Stance Detection for the Fake News Challenge: Identifying Textual Relationships with Deep Neural Nets

Ali K. Chaudhry
Stanford University
Graduate School of Business
alikc@stanford.edu

Darren Baker
Stanford University
School of Engineering &
Graduate School of Business
drbaker@stanford.edu

Philipp Thun-Hohenstein
Stanford University
Graduate School of Business
pthun@stanford.edu

Abstract

The issue of "fake news" has arisen recently as a potential threat to high-quality journalism and well-informed public discourse. The Fake News Challenge was organized in early 2017 to encourage development of machine learning-based classification systems that perform "stance detection" -- i.e. identifying whether a particular news headline "agrees" with, "disagrees" with, "discusses," or is unrelated to a particular news article -- in order to allow journalists and others to more easily find and investigate possible instances of "fake news." We developed several deep neural network-based models to tackle the stance detection problem, ranging from relatively simple feed-forward networks to elaborate recurrent models featuring attention and multiple vocabularies. We ultimately found that an LSTM-based bidirectional conditional encoding model using pre-trained GloVe word embeddings delivered the best performance: greater than 97% classification accuracy on the dev set.

1 Introduction

Over the past several months, the issue of "fake news" -- defined by the New York Times as "made-up stories written with the intention to deceive" and published in formats similar to those of traditional "real" news -- has arisen as a threat to high-quality journalism and well-informed public discourse. In particular, fake news has been accused of increasing political polarization and partisan conflict in the United States during the divisive 2016 presidential campaign and the early days of the Donald Trump administration.

For our CS224N final project, our team chose to work toward finding a solution for the classification task presented in the Fake News Challenge[1] (FNC), which was organized in early 2017 by a group of academics and contributors from the journalism industry with the goal of applying artificial intelligence techniques toward the goal of improving news accuracy. As the FNC organizers describe on their web site, the goal of the challenge is not to directly identify whether a headline or article is "fake" or not, which is arguably a highly subjective question, and one that even skilled humans may have difficulty answering. Instead, the challenge is organized around the more well-defined problem of "stance detection," which involves comparing a headline with a body of text from a news article to determine what relationship (if any) exists between the two. There are 4 possible classifications:

1. The article text **agrees with** the headline.
2. The article text **disagrees with** the headline.

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

3. The article text **is a discussion of** the headline, without taking a position on it.
4. The article text **is unrelated to** the headline (i.e. it doesn't address the same topic).

Presumably, a classifier that can solve the stance detection problem with high accuracy might effectively be used either as a tool for humans working to identify fake news (e.g., retrieving articles that agree, disagree and discuss a headline), or as a building block for a more elaborate future AI system that would try to identify the actual truthfulness of news stories (e.g., using credible sources to classify).

2 Background and Literature Review

Over the past few years, many research efforts in NLP have focused on the application of deep neural network models to various sequence-based tasks. A frequent template for such tasks involves taking two text sequences, encoding them in some form, and then attempting to classify their relationship, as in the "natural language inference" task discussed by Bowman et al. [3]. In this task, the authors shared a new corpus of labeled sentence pairs and explored approaches for classifying them based on semantic relationships like "entailment" and "contradiction."

Another common sequence-based task in NLP is machine translation (MT), which is perhaps the canonical application for recent work on "sequence-to-sequence" (seq2seq) structures. These models encode one sequence of words or tokens (for example, a sentence in English) and then attempt to "decode," step by step, a related sequence of output tokens (e.g. the French equivalent of the English sentence). Sustkever et al. from Google made one of the first applications of seq2seq2 models to MT [9] just a few years ago. Others, including Bahdanau et al. [2] and Luong et al. [7] have extended the Google team's work by introducing new mechanisms such as "attention," which allows the model's decoder to focus on certain portions of the encoded input sequence at each step of the output in order to make the best possible predictions. Since recurrent neural networks are often a central component of sequence-based NLP models, most of these efforts (and ours as well) have relied on "memory" units like the LSTM (long short-term memory, Hochreiter and Schmidhuber, [6]) and the GRU (gated recurrent unit, Chung et al., [4]).

