NLP1 Practical II
Sentence Sentiment Classification with Neural Models

Jochem Soons
11327030
jochem.soons@student.uva.nl

Darius Barsony
11234342
darius.barsony@student.uva.nl

1 Introduction

Automatically identifying sentiment from text is an application within Natural Language Processing that has recently gained much attention. Applications of sentiment analysis, also termed opinion mining, include hate speech detection on social media, brand monitoring and reputation management, and product analysis (Feldman, 2013). In this report, we consider the task of sentence-level sentiment analysis, meaning that our main task is to extract sentiment from sentences correctly.

We will analyze and compare performances of a variety of neural network techniques on the task of sentiment classification and we will deploy these techniques on the Stanford Sentiment Treebank (SST) dataset (Socher et al., 2013). This dataset consists of sentences that are classified in 5 different sentiment categories. The techniques discussed in this report can be categorized into two main approaches. Firstly, variants of a general Bag-of-Words (BOW) modeling approach for sentiment analysis (Pang et al., 2002) will be investigated. Secondly, there will be looked into implementing Long short-term memory (LSTM) networks (Greff et al., 2016), as well as using a variation of LSTMs, namely Tree-LSTMs, that can encode syntactic information of sentences (Tai et al., 2015).

We will explore and analyze various aspects of the posed task of sentence sentiment classification. This report aims to answer the following research questions:

RQ1. What is the importance of word order for the task of sentiment classification?

RQ2. Does a Tree-LSTM improve performance compared with using regular LSTMs?

RQ3. How does performance depend on the length of sentences to be classified?

RQ4. Will the supervision of the sentiment in every tree node improve performance?

RQ5. How does performance differ when using Word2Vec or GloVe pretrained embeddings?

RQ6. How does performance compare between using a binary Tree-LSTM and a Child-Sum Tree-LSTM?

We will briefly motivate the importance of each question, after which we will also express some expectations. A detailed explanation of our methods to answer all questions follows in section 3.

Firstly, concerning RQ1, word order accounts for the context that the words usually occur in, and it is interesting to investigate whether this information holds any value for the classification task at hand. The BOW models do not account for this information, while LSTMs are designed to incorporate sequence information. Therefore, we expect the LSTMs to outperform the more straightforward BOW methods.

Secondly, previous research has already highlighted the potential of using syntactic information for semantic analysis (Vilares et al., 2015). Therefore, it is interesting to examine differences in performance of LSTMs and Tree-LSTMs as stated in RQ2. We expect that the syntactical information will benefit Tree-LSTMs to outperform regular LSTMs.

Thirdly, following RQ3, lengths of sentences can vary strongly, which might be a factor that could significantly impact performance. As shorter sentences are often more ”to the point” than longer sentences, we expect the models to perform relatively better when sentence length is short.

Fourthly, as syntactic tree structures contain semantic label information for every node, we investigate the effect of using this extra information for increased supervision (RQ4). As the amount of training data increases significantly from using node-level supervision, we expect the Tree-LSTM's performance to increase.

Fifthly, looking at RQ5, there are different popular words embeddings to be found in literature

one can implement for our task. Therefore, we find it interesting to compare the performance of two widely used pretrained embeddings.

Lastly, we implement two different variants of the Tree-LSTM technique, as mentioned in RQ6, such that we can further investigate the potential of applying LSTM-based methods that can exploit syntactical information of sentences.

2 Background

BOW models and word embeddings. Firstly, we consider the most straightforward **BOW** model possible. Here, each word in the model is represented by an embedding vector that contains the sentiment conveyed by the word. These vectors are of dimension $d = 5$, equal to the number of classes we have. To perform classification, vectors of words within the sentence are summed and an argmax operation is performed, resulting in the predicted sentiment class of the sentence. Secondly, we implement a Continous BOW model (**CBOW**), which is similar to the simple BOW model, only now word embeddings are of dimension $d = 300$ (Mikolov et al., 2013). Using a higher dimensionality, we can learn more aspects of each word. Thirdly, we deploy a **DeepCBOW** model that learns word embeddings of $d = 300$ in a more complex manner, through the addition of extra layers and non-linear activation functions. In the previous BOW models, all weights are initialized randomly. However, instead of randomly initializing the weights we also implement a Deep CBOW model using the pretrained GloVe (Pennington et al., 2014) word embeddings and the pretrained Word2Vec (Mikolov et al., 2013) embeddings with $d = 300$ (**PT-DeepCBOW**). These embeddings help the model start from a point where similar words are already close to one another in the distributional semantic space.

