Sentence Representation using Neural Models for Sentiment Classification

Adriana Cabrera
Student Number: 14447630
Graduate School of Informatics
University of Amsterdam
adriana.cabrera@student.uva.nl

Luca Pantea
Student Number: 14651769
Graduate School of Informatics
University of Amsterdam
luca.pantea@student.uva.nl

1 Introduction

Due to an increasing amount of user-generated content such as reviews, testimonials, as well as social media content published to the internet, the use of sentiment classification has become increasingly important. In Natural Language Processing (NLP), sentiment classification allow us to classify the polarity of textual data using Machine Learning (ML) techniques (Deho et al., 2018).

With one of its earliest applications being in the early 2000s (Pang et al., 2002), the uses of sentiment classification are endless. Previous work with sentiment classification can be seen in the medical sciences domain, where (Saha and Munmun, 2017) focuses on the detection of stress within university students by analyzing their linguistic style in social media posts. Other work on sentiment classification has been done with social media content to predict election outcomes (Chauhan et al., 2020).

One of the most traditional methods of conducting sentiment classification the Bag-of-Words (BOW) model (Pang et al., 2002). While the traditional BOW model of has proven to be quite successful; it does have its limitations. For example, it fails to capture word order and variable-length text which is crucial to understand the context and semantic meaning of sentences (Mikolov et al., 2013).

These limitations have led to the development of new models that resolve these issues. For example, word embeddings such as *Word2Vec* and *GloVe* have become popular techniques for representing words with similar semantic meanings (Mikolov et al., 2013). Similarly, neural models such as Long Short Term Memory (LSTMs) are able to capture long sequential data and has proven to generate higher-quality results within sentiment classification (Hochreiter et al., 1997).

In this paper, we compare a variety of models for sentiment classification. We examine the results from the traditional BOW model, as well as from more updated models such as the Continuous BOW (CBOW), LSTMs, and Tree-LSTMs. Our main objective in this paper is to discover what role word order, variable sentence length, as well as tree-structured data plays in sentiment classification tasks. We also further investigate how our model performs using a transformer based model, *BERT*.

We expect that that the introduction of these models will enhance performance and thus achieve higher accuracy results as we keep more valuable information in our phrases and sentences, such as the semantic relationships with neighboring words.

Our findings prove that the CBOW and DeepCBOW models indeed led to an increase in accuracy of nearly 8% compared to the traditional BOW model. Using pre-trained word embeddings in the DeepCBOW model led to another improvement of 8%. When using the LSTM models, we saw a significant improvement compared to the BOW of nearly 16-19%, and a 20% improvement when using Tree-LSTM. Finally, we investigate the performance of a pre-trained language model and conclude that it provides a modest modest improvement of 10% over the other variants.

2 Background

2.1 Bag-of-Words

A traditional approach for sentiment classification is the Bag-of-Words (BOW) model (Pang et al., 2002). Here, words in a text corpus are viewed as a fixed-length vector and the frequency of a word is counted. The word frequency is then stored in a feature vector, which can then be used to classify the sentiment of a text corpus.

However, one shortcoming of the BOW model is that it is unable to grasp word order and relationships between words. This is especially important because the position of a word in a sentence can change the context and meaning of a text corpus. For example: *”The cat chased the mouse”* versus *”The mouse chased the cat”*. Without being able to grasp word order, we can lose valuable semantic information from our textual data, further leading to an inaccurate sentiment classification model (Mikolov et al., 2013).

2.2 Word Embeddings

In order to better represent sentences in sentiment classification, we can use a popular concept in NLP called word embeddings (Mikolov et al., 2013; Pennington et al., 2014). Using this approach, we can capture the syntactic and semantic relationships between words.

There are two popular word embedding techniques: Word2Vec (Mikolov et al., 2013) and Global Vectors for Word Representation (GloVe) (Pennington et al., 2014). Using word embeddings we can represent words in vectors of a high-dimensional space, with the vector values depicting the promiximity of meaning similarities between words.

As part of the Word2Vec approach, the Continuous Bag-of-Words (CBOW) model was introduced to learn word embeddings (Mikolov et al., 2013). This model aims to predict the target word based on its surrounding words. For example, the model will take the words *cat* and *tree* as input to predict the context word, *tree*. Given the CBOW’s ability to capture context words, (Mikolov et al., 2013) has attributed this model with higher quality and sentiment classification accuracy compared to the traditional BOW model.

2.3 Long Short-Term Memory

Another popular model is the Long Short-Term Memory (LSTM) model. This type of Recurrent Neural Network (RNN) is able to learn long-term dependencies and capture variable-length sequential data as well, making it a powerful tool for sentiment classification tasks (Hochreiter et al., 1997).

LSTMs makes use of a ”memory” which allows the model to retain and use information from previous inputs (Zhang et al., 2018). They do this by having two states: a *hidden state* and a *cell state*, as well as three main gates: a *forget gate*, an *input gate*, and an *output gate*. The forget gate decides which information can be kept for the cell state and which information can be forgotten, the input gate updates the cell state, and the output gate determines what information the hidden state should carry.

