A Comparison of DCBOW, LSTM and BTLSTM

Rob Hesselink
10587454

Raymond Koopmanschap
11925582

1 Introduction

Sentiment classification is a challenging task in natural language processing. It has seen an increase in interest since a growing portion of human communication is digital and publicly accessible. Examples of this are reviews, blog posts, twitter feeds and comment sections. Recently, several models have gained increased performance by incorporating word order and sentence structure. In this work, we replicate the Binary Tree Long Short-Term Memory unit (BTLSTM) of (Tai et al., 2015) and analyse its performance under several different conditions. It must be noted that similar models were developed simultaneously by (Le and Zuidema, 2015) and (Zhu et al., 2015). The BTLSTM is an extension of the LSTM unit, which is itself a specific version of a Recurrent Neural Network. These networks are capable of incorporating previous inputs in their assessment of the current input. This means that these models are capable of incorporating word order. The Binary Tree LSTM extends this by also incorporating grammatical structure in the form of a parse tree.

Our goal is to analyse and statistically compare the results of a simple bag-of-words (BOW) model, an LSTM and a BTLSTM. The expectation is that models that include more features will be able to make more accurate classifications. We evaluate significance by the Wilcoxon ranked sign test as suggested by Benavoli et al. (Benavoli et al., 2016). Moreover, we wish to know whether performance is dependent on the sentence length or the depth of the parse tree, as both are an indication of sentence complexity. Lastly, we investigate the effect of including subtrees in the training data, supervising the models at subsections of the sentences. Models are tested on the Stanford Sentiment Treebank (SST).

While there have been earlier investigations into deep learning models for sentiment classification by (Lee and Dernoncourt, 2016), (Lu et al., 2017), these have not included significance testing. While (Tai et al., 2015) and (Zhou et al., 2016) have investigated the effect of tree complexity, these did not include models without memory, such as the BOW model discussed here.

Our results show that BTLSTMs outperform LSTMs, which outperform a bag-of-words approach, which is in accordance with earlier results. Statistical significance testing shows that legitimate questions appear when comparing the performance between different types of models and even within a type of model. Investigations with respect to tree complexity suggest that tree depth and sentence length are poor indicators of model performance, suggesting unknown variables that govern classification complexity. Lastly, word order was shown to be not significant for the performance of LSTMs.

2 Background

We will shortly introduce the main techniques used in this paper and explain the specific models that are used.

2.1 Word Embeddings

Word embeddings in neural models were suggested by (Bengio et al., 2003) to counteract the *curse of dimensionality*. A large vocabulary represented as one-hot vectors is computationally unfeasible. These sparse representations are transformed to dense lower-dimensional representations according to some criterion. This criterion is often derived from the distributional hypothesis, which states that *similar words appear in similar contexts*. These embeddings have been shown to convey multiple forms of semantic relatedness, for example in (Mikolov, 2013).

2.2 Bag-of-Words

A BOW model discards all syntactic structure and views text as merely a combination of words. Examples include support vector machines, naive Bayes classifiers (Pang et al., 2002) or neural models. Sentences can be represented as the sum of individual embedded words, as was done by (dos Santos and Gatti, 2014) to classify short texts.

2.3 LSTM

In order to incorporate word order in our model we will be using Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997). LSTMs are build on the idea of recurrent neural networks (RNN) (Elman, 1990) (Mikolov, 2012) which is a class of neural networks where each neuron, also called hidden state, has a transition to itself, this gives the ability to handle sequences of data by recursively applying the transition function together with an activation function, allowing the network to use parts of previous words. This is often visualised as a chain where each time step a new word is put in. 1. A major shortcoming of RNNs is that by applying an activation function recursively vanishing or exploding gradients will occur (Pascanu et al., 2012). This greatly reduces the capability of learning long-term dependencies. The LSTM is introduced to handle this shortcoming. It does this by using a cell state next to the hidden state. The cell state corresponds to the long-term memory while the hidden state is used for short term memory. The cell state does not have activation functions and is thereby not prone to the vanishing and exploding gradients problem. The cell state uses the information of the hidden state together with the data input to update its state. This updating is done by incorporating a forget, input, candidate and output gate. The forget gate determines how much of the previous cell state, which stores the relevant long-term information, is forgotten. It does this by first filtering for relevant information of our current input and the previous hidden state and multiplies this information to the current cell state. Second, the input together with the candidate gate decides which current relevant information is added to the cell state. Finally, the output gate combines the cell state with the hidden state to combine the newly selected relevant information. The mathematical formulas can be found in (Tai et al., 2015) By using such a structure we allow the cell state to store relevant information for a long time which makes it possible to learn long-term dependencies.