Long short-term memory (LSTM). LSTM's are a special type of Recurrent Neural Network (RNN) that are capable of learning long-term dependencies of sequential input (Hochreiter and Schmidhuber, 1997). LSTMs have become widely used for various NLP tasks in the domain of language modelling (Mikolov et al., 2010; Sundermeyer et al., 2012), parsing (Kiperwasser and Goldberg, 2016), machine translation (Chorowski et al., 2015), image captioning (Bernardi et al., 2016), visual question answering (Antol et al., 2015) and many other tasks. More specifically, LSTM networks are an adaptation to vanilla RNN networks by introducing several gating mechanisms within each LSTM cell, thereby explicitly modeling what to forget and remember from the sequence of input. By having this flow of information through the inner cells of the network, LSTMs overcome the problem of vanishing gradients and are capable of learning long-term dependencies in data (Greff et al., 2017).

Tree-LSTM. An important extension of the original LSTM model is the Tree-LSTM model. (Tai et al., 2015) introduces an extension to the original linearly structured network of the original LSTM model. Tree-LSTM's have shown remarkable improvement on baselines of LSTM models on the task of sentiment classification (Tai et al., 2015).

Tree-LSTM's unlike regular LSTMs are structured in a tree, which makes them useful to exploit the tree-like structure of parsed sentences. Multiple variations of the Tree-LSTM exist. Here the Child-Sum Tree-LSTM's and N-ary Tree-LSTMs are considered. Both versions allow for incorporating information provided by the children of every node in the tree.

In general, Tree-LSTM's traverse the tree from leaves to the parent node. The difference between regular LSTMs is that gating vectors and memory cell updates are now dependent on the states of possibly many children (in this research paper, only binary Tree-LSTMs are considered but note that it could be many). Additionally, Tree-LSTMs have a forget gate layer for every child instead of having one forget gate layer. This allows for the model to learn to forget different information per child.

In N-ary Tree-LSTMs, the gates are the same as in the original LSTM except that now each child has its own parameter matrix. In the Child-Sum Tree-LSTM, the output of the hidden layer is computed by summing over parameters of all the hidden layers that are traversed when reaching the parent node. For both the N-ary Tree-LSTM and the Child-sum LSTM, the model simply becomes a chain if the cells of the model go from top to bottom and the number of children per parent node is simply 1.

3 Methods

In this section, we will discuss implementation details of our models, training procedure, the dataset and experimental setup.

Model architectures. For all methods except the simple BOW method, we use word embeddings

of dimension $d = 300$. For the DeepCBOW and PT-DeepCBOW models, we implemented the feed-forward network with linear layers of size (300, 100), (100, 100) and (100, 5), where we include bias vectors for each layer and implement Tanh activation functions between every layer. For the vanilla LSTM model, we implement every LSTM weight with size (300, 168), where we use pre-trained word embeddings that we fix (i.e. we do not perform finetuning), and we have a final output layer of size (168, 5). Because a strong expressive model like the LSTM is likely to overfit the training data on small datasets such as SST, we implement a dropout layer with probability 0.5 before the final output layer. Finally, for the TreeLSTM model, each gate weight within the LSTM is of size (150, 300), resulting of a total reduce layer of size (750, 300). Again, a dropout layer is applied with probability 0.5 before outputting the class scores.

Dataset. We implement our models on the Stanford Sentiment Treebank (SST) dataset (Socher et al., 2013), consisting of sentences that are classified in 5 different sentiment categories (”very negative”, ”negative”, ”neutral”, ”positive”, ”very positive”). After splitting the data, the training, validation and test set consist of 8544, 1101 and 2210 samples, respectively. Additional statistics regarding the SST dataset can be found the appendix.

Training procedure. We use stochastic gradient descent (SGD) to optimize using the Adam optimizer (Kingma and Ba, 2014), based on Cross-Entropy loss. Moreover, to speed up training, we feed all LSTM models minibatches of size 25, where we make use of padding to deal with sentences of inconsistent length. All BOW methods are training using a learning rate of 0.0005 and all LSTM methods have a learning rate of $2e-4$. We train all BOW methods for 30.000 iterations, all LSTM methods for 25.000 iterations, we evaluate on a validation set every 500 iterations and subsequently pick the best performing model during training to evaluate on a held out test set. Our evaluation metric is classification accuracy. To produce robust results, we evaluate all models on three separate runs using different seeds.

