NLP 1: Report on Neural Sentiment Classification

Dirk Hoesktra
University of Amsterdam
dirk.hoekstra@student.uva.nl

Philipp Lintl
University of Amsterdam
philipp.lintl@student.uva.nl

Abstract

This paper compares several neural sentiment classification methods on sentences. It is central to identify features, such as word order, sentence structure or sentence length, that are crucial to performance. We pursue this by applying several models that specifically take into account some of those features on the Sentiment Treebank task. Our findings mostly coincide with the original papers introducing the models. Namely, that especially considering word order and to some extent syntactic order boosts performance. Furtherly using pretrained word embeddings increases accuracy and for some models the sentence length is relevant to its classification quality.

1 Introduction

This paper deals with identifying sentiments within language, which can be of many forms, such as positive, negative or sad. In fact, sentiment detection is becoming an increasingly relevant NLP task, as the amount of textual data grows exponentially with an advancing digitization and the rise of the internet.(1) Relevant applications of sentiment detection include the analysis of product reviews, opinions on social media, newspaper articles or even travel destinations. (3)
Traditionally, rather basic supervised text classification techniques, such as Naive Bayes or Support Vector Machines have been succesfully applied. (8) However, as they mostly rely on the word order insensitive Bag-of-Words (BOW) assumption, shortcomings have been noticed. For once there is a computational limitation, as BOW based models use the entire vocabulary as feature space, which is highly sparse. Secondly, semantical comprehension can not be exhausted, as word- and syntactic order are not taken into account and thus for instance negations or long distance dependencies are not considered. (2)
Neural models, on the other hand inherit dimension reduction by using dense representation vectors. In that regard we apply order insensitive models. Sequence models that use sentence representations which incorporate the word order, are considered next. Models of that category rely on recurrent neural networks (RNNs), due to their sequencial processing of input vectors. One particular RNN, the Long-Short-Term-Memory (LSTM) network, captures long term dependencies and is thus applied. In order to even furtherly incorporate syntactic structure it is expanded for the so called Tree-LSTM.
In this paper, we will particularly research the effect different features have on the final sentiment classification accuracy.

We investigate the importance of word order by comparing models that consider it compared to BOW based models. We expect a significant increase in performance, as this is a central shortcoming of simple sentiment classification techniques.
Another feature we scrutinize, is the syntactic structure of a sentence. This is motivated by the fact that ordinary LSTM models only cover strict sequential processing of information.(2) It is of interest, if taking into account the roles of words within a sentence influences the sentiment detection. In particular, we compare a tree based LSTM model with an ordinary LSTM model to answer this question.
Additional to solely incorporate the sentence structure, it is of interest, if supervising the sentiment at each subpart of the sentence can increase the models performance. In particular, we therefore train the model not only on the whole sentences

themselves but also all of the syntactic subparts.
Furthermore, we look into the effect of sentence length on each of the models performance. This is particularly of interest, as models which cover word order should be able to identify long order dependencies. In general it is investigated, whether longer sentences and thus more words and information imply a better performance across all the applied models.
All of the tested models are evaluated on the Stanford Sentiment Treebank (SST) (5) and averaged over several runs.

2 Background

We can divide our models into two groups.
The first group consists of BOW based models. In a BOW model, the occurrences of words are encoded binarily (one-hot) for each movie review. As only the occurrences are counted, word ordering is not taken into account. Hence the naming "bag of words". Our BOW is then fed to a neural network for classification.
The second group consists of the LSTM based models. LSTM networks are a special kind of Recurrent Neural Networks or RNNs for short. RNN networks have loops in them. This allows information to persist through the network. The same holds for LSTM networks. However, the key difference between RNN and LSTM is that LSTM models have the ability to model long-term dependencies. This can be attributed to the fact that in LSTM models it is easier for data to flow through a cell without the information being changed. This means that data learned earlier in the network has a higher change of persisting until the end.
We extend both our BOW and LSTM models by using word embeddings. Word embeddings are capable of capturing the semantic similarity between words. We use the pretrained *word2vec* word embeddings.(7) Each word is now represented not in a one-hot fashio, but by a vector $\boldsymbol{v} \in \mathbb{R}^{300}$ where each $\boldsymbol{v}_n$ represents the semantic similarity of the word in that dimension.
Finally, we also extend our LSTM model by using syntactic trees as input features. The tree structure has a word at each leave node and a sentiment score at the other nodes. As our sentence structure is preserved the Tree LSTM should be able to learn the dependencies of the words in a sentence.

3 Models

In this section we describe the architecture of our models.

BOW
Our simplest model is the BOW model. Here we use a word embedding with 5 dimensions. Where each dimension represents a sentiment class. We then sum all the vectors of a movie review plus a bias vector. The classification is then the arg max of the resulting vector. For this model we use the cross-entropy loss function.

CBOW
For the CBOW model we use a word embedding with 300 dimensions. The calculations are the same as for the BOW with the exception that we multiply our parameter matrix $\boldsymbol{W} \in \mathbb{R}^{5\times D}$ with a vector $\boldsymbol{x} \in \mathbb{R}^{D\times 1}$ so that our result is once a gain a vector where each element represents a sentiment class.

Deep CBOW
For the Deep CBOW model we make a neural network with 2 hidden layers of 100 units each. We once again use a word embedding with 300 dimensions. We use the tanh activation function for the hidden layers and a 5 class softmax on the output layer. And once again, the arg max of the output layer is the classification.

Pretrained Deep CBOW
We extend our Deep CBOW by using the pretrained word embeddings from the *word2vec* dataset. The rest of the network architecture remains the same as in the Deep CBOW.

LSTM
In our LSTM cell each word is fed through the following 4 formulas:

$$i = \sigma(W_{ii}x + b_{ii} + W_{hi}h + b_{hi}) \quad (1)$$
$$f = \sigma(W_{if}x + b_{if} + W_{hf}h + b_{hf}) \quad (2)$$
$$g = \tanh(W_{ig}x + b_{ig} + W_{hg}h + b_{hg}) \quad (3)$$
$$o = \sigma(W_{io}x + b_{io} + W_{ho}h + b_{ho}) \quad (4)$$

Then the c and h values are updated to c' and h'

$$c' = f * c + i * g \quad (5)$$
$$h' = o \tanh(c') \quad (6)$$

	Parameter	Value
Both	Learning rate	0.0002
	Embedding size	300
	Hidden size	100
BOW	# Iterations	30000
	Evaluation interval	1000
LSTM	# Iterations	5000
	Batchsize	25
	Evaluation interval	250

Table 1: Hyperparameters for BOW (top) and LSTM (bottom) based models.

We then feed our sentences one word at a time to our LSTM cell. Furthermore, we use a dropout of 0.5 and a cross-entropy loss function.

Tree LSTM
Our final model is the Tree LSTM model. We encode each tree with a transition sequence consisting of *REDUCE* and *SHIFT* actions. (9) Next, we modify our LSTM cell such that it returns a new state when provided with 2 words. Finally we walk through each transition sequence and if we find a *REDUCE* action we pass the 2 words to the Tree LSTM cell. The rest of the network architecture remains the same as in the LSTM cell.

4 Experiments

We evaluate our models on the task of sentiment classification of sentences sampled from movie reviews. We ensure a fair and meaningful comparison by evaluating the experiments on several runs, as well as having the same untuned hyperparameters: learning rate, embedding size, hidden size. An overview is shown in Table 1

Sentiment Classification

In our only task, we predict the sentiment of sentences sampled from movie reviews based on the Stanford Sentiment Treebank.(5) The possible output classes consist of: *very negative, negative, neutral, positive* and *very positive*. The data is split into train/dev/test parts of 8544/1101/2210 observations.
During training, all models were optimized according to five class cross entropy loss on the outputs generated. The according optimization algorithm deployed, was Adaptive Moment Estimation (Adam), which is a special gradient descent method that computes adaptive learning rates for each parameter.(4) BOW based models were sequentially fed and evaluated every 1000 iterations, whereas LSTM models were computed with mini-batches of size 25 and evaluated every 250 iterations. The number of iterations were set after assessing convergence on the development set and can be seen in the appendix graphs. The final evaluation metric was chosen to be the mean accuracy (percentage of correctly assigned labels) of models on test-data sentences for three random seeds.

5 Results and Analysis

The results of our experiments are divided into our research questions raised in the beginning.

Word order
Concerning the importance of word order for sentiment detection, we compare BOW based models with LSTM based models. The best performing BOW based model (PTDeepCBOW) is not significantly worse than any of the LSTM based models, which implies that it is not a highly crucial feature. However, except for the one incorporating pretrained embeddings, most of the simpler BOW based models perform significantly worse. A conclusion might be that word order does impact the sentiment of a sentence, when pretrained embeddings are not drawn on. These results coincide with literature.

