Sensing the Sentiment: A Comparison of Bag of Words Models with LSTMs and Tree-LSTMs

Arvid Lindström
UVA-ID: 12365718
Arvid.Lindstrom@gmail.com

David Biertimpel
UVA-ID: 12324418
david.biertimpel@student.uva.nl

Abstract

Identifying the sentiment of texts has recently received great interest, especially due to the rise of online marketing or hate speech in social networks. In this paper we examine models based on the bag of word (BOW) approach and the more sophisticated LSTM networks on the task of classifying sentiment in movie reviews. Among others we investigate how considering a sentence's tree structure or the sentence length affects classification.

1 Introduction

In recent years, sentiment analysis methods have evolved dramatically, driven by access to training data, computing resources, and the general demand to automatically classify texts regarding their affective meaning. Sentiment analysis refers to the classification of emotional connotations in texts such as movie reviews. One prominent approach of the past are BOW models which create a vocabulary utilizing features extracted from individual words. A successful example was shown by [15] who predicted whether news articles in the Dutch 2006 election were positive or negative. It has been shown recently that although models have improved, most progress can be attributed to the quality of the training data [9]. Given that Harris' Distributional Hypothesis is still widely accepted by computational linguists [2], modern models should benefit from more linguistically motivated methods.

This paper uses BOW methods as a baseline for comparison with two recurrent neural network models, LSTMs [4] and TreeLSTMs [14, 7, 16]. All models considered utilize embeddings (real valued vector representations of words) which may capture semantic relations between words. The dataset used is the crowd-annotated Standford Sentiment Treebank [13] containing a total of 11855 manually annotated sentences. The LSTM based models are capable of modelling the sequential properties of sentences, using the order in which words appear. Unlike BOW models, LSTMs implicitly maintain a history of words seen in a sentence, which contributes to their understanding of semantic relations.

Our research questions are formulated as follows: Given the task of movie review sentiment classification,

- How does the accuracy of BOW models compare to more sophisticated approaches that implicitly preserve word order and history like the LSTM and Tree-LSTM?
- Does a Tree-LSTM improve performance and is it beneficial to consider the sentiment at each tree node while training (in contrast to only considering the root node)?
- How does the performance of the models depend on the length of a sentence to be classified?
- Are sentences of a particular sentiment harder to classify than others, and which models struggle with a sentiment in particular?

To approach these questions, a cross-comparison study is performed involving five distinct models: A neural BOW model, a continuous BOW, a so called Deep-continuous BOW using a multilayer perceptron [5], a LSTM and a tree-based LSTM model.

2 Background

2.1 Representing words as vector embeddings

For NLP tasks, learned distributed representations of words allows the use of machine learning algorithms to better analyze words. An embedding, in this context, refers to a real-valued vector representing a word. One of the first examples of this technique used vectors to represent similarities between English and Italian names [11]. By mapping a word to a continuous vector space, we can represent words with a high degree of co-occurrence as neighboring points in that vector space. In this paper we use the publicly available GloVe vector embeddings by [10].

2.2 Bag of words models

The Bag of Words (BOW) models used in this paper are the neural BOW [12], the continuous BOW (CBOW) and the Deep Continuous BOW (DCBOW) [8] models. The neural BOW model performs sentiment prediction of a sentence S using the word-embeddings and summing over the individual dimensions, choosing the class corresponding to the largest sum. The assigned classes for sentences in our experiments are ["Very Positive", "Positive", "Neutral", "Negative", "Very Negative"]. Due to this summation, the order in which the word appears in the sentence is effectively discarded. The continuous bag of words model is inspired by the architecture proposed by [8]. Unlike the neural BOW, CBOW represents each word as an embedding of an arbitrarily chosen size, to capture more nuances of the words. The resulting embeddings of a given sentence are summed across all dimensions and then mapped to sentiment space where prediction is performed by choosing the maximum argument.
The DCBOW is explained further in Section. 3.