Two research papers were noted by the Fake News Challenge organizers for having laid useful groundwork for the stance detection problem. Ferreira and Vlachos [5] used the "Emergent" dataset to compare rumored claims against news articles that had been previously labeled by journalists with an estimate of their veracity, with the goal of predicting the stance of the article towards the rumor. This team summarized each article into a headline and used a logistic regression model with features representing the article and claim to classify the combination of article and claim as either "for," "against," or "observing," with a final accuracy level of 73%.

Augenstein et al. [1] took on a similar task of stance detection, though on a different dataset with somewhat shorter text strings. They tried to predict whether a tweet was "positive", "negative" or "neutral" towards a short topic string (e.g. "Legalization of Abortion") that they labeled the "target." They explored several models that employed a pair of LSTMs in different arrangements. In the model they termed "independent encoding," one LSTM encoded the target string while another LSTM encoded the tweet, and the final hidden state vectors of the two LSTMs were then passed through a single feed-forward and softmax layer to make a prediction. In the "conditional encoding" model, the two LSTMs were arranged sequentially: the tweet encoder state was initialized using the final hidden state of the target encoder, and prediction was based on the final hidden state of the tweet encoder only. A final variation was "bidirectional conditional encoding," which extended the previous model by encoding the tweet in both directions and then using the two final hidden state vectors for prediction. This group's results showed that conditional encoding delivered a meaningful performance boost over an independent encoding model, with bidirectional conditional encoding delivering some small additional gains.

3 Dataset overview

The datasets for our task are provided by the Fake News Challenge organization, and can be downloaded on their Github page.[2] The complete training set consists of just under 50,000 "stance" tuples, with each tuple consisting of:

- A headline that is to be compared against an article to determine its stance toward the article. Word counts for the headlines range from 2 to approximately 40, with an average length of ~11.
- The (integer) ID of an article against which the headline is to be compared, which can be used to find the text of the article body in a separate file. Article lengths range from 2 to nearly 5000 words, with an average length of around 360 words.
- The true stance of the headline with respect to the article. (This is one of the four classes outlined earlier: agree, disagree, discuss, and unrelated.)

The total number of unique headlines and articles is 1683 of each, though because most headlines are matched against many different articles, the number of individual training samples is much larger. The true class breakdown for the training set is roughly 73% "unrelated," 18% "discuss," 7% "agree," and 2% "disagree."

In order to build our classification models, we selected a set of roughly 3000 stances to serve as a development set (for performance evaluation and hyperparameter tuning), and left the remainder for training. Two key observations about this development set:

- We did not select an additional test set of datapoints for final evaluation of our model(s), because the FNC organization will be releasing such a test set soon, and we believed our dev set is sufficient for evaluation purposes until that test set becomes available.
- We initially used a random sample of training examples to select the members of our dev set, but we worried that the fact that unique headlines and articles are used repeatedly in various training examples could lead to "leakage" of training data at evaluation time. Therefore, we also experimented with explicitly segregating unique headlines to belong to either the training or dev sets, but never both. (We provide more details below on the results of these experiments with respect to our performance on the classification task.)

We also preprocessed our dataset extensively in order to split sentences, normalize casing, handle punctuation and other non-alphabetic symbols, and otherwise improve token consistency. We also built vocabulary lists for terms occurring in either the headline or article texts, and extracted corresponding pre-trained word embeddings for these tokens from the Stanford GloVe corpus (50d vectors from 6B corpus, Pennington et al., [8]).[3]

4 Modeling approaches

Our goal in approaching the stance detection problem was to experiment with a wide range of the deep learning and NLP techniques that we have learned this quarter in CS224N, including:

- Dense vector embeddings of words, tokens, and sequences
- Multi-layer feed-forward networks
- Recurrent neural networks, including LSTM and GRU "cells"
- Attention mechanisms

The following subsections give an overview of the models that we implemented.

[2] https://github.com/FakeNewsChallenge/fnc-1
[3] http://nlp.stanford.edu/data/glove.6B.zip

4.1 Naive baseline (Jaccard similarity)

The FNC scores results in a two-stage progress, granting a small amount of credit for successfully identifying "related" vs. "unrelated" headline/article pairs (which is presumed to be easier task), and much more credit for correctly classifying the "agree" / "disagree" / "discuss" relationship between pairs that are deemed to be related. Thus, to establish a simple performance baseline, we first implemented a quick Jaccard similarity scorer that compared headlines with individual sentences from their paired article. By finding maximum and average Jaccard similarity scores across all sentences in the article and choosing appropriate threshold values, we were already able to achieve ~90% accuracy on the related/unrelated task, so we hoped for very high accuracy when moving to deep learning.

