Exploring neural models for sentence encoding and sentiment analysis

Yke Rusticus
University of Amsterdam
Student nr.: 11306386
yke.rusticus@student.uva.nl

Ikira Schielke
University of Amsterdam
Student nr.: 12755087
ikira.schielke@student.uva.nl

1 Introduction

Sentiment analysis is a rapidly growing research field as it has grown apparent that gathered information can be used as a proxy for the users emotion, to mine opinions or to gain insights about the polarity and thoughts about a given topic. Businesses are interested how users perceive their brand or provided service and users cherish relevant and customised content.

Amid the academia surpasses itself with novelty where papers like *Twitter sentiment analysis. The good the bad and the omg* by (Kouloumpis et al., 2011) are welcomed. Mining information about language creativity, emoticons and pursuing other than solely token based avenues have proven themselves to be more feasible in the task of sentiment analysis. We expect, that in order to properly capture the sentiment of a movie review, a sentence based approach using vast pre-trained word embeddings like GloVe, firstly introduced by (Pennington et al., 2014), will lead to auspicious results.

Hereby we will approach this task by following best practices. Assessing the distorting impacts of word order is done by comparing BOW and LSTM models (Kingma and Ba, 2014a) (Greff et al., 2017), the influence of sentence lengths (Jagtap, 2013) and the effects of exploiting the syntactical tree structure of sentences whilst incorporating different strategies of sentiment weight over all models. Approaching these issues show the relevance of word order as a review may discuss positive and negative issues of a topic in which rule- and lexicon-based models will fail to depict the nuances of sentiment.

This work, in addition to other literature, provides comparisons between several models in a relatively simple setting, giving more insight into the behaviours of each of the models in relation to each other. Our experiments were conducted on the STT data set (Socher et al., 2013), and pre-trained word embeddings were used from GloVe (Pennington et al., 2014).

Our findings, as presented in Section 6, show that LSTM-models slightly outperform BOW-models in our experimental setup. Variation in sentence length did not clearly affect the performance of the models. However, pre-trained word embeddings, over self-learned embeddings, do seem to have a relatively large impact on the performance of the models.

2 Background

2.1 Word embeddings

Word embeddings are a form of text representation in which words are mapped to numerical vectors. This process often involves the reduction of dimensions. The two most common word embeddings are word2vec and Global Vectors (GloVe). Both methods represent word distributions in a vector space as the word is mapped with its co-occurrences in a given context of a sentence. Word2vec (Mikolov et al., 2013) - being a predictive model - learns and improves its vector representation by minimizing the loss of predicting the target word from the context word. In this paper, we chose to use the statistical model GloVe with its continuous word embeddings exploiting global information. It constructs a global matrix with co-occurrence information by counting the context frequency of each word. Log-smoothing normalizes the matrix for computations. As frequent words occur in many contexts, matrix factorization is applied to yield a lower-dimensional matrix, where each row now relates to a vector representation of a word instead of solely to the word. Reconstruction loss reduces dimensions by trying to find the optimal lower-dimensional rep-

resentation which can explain most of the variance in higher dimensionality.

We chose GloVe over word2vec due to several reasons, first being the vocabulary size. The provided pre-trained GloVe embeddings covered a larger vocabulary than word2vec. Furthermore, as assessed in rexperiments by (Wang et al., 2019), GloVe overcomes imbalances in word frequency, sparsity by having a much more efficient loss function contains nonzero elements of the co-occurrence matrix thus, scoring more accurate results.

2.2 Techniques

The bag-of-words (BOW) approach is one of the most simple methods for extracting features in texts (Kingma and Ba, 2014a). It describes the occurrence of words within a given document, also providing information about the vocabulary of the document and the measure of the presence of words. The model does not provide information about the order or structure of words, hence the bag analogy.

Whereas the BOW model captures whether words occur in the document, the continuous BOW (CBOW) captures the context of a target word over a given window size (Mikolov et al., 2013).