2.3 Recurrent Neural Network methods

Originally introduced by [4], the LSTM model is a form of recurrent neural network in which a single LSTM unit, called a cell, maintains an internal state representing the type of word or phrase used for prediction. The internal state at time-step C_t is maintained by forget gates f_t which, given an input word w_t, uses a sigmoid activation to scale the previous cell state C_{t-1}, effectively learning to forget information [1]. We train our LSTMs using the dropout method proposed in [3] to mitigate the problem of overfitting on our relatively small dataset. The LSTM works sequentially by considering the words in the order that they appear, and for each word w_t, outputting an activation h_t which $\forall_t w_t \in$ S contribute towards the final cell state C_{end}. This final state, generated after the last word in the sentence has been seen, will depend on the previous cell state at that time-step C_{end-1} and is mapped linearly to the sentiment space. Note that the LSTM takes as input the word embedding $E(w_t)$ for each word in sentence S.
The final model used in this paper is a Binary-Tree-LSTM model [14], [16] and [7]. In contrast to the strictly sequential information flow of an LSTM, a Tree-LSTM preserves the inherent hierarchical structure of a sentence while parsing it. In this paper we use a binary-Tree LSTM where each sentence to be classified is parsed as binary sub trees. As such, the forget gates bounding the internal state of the Tree-LSTM cells are dependent on the input from the cell's left and right children. This allows our model to learn which children contribute sentiment related information which may increase the classification performance. The words from a sentence enter as input to the leaf-nodes in our Tree-LSTM cells and using a transition table defined by [13], outputs are fed to higher nodes in the tree until eventually a sentiment score is obtained from the root node.

3 Models

All neural BOW based models have in common that we randomly initialize the word embeddings, which are trained afterwards using stochastic gradient descent. With regard to a loss function, the cross entropy loss is used for all models. In the case of the plain neural BOW, each word embedding is a vector with 5 entries corresponding to the 5 sentiment gradations. This leads to an embedding matrix $\boldsymbol{E} \in \mathbb{R}^{V\times 5}$, where V is the size of the vocabulary. The CBOW extends this approach by adding a linear layer to the model that maps the resulting vector after the summation of size 300 (as $\boldsymbol{E} \in \mathbb{R}^{V\times 300}$) back to the 5 sentiment classes. The DCBOW adds more complexity to the model by putting between the summation and the prediction a non-linear network consisting of two linear layers with a bias and a Tanh activation function each. In all these models we make a prediction by taking the *argmax* of the resulting vector.

Now we turn to recurrent structures and thus to the LSTM and Tree-LSTM. In the LSTM we feed one input batch to the LSTM cell, where it interacts with the matrices $\boldsymbol{W}_i$, $\boldsymbol{W}_h$ and the biases. After these two linear transformations, we obtain the new values for h and c. In order to make a prediction, the output of the last recurrent layer is fed to a linear transformation mapping h onto 5 dimensions. This step includes a dropout component (p=0.5) for preventing overfitting. After a prediction is made by applying the *argmax* onto the resulting vector, the error is back propagated through the model and the weights are updated with SGD.

For the Tree-LSTM the classification works similar to the normal LSTM with the difference that the Tree-LSTM cell receives input from two children (each $B \times 150$). These are concatenated before being passed through a reduce layer (300×750), which results in a tensor of shape $B \times 750$. Based on this we calculate the new values for h and c. Classification again takes place through an output layer that maps h to the 5 sentiment classes including a dropout component.

4 Experiments

To answer our research questions, we train our models using the Stanford Sentiment Treebank (SST) [13] dataset, either including or excluding the use of the pretrained Glove word embeddings [10]. We divide the SST dataset in 8544 training; 1101 validation and 2210 test observations. We train all the BOW based models over 30000 iterations and the LSTM and Tree-LSTM over 10000 iterations.

The model architectures can be seen in Table. 1. Note that the neural BOW and CBOW are solely trained on the SST dataset. For the DCBOW there exists a version with- and without using the pretrained Glove embeddings. The LSTM and Tree-LSTM are both only trained on the Glove embeddings. The first Tree-LSTM is trained on the regular tree structure of a sentence, the second one is further trained on each sub tree and therefore considering the inner tree nodes. For all training processes the optimization algorithm Adam with a learning rate of $\eta = 0.005$ is used [6]. To obtain stable results we train each model three times with different seeds and provide mean and standard deviation of the final training accuracy. In terms of evaluation we assess the models solely on their accuracy. After training, we partition our test data into bins with the following sentence lengths: [≤ 3; $4-7$; $8-14$; $15-21$; $22-28$; > 28] to investigate whether the models performance is influenced by sentence length. We also partition the test data w.r.t. the sentiment classes to answer if some sentiments are more difficult to classify than others.

Model	**Shape Linear Layers**
CBOW	300×5
DCBOW & Pretrained DCBOW	300×100, 100×100, 100×5
LSTM	$\boldsymbol{W}_i : 672 \times 300$, $\boldsymbol{W}_h : 672 \times 168$ output: 168×5
Tree LSTM & Tree LSTM ST	children: $B \times 150$ reduce layer: 750×300 projection: $B \times 750$ output: 150×5

Table 1: Summary of the most important architecture elements of the used models.

