CS 224D Final Project: Neural Network Ensembles for Sentiment Classification

Tri Dao
Department of Computer Science
Stanford University
trid@stanford.edu

Abstract

We investigate the effect of ensembling on two simple models: LSTM and bidirectional LSTM. These models are used for fine-grained sentiment classification on the Stanford Sentiment Treebank dataset. We observe that ensembling improves the classification accuracy by about 3% over single models. Moreover, the more complex model, bidirectional LSTM, benefits more from ensembling.

1 Introduction

In language modeling and machine translation, predictions are made by taking average of around 5 models that have been trained independently. Ensembling can reduce variance (in the case of bagging) or both bias and variance (in the case of boosting). These techniques have been very popular in the context of decision trees. We hope that ensembles of a larger number of models (around 50) for tasks with smaller data sets can improve the performance of neural network models for these tasks. In particular, we investigate the effectiveness of emsembling a large number of simple long short-term memory (LSTM) models for sentiment classification.

2 Problem statement

For the task of sentence sentiment analysis, we use the Stanford Sentiment Treebank dataset [5] with 11,855 single sentences extracted from movie reviews. The sentences in the treebank were split into a train (8544), dev (1101) and test splits (2210).

Given a sentence from a movie review, we aim to predict the sentiment of the review. In the binary classification task, we predict whether the review is positive or negative. However, in this project, we will focus on the fine-grain classification task. That is, we predict whether the review is very negative, negative, neutral, positive, or very positive.

We evaluate the models by the prediction accuracy on the test set. We compare the accuracy of the ensemble (bagging) with that of a single model to assess the effectiveness of these techniques.

3 Related Work

Socher et al. [5] introduced Recursive Neural Tensor Network for the task of sentence sentiment analysis. The model pushes the state of the start in binary sentiment classification (from 80% to 85.4%). They also introduce and analyze fine-grained sentiment classification (5 classes), and achieve 45.7% accuracy. Making use of sentence parses and classification for each phrase improves the accuracy of the overall classification task.

Tai et al. [6] developed tree-structured LSTM for the same task. This is a generalization of LSTMs to tree-structure network typologies, and in particular sentence parses. This again improves the prediction accuracy of both the binary and fine-grained sentiment classification to 88.0% and 51.0% respectively.

Both of these works use the sentiment classification of phrases in the sentences as training data. There are 215,154 unique phrases in the dataset compared to just 11,855 sentences. Hence the actual number of training examples is much larger than the number of sentences. In this project, we do not make use of the phrases because we want to investigate the effectiveness of ensembling in the context of small dataset. That is, our dataset only consists of 11,855 sentences, 8544 of which are used as training data.

4 Approaches and models

4.1 LSTM

Traditional recurrent neural network often runs into the problem of vanishing gradient, since the gradient is multiplied a larger number of times by the weight matrix. For long sequences (such as long sentences), this makes the network unable to learn long-term dependencies.

This motivates the LSTM model introduced by Hochreiter and Schmidhuber [2] which contains a new structure called a memory cell (see Figure 1 below). A memory cell consists of an input gate, a neuron with a connection to itself, a forget gate, and an output gate. The memory cells can keep information intact, unless the inputs make them forget it or overwrite it with new input.

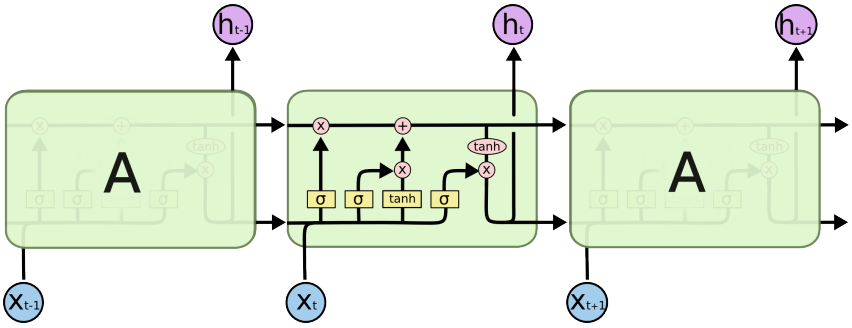

Figure 1: The repeating model in an LSTM contains four interacting layers, from [3].