4.2 Multi-layer feed-forward network

Our first NN-based model used simple transformations (averaging, concatenation, etc.) on pre-trained word embeddings to construct independent sets of features for headlines and articles, which we then sent through a multi-layer network to get a prediction result. This method trained very quickly and delivered solid classification results, with overall accuracy levels in the range of 90-95% across the 4 classes we tried to predict. However, we believed that recurrent models might be able to capture nuances of word order and other such relationships in a more robust way, so we focused much of our energy on the more elaborate architectures described next.

4.3 LSTMs with independent, conditional, and bidirectional conditional encoding

Taking inspiration from the approach of Augenstein et al. [1], we explored the use of multiple recurrent network layers to encode headlines and articles before classifying the resulting state vectors with a softmax transformation. Our first attempt involved separate (parallel) LSTM encodings of the headline and article and a subsequent classification using the final hidden states of each encoder to make a prediction. (This is what Augenstein labeled **"independent encoding."**) Since the vocabulary over our training set was relatively small -- around 3,000 distinct types in headline strings and 24,000 types in articles -- we used pre-trained word embedding vectors from the Stanford GloVe dataset to give us a boost toward capturing token semantics. Therefore, the input to each LSTM at each step was a 50-dimensional vector representation of the current token, based on GloVe but also trained further in our model(s) to catch any adjustments that might be specific to our classification task.

Next, we moved on to a **conditional encoding**, which involves recurrent cells placed in sequence rather than in parallel. We first sent the headline text through an LSTM layer (call this $\text{LSTM}^{\text{Headline}}$), and then initialized another LSTM layer (call this $\text{LSTM}^{\text{Article}}$) with the final hidden state vector of $\text{LSTM}^{\text{Headline}}$. The final prediction was then made using only the final hidden state of $\text{LSTM}^{\text{Article}}$-- though of course the conceptual goal here is that this state also captures information about the headline and its relationship to the article, based on its initialization.

Finally, we tried a **bidirectional conditional encoding** consisting of 4 different LSTM layers. $\text{LSTM}^{\text{Headline-Forward}}$ connects to $\text{LSTM}^{\text{Article-Forward}}$, as in the previous model, while $\text{LSTM}^{\text{Headline-Backward}}$ and $\text{LSTM}^{\text{Article-Backward}}$ have the same relationship, but with their inputs supplied in reverse order. The final states of $\text{LSTM}^{\text{Article-Forward}}$ and $\text{LSTM}^{\text{Article-Backward}}$ are then averaged and fed into a final softmax prediction layer.

4.4 Sequence-to-sequence recurrent model with attention

All of the previous models were essentially custom-built in Python and Tensorflow, though we did take advantage of some useful building blocks from the Tensorflow library, such as the BasicLSTMCell class. For our final model, we decided to experiment with some of the richer library code available for Tensorflow. Specifically, we adapted the sequence-to-sequence neural machine translation (NMT) tutorial[4] in order to

[4] https://www.tensorflow.org/tutorials/seq2seq

create an attentive seq2seq model using Tensorflow's "embedding_attention_seq2seq" function. This model allowed us to use either LSTM or GRU cells easily, and also provided the advantage of adding an attention layer on top of the conditional encoding that is standard in seq2seq models. However, it also came with two possible liabilities:

- Although the model did use word embeddings for training, it did not easily allow us to supply pre-trained word vectors for the tokens in our headlines and articles. (Instead, it simply accepted parameters for the size of the vocabulary and the desired embedding size, and trained new embeddings from scratch.) Given that our training set was relatively small -- a few megabytes vs. the tens of gigabytes that might be used in an application like NMT -- we believe that the lack of pre-trained embeddings would be a drag on performance.
- Additionally, the model was originally designed as a true sequence-to-sequence structure, meaning that it was intended to produce outputs at each step of the "decoder." In our application, by contrast, step-by-step "decoding" was not relevant: we only wanted a final classification output after the article text had been fully fed in. We wondered whether a loss function based only on this final output and not on the step-wise output would compromise the model's performance.

5 Experimental process and results

We ran a range of experiments for each of the models outlined above, though we directed most of our time and effort toward the recurrent neural network approaches. This section will briefly outline the major hyperparameters in our experiments and present our results.