In comparison to BOW related models Long Short-Term Memory (LSTM) networks do capture word order and semantic features (Greff et al., 2017). LSTMs are a modification of recurrent neural networks which subsume long-term dependencies in data. They make use of three different kinds of junctions. Information about the i.e. word counts is passed through the cell state with minor linear interaction. The first step is the decision of the forget gate which information of the cell state should be discarded and thus not further passed. The second step is activating the sigmoid input gate layer which decides which information will be stored in the cell state and a tanh-layer creates a probable vector of new candidate values that could be added to the cell state. The last step is updating the value of the old cell state by multiplying it by the forget layer and adding the input layer with the multiplied candidate values. A final decision is made at the output gate being a sigmoid layer which decides which parts of the cell state will be passed onto the tanh-layer to be multiplied with the result of the output gate.

Instead of treating a sentence as a flat sequence the Tree-LSTM includes the syntactic information given by the tree structure of the sentence thus composing the meaning of a vector by the meanings of a word and the rules that combine these words (Zong and Strube, 2015).

3 Models

Every of the following models uses cross-entropy loss, which increases as the probability deviates from the actual label.

3.1 BOW

The first model is an one-layered neural network (NN) which takes the vocabulary as an embedding, sums up the vectors of sentences losing the word order. The predicted label is the argmax of that sum and the corresponding sentiment label of the regarding the word.

3.2 CBOW

The continuous bag-of-words model (CBOW) learns word embeddings of arbitrary size. The arbitrariness is repercussed by word embeddings with different dimensions. Again the vector summation and a dimension reduction from the vector size V down to 5 units is applied.

3.3 Deep CBOW

The deep CBOW is more advanced in implementation and performance than the prior model. The network incorporates three layers with two tanh-activations. Compared to the sigmoid-activation function the tanh-function ranges from [-1,1] instead of [0,1] thus mapping gradient strength more strongly as the derivatives are steeper.

3.4 PTDeepCBOW

Pre-trained Deep CBOW uses the same architecture as the Deep CBOW model, however, it makes use of the pre-trained GloVe word embeddings, which are held fixed during training.

3.5 LSTM

The LSTM classifier makes use of a LSTM cell which updates the hidden state one word at a time. The LSTM cell builds up from two linear layers which outputs have been concatenated for an efficient computation of the matrix multiplication. The classifier, however, takes the vocabulary, an instance of the LSTM cell and needed dimensions to predict the final hidden state of each sentence in

the corpus. To prevent overfitting a Dropout layer is included, which excludes neurons from computing with a probability of 50 percent. As sentences have different lengths padding is applied where needed.

3.6 Tree-LSTM

Compared to the Tree-LSTM, the LSTM can be considered a sequential Tree-LSTM where each internal node has one child. In this paper we are using binary Tree-LSTM, thus two leaves (words) or two nodes (consisting of subtrees) are combined to a common vector and gating vectors and computing memory cell updates are dependent on the states of child units. Furthermore, instead of having a single forget cell, the Tree-LSTM contains forget gates for each child, allowing the model to learn relations between semantic heads and its dependents.

4 Experiments

For each of the conducted experiments, the underlying task for the models was to correctly predict the sentiment of a given input sentence. We used the Stanford Sentiment Treebank (SST), which consists of sentences, their tree structure, and sentiment scores at each node of the tree ranging from 0 (”very negative”) to 4 (”very positive”). The training, validation and test subsets had relatively small sizes of resp. 8544, 1101 and 2210. We trained the models using stochastic gradient descent, where we relied on automatic differentiation and the optimization algorithm Adam (Kingma and Ba, 2014b). The only supervision signals the models received were from predicting the sentiment of the root node of a given sentence. The performance of each model was evaluated by its accuracy of correct predictions on the test set. Each model was trained three times and tested on this test set to obtain an average accuracy of correct predictions per model. Using a similar setup, we also tested one of each model on 10 different testsets, each containing sentences of different lengths (set 1: 0-5 tokens, set 2: 5-10 tokens, etc.), in order to investigate the relation between accuracy and sentence length. As a final experiment we split each training example into sub-trees and added these to our training set, resulting in 155019 training examples. We then again trained three different models of each kind and evaluated them as described above. The model parameters that were used for each experiment are given in Table 1. For the BOW- and the LSTM-models batch sizes of 1 resp. 25 were used, using padding to account for unequal sentence lengths. The code that was used for this work has been made available to allow for repoduction of the experiments[1].