4.2 Bidirectional LSTM

Graves et al. [1] introduce the bidirectional LSTM model. Figure 2 shows the structure of the bidirectional LSTM, where there is one forward LSTM and one backward LSTM running in reverse time. Their features are concatenated at the output layer, enabling information from both past and future to come together and make more accurate prediction.

4.3 LSTM model for sentiment classification

We use a simple LSTM model as described in Figure 1. For each word in the input sentence, we look up its corresponding word vector pre-trained with GloVe [4]. We then pass these vectors into a layer of LSTM cells, then apply a softmax layer to the output of the last LSTM. This outputs a probability for each class which can be used to predict the sentiment of the sentence. In the case of bidirectional LSTM, we follow the same procedure but we concatenate the output of the forward and the backward LSTM layers before applying softmax.

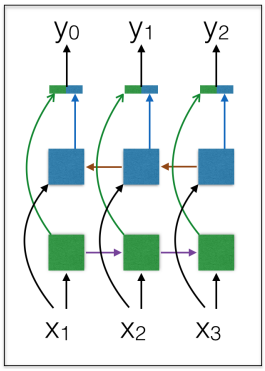

Figure 2: Bidirectional LSTM contains one forward and one backward LSTM, from [7].

4.4 Bagging

We train multiple neural network models (around 50) on bootstrap samples from the Stanford Sentiment Treebank dataset. That is, for each single model, we obtain a sample with replacement of size n from the training dataset, where n is the number of training examples in the training dataset. This means that some training examples might appear more than once in the bootstrap training set. To make a prediction, we either take the majority vote of all the models or average the probabilities output from each LSTM model to classify a new sentence to one of five sentiment classes. This has the effect of reducing the variance of the ensemble model. Therefore we do not apply as much regularization to the individual LSTM model. In particular, we do not use early stopping but allow each individual LSTM model to train to completion.

We train each single model on bootstrap samples from the training dataset instead of the whole dataset to reduce the correlation between the models. Since we will take the average of the predictions, the variance of the ensemble depends strongly on the correlation between the models. The less correlated the models are, the bigger the variance reduction bagging will bring.

5 Experimental results

5.1 Single LSTM model

We plot the train and dev accuracy of a single LSTM model against number of training epoch in Figure 3. Note that this model is trained on the original training set, not the bootstrap one.

We see that the training accuracy keeps increasing while the validation accuracy increases and then decreases. This suggests that the model overfits the data. However, since we will combine many such single models in an ensemble to reduce the variance, each model overfitting is not a problem. This is similar to bagging many decision trees where we grow the trees fully and do not prune them (a form of regularization.)

5.2 Single bidirectional LSTM model

Now we plot the train and dev accuracy of a single LSTM model against number of training epoch in Figure 4. Again, this model is trained on the original training set, not the bootstrap one.

We see that the performance is not unlike that of the single LSTM model without the backward flow, even though we are using a more powerful model. This suggests that the prediction error is coming from the variance instead of the bias. That is, with such a small training size (8544 examples), the variance dominates the error in both cases, leading to lower accuracy.

