Applications of Deep Learning to Sentiment Analysis of Movie Reviews

Houshmand Shirani-Mehr
Department of Management Science & Engineering
Stanford University
hshirani@stanford.edu

Abstract

Sentiment analysis is one of the main challenges in natural language processing. Recently, deep learning applications have shown impressive results across different NLP tasks. In this work, I explore performance of different deep learning architectures for semantic analysis of movie reviews, using Stanford Sentiment Treebank as the main dataset. Recurrent, Recursive, and Convolutional neural networks are implemented on the dataset and the results are compared to a baseline Naive Bayes classifier. Finally the errors are analyzed and compared. This work can act as a survey on applications of deep learning to semantic analysis.

1 Introduction

Sentiment analysis or opinion mining is the automated extraction of writer's attitude from the text [1], and is one of the major challenges in natural language processing. It has been a major point of focus for scientific community, with over 7,000 articles written on the subject [2]. As an important part of user interface, sentiment analysis engines are utilized across multiple social and review aggregation websites. However, the domain of the applications for Sentiment Analysis reaches far from that. It provides insight for businesses, giving them immediate feedback on products, and measuring the impact of their social marketing strategies [3]. In the same manner, it can be highly applicable in political campaigns, or any other platform that concerns public opinion. It even has applications to stock markets and algorithmic trading engines [4]-[5].

It should be noted that adequate sentiment analysis is not just about understanding the overall sentiment of a document or a single paragraph. For instance, in product reviews usually the author does not limit his view to a single aspect of the product. The most informational and valuable reviews are the ones that discuss different features, and provide a comprehensive list of pros and cons. Therefore, it is important to be able to extract sentiments on a very granular level, and relate each sentiment to the aspect it corresponds to. On the more advanced level, the analysis can go beyond only positive or negative attitude, and identify complex attitude types.

Even on the level of understanding a single sentiment for the whole document, sentiment analysis is not a straightforward task. Traditional approaches involve building a lexicon of words with positive and negative polarities, and identifying the attitude of the author by comparing words in the text with the lexicon [6]. In general, the baseline algorithm [7] consists of tokenization of the text, feature extraction, and classification using different classifiers such as Naive Bayes, MaxEnt, or SVM. The features used can be engineered, but mostly involve the polarity of the words according to the gathered lexicon. Supervised [8] and semi-supervised [9] approaches for building high quality lexicons have been explored in the literature.

However, traditional approaches are lacking in face of structural and cultural subtleties in the written language. For instance, negating a highly positive phrase can completely reverse its sentiment, but unless we can efficiently present the structure of the sentence in the feature set, we will not be able to

capture this effect. On a more abstract level, it will be quite challenging for a machine to understand sarcasm in a review. The classic approaches to sentiment analysis and natural language processing are heavily based on engineered features, but it is very difficult to hand-craft features to extract properties mentioned. And due to the dynamic nature of the language, those features might become obsolete in a short span of time.

Recently, deep learning algorithms have shown impressive performance in natural language processing applications including sentiment analysis across multiple datasets [10]. These models do not need to be provided with pre-defined features hand-picked by an engineer, but they can learn sophisticated features from the dataset by themselves. Although each single unit in these neural networks is fairly simple, by stacking layers of non-linear units at the back of each other, these models are capable of learning highly sophisticated decision boundaries. Words are represented in a high dimensional vector space, and the feature extraction is left to the neural network [11]. As a result, these models can map words with similar semantic and syntactic properties to nearby locations in their coordinate system, in a way which is reminiscent of understanding the meaning of words. Architectures like Recursive Neural Networks are also capable of efficiently understanding the structure of the sentences [12].These characteristics make deep learning models a natural fit for a task like sentiment analysis.

In this work, I am going to explore performance of different deep learning architectures for semantic analysis of movie reviews. First, a preliminary investigation on the dataset is done. Statistical properties of the data are explored, a Naive Bayes baseline classifier is implemented on the dataset, and the performance of this classifier is studied. Then different deep learning architecture are applied to the dataset, and their performance and errors are analyzed. Namely, Deep dense networks with no particular structure, Recurrent Neural Networks, Recursive Neural Networks, and Convolutional Neural Networks are investigated. At the end, a novel approach is explored by using bagging and particularly random forests for convolutional neural networks.