5.1 Choosing appropriate hyperparameters

Though our models included up to a dozen hyperparameters to support experiments with capabilities like learning rate decay and gradient clipping, (see results in Appendix I), we ultimately focused primarily on the following subset:

- **Learning rate:** Many of our early experiments with the recurrent models were unsuccessful (i.e. classification accuracy was poor) because we were using a comparatively large learning rate (0.5) that had been the default setting in the Tensorflow NMT tutorial on which some of our code was based. We realized later that this learning rate was at least an order of magnitude too large to produce meaningful results for our task, and ultimately found that a learning rate of around 0.001 provided the best balance between training speed and final dev set performance.
- **Token length for headlines and articles:** As noted in section 3 above, some headlines in our training data were as long as 40 words, while some articles approached 5000 words in length. Though we could conceivably have padded all our inputs to these lengths, or used length bucketing to improve training speed, our inspection of many headline/ article pairs suggested that a human would likely be able to classify most pairs by focusing on the first 10-20 words of an article and the first 40-50 (or even fewer) words of an article. We ultimately settled on truncating headlines after 15 tokens and truncating articles after 40 tokens in most versions of our models (while padding any headlines/articles that did not reach these lengths), as longer lengths did not appear to provide any meaningful performance gains in our testing.
- **Including punctuation:** We prepared parallel versions of our training data: one that kept most punctuation and one that stripped it out. Our experiments consistently showed that neither one performed better than the other in classification, so we mainly used the training data without punctuation for our final models.
- **Running 2-, 3-, or 4-class classification:** Given the structure of the stance detection problem, we considered whether it would be more effective to build a single classifier that could handle all classes (agree, disagree, discuss, unrelated) at the same time, or whether we might more profitably employ a two-stage approach that first performed related/unrelated classification and then used a different model to bucket the "related" pairs into the more granular agree/disagree/discuss categories. Since the training set is somewhat unbalanced ("unrelated" points are 73%, while

"disagree" points are just under 2%), we initially believed that the two-stage approach might be more effective. However, after tuning our models, we found that the sequential combination of 2-class and 3-class classification did not give meaningfully better results than simply training all 4 classes at the same time, so the numbers we report below are all based on a single 4-class model.

We did invest significant effort in tuning other hyperparameters related to the architecture of our network, such as the number of network layers or the number of recurrent units in each layer. However, tuning these values didn't ultimately cause major variations in our results. Some simple plots showing sensitivity analysis are included in the Appendix for the interested reader.

5.2 Results and discussion

Recall from section 3 above that we used two different approaches to select a development set of stances for evaluation. Understandably, the model performed slightly better on the set where dev stances were randomly chosen, and therefore some headlines might have appeared in both the training and dev sets, though paired with different articles. However, the gap between performance on this dev set and the one where headlines were strictly segregated between dev and training was not very large -- a few percentage points at most. We report statistics for both development sets in Table 1.

Table 1: Overall classification accuracy for 4-class problem

('Set 1' refers to random split in training/dev examples; in 'Set 2', headlines are strictly segregated between training/dev)

	Set 1		Set 2	
Model	**Training accuracy**	**Dev accuracy**	**Training accuracy**	**Dev accuracy**
Independent encoding	88.1%	84.5%	88.7%	84.1%
Conditional encoding	99.8%	**97.3%**	99.8%	94.7%
Bidirectional conditional encoding	99.8%	**97.3%**	99.8%	95.3%
Seq2seq with attention	99.0%	96.5%	99.1%	93.9%

For our best-performing model (the bidirectional conditional encoding approach), Table 2 shows the final per-class precision, recall, and F1 scores that we observed when we reached our highest level of classification accuracy on the randomly selected dev set.

Table 2: Per-class precision/recall statistics for best-performing model

Class label	**Proportion of original training examples**	**Precision**	**Recall**	**F1 score**
Unrelated	72.6%	98.7%	98.7%	**98.7%**
Discuss	17.6%	96.1%	97.0%	**96.5%**
Agree	8.0%	89.2%	89.5%	**89.4%**
Disagree	1.8%	89.6%	78.2%	**83.5%**

Figure 1 shows a plot of overall dev accuracy and per-class F1 scores as a function of the training epoch. As the plot shows, performance for each of these measures generally plateaus after roughly 20 epochs.