Model	Learn. rate	Embed. dim	Hid. dim
BOW	$1 \cdot 10^{-2}$	-	-
CBOW	$4 \cdot 10^{-4}$	300	-
DeepCBOW	$2 \cdot 10^{-4}$	300	100
PTDeepCBOW	$5 \cdot 10^{-5}$	300	100
LSTM	$8 \cdot 10^{-5}$	300	168
Tree-LSTM	$8 \cdot 10^{-5}$	300	150

Table 1: Model parameters: learning rate, embedding dimension and hidden dimension.

5 Results and Analysis

Figures 1a and 1b show validation accuracies of the BOW-models and LSTM-model respectively over training iterations, where each iteration corresponds to training on one batch. From these figures and the test accuracies presented Table 2, we can notice several things.

First of all, all models that make use of pretrained word embedding have an accuracy of around 10 percent higher than models that do not. This is likely due to the fact that our only error signals came from predicting the root node of the tree, which made it hard to achieve good word embeddings through training. One could expect this effect to mitigate when training on all subtrees, since then we are training ”closer” to the words themselves. However, the results show no improvement for any of the models when we use all subtrees in training. In fact, we see a slight decrease in performance for each of the models.

Word order seems to be a useful property to achieve higher performance of the models. Both LSTM-models, that take word order into account, score higher than the BOW-models that do not take word order into account. However, the difference between the highest scoring BOW-model (PTDeepCBOW) and the lowest scoring LSTM-model (LSTM) is only 1%.

The tree structure of the data seems to have little influence on the performance of the models, as the LSTM and Tree-LSTM lie close to each other in terms of validation curves and test accuracies.

[1] https://gist.github.com/ykerus/c1b9623b1c1913f85c76a607ed93de11

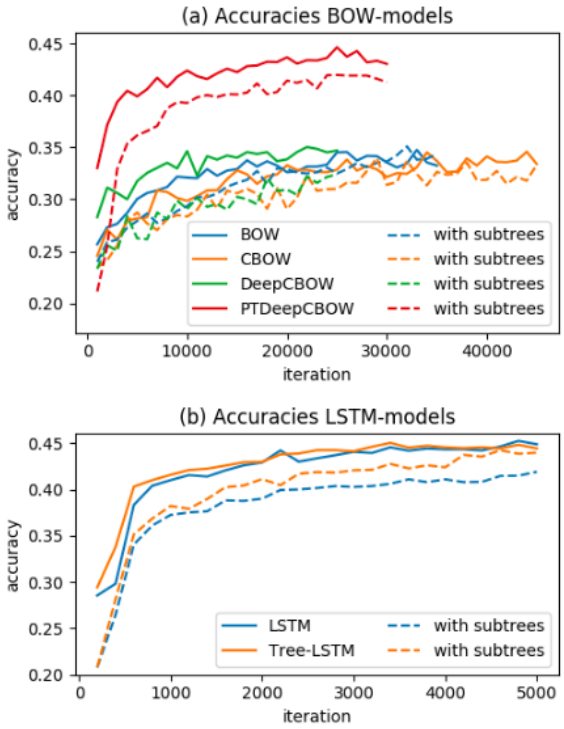

Figure 1: Validation accuracies per model averaged over three separate runs with different initialized weights. Solid lines represent models trained only on the root of the sentence trees, dashed lines represent models trained on all the subtrees.