Experiments. In order to answer the research questions posited in the introduction various experiments were conducted.RQ1 and RQ2 can be answered by comparing accuracy scores for all models. Additionally we further test RQ1 by also conducting an experiment in which we randomly permute our training input, so that we can validate what the actual effect is of losing word order information. Secondly, to find an answer to RQ3, we split the test dataset in four bins, determined by sentence lengths, and subsequently compare performance of all models. Thirdly, to investigate RQ4, we treat every node in the syntactic tree of sentences as a separate tree. To do this, we create a function that extracts all subtrees given a treestring, and we subsequently add these subtrees to our training set. Fourthly, to examine RQ5, we implement all methods that use pretrained embeddings both with Word2Vec and GloVe vectors. Lastly, to address RQ6, we implement both the binary Tree and Child-Sum LSTM and compare performances.

4 Results and Analysis

We will now analyze and discuss our results. Accuracy results for all the different models using Word2Vec and GloVe vectors can be found in tables 1 and 2, respectively. We will discuss our results in the same order as our research questions.

First, we notice that the LSTM based models clearly outperform the BOW-based models, with the exception of the PT-DeepCBOW model that performs only slightly worse. This is in line with what we expected, and this result strongly hints towards word order being of great importance for the task of sentiment classification. However, from both tables it is also noticeable that the LSTM with randomly shuffled input performs almost as good as the LSTM trained on regular data. Therefore, we can conclude that word order does not seem to be of great importance for the classification task on the SST dataset. Instead, we hypothesize that the leap in performance of the LSTM and PT-DeepCBOW models is due to the usage of the pretrained embeddings and the more complex modelling structure inherent to LSTMs.

Second, we find that using a Tree-LSTMs does improve results when compared to their vanilla counterpart. This shows that exploitation of syntactical information might be beneficial for the task of sentiment analysis. However, results are not too convincing as there is only a small improvement of accuracy.

Third, analyze the effect of sentence length on performance by looking at the averaged accuracy scores plotted for different sentence lengths, in figure 1. We notice the LSTM models outperforming the other models in almost every sentence length

group, with the exception of short sentences in which the PT-DeepCBOW model performs best. Relative differences of performance between models are small. However, it is noticeable that general performance of all models tends to decrease when sentence length increases. This is in line with our expectation mentioned in section 1.

Fourth, we notice that implementing node supervision on the Tree-LSTM clearly improves performance, especially when used with GloVe embeddings (highest overall score). We reason that happens because this method of treating every node in the syntactic tree as a separate tree ensures that we enlarge the dataset significantly, which ensures that the Tree-LSTM learns the more intricate structure of sentences.

Fifth, when comparing results of tables 1 and 2, we notice no significant differences in results, except for the high score using node supervision in combination with GloVe embeddings. This shows that for our task of sentence-level sentiment analysis, it does not seem to much if we apply either Word2Vec or GloVe word embeddings.

Sixth, when comparing performances of the Binary Tree-LSTM and the Child-Sum Tree-LSTM, we notice that the former slightly outperforms the latter, albeit with small margins. Therefore, we can conclude that it does not bring significant benefits to sum children nodes instead of concatenating.

Model	**Acc. + std. (%)**
BOW	24.30 ± 0.99
CBOW	32.38 ± 1.39
DeepCBOW	37.16 ± 1.30
PT-DeepCBOW	44.65 ± 0.74
LSTM	45.81 ± 0.26
LSTM (random shuffle)	45.10 ± 0.48
Binary Tree-LSTM	47.15 ± 0.50
Tree-LSTM (node superv.)	48.21 ± 0.51
Child-Sum Tree-LSTM	46.30 ± 0.80

Table 1: Averaged accuracy scores of all models using Word2Vec embeddings (n=3).

Model	**Acc. + std. (%)**
PT-DeepCBOW	43.24 ± 1.03
LSTM	46.97 ± 1.02
LSTM (random shuffle)	46.43 ± 0.36
Binary Tree-LSTM	47.47 ± 1.10
Tree-LSTM (node superv.)	50.58 ± 0.72
Child-Sum Tree-LSTM	46.86 ± 0.47

Table 2: Averaged accuracy scores of all models using GloVe embeddings (n=3).