Finally, Table 3 and Table 4 show confusion matrices for our model across the four class labels, with raw counts and percentages (of true class counts), respectively.

Table 3 (left) and Table 4 (right): Four-class confusion matrices for best-performing model

		Predicted Class			
		Unrelated	Agree	Disagree	Discuss
True Class	Unrelated	2151	11	2	15
	Agree	16	214	3	6
	Disagree	2	10	43	0
	Discuss	11	5	0	511

		Predicted Class			
		Unrelated	Agree	Disagree	Discuss
True Class	Unrelated	98.7%	0.5%	0.1%	0.7%
	Agree	6.7%	89.5%	1.3%	2.5%
	Disagree	3.6%	18.2%	78.2%	0.0%
	Discuss	2.1%	0.9%	0.0%	97.0%

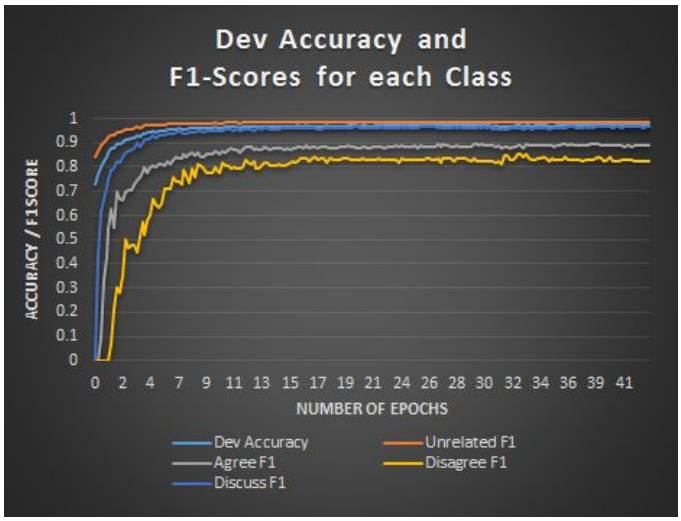


Figure 1: Dev set performance metrics by training epoch for best-performing model

Some observations about these results:

- As mentioned previously, we were unsure whether to expect better results from non-attentive models that used pre-trained GloVe word embeddings, or from our attentive sequence-to-sequence model that used randomly initialized embeddings. The results show that the performance gap between the two sides is not large, but in the end the non-attentive models won out. We presume this is a result of the extra predictive power offered by pre-trained embeddings in the context of our relatively small training dataset.
- Of the four classes, the "disagree" class and the "agree" classes have the the lowest F1 scores (with 83.5% and 89.4%, respectively). These two classes were relatively underrepresented in the training set, with "disagree" making up only 2% of examples and "agree" only another 8%. We were pleased with the relatively high precision we were able to achieve for these classes even with limited training data, but more data for these two classes would almost certainly allow us to improve our F1 scores for each.

5.3 Error analysis

Although the accuracy figures show that our model made classification errors on the dev set only relatively infrequently, we examined several failure cases to identify possible causes. Notable error classes included:

- **Difficult distinction between "discuss" and "agree"**: Even for a skilled human reader, assessing the boundary between the "discuss" and "agree" classes can be nuanced and challenging. For example, our model predicted "agree" for a headline claiming "*U.S. airstrike targets Al-Shabaab leader*" and associated article text "*Pentagon confirmed that the leader of Al Shabab terror group who was the target of a U.S. airstrike was killed*." Our team felt that a classification of "agree" was reasonable, but the training label for this pair was "discuss."
- **Sarcasm, metaphors, and other subtle semantic elements:** Occasionally, news headlines often contain sarcasm and metaphors that can be difficult for the model to properly comprehend. For example, the headline "*If only a bird really did poop on Putin*" implies that the event in question did not actually happen -- though even a typical human reader might have to parse this headline more than once to catch the meaning. Similarly, other headlines use metaphorical language -- for example, "*Watch bird launch airstrike on Putin's shoulder*." The semantics of phrases like this are likely hard for the model to learn without a variety of similar training examples to consider.
- **Similar topics, but related to different entities**: Some misclassifications occurred when the headline and article were talking about the same topic but related to different specific entities. For

instance, the model classified a headline "*Macaulay Culkin has not died despite what everyone is saying on facebook*" as agreeing with an article that talks about "*Jedd Nelson rebuffs internet rumors that he died of a drug overdose.*" While both texts talked about refuting rumors of an individual's death, they were talking about different individuals -- a fact that the model didn't accurately capture.

- **General ambiguities in relationships between news events:** Some individuals and entities appear regularly in the news, and in the absence of richer external context, it can be difficult to evaluate whether mentions of (say) a well-known person are truly related in the sense that they describe the same news "event." For instance, the training data indicated that the headline "*Kim Jong Un had ankle surgery*" was "unrelated" to the article text "*The younger sister of Kim Jong Un has allegedly taken over state duties while her brother receives medical treatment*" -- a label that a human reader might reasonably consider either to be accurate or inaccurate, depending on the level of specificity expected from "related" pairs.
- **Incorrect labels in the original data**: A small number of incorrect predictions were driven by training examples that were mislabeled (based on our judgment). Fortunately, our model behaved as desired in these cases -- that is, it chose the label that seemed appropriate to us.

One common thread in these error classes is that many of them might best be addressed by more or better training data, though of course that avenue may be mostly out of our team's control, at least with respect to the FNC task specifically. However, it's possible that we could incorporate some other kinds of feature data on top of the existing training set to eke out a small amount of additional performance -- for example, by incorporating information like parts of speech, dependency relationships, or named entity labels.

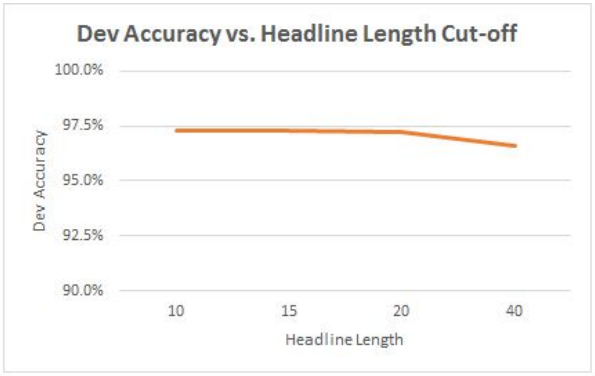
6 Conclusions and future work

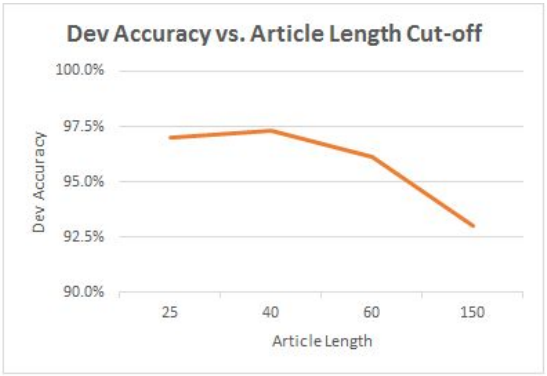
Our experiments showed clearly that recurrent neural network models with "memory-enabled" units like LSTMs significantly outperform non-recurrent models on the sequence-based task of stance detection. Furthermore, we learned that adding conditional encoding delivers a significant boost in prediction performance, while the addition of bidirectionality and/or attention seems to have a more modest impact -- at least in the context of this specific task. The conditional encoding allows the network to base its predictions not only on the content of the independent text strings (headline and article), but also on their relationship to each other, leading to far greater accuracy than can be achieved with other forms of independent or parallel encoding.

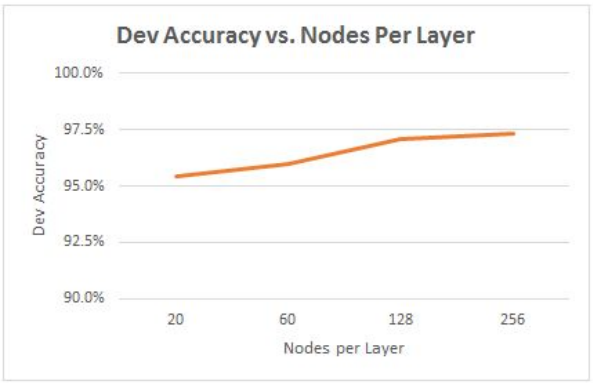
Although we are pleased with the high levels of classification accuracy delivered by our model on this task, we are interested in considering opportunities for further performance gains. We consider the following to be promising avenues for exploration:

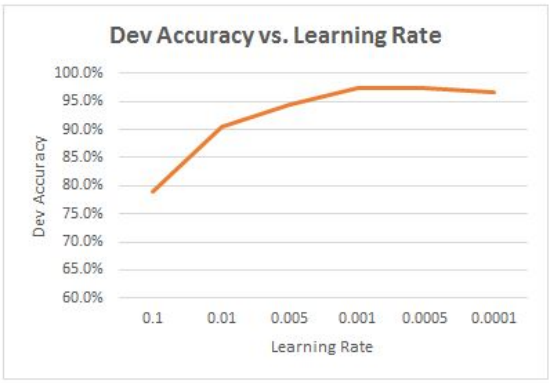
- **Obtaining additional training data:** Given the relatively small size of our training set, it seems likely that additional data (especially for the less frequent classes, like "agree" and "disagree") would help our model learn to generalize better. Of course, this is easier said than done, since presumably the training data provided here must be hand-labeled, and seeking out "fake news" to serve as a source of "disagree" cases may be a non-trivial exercise.
- **Using higher-dimensional pre-trained embeddings:** We used 50-dimensional pre-trained GloVe word embeddings, but it would be worth experimenting to determine whether higher-dimensional vectors might add more nuance to our model's encoding classification capabilities.
- **Combining an attention mechanism with pre-trained embeddings:** As discussed, our custom-built conditional encoding model used pre-trained embeddings but didn't include attention, while our seq2seq model included attention but trained its own embeddings for each word in the training vocabulary. Combining these might allow us to squeeze out some small additional performance gains.
- **Incorporating additional syntactic features:** We would be interested in evaluating whether additional language features, such as grammatical dependencies or named entity labels, would improve the model's understanding enough to help it distinguish between classes in the minority of cases where it commits errors today.

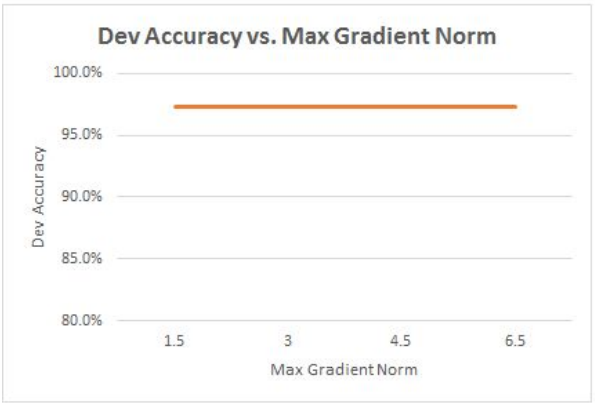
Appendix 1:
Sensitivity Analysis on Hyperparameter Values

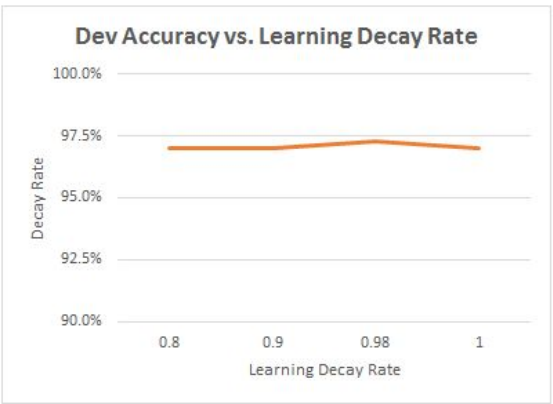












References

[1] Augenstein, Isabelle, et al. "Stance detection with bidirectional conditional encoding." *arXiv preprint arXiv:1606.05464,* 2016.

[2] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *arXiv preprint arXiv:1409.0473,* 2014.

[3] Bowman, Samuel R., et al. "A large annotated corpus for learning natural language inference." *arXiv preprint arXiv:1508.05326,* 2015.

[4] Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." *arXiv preprint arXiv:1412.3555,* 2014.

[5] Ferreira, William, and Andreas Vlachos. "Emergent: a novel data-set for stance classification." *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* ACL, 2016.

[6] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8, 1997 (p. 1735-1780).

[7] Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." *arXiv preprint arXiv:1508.04025,* 2015.

[8] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global Vectors for Word Representation." *EMNLP,* Vol. 14, 2014.

[9] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." *Advances in neural information processing systems*, 2014.