Other variants of the LSTM model have been introduced as well, such as the Tree-LSTM (Tai et al., 2015; Le and Zuidema; Zhu et al., 2015). The underlying concept of the Tree-LSTM is quite similar to that of a standard LSTM. However, this variant is more compatible with dependency tree structured data, rather than only sequential data. As a result, the Tree-LSTM proved to be more successful at representing semantic relationships compared to a standard LSTM (Tai et al., 2015; Le and Zuidema; Zhu et al., 2015).

3 Models

3.1 BOW

We first experiment with a BOW model as our baseline. Here, we associate every word with a multi-dimensional vector that captures the sentiment of each word. After assigning every word a multi-dimensional vector based on its sentiment label, we sum the vectors of the words in the sentence using a bias vector, resulting in our sentiment classification.

3.2 CBOW

We also experiment with a CBOW model, where the embeddings are a dimension of *arbitrary* size. This is useful as we can select higher dimensionality and thus learn more aspects of each word. We sum the resulting word vectors, and learn a parameter matrix and multiply it by the sum vector: Wx. In this model we utilize one linear layer and a bias term.

3.3 Deep CBOW

In order to improve the performance of the CBOW model, we used a Deep CBOW model where we added more layers and non-linearities so we can learn more aspects of the data. Here, we have three linear layers with two `Tanh` functions.

3.4 Deep CBOW with pre-trained embeddings

We also trained our DeepCBOW model with pre-trained word embeddings using Word2Vec an GloVe. Using these pre-trained word embeddings, we are able to start from a point where words that have close proximities are already close in a distributional semantic space.

3.5 LSTM

We represent sentences as a sequential set of tokens using LSTM. Our model includes an LSTM cell, in which word embeddings are sequentially entered and its hidden state is updated one word at a time. In order to make a prediction from the final hidden state in the LSTM cell, we then apply the LSTM classifier, which is comprised a linear layer and a Dropout layer. The Dropout layer serves to mitigate the problem of overfitting of our model.

3.6 Tree-LSTM

We also explore the Tree-LSTM model. The structure of the Tree-LSTM variant contains a forget gate for each child to learn semantic heads and relatedness.

We specifically implement a Binary Tree-LSTM in which every node has two children. In our Binary Tree-LSTM cell, we return a new state when provided with the node (e.g. two children). We then apply our Tree-LSTM cell to our Tree-LSTM class to encode a complete sentence and return the root node. The Tree-LSTM classifier takes the hidden state at the node as input to predict the label for a given node.

4 Experiments

We present our experimental results from the comparison of the models described in Section 3, under a range of hyperparameter configurations to achieve better model performance. We justify our results by comparing the versions of the tuned models with their baseline configuration (i.e. given in the notebook). This section outlines the experimental setup and briefly describes the prediction task and the dataset used. Finally, we present the hyperparameter considered in this work and the evaluation criterion used for the results. For checking out the entirety of our experiment logs and analyse runs in detail, we point the reader to our online run log directory[1].

4.1 Experimental setup

We aim to perform sentiment classification for a given sentence. The underlying dataset used is the Stanford Sentiment Treebank (SST) (Socher et al., 2013), which provides sentences structured as a binary tree, with fine-grained sentiment scores at each node, and the root node containing the sentence sentiment score. An illustration is presented below in Figure 2, in Appendix A. The training, validation and test sets are rather small, with 8544, 1101 and 2210 samples in each set, respectively.

The Adam optimizer is used to optimize the models, and we apply a baseline learning rate of 0.0005. To assess the performance of the algorithms, we minimise the Cross-Entropy between the target and the input probabilities (Equation 1). To evaluate each model, we track the validation accuracy across training iterations and use the best-performing model (i.e. best validation accuracy) during training on the held-out test set and measure the Top-1 accuracy.

To increase the classification accuracy of the benchmarked models, we run experiments with a variety of hyperparameters to validate and ablate different network design decisions. Concretely, we vary the model learning rate η between 0.001, 0.0005 and 0.0001. We experiment with batch sizes **B** of 1, 25, 32 and 64. Finally, we create word embeddings using methods from literature, namely *Word2Vec*, *GloVe* and *BERT* (Devlin et al., 2018). The latter was only used when testing the model performance of BERT. All networks are regularized by adding a dropout layer with $p = 0.5$ before the final output layer. We perform each experiment using three randomly generated seeds and take the average of the performance to obtain a more robust measure of the algorithms' performance. We additionally present the maximum obtained accuracy.

5 Results and Analysis

In this section, we provide our empirical findings and outline their significance in answering the research questions in the introduction. All results showcased here are within a 95% confidence interval, with the standard error plotted alongside the mean across 3 runs with different random seeds. Table 1 shows the standard baseline model parameters.

5.1 On the Importance of Word Order

To asses word order, we observe the performance of the models that take word order into account (LSTM-based models) as opposed to those which do not (BOW-based models). All Variants of the LSTM models perform better during training and generalise better on the test set, indicating that the word order is a useful feature in this instance of sentiment classification. One noteworthy outcome of the evaluation is that the top-performing BOW

[1] Runs available at: WandB instance.