Syntactic structure
In our experiments, syntactic sensitive models have not found to be outperforming the basic LSTM models, which are only order sensitive but discard the syntactic structure of sentences. Though, the slightly higher scores indicate that for other data situations and tuned hyperparameters, the syntactic structure could in fact lead to significant performance boosts compared to other LSTM baseline methods. Similar results have been obtained in the original paper.(2)

Supervising Subtrees
Looking at the results obtained by additionally supervising all subparts of a sentence while training, it is assessed to be worsening the performance compared to the ordinary Tree-LSTM model. We explain this behaviour by the fact, that increasing the amount of training instances leads to a model that learns the training data and thus overfits. Thus, simply having more training instances does not necessarily improve performance. We note at this point that the low accuracies might be based upon flawful implementations on our behalf.

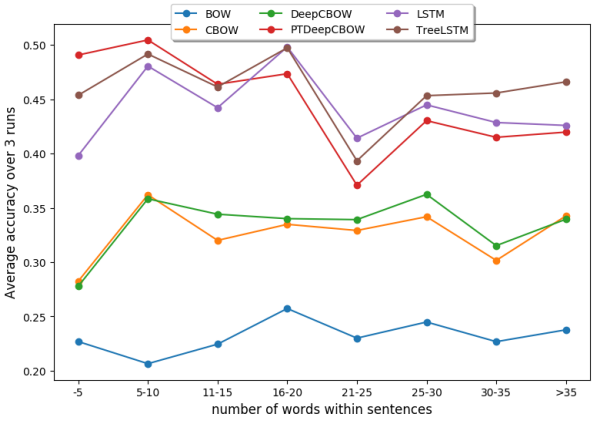

Figure 1: Averaged accuracy over 3 runs on length divided test-data.

Sentence length

Depicted in Figure 1, we observe hardly any trends for the basic BOW models. The DeepCBOW and CBOW show a spike for sentences between 5-10 words long. The PTDeepCBOW model shows a small decreasing trend the longer the sentences are. Both LSTM models show an increasing accuracy until a drop for sentences of length 21-25 and a repeated increase for sentences longer than that. Interestingly, the TreeLSTM model outperforms the LSTM model for almost all lengths and especially larger than 25 words.

General findings

Comparing the models, it is evident that except for the PTDeepCBOW model, basic BOW based models detect sentiments significantly less. The LSTM based models entail an obvious boost in performance, though among them not much difference is observable. Most of the LSTM based models show a clear decrease in performance after around 5000 iterations. This implies overfitting, as the model performs worse on the dev set, despite being exposed to substantially more training.

Limitations

As no hyperparameter tuning was applied, only the relative compared results should be considered. Tuning parameters like the learning rate or number of hidden nodes, should boost the individual model performances. Using similar hyperparameter values for all models was consistent with our understanding of a fair and meaningful comparison. Nonetheless, due to limited computational and temporal resources, we opted for this restrictive measure.

Method	Accuracy	StD
BOW	0.2325	(0.0260)
CBOW	0.3329	(0.0161)
DeepCBOW	0.3427	(0.0153)
PTDeepCBOW	0.4469	(0.0062)
LSTM	0.4519	(0.0070)
Tree-LSTM	0.4599	(0.0067)
Subtree-LSTM	0.3012	(0.0096)

Table 2: Test set results on the sentiment classification task, we report mean scores over 3 runs (standard deviations in parentheses). Results are grouped as follows: (1) BOW based models (2) LSTM models

Significant differences in correct classifications
BOW vs. CBOW (*)
DeepCBOW vs. PTDeepCBOW (**)

Table 3: Pairwise sign tests on individual classifications (insignificant not shown). (*) BOW signif. lower then all other. (**) DeepCBOW vs. all LSTMs significant.

6 Conclusion

To summarize our key findings, our results coincide with literature in the fact that word order is in fact important for sentiment detection. Syntactic structure, on the other hand, benefits performance slightly but not significantly compared to basic order sensitive models. Surprisingly, using all subparts of sentences for training does not lead to an increased performance, but contrary overfitting. Also counterintuitive to us, was the observation of LSTM models requiring less iterations than BOW models before overfitting. Furtherly, sentence length does influence the performance for some models. Based on our and the original Tree-LSTM paper, we suggest to furtherly investigate possible impacts of incorporating syntactic structure. The importance of pretrained word embeddings could also be incorporated into more basic sentiment detection tasks to allow better performance on smaller or different settings, such as real time application which does not allow for extensive training but rather constantly updates the model efficiently. Considering the high accuracy of our PTDeepCBOW model, it might additionally be of interest to train word embeddings like *word2vec* or *glove* within training or with respect to the entire data set. That way, general representations can be specifically adjusted for the case of movie reviews.