The dataset used for this work is the Stanford Sentiment Treebank dataset [13], which contains 11,855 sentence extracted from movie reviews. These sentences contain 215,154 unique phrases, and have fully labeled parse trees. The sentences are already parsed by Stanford Parser and the semantic of each phrase on the tree is provided. The dataset has five classes for its labels, and a cross-validation split of 8,544 training examples, 1,101 validation samples, and 2,210 test cases is already provided with the data. Figure 1 shows a sample of this dataset.

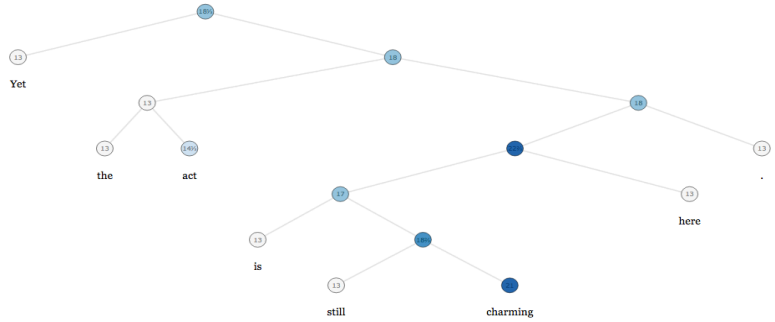

Figure 1: Structure of a sample from Stanford Sentiment Treebank dataset.

This report is organized as follows: Section 2 explains the preliminary results on the dataset using Naive Bayes classifier. In sections 3-6 different deep learning models are studied and their performance is analyzed. Finally, section 7 compares the results of the models and conclude the paper.

2 Preliminary Analysis & Baseline Results

The first step in exploring performance of different classifiers on a dataset is to identify an effective performance measure. In many cases, especially when the dataset is heavily biased towards one of the label classes, using accuracy is not the best way to measure performance. However, as shown in figure 2, the distribution of sample labels in Stanford Sentiment Treebank (SST) dataset is not dominated by any single class. Additionally, predicting none of the classes carries bigger weight

compared to the others. The distribution of labels in the validation set shows same structure. Therefore, accuracy can be used here as an effective measure to compare results of different classifiers.

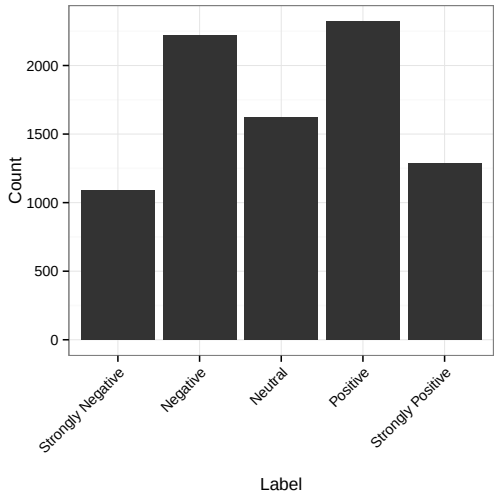

Figure 2: Distribution of labels in the training set of SST dataset.

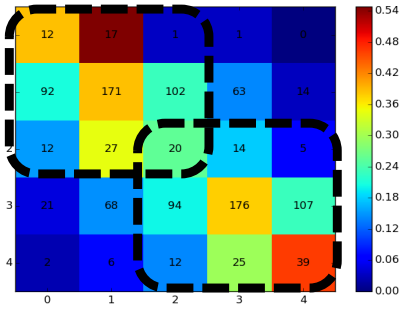

Figure 3: The confusion matrix for Naive Bayes Classifier applied to SST.

Although SST provides sentiments of phrases in the dataset as well, and we are able to train our models using that information, sentiment analysis engines are usually evaluated on the whole sentence as a unit. Therefore, in this work the final performance is measured for the sentences, which corresponds to the sentiment at the root of a tree in SST.

To have a baseline result for comparing how well the deep learning models perform, and to get a better understanding of the dataset, a Naive Bayes classifier is implemented on the data. The results of this classifier is shown in table 1. While the training accuracy is high, the test accuracy is around 40%. Figure 3 is a visualization for the confusion matrix of the classifier. The figure shows that Naive Bayes classifier performs relatively well in separating positive and negative sentiments, however it is not very successful in modeling the lower level of separation between "strong" and regular sentiment. Therefore, making the decision boundaries more complex seems like a viable option for improving the performance of the classifier. This option is explored in following sections.