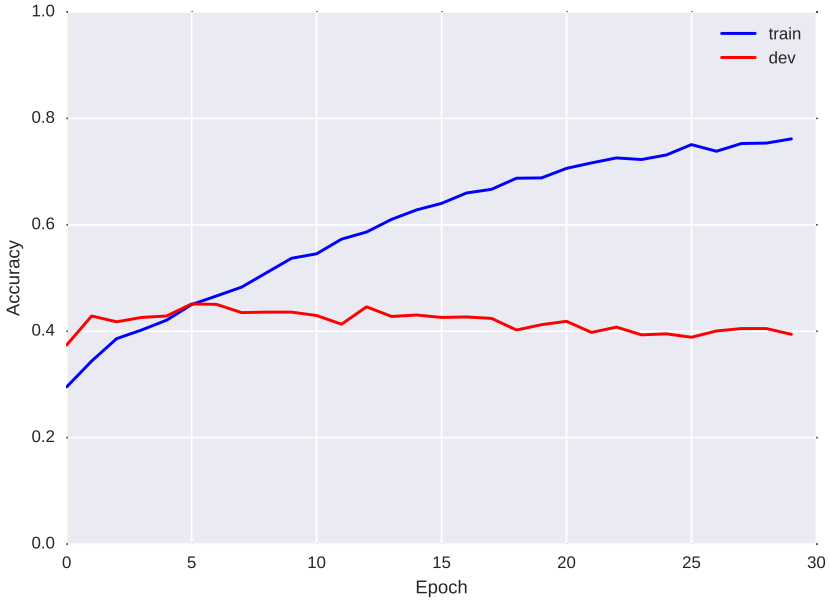

Figure 3: Accuracy of a single LSTM model.

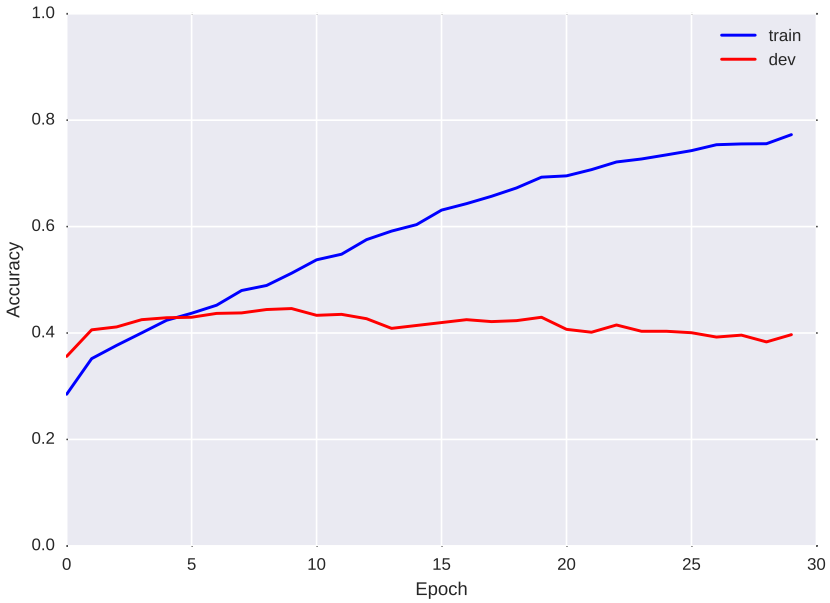

Figure 4: Accuracy of a single bidirectional LSTM model.

5.3 Ensemble

We plot the accuracy of the ensemble model on the dev set against the ensemble size in Figure 5 and Figure 6.

We see that in both cases, the ensemble's performance increases by about 3% compared to the performance of a single model. In the case of ensemble consisting of LSTM models, the benefit of ensembling is only evident for ensemble size of up to 10. After that, there is virtually no benefit to having a larger ensemble. In the case of ensemble consisting of bidirectional LSTM models, the performance of the ensemble keeps increasing even as the ensemble size gets to 50. This might be

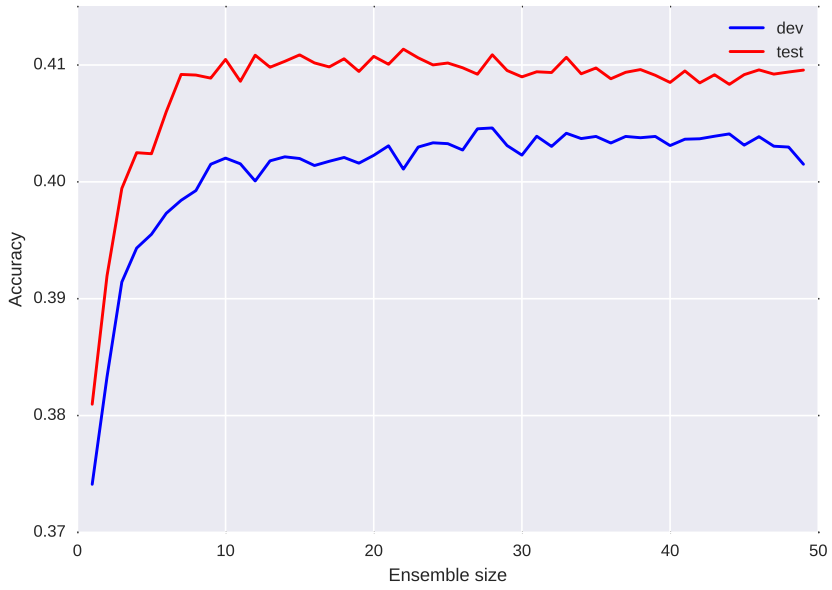

Figure 5: Accuracy of the ensemble consisting of LSTM models.