References

[1] Bo Pang, Lillian Lee and Shivakumar Vaithyanathan *Thumbs up? Sentiment Classification using Machine Learning Techniques*, Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Philadelphia, July 2002, pp. 79-86.

[2] Kai Sheng Tai, Richard Socher and Christopher D. Manning *Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks*, 2015,

[3] Qiang Ye, Ziqiong Zhang and Rob Law *Sentiment classification of online reviews to travel destinations by supervised machine learning approaches*, Expert Systems with Applications, Volume 36, Issue 3, Part 2, April 2009, pp. 6527-6535.

[4] Kingma, D. P., Ba, J. L. *Adam: a Method for Stochastic Optimization, International Conference on Learning Representations*, 1âĂŞ13 (2015)

[5] Richard Socher et al. *Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank*, Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)

[6] Joseph Lilleberg et al. *Support vector machines and Word2vec for text classification with semantic features*, 2015 IEEE 14th International Conference on Cognitive Informatics Cognitive Computing (ICCI*CC)

[7] Tomas Mikolov et al. *Distributed Representations of Words and Phrases and their Compositionality*, Advances in neural information processing systems. 2013.

[8] D. Jurafsky and J. H. Martin. *Speech and Language Processing*, 2nd Edition. Prentice Hall, 2008.

[9] Muhua Zhu et al. *Fast and Accurate Shift-Reduce Constituent Parsing*, Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pp. 434-443

7 Remark

Due to technical difficulties regarding the used Google Colab environment, the final version of our notebook does not include all of the interim results which are part of this paper. We hardly managed to get the subtree LSTM run on several random seeds due to several terminations of our sessions in colab. We excuse our lack of scientific depth in that regard.

8 Appendix

https://colab.research.google.com/drive/1RJGcIObIKWnXg7kQcRRNOQBrsc-5Owm7#scrollTo=HuCBjFrwdSlq

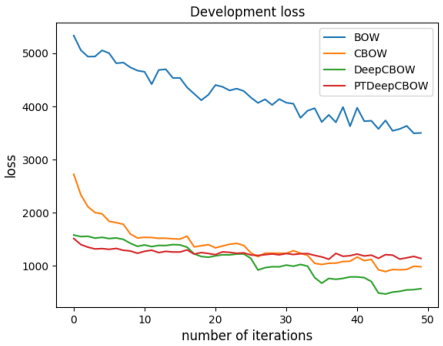

Figure 2: Development loss BOW models

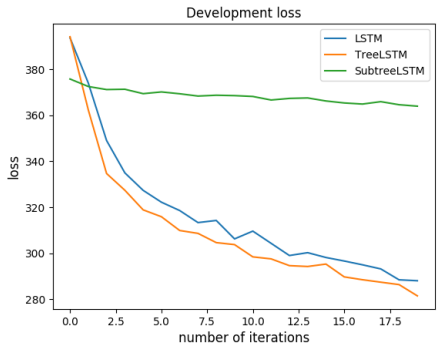

Figure 3: Development loss LSTM models

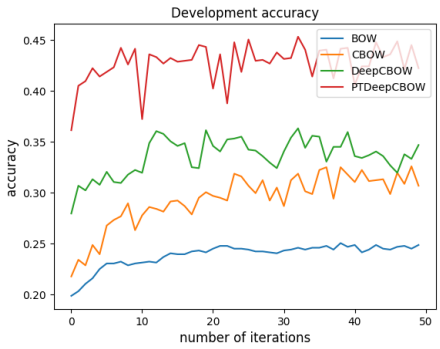

Figure 4: Development accuracy BOW models

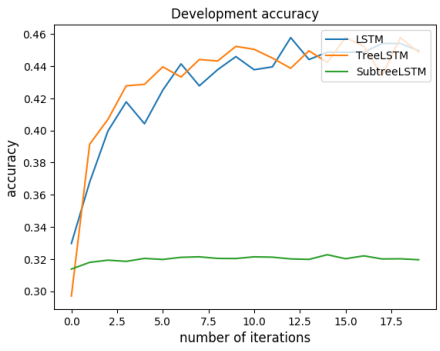

Figure 5: Development accuracy LSTM models