3 Word2vec Averaging and Deep Dense Networks

The simplest model to apply to the sentiment analysis problem in deep learning platform is to use an average of word vectors trained by a word2vec model. This average can be perceived as a representation for the meaning of a sentence, and can be used as an input to a classifier. However, this approach is not very different from bag of words approach used in traditional algorithms, since it only concerns about single words and ignores the relations between words in the sentence. Therefore, it cannot be expected from such a model to perform well. The results in [13] show that this intuition is indeed correct, and the performance of this model is fairly distant from state-of-the-art classifiers. Therefore, I skip this model and start my implementation with more complex ones.

The next natural choice is to use a deep dense neural network. As the input, vectors of words in the sentence are fed into the model. Various options like averaging word vectors or padding the sentences were explored, yet none of them achieved satisfactory results. The models either did not converge or overfit to the data with poor performance on validation set. None of these models achieved accuracy higher than 35%. The intuition for these results is that while these models have too many parameters, they do not effectively represent the structure of the sentence and relations between words. While in theory they can represent very complex decision boundaries, their extracted features do not generalize well to the validation and test set. This motivates using different classes of neural networks, networks that using their architecture can represent the structure of the sentences in a more elegant way.

4 Recurrent Neural Networks

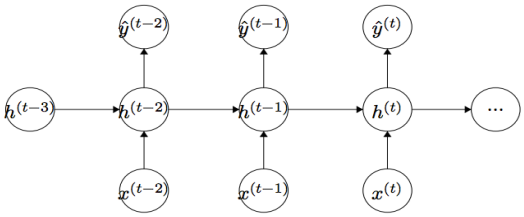

Figure 4: The structure of a Recurrent Neural Network.

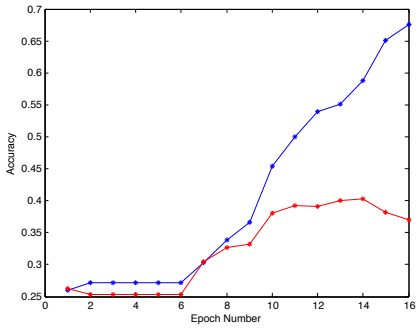

Figure 5: Learning Curve of implemented Recurrent Neural Network

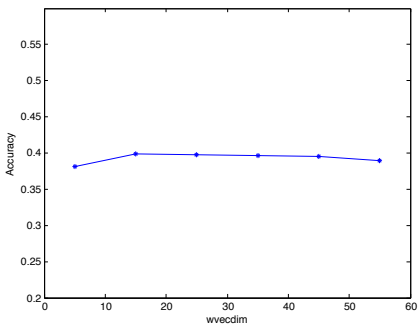

Figure 6: Recurrent Neural Network: Effect of Word Vector Dimension

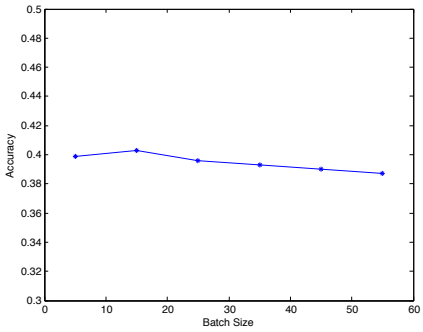

Figure 7: Recurrent Neural Network: Effect of Batch Size

Recurrent neural networks are not the most natural fit for representing sentences (Recursive neural networks are a better fit to the task for instance), however it is beneficial to explore how well they perform for classifying sentiments. Figure 4 [1] shows the structure of a vanilla recurrent neural network. The inputs are the successive word vectors from the sentence, and the outputs can be formulated as following:

$$h^{(t)} = \hat{f}(Hh^{(t-1)} + Lx^{(t)})$$
$$\hat{y}^{(t)} = \text{softmax}(Uh^{(t)})$$

Where $\hat{f}$ is the non-linearity which is initially the sigmoid function, and $\hat{y}^{(t)}$ is the prediction probability for each class. One possible direction is to use $\hat{y}$ at the last word in the sentence as the prediction for the whole sentence, since the effect of all the words have been applied to this prediction. However, this approach did not yield higher than 35% accuracy in my experimentations.