Lastly, no clear effect was measured on the performance of the models on examples of different sentence lengths. Figure 2 shows some fluctuations for most models, but no model seems to follow a trend up or down. The fact that we had only a few hundred, and sometimes a few tens of examples per test set was most probably a strong negative factor on the results that we obtained. Averaging the BOW-models and the LSTM-models respectively also did not result in clear trends.

Model	Test accuracy	Test accuracy (trained on subtrees)
BOW	0.355 ± 0.010	0.345 ± 0.011
CBOW	0.365 ± 0.027	0.330 ± 0.012
DeepCBOW	0.352 ± 0.004	0.305 ± 0.004
PTDeepCBOW	0.454 ± 0.002	0.424 ± 0.009
LSTM	0.468 ± 0.006	0.436 ± 0.001
Tree-LSTM	0.469 ± 0.005	0.445 ± 0.005

Table 2: Test accuracies averaged over three separate runs per model, with standard deviation.

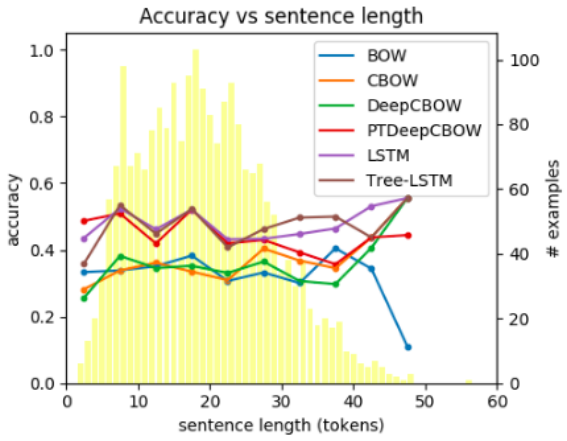

Figure 2: Single model accuracies for varying sentence lengths. The barplot indicates how many test examples were available per sentence length.

6 Conclusion

The experiments have shown results mostly in the line of expectation. Usage of pre-trained word embeddings results in a boost in performance, as learning them from a sparse data set is a difficult task. It has come as a surprise however, that using the order that is present in language, such as word order or the tree structure, had only a small effect on the performance. This brings up the question of whether humans do in fact use this order in the way that we try to model it. Li and Qian (2016) showed however, that an LSTM can achieve accuracies of around 90 percent in just predicting whether a given input is positive or negative. In our task, we also looked at how positive or negative an input was, with a fairly small data set. In future work, it would therefore be necessary to conduct the experiments on either more annotated data, or perhaps in a more self-supervised setting, in order to truly explore the capabilities of LSTMs over BOW-models in this task.

References

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bastiaan Steunebrink, and Jürgen Schmidhuber. 2017. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28:2222–2232.

Vandana Jagtap. 2013. Analysis of different approaches to sentence-level sentiment classification. *International Journal of Scientific Engineering and Technology (ISSN: 2277-1581)*, 2:164–170.

Diederik Kingma and Jimmy Ba. 2014a. Adam: A

method for stochastic optimization. *International Conference on Learning Representations*.

DP Kingma and JL Ba. 2014b. Adam: A method for stochastic optimization. arxiv 2014. *arXiv preprint arXiv:1412.6980*.

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. 2011. Twitter sentiment analysis: The good the bad and the omg!

Dan Li and Jiang Qian. 2016. Text sentiment analysis based on long short-term memory. In *2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI)*, pages 471–475. IEEE.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

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, Doha, Qatar. Association for Computational Linguistics.

R. Socher, A. Perelygin, J.Y. Wu, J. Chuang, C.D. Manning, A.Y. Ng, and C. Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. *EMNLP*, 1631:1631–1642.

Bin Wang, Angela Wang, Fenxiao Chen, Yunchen Wang, and C-C Jay Kuo. 2019. Evaluating word embedding models: Methods and experimental results. *arXiv preprint arXiv:1901.09785*.

Chengqing Zong and Michael Strube, editors. 2015. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China.