Model	Learn. Rate	Embed. dim.	Hidden dim.	*Acc*
BOW	$5 \cdot 10^{-5}$	-	-	0.247
CBOW	$5 \cdot 10^{-5}$	300	-	0.349
DeepCBOW	$5 \cdot 10^{-5}$	300	100	0.366
DeepCBOW^p	$5 \cdot 10^{-5}$	300	100	0.441
LSTM	$5 \cdot 10^{-5}$	300	168	0.459
LSTM Mb	$5 \cdot 10^{-5}$	300	168	0.47
TreeLSTM	$5 \cdot 10^{-5}$	300	168	0.473

Table 1: Baseline configurations for all models within this work, with the corresponding test set accuracy. To this, we further add a pre-trained bidirectional transformer, namely BERT, with the following default parameters: learning rate: 0.0005 and batch size 25.

model, namely DeepCBOW with word embeddings is only 1.8% shy away from the baseline LSTM model, and a 20.5% increase from its counterpart using the dataset embeddings.

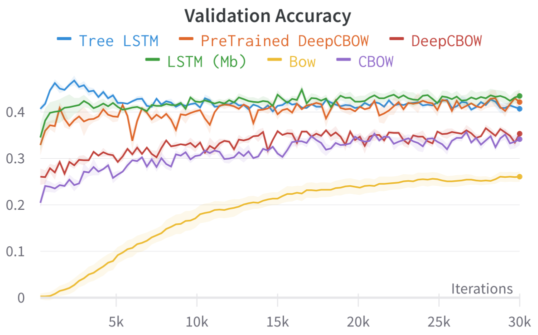

Figure 1: A comparison of the validation performance for the baseline models across training iterations.

5.2 Leveraging Recursive Data Structures

The underlying tree structure appears to have only marginally influenced the performance of the algorithms at test time, as both Tree-LSTM and LSTM offer relatively similar performance, as showcased in Table 1 (only a 0.6% relative improvement). The gap could potentially be expanded by training the model with a larger dataset, as LSTMs have been shown to generalise better under a more extensive dataset (Van Houdt et al., 2020). We initially attempted to extend our training dataset with subtrees and their corresponding node-level sentiment to increase the overall accuracy (resulting in around 155k training samples). However, this direction did not lead to an improvement in generalisation, as the models were trained using ’slices’ of sentences, which would often lack the appropriate context (e.g. ”A film worth throwing away” - ”A film worth”, ”throwing away”).

5.3 Variable Sentence Length Evaluation

We also evaluated the model performances on variable-length sentences. We partition the test dataset (short and long sentences), by observing the distribution of the dataset. Conveniently, as depicted in Figure 3, we split the dataset using the frequency-weighted sentence length mean, which yields a balanced partition of the test dataset. The results on long and short sentence prediction using the baseline model configuration can be found in Table 2. The results obtained indicate that the model achieves a (marginally $\sim 2\%$ on average) better accuracy on the longer sentence length. This behaviour is expected as the model is able to capture more context and a more meaningful semantic representation of the sentence.

Variable Sentence Length Evaluation			
Model	Short *Acc*	Long *Acc*	*Acc**
BOW	0.267	**0.272**	0.247
CBOW	0.349	**0.353**	0.349
DeepCBOW	0.342	**0.357**	0.366
DeepCBOW^p	0.426	**0.47**	0.441
LSTM	0.457	**0.497**	0.459
LSTM Mb	0.424	**0.466**	0.47
TreeLSTM	0.461	**0.503**	**0.473**

Table 2: results for the test set variable sentence length experiment, compared to the model baseline accuracy (last column: *Acc*∗).

5.4 Hyperparameter Tuning

We also explore hyperparameter tuning our model in order to achieve a better test set accuracy. The experiment results are displayed in the parallel coordinate plot in Appendix A, Figure 4, with the best configuration outlined in Table 3. For BERT, we perform further experiments with the hyperparameters suggested by its authors. We observe decent performance improvements across the board (except for TreeLSTM), by mainly altering the learning rate of used in the model’s training.

5.5 Fine-tuning BERT for sentence classification

Finally, we fine-tune a pre-trained instance of BERT with one additional layer in order to adapt it to the sentence classification task at hand. Our findings suggest that even with 10k iterations, the model outperforms the others by a relatively significant margin ($\sim 10\%$). However, the model’s

110M trainable parameters introduce two limitations. First, the SST dataset might not fully engage the complexity of the model and thus lead to unsatisfactory generalisation performance. And subsequently, a computational bottleneck, as the training took around 6X as much as the LSTM-based models and required considerably more memory.

6 Conclusion

This paper provided a comparison between different models for sentiment classification tasks. Our results show that models that capture variable-length sequential data as input indeed outperform the BOW model. However, is notable that we did not see as big of an improvement with the LSTM when compared to DeepCBOW with pre-trained word embeddings. We expected that this model's additional ability to capture variable-length text and tree-structured data would significantly outperform the others. Perhaps, we can then infer that capturing tree-structured data does not