[1] From CS224D Slides

Motivated by [14], I added a pooling layer between softmax layer and the hidden layer, which increases the accuracy to 39.3% on the validation set. The pooling is done on $h(t)$ values, and mean pooling achieves almost 1% higher accuracy compared to max pooling. As a further improvement, LSTM unit was used as the non-linearity in the network. With only this change, the performance does not improve, and the model overfits due to more parameters in the LSTM unit. However, by using Dropout [19] as a better regularization technique, the model is able to achieve 40.2% accuracy on the validation set and 40.3% accuracy on the test set. This accuracy is almost the same as the baseline model.

Figure 5 shows the learning curve for the recurrent neural network model, and figures 6 and 7 show the effect of changing different hyperparameters on the accuracy of the model.

5 Recursive Neural Networks

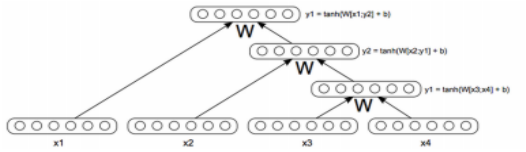

Figure 8: The structure of a Recursive Neural Network.

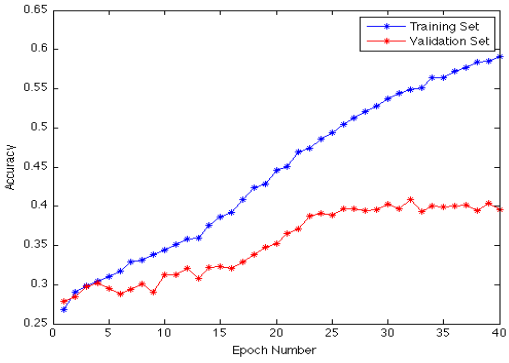

Figure 9: Recursive Neural Network: Learning Curve

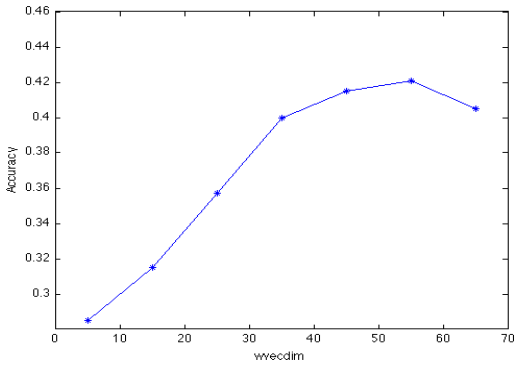

Figure 10: Recursive Neural Network: Effect of Word Vector Dimension

Figure 8 [2] shows the structure of a recursive neural network. The structure of the network is based on the structure of the parsed tree for the sentence. The vanilla model for this network can be formulated as follows:

$$h = \hat{f}(W \begin{bmatrix} h_{\text{Left}} \\ h_{\text{Right}} \end{bmatrix} + b)$$

$$\hat{y} = \text{softmax}(W^{(s)}h + b^{(s)})$$

Since this model is already studied in detail in the assignments, and specially since Convolutional Neural Networks achieve higher accuracy, I did not experiment with Recursive neural networks in extent. The learning curve and some experimentations on the hyperparameters of the model are shown in figures 9 and 10. The accuracy of the model is 42.2% on the test set, which is higher than recursive neural networks and the baseline results.

6 Convolutional Neural Networks

In convolutional neural networks, a filter with a specific window size is run over the sentence, generating different results. These results are summarized using a pooling layer to generate one vector as

[2] From CS224D Slides

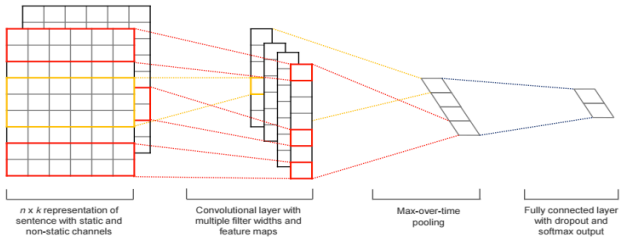

Figure 11: The structure of a Convolutional Neural Networks.

the output of the filter layer. Different filters can be applied to generate different outputs, and these outputs can be used with a softmax layer to generate prediction probabilities. Figure 11 [3] shows the structure of this network. The model can be described using following equations:

$$c_i^{(j)} = \hat{f}^{(j)}(W x_{i:i+h-1} + b)$$
$$\hat{c}^{(j)} = \max(c_1^{(j)}, c_2^{(j)}, \ldots, c_{n-h+1}^{(j)})$$
$$\hat{y} = \text{softmax}(W^{(s)}\hat{\mathbf{c}} + b^{(s)})$$

Where h is the length of the filter. For this work, I have used the model proposed by Kim [20], which uses Dropout and regularization on the size of gradients as approaches to help the model converge better.

Figure 12 shows the learning curve of the Convolutional neural network, and figure 13 shows that 50 is the local optimal dimension for word vectors used in the model.

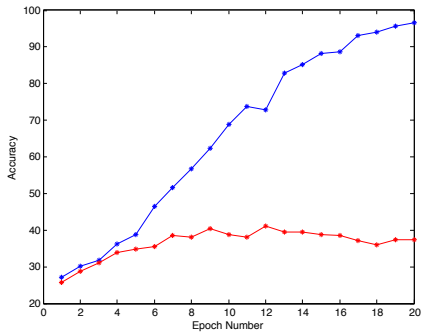

Figure 12: Convolutional Neural Network: Learning Curve

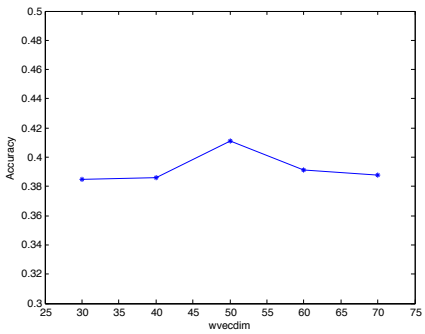

Figure 13: Convolutional Neural Network: Effect of Word Vector Dimension

While we observe a slight improvement over Recurrent neural networks, the results are not significantly better than Baseline classifier. The significant gap between the training error and test error shows that there is a serious overfitting in the model. As a solution, instead of training the word vectors along other parameters using samples, predefined 300-dimensional vectors from word2vec [4] model are used, and are kept fixed during the training phase. These vectors are trained based on a huge dataset of news articles. The resulted model shows a significant improvement in the accuracy. Figure 14 shows the learning curve for this model. The model trains very fast (highest validation accuracy is at epoch 5) and the final accuracy on test set is 46.4%.

7 Conclusion & Analysis of Results

Table 1 shows the comparison of results for different approaches explored in this work.

[3] from [20]

[4] Available from https://code.google.com/p/word2vec/

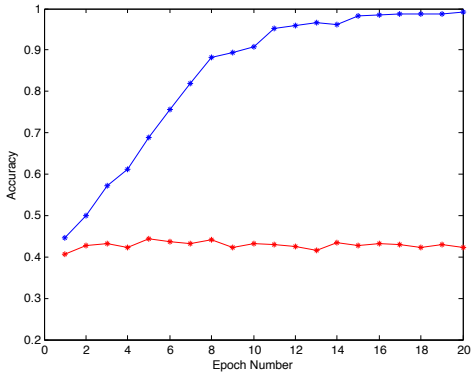

Figure 14: Convolutional Neural Network with word vectors fixed from word2vec model: Learning Curve

Recurrent neural networks are not an efficient model to represent structural and contextual properties of the sentence, and their performance is close to the baseline Naive Bayes Algorithm.

Recursive neural networks are built based on the structure of the parsed tree of sentences, therefore they can understand the relations between words in a sentence more adequately. Additionally, they can use the phrase-level sentiment labels provided with the SST dataset for their training. Therefore, we expect Recursive networks to outperform Recurrent networks and baseline results.

Convolutional neural network can be assumed as a generalized version of recursive neural networks. However, like recurrent neural networks, they have the disadvantage of losing phrase-level labels as training data. On the other hand, using word vectors from word2vec model results in a significant improvement in the performance. This change can be contributed to the fact that due to large number of parameters, neural networks have a high potential for overfitting. Therefore, they require a large amount of data in order to to find generalizable decision boundaries. Learning the word vectors along other parameters from sentence-level labels in SST dataset results in overfitting and degrade performance on the validation set. However, once we use pre-trained word2vec vectors to represent words and do not update them during the training, the overfitting decreases and the performance improves.

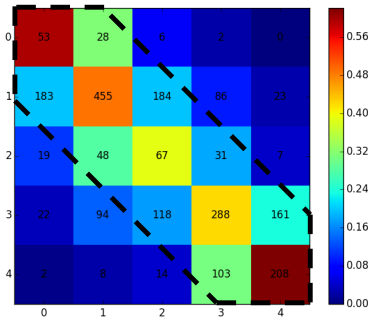

Figure 15: Confusion Matrix: Convolutional Neural Network with fixed word2vec word vectors

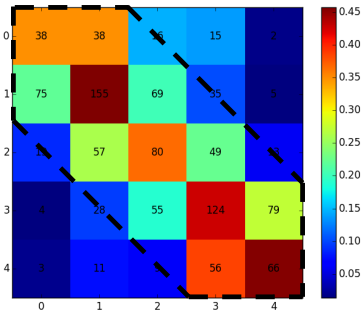

Figure 16: Confusion Matrix: Recursive Neural Network

Figures 15 and 16 show the confusion matrix of the two best model from the experimentations. Comparing to the confusion matrix for Naive Bayes, we can see that the correct predictions are distributed more evenly across different classes. Naive Bayes classifier is not as consistent as deep learning models in predicting classes on a more granular level. As mentioned before, this is due to capacity of deep neural networks in learning complex decision boundaries. While it is possible to engineer and add features in such a way that the performance of Naive Bayes classifier improves, the deep learning model extracts features by itself and gain significantly higher performance.

Model	Training Accuracy	Validation Accuracy	Test Accuracy
Naive Bayes	78.3	38.0	40.3
Recurrent Neural Network	56.8	40.2	40.3
Recursive Neural Network	54.0	38.6	42.2
Convolutional Neural Network	72.7	41.1	40.5
Convolutional Neural Network + word2vec	88.2	44.1	**46.4**

Table 1: Summary of Results

References

[1] Pang et al. (2008) Opinion mining and sentiment analysis. *Foundations and trends in information retrieval* 2.1-2: 1-135.

[2] Feldman et al. (2013) Techniques and applications for sentiment analysis. *Communications of the ACM*.

[3] Pang et al. (2008) Opinion mining and sentiment analysis. *Foundations and trends in information retrieval* 2.1-2: 1-135.

[4] Bollen et al. (2011) Twitter mood predicts the stock market. *Journal of Computational Science* 2.1: 1-8.

[5] Groenfeldt, Tom. Trading On Sentiment Analysis – A Public Relations Tool Goes To Wall Street. Editorial. *Forbes*. N.p., 28 Nov. 2011. Web.

[6] Agarwal, Basant, et al. (2015) Sentiment Analysis Using Common-Sense and Context Information. *Computational intelligence and neuroscience*.

[7] Pang et al. (2012) Thumbs up?: sentiment classification using machine learning techniques. *Proceedings of the ACL-02 conference on Empirical methods in natural language processing*-Volume 10. Association for Computational Linguistics.

[8] Baccianella et al. (2010) SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. LREC. Vol. 10.

[9] Hatzivassiloglou et al. (1997) Predicting the semantic orientation of adjectives. *Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the european chapter of the association for computational linguistics*. Association for Computational Linguistics.

[10] Collobert, Ronan, et al. (2001) Natural language processing (almost) from scratch. *The Journal of Machine Learning Research* 12: 2493-2537.

[11] Mikolov, Tomas, et al. (2013) Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*. 2013.

[12] Socher, Richard, et al. (2011) Parsing natural scenes and natural language with recursive neural networks. *Proceedings of the 28th international conference on machine learning (ICML-11)*.

[13] Socher, Richard, et al. (2013) Recursive deep models for semantic compositionality over a sentiment treebank. *Proceedings of the conference on empirical methods in natural language processing*. Vol. 1631.

[14] Graves, Alex. (2012) *Supervised sequence labelling with recurrent neural networks*. Vol. 385. Heidelberg: Springer.

[15] Bastien et al. (2012) Theano: new features and speed improvements. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

[16] Bergstra et al. (2010) Theano: a CPU and GPU math expression compiler. *In Proceedings of the Python for Scientific Computing Conference (SciPy)*.

[17] Hochreiter et al. (1997). Long short-term memory. *Neural computation*, 9(8), 1735-1780.

[18] Gers et al. (2000). Learning to forget: Continual prediction with LSTM. *Neural computation*, 2451-2471.

[19] Srivastava, Nitish, et al. (2014) Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15.1: 1929-1958.

[20] Kim, Yoon. (2014) Convolutional neural networks for sentence classification.*arXiv preprint arXiv*:1408.5882 (2014).

[21] Breiman, Leo. (2001) Random forests. *Machine learning* 45.1: 5-32.