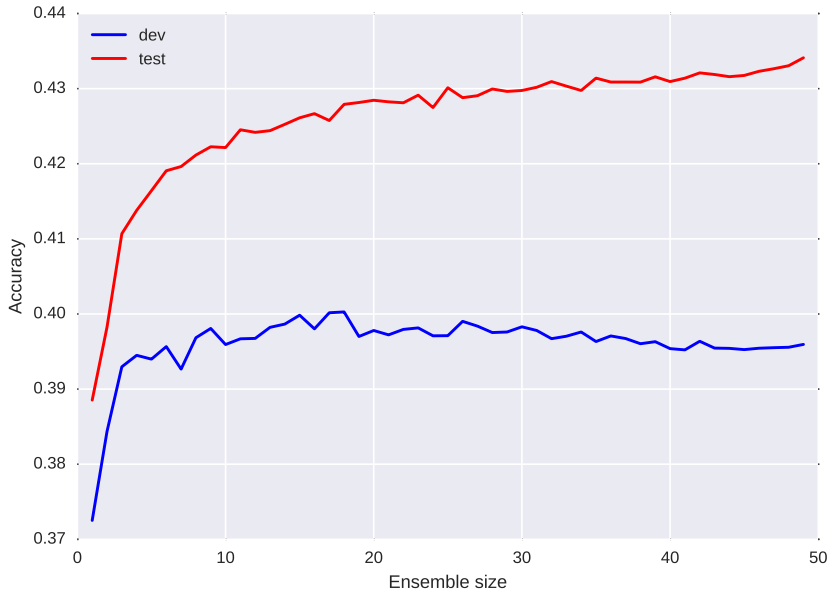

Figure 6: Accuracy of the ensemble consisting of bidirectional LSTM models.

because the bidirectional LSTM model is more powerful, with lower bias and higher variance. Thus ensembling helps reduce this higher variance while maintaining a lower bias. Therefore complex models benefit more from ensembling.

6 Conclusion

We investigate the effect of ensembling on two simple models: LSTM and bidirectional LSTM. We observe that ensembling improves the fine-grained sentiment classification accuracy by about 3%. Moreover, the more complex model, bidirectional LSTM, benefits more from ensembling. We see that the ensemble's performance keep increasing even when the ensemble size gets up to 50.

The improved performance on a small dataset (8544 training sentences) is encouraging. However, we are not able to match the performance of other models ([5], [6]) that are trained on a larger dataset (215,154 phrases). It would be interesting to see the performance of ensemble models trained on the phrases, i.e a larger effective training corpus.

In the future, we will also explore other ensemble methods. For example, boosting can be used sequentially train a larger number of neural networks. Another way to combine multiple models is to concatenate the intermediate representation learned by each model (i.e. the last LSTM cell) and apply a softmax layer on top of that to make the prediction. The weight matrix for the softmax layer will be learned from the data.

Acknowledgments

We thank Kevin Clark for allowing us to build off of his LSTM codebase and for giving valuable advice.

References

[1] Alan Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE, 2013.

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8): 1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `http://dx.doi.org/10.1162/neco.1997.9.8.1735`.

[3] Christopher Olah. Understanding lstm networks, 2015. URL `http://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-chain.png`.

[4] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[5] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Stroudsburg, PA, October 2013. Association for Computational Linguistics.

[6] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China, July 2015. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P15-1150`.

[7] Wu Zhen Zhou. Bidirectional rnn, 2016. URL `https://github.com/hycis/bidirectional_RNN/blob/master/item_lstm.png`.