5 Results and Analysis

In order to answer the research questions posed at the beginning, we now present and analyze our experimental results. We first look at the mean accuracies of each model over the three runs, presented in Table 2:

BOW	$25.19\% \pm 2.23\%$
CBOW	$22.73\% \pm 1.41\%$
Deep CBOW	$36.80\% \pm 0.87\%$
Pretained D. CBOW	$44.18\% \pm 1.43\%$
LSTM	$46.84\% \pm 0.48\%$
Tree LSTM	$47.07\% \pm 1.06\%$
Tree LSTM ST	$47.13\% \pm 0.63\%$

Table 2: Depiction of the models mean accuracies over three runs together with their standard deviation.

We first notice that the LSTM models perform substantially better than the neural BOW based models. While this is especially true for the BOW and CBOW model, the DCBOW models show an increase in performance and the pretrained DCBOW even approaches the LSTMs models. However, these remain constant at values around 47% and thus show even better results. Since the BOW models do not consider word order, this result suggests that taking word order into account has a positive effect on the models performance.

When we compare the performance of the LSTM with the binary Tree-LSTM, we find that the Tree-LSTM does not perform decisively better than the LSTM (46.84% vs. 47.07%). We see also no major difference between the Tree-LSTM and the Tree-LSTM trained on subtrees (47.07% vs. 47.13%). From this we can conclude that regarding the tree structure of a sentence does not have a positive effect on the model's performance. The same applies to incorporating the subtrees into the training of the Tree-LSTM.

To investigate the influence of different sentence lengths on our trained models, we use the test data partitioned into bins based on sentence length as explained in Sec. 4. For each bin we compute the accuracy with the following trained models: Pretained DCBOW, LSTM, Tree-LSTM, Tree-LSTM ST (ST = Subtree). The results of this procedure are shown in Figure 1.

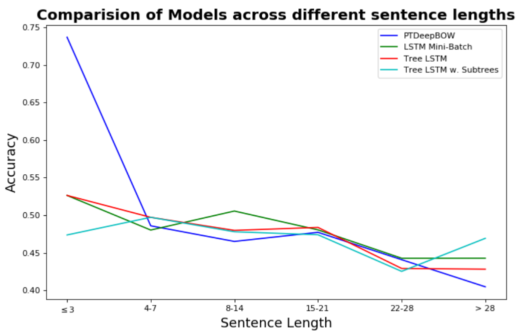


Figure 1: Illustration of the models performance accross different sentence lengths.

First we find that the four models behave approximately the same way with the trend that the the longer the sentence length the more accuracy decreases. However, it is striking that the pretrained DCBOW performs vastly better then the LSTM models with sentences length of ≤ 3.

Lastly, we partition our test data w.r.t. annotated sentiment label into five different subsets. With each of our trained models, we then calculate the accuracy for each of these subsets. The complete results of this can be found in Table 3 in the appendix (see section 7). When looking at the average accuracy values of the individual sentiment classes, we notice that the values differ significantly. While the classes "Positive" and "Negative" show similar accuracy's (52.02% vs. 52.27%), the other classes are noticeably lower. The class "Very Positive" displays an accuracy of 32.98%, "Very Negative" 21.2% and "Neutral" 21.2%. The models BOW, CBOW and Tree-LSTM perform worst for "Very Negative" sentences and DCBOW, Pretrained DCBOW, LSTM and Tree-LSTM ST models perform worst in "Neural" sentences. In addition, the "Neutral" class has the lowest maximum value compared to all other classes (27.51%). These insights suggest that the "Neutral" class is the most difficult to predict. However, it can also be seen that the models have problems with the gradation between a weak and a stronger sentiment.

6 Conclusion

In this paper we have investigated the performance of the models neural BOW, Continuous BOW, Deep Continuous BOW, LSTM and Binary-Tree-LSTM on the task of classifying sentiments from the SST dataset. We find that the preservation of word order utilized by LSTMs and binary Tree-LSTMs increases classification accuracy. In terms of the sentence length, we find it evident that longer sentences are harder to classify than shorter ones. With regard to our final research question, our findings suggest that certain sentiment classes are remarkably more difficult to predict than others. Our comparison across classes shows that the performance for all models decreases drastically when considering sentences of "neutral" sentiment. This may be a domain related problem in the scope of movie reviews, or might refer to neutral sentences being more ambiguous due to a lack of polarizing words. For future research, a natural extension is to perform the same training across multiple corpora with different sentiment classes. Typically the sentiments used in literature are annotated with varying degrees of binary sentiments (such as positive versus negative). A more interesting study would be sentiment analysis across more widely defined sentiments or emotional states such as "angry", "sad" or "joyful" reviews.