2.4 N-ary Tree LSTM

To incorporate not only word order but also sentence structure, the LSTM will be extended by a Tree LSTM. Instead of representing the LSTM as

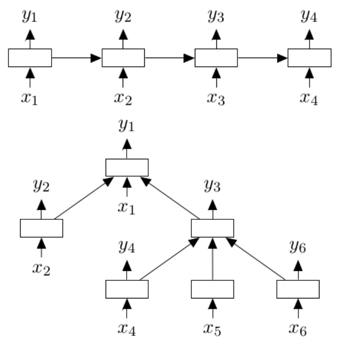


Figure 1: A comparison of LSTM (top) and N-ary Tree LSTM (bottom), courtesy of Tai et al.

a chain of LSTM units, as we did with RNNs, we can represent them by using a tree structure. The BTLSTM recursively simplifies the parse tree, until it concludes at the root node and outputs a classification. The BTLSTM combines a set of inputs from the previous layer, meaning multiple children can influence the output of the common parent. The tree structure is well suited to the hierarchical nature of natural language, where e.g. the preposition “until” and the noun “tomorrow” combine to form “until tomorrow”. Combining the LSTM units as a tree give the ability to capture syntactic properties of the sentence which can increase the correct understanding of those sentences. The mathematical details of the Tree LSTM can be found in (Tai et al., 2015).

3 Models

The following sections describes the models used in this experiment[1]. We will be using three main models for our experiments, namely the deep continuous bag-of-words (DCBOW), the standard LSTM and the binary Tree LSTM (BTLSTM) which is a specific version of the N-ary Tree LSTM explained in 2.4. The DCBOW is used mainly as a baseline to compare it to the effect of features that the other two models implement: word order for LSTM and grammatical structure for BTLSTM. All our models will use the 300

[1]Code can be found at github.com/robdhess/nlp1

dimensional pretrained Glove word embeddings (Pennington et al., 2014) as input for the first layer. The final layer consists of five units, corresponding to the sentiment classes. The cross entropy is taken as the loss function. The DCBOW consists of two hidden layers, with tanh activation functions. The LSTM model consists of a hidden layer with 168 LSTM units. The BTLSTM has a hidden layer with 150 LSTM units. All LSTM-type models include a dropout layer as defined by (Srivastava et al., 2014).

4 Experiments

The task at hand is to classify the sentiment of single sentences on a scale containing very negative, negative, neutral, positive and very positive. The data set used for this experiment is the Stanford Sentiment Treebank (SST) (Socher et al., 2013). The data was split into a training, development and validation set of 8544, 1101 and 2210 sentences respectively. Models are created and trained using PyTorch and the ADAM algorithm (Kingma and Ba, 2014). The BOW model is trained for 30000 iterations using a learning rate of $5e-4$, the LSTM-type models are trained for 25000 iterations with learning rate $3e-4$.

Models are scored on their accuracy and standard deviation, which were collected over five separate trials. The significance between different models is determined by comparing the predictions on the test data. We test significance by the non-parametric Wilcoxon ranked sign test, as suggested by (Benavoli et al., 2016). Since the multiple comparisons increase the family-wise error rate, we apply the Bonferroni correction to the significance level α which leads to

$$\alpha_c = \frac{\alpha}{m} \tag{1}$$

Here m is the number of comparisons. The resulting α_c value for the significance level 0.05 is 2.78e-4. The pair-wise Wilcoxon test is preferred over methods such as ANOVA, since the normality and sphericity requirements are unlikely to be met by these models (Demšar, 2006). The Friedman test is avoided as its conclusions depend on the pool of chosen algorithms, leading to potentially paradoxical results as seen in (Benavoli et al., 2016).

The effects of word order and subtree supervision are investigated by modifying the training data. The former scrambles sentences in the training data, removing syntactical structure. The latter adds all subtrees to the training data.

We investigate the effect of tree complexity on performance according to two definitions of complexity. The first defines complexity of a tree as the number of nodes. However, since we are dealing with binary parse trees, this is equivalent to investigating the effect of sentence length on accuracy. The second definition equates complexity to the depth of the deepest node. This is equal to the maximum amount of times a word is run through the Tree LSTM.

5 Results and Analysis

Model	Accuracy	subtree Accuracy
DCBOW	0.413 ± 0.011	0.390 ± 0.016
LSTM	0.437 ± 0.009	$\mathbf{0.439} \pm 0.003$
Tree LSTM	$\mathbf{0.455} \pm 0.003$	0.425 ± 0.009
S-LSTM	0.423 ± 0.004	

Table 1: Sentiment classification results. We report average accuracy and standard deviation over 5 runs. Best scores indicated in bold.

