Sentiment Classification using Neural Network Models

Tsatsral Mendsuren
14530775

Didier Merk
11037172

1 Introduction

Sentiment classification is a technique of classifying emotions or sentiments of text using machine learning techniques. It is useful for business intelligence applications, recommender systems (Pang et al., 2002) and can be a powerful marketing tool (Rambocas, 2013). In this paper multiple different variations of bag-of-words (BOW) and Long Short-Term Memory (LSTM) neural networks are trained and tested on the Stanford Sentiment Treebank dataset (Socher et al., 2013).

Each model attempts to accurately classify the sentiment of an unseen movie review as one of 5 classes: very negative, negative, neutral, positive or very positive. The goal of this paper is to find which models have the highest accuracy and which aspects of sentence structure are important in sentiment classification of text. To accomplish this, the following five questions will be answered:

1. How important is word order for sentiment classification?
2. Does a tree structure help to achieve a higher accuracy in sentiment classification?
3. How does sentence length affect the performance of the sentiment classification models?
4. Does the performance of models improve when supervising sentiment at each node in the tree?
5. How does the N-ary Tree LSTM compare to the Child-Sum Tree LSTM?

The first question is answered by introducing variations of bag-of-words models, which do not incorporate word order in their training. These are compared to different LSTM models, which are able to incorporate word order and sentence structure. Zaremba et al. (2014) found that in tasks such as language modelling and translation (Luong et al., 2014) these LSTMs have higher accuracies than the more basic BOW models. The expectation is that the LSTM models will similarly outperform BOW models in sentiment analysis tasks.

Tai et al. (2015) found that making use of the syntactic tree-like structure of sentence in a Tree-LSTM model helped to achieve higher accuracy compared to the previous *vanilla* LSTM. In this paper the Tree-LSTM will be compared to the LSTM model to answer whether a tree structure indeed results in a better performance.

Similarly Tai et al. (2015) found that LSTM models perform better on shorter sentences than on long sentences. To test this hypothesis, the accuracy of the models will be tested on sentences of short, middle and long lengths. Additionally, the paper compared the *N*-ary Tree LSTM model to the Child-Sum Tree LSTM model and found that the *N*-ary Tree LSTM outperformed the Child-Sum Tree LSTM on sentiment classification tasks. This research expects to produce the same results.

Whether performance of models improves when supervising sentiment at each node in the tree has not been discussed in the literature yet, and this research aims to answer that question by treating every node in each tree as a separate tree.

Together, the answers to these questions demonstrate a global idea behind why some models perform better than others and which aspects in sentiment classification are important.

The best performing model for sentiment classification of movie reviews is the Child-Sum Tree-LSTM model. The Subtree-LSTM and binary Tree-LSTM follow closely with a slightly lower accuracy; after these, the mini-batch LSTM and regular

LSTM model follow. Each LSTM model has a higher accuracy than each BOW model, indicating that word order is important for sentiment classification. The Tree models also outperform all non-tree models displaying that tree structure helps achieving a higher accuracy. In addition, all models except the standard BOW model performs better on shorter sentences.

2 Background

In this section the main techniques used in this paper will be covered and a short explanation of the techniques and the relationships between them will be given.

2.1 Word embeddings

A common technique in NLP research that is central to this paper is that of word embeddings. In word embeddings each word is associated with a multi-dimensional vector, which represents the meaning or sentiment conveyed by a word. The words with a similar meaning, will have similar word embeddings. In this paper, the text used for training and testing is always represented by the embeddings of the words.

2.2 Bag-of-Words models

The first variations of models are implemented with Bag-of-Words methods to parse the sentences for sentiment analysis. The models uses different architectures, but they are all used to learn the weights of word embeddings of the training data in relation to the sentiments. All the word embeddings in a sentence are then summed together with a bias vector. However, the summing of these vectors means that the model does not incorporate word order into its prediction. Finally the argmax function can be applied to predict the most likely sentiment. Besides the baseline BOW model, two variations of BOW models are implemented in this research to learn more complex functions, including increasing the size of word embeddings (CBOW) and bringing in non-linearility (Deep CBOW). A detailed description of all models will be given in section 3.

2.3 Long Short-Term Memory models

BOW models do not take word order into account in their sentiment classification, making them unable to incorporate the dependencies in sequence data. LSTM models are able to capture the order of words in a sentence and the dependencies between words.

Greff et al. (2017) describe the *vanilla* LSTM that accomplishes this by having a cell state that runs through the whole training of a sequence of data where information just flows and maintains. Another cell state in the LSTM is the hidden state, which incorporates four non-linear layers or gates: the block input layer, input gate layer, forget gate layer and output gate layer with either a sigmoid or tanh activation function. The hidden state non-linearly transforms the current input data and adds the transformed information to the cell state through element-wise operation. When the next instance in the sequence data is processed, its final hidden state is both influenced by its own hidden state and the hidden state, cell state of the previous iteration. This explains why the LSTM module can maintain and ”memorize” the long-term dependencies. In human language sentences, the word in a sentence flows in an order comparable to sequential data and the order of the words does influence the semantic meaning of the sentence. Therefore it makes sense to use LSTM models to handle the dependencies in our data.

Tree-LSTM models are different in the sense that they make use of the syntactic structure of a sentence. This means that the dependencies in the data are parsed using a tree structure (Tai et al., 2015). Closely dependent instances are treated as the children of the same node; a set of instances can be established as a tree structure with a dependency decreasing from the bottom to the top. An LSTM cell is then used to encode the information of the data from bottom to top of the tree. The tree-like structure may be helpful for sentiment classification, since natural sentences may be intuitively interpreted by combining the meaning of smaller phrases into larger morphological structures before combining them into sentences. In this report a Binary-Tree LSTM and Child-Sum Tree LSTM are analysed.

3 Models

All models are implemented with PyTorch (Paszke et al., 2019). The cross-entropy loss is used as the loss function for all models. Random initialization is used for the weights.

3.1 BOW

The baseline BOW model uses word embeddings of the size of the sentiment classes, which has dimension of size 5. The training enables the model to learn the weights of the word embeddings in relation to the sentiment of a sentence treated as a bag of words. Therefore, the prediction of a sentence's sentiment is the most probable sentiment class in the sum of the predictions per word.

3.2 CBOW

In order to grasp more information from the data, larger word embeddings of size 300 are used. Hence, a linear layer of size (300, 5) is applied after the embeddings in order to project the results to the number of sentiment classes.

3.3 DeepCBOW

On top of the CBOW model, non-linearity is added in the Deep CBOW model, to learn a more complex mapping function from the word embeddings to the sentiment of the sentence. Two hidden linear layers of output size 100 followed by a Tanh activation function are added after the word embeddings of size 300. A linear layer of size (100, 5) is applied after the hidden layers for an output of the correct size.

3.4 LSTM

Similar to the other models, a word embedding of size 300 is used. After the embeddings, an LSTM cell is implemented using the PyTorch LSTM formulation. Specifically, 8 separated linear layers are used for the input gate, forget gate, block input and output gate which are used for the input and the previous hidden state. The output size of these linear layers is 168. To predict the sentiment of a sentence, the LSTM model is fed one word at a time, hence, the hidden state incorporates the information of the whole sentence when there is no word left. The last hidden state is, therefore, used for sentiment prediction.

3.5 Binary-Tree LSTM

This model is based on the model described by Tai et al. (2015). Unlike the previous LSTM model, in the Tree-LSTM a sentence is parsed in a binary tree structure which combines words into natural phrases, then into longer phrases and finally into the whole sentence. At a certain node, the main transformation of the previous hidden state is done on both left-child and right-child; the final hidden state is the sum of the hidden states from both children. The hidden size used is 150. Similarly, the last hidden state is used for sentiment prediction.

3.6 Child-Sum Tree-LSTM

The Child-Sum Tree LSTM cell is a different variant of the Tree-LSTM. Instead of giving hidden states of the left and right child their own weights at a node, the Child-Sum Tree LSTM first sums the two hidden states and then performs the main LSTM transformations and activations on the summed hidden state. Again, a hidden size of 150 is used and the last hidden state is used for sentiment prediction.

4 Experiments

The experiments were designed to answer the research questions in the introduction. A comparison was made between the test performance of BOW model variations and LSTM models; Tree-LSTM model variations and the primary LSTM model; and Child-Sum Tree-LSTM and Binary-Tree LSTM. The test dataset was divided into three subsets based on sentence length. To supervise sentiment at each node in the tree, all possible subtrees of the sentences in the training data were added to the training. The research questions were answered by comparing the test performance of the different models.

4.1 Data

The dataset used for training and testing in this paper is the Stanford Sentiment Treebank (SST) dataset. For a sentence, this dataset provides the tokens, the binary tree structure and the sentiment score of the sentence. The sizes of the training set, validation set and the test set are 8544, 1101, 2210. For the BOW- and CBOW model, no pre-trained word embedding is used, so the vocabulary for word embeddings are also extracted from the SST dataset.

Word embeddings learnt from a relatively small dataset such as SST, may have a result that will not generalise well enough. To combat this, word embeddings pre-trained on a larger dataset are imported. Together with corresponding vocabulary from the Word2Vector tool (Mikolov et al., 2013), trained using a skipgram model. These pre-trained word embeddings are used for the training of the Deep CBOW model and the LSTM model variations.

4.2 Training and evaluation

To train the BOW, CBOW and DeepCBOW and LSTM model, the Adam optimizer with learning

Model	**Iterations**	**3-Run acc.**	**Short sent.**	**Medium sent.**	**Long sent.**
BOW	28000	0.26 (0.015)	0.25 (0.012)	0.26 (0.020)	0.26 (0.016)
CBOW	26666	0.35 (0.015)	0.35 (0.012)	0.35 (0.003)	0.35 (0.025)
Deep CBOW	27000	0.37 (0.022)	0.39 (0.011)	0.37 (0.029)	0.36 (0.023)
PT Deep CBOW	11666	0.43 (0.007)	0.48 (0.011)	0.43 (0.014)	0.41 (0.017)
LSTM	26000	0.45 (0.001)	0.48 (0.005)	0.46 (0.004)	0.44 (0.008)
Batched LSTM	5133	0.46 (0.011)	0.49 (0.016)	0.47 (0.013)	0.43 (0.008)
Tree-LSTM	2466	0.46 (0.009)	0.47 (0.009)	0.48 (0.006)	0.44 (0.011)
Subtree-LSTM	17000	0.46 (0.009)	0.49 (0.007)	0.47 (0.013)	0.44 (0.010)
CS Tree-LSTM	**4266**	**0.47 (0.005)**	**0.50 (0.007)**	**0.48 (0.001)**	**0.46 (0.009)**

Table 1: Accuracies of the BOW and LSTM models and their variations. Amount of iterations until convergence, 3-run average accuracy on full, short, medium and long sentences are displayed, with the best model in bold.

rate of 0.0005 is used. The models run for 30000 iterations and the test performance of the model with highest validation accuracy is recorded (where the evaluation is done every 1000 iterations). The LSTM and Tree LSTM models with batch size 25 are all trained for 7000 iterations and evaluated every 200 iterations, given that they tend to converge faster. The LSTM models are also trained using the Adam optimizer, with a learning rate of 0.0003. The evaluation metric for test performance is the accuracy. The average accuracy of 3 runs will be calculated.

5 Results and Analysis

The full results are summarised in table 1. The model with the highest accuracy and best suited for sentiment classification on the SST dataset is the Child-Sum Tree-LSTM model with an accuracy of 47% (standard deviation of 0.005).

All LSTM models outperform all BOW models. As discussed, the LSTM models incorporate word order, where the BOW models do not. This indicates that word order is an important factor in sentiment classification. Intuitively this makes sense, since word order is an important aspect of word meaning and the order of words can change the meaning of a sentence.

The Tree-LSTM models (Tree, Subtree and Child-Sum Tree) all perform better than the LSTM models that did not incorporate the syntactic structure. This seems to be evidence that using the tree structure helps achieving a higher accuracy.

In addition, all models except the first BOW model perform better on short sentences than on long sentences. A reason for this might be that the morphological structures of short sentences tend to be more consistent with the sentiment of the whole sentence. Another reason is that short sentences are less complex, and their morphological and semantic information can simply be learnt better.

The performance of the Tree-LSTM does not improve when supervising sentiment at each node in the tree. A possible reason for this might be that sub-trees are in fact incomplete sentences, meaning the model trained on them will not generalize well to complete sentences.

The Child-Sum Tree-LSTM performs better than the Binary Tree-LSTM. This result does not correspond with previous studies that found the constituency Tree-LSTM to outperform Child-Sum Tree-LSTM. A possible explanation for this is that the previous study trained the Child-Sum Tree-LSTM on far less data than the Binary Tree-LSTM (Tai et al., 2015).

6 Conclusion

It can be concluded that LSTM models are a significant improvement over Bag-of-Words models for sentiment classification problems. Every LSTM variation outperforms every Bag-of-Words variation, word order and syntactic structure seems to be very important.

The best model tested in this paper is the Child-Sum Tree-LSTM model with an accuracy of 47%, which corresponds to the literature. The other results were not unexpected and correspond to earlier studies done in sentiment analysis and other language research fields.

In recent years the new transformer architecture has become the state of the art in NLP research (Wolf et al., 2020). A next step in this research could be to compare the LSTM models to this transformer architecture in sentiment analysis.

References

Klaus Greff, Rupesh K. Srivastava, Jan Koutnik, Bas R. Steunebrink, and Jurgen Schmidhuber. 2017. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232.

Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2014. Addressing the rare word problem in neural machine translation.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Meena Rambocas. 2013. Marketing research: The role of sentiment analysis. *FEP WORKING PAPER SERIES*.

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, Seattle, Washington, USA. Association for Computational Linguistics.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization.