgments

Many thanks to Dr. Richard Socher, who advised me on this project. The TAs and instructors of CS 224N deserve high praise for making the course highly engaging, despite the logistical challenges posed by the sheer size of the class.

Thanks also to the course staff for use of some boilerplate code from the homework assignments as a starting point for the project implementation.

References

[1] Allcott, H., and Gentzkow, M., *Social Media and Fake News in the 2016 Election*, `https://web.stanford.edu/~gentzkow/research/fakenews.pdf`, January, 2017.

[2] Kaggle, *Getting Real About Fake News*, `https://www.kaggle.com/mrisdal/fake-news`, URL obtained on March 2, 2017.

[3] Signal Media, *The Signal Media One-Million News Articles Dataset*, `http://research.signalmedia.co/newsir16/signal-dataset.html`, URL obtained on March 2, 2017.

[4] Manning, C., Raghavan, P., and Schütze, H., *Introduction to Information Retrieval*, `http://informationretrieval.org/`, 2008.

[5] Pennington, J., Socher, R., and Manning, C., *GloVe: Global Vectors for Word Representation*, 2014.

[6] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T., *Bag of Tricks for Efficient Text Classification*, August, 2016.

[7] Hochreiter, S., and Schmidhuber, J., *Long Short-Term Memory*, Neural Computation, 1997.

[8] Mohammadi, M., Mundra, R., Socher, R., and Wang, L., *CS224N Lecture Notes, Part V*, 2017. `http://web.stanford.edu/class/cs224n/lecture_notes/cs224n-2017-notes5.pdf`, URL obtained on March 3, 2017.

[9] Jozefowicz, R., Zaremba, W., and Sutskever, I., *An Empirical Exploration of Recurrent Network Architectures*, Proceedings of the 32nd International Conference on Machine Learning, 2015.

[10] Chung, J., Gulchere, C., Cho, K., and Bengio, Y., *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*, 2014.