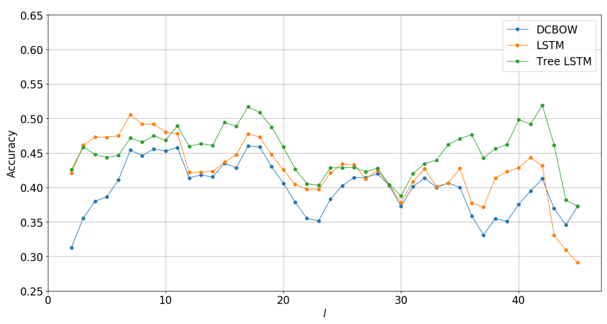
Table 1 shows the results of our models on the sentiment classification task. A general trend is that as more features are added accuracy improves and variance decreases.

5.1 Significance

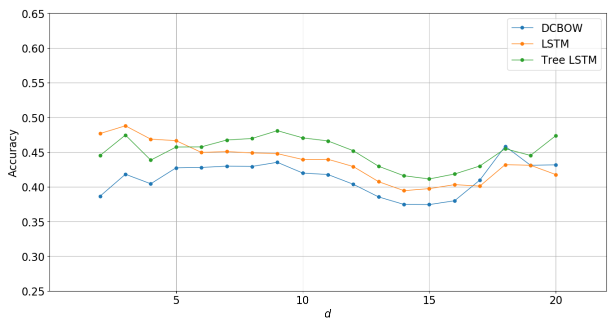
	DCBOW	LSTM	BTLSTM
DCBOW	2/10		
LSTM	2/25	3/10	
BTLSTM	2/25	9/25	1/10

Table 2: Significance testing among models. Numbers indicate the fraction of models which were found *not* to significantly differ.

Table 2 shows the result of 180 two-way comparisons using the Wilcoxon signed rank test. There are two important points here. The first is that models of different types often do not make significantly different predictions. Secondly and perhaps more interesting, models of the same type differ significantly amongst themselves. While variation on accuracy is quite low, models often disagree on the classification of the same sentence.



(a) Accuracy as a function of sentence length.



(b) Accuracy as a function of tree depth

Figure 2: Results for tree complexity. Note that values are aggregated in overlapping bins [x-2, x+2]. The final bin contains aggregated data beyond that length/depth.

5.2 Word Order

The effectiveness of the LSTM trained on scrambled sentences shows very little decrease. The increase in accuracy between BOW and LSTM is however quite large. This suggests that the LSTM does not need word order, but it needs to know which words are in the same sentence. Similar to how humans can reconstruct a scrambled sentence to its original, so do LSTMs seem able to combine words to form a coherent message.

5.3 Tree Complexity

Figure 2a shows a slight decrease in accuracy for DCBOW and LSTM. The Tree LSTM remains fairly constant, even increasing in performance near the end. This suggests that the tree structure is beneficial when faced with longer sentences. As seen in Figure 2b, models show a similar curve with respect to tree depth, although BTLSTM generally outperforms the others. However, the difference between BOW and LSTM does not grow significantly for longer sentences, suggesting that the effect of memory is limited.

All these observations are quite tentative, as noise is significant even after smoothing. In general, the curves in both figures seem to follow a very similar path. This suggests that there exist properties of the test data that all three models find difficult to capture.

5.4 Subtrees

The effect of including subtrees to the data is fairly minimal, with only the LSTM increasing in performance. The models trained on subtrees showed a very high accuracy on the training data (~ 0.7), with much lower accuracy on dev/test data. This suggests that this approach leads to overfitting and adding subtrees should therefore be treated with care.

6 Conclusion

The results have shown that there is benefit to be gained by including grammatical structure on the sentiment classification task. This is in agreement with the original results by (Tai et al., 2015). However, statistical significance testing shows that models of their own type may be dissimilar, while those of different types might be alike. This creates questions of how models should be correctly compared. Results regarding tree complexity show that complexity as defined here is a poor predictor of model accuracy. The addition of subtrees did not lead to better results, but instead to overfitting.

References

Alessio Benavoli, Giorgio Corani, and Francesca Mangili. 2016. Should we really use post-hoc tests based on mean-ranks? *The Journal of Machine Learning Research* 17(1):152–161.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* 3(Feb):1137–1155.

Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research* 7(Jan):1–30.

Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. pages 69–78.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science* 14(2):179–211. https://doi.org/10.1207/s15516709cog1402_1.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. *arXiv preprint arXiv:1503.02510* .

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. *arXiv preprint arXiv:1603.03827* .

Chi Lu, Heyan Huang, Ping Jian, Dan Wang, and Yi-Di Guo. 2017. A p-lstm neural network for sentiment classification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, pages 524–533.

Tomas Mikolov. 2012. *Statistical Language Models Based on Neural Networks*. Ph.D. thesis, Brno University of Technology. http://www.fit.vutbr.cz/ imikolov/rnnlm/thesis.pdf.

Thomas Yih Wen-Tau Zweig Geoffrey Mikolov. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics*. pages 746–751.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, pages 79–86.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *CoRR, abs/1211.5063* .

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. 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.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* .

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639* .

Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *International Conference on Machine Learning*. pages 1604–1612.