GITHUB GIST LINK: https://gist.github.com/d4vidbiertmpl/19d2d23e340cdd519ce350b3a581f1cc

References

[1] F. A. Gers, J. Schmidhuber, and F. A. Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12:2451–2471, 2000.

[2] Z. S. Harris. Distributional structure. *WORD*, 10(2-3):146–162, 1954.

[3] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[4] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997.

[5] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, July 1989.

[6] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[7] P. Le and W. H. Zuidema. Compositional distributional semantics with long short term memory. *CoRR*, abs/1503.02510, 2015.

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[9] I. Mozetic, M. Grcar, and J. Smailovic. Multilingual twitter sentiment classification: The role of human annotators. *PLOS ONE*, 11(5):1–26, 05 2016.

[10] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *In EMNLP*, 2014.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Neurocomputing: Foundations of research. In J. A. Anderson and E. Rosenfeld, editors, *Neurocomputing: Foundations of Research*, chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988.

[12] I. Sheikh, I. Illina, D. Fohr, and G. Linares. Learning word importance with the neural bag-of-words model. In *Learning Word Importance with the Neural Bag-of-Words Model*, pages 222–229, 01 2016.

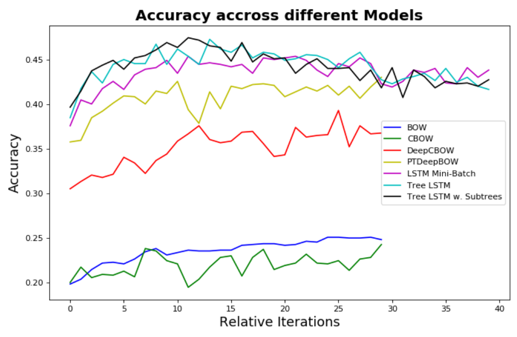
[13] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, 2013.

[14] K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *CoRR*, abs/1503.00075, 2015.

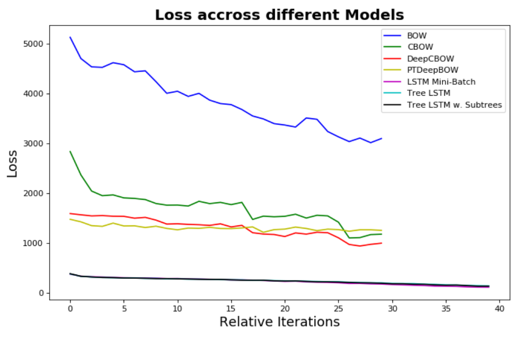
[15] W. van Atteveldt, J. Kleinnijenhuis, N. Ruigrok, and S. Schlobach. Good news or bad news? conducting sentiment analysis on dutch text to distinguish between positive and negative relations. *Journal of Information Technology & Politics*, 5(1):73–94, 2008.

[16] X. Zhu, P. Sobhani, and H. Guo. Long short-term memory over tree structures. *CoRR*, abs/1503.04881, 2015.

7 Appendix



(a)



(b)

Figure 2: Depiction of the accuracy (a) and loss (b) during training. The x-axis shows *relative iterations*, as the BOW models were evaluated every 1000 iterations and the LSTM based models every 250 iterations. Thus the iterations of the models do not directly correspond to each other, but show the overall trend.

	Very Negative (%)	*Negative (%)*	*Neutral (%)*	*Positive (%)*	*Very Positive (%)*
BOW	12.54	31.28	18.51	32.16	20.55
CBOW	6.09	33.18	14.14	38.63	7.77
Deep CBOW	18.64	46.45	16.45	60.59	28.82
Pretained D. CBOW	24.37	**69.19**	16.2	50.78	**45.86**
LSTM	27.96	69.04	10.03	60.78	45.11
Tree LSTM	24.01	59.08	**27.51**	61.37	40.6
Tree LSTM ST	**34.77**	55.92	24.16	**61.57**	42.11
Average	21.2	52.02	18.14	52.27	32.98

Table 3: The accuracies of the models accross the different sentiment classes. In each sentiment class the maximum value is highlighted.