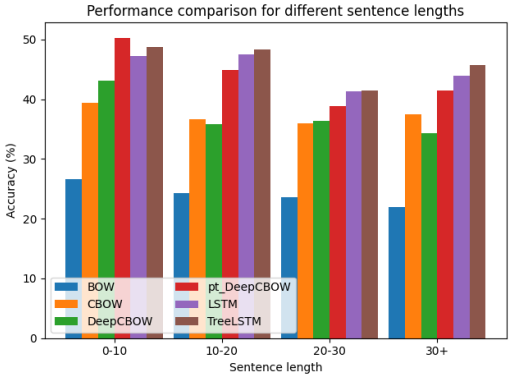

Figure 1: Averaged accuracy scores of all models plotted for different sentence lengths (using Word2Vec embeddings, n=3).

5 Conclusion

In this report, we have implemented various neural models for the task of sentence-level sentiment classification. We found that using more complex LSTM-based methods increase performance when compared to simpler BOW-based methods. Moreover, we noticed that using pretrained word embeddings offered the biggest improvement in accuracy. Surprisingly, and not entirely in line with our expectations and previous research, explicitly modeling of word order did not seem to offer much improvement of performance. Incorporating syntactic information also did not offer that much improvement, except for the case where we implemented increased supervision on every node in the tree. We reason this might be because the limited size of our dataset, and we hypothesize that benefits of using LSTMs will become more apparent with larger and more complex datasets.

Because of our findings, we recommend implementing the models discussed in this report on a larger dataset that might be more suited for neural models. Also, we feel that future work might serve to investigate why models experience more difficulties when being applied to longer sentences, as LSTMs should be able to model long-term dependencies. Lastly, we believe that a qualitative analysis that investigates what kind of classification errors are made by a model can also be of importance for future research. In the end, sentiment analysis being applied in real-life has real-life effects that can possibly harm human beings, and therefore, we believe that simply comparing accuracy scores is not enough to produce meaningful results.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank. 2016. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *Journal of Artificial Intelligence Research*, 55:409–442.

Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. *arXiv preprint arXiv:1506.07503*.

Ronen Feldman. 2013. Techniques and applications for sentiment analysis. *Communications of the ACM*, 56(4):82–89.

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2016. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232.

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.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

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

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. *arXiv preprint cs/0205070*.

Jeffrey Pennington, Richard Socher, and Christopher D 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 Y 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.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*.

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*.

David Vilares, Miguel A Alonso, and Carlos Gómez-Rodríguez. 2015. On the usefulness of lexical and syntactic processing in polarity classification of t witter messages. *Journal of the Association for Information Science and Technology*, 66(9):1799–1816.

A Appendices

A.1 Data statistics

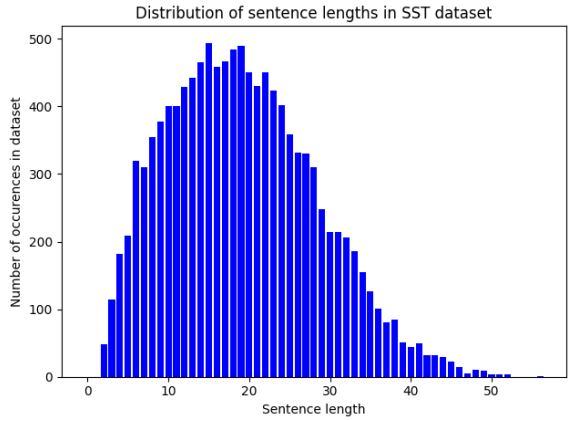

Figure 2: Bar plot of number of occurences per sentence length in the SST dataset.

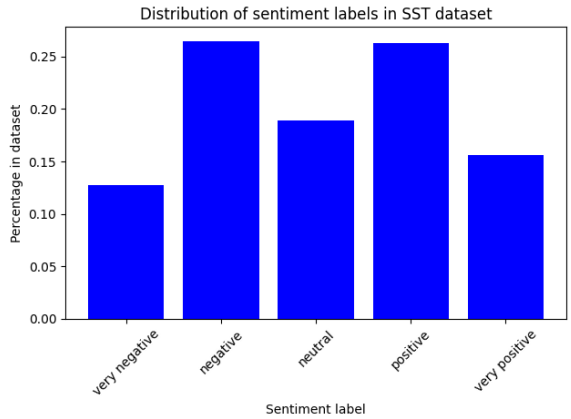

Figure 3: Barplot of fraction of sentences having one of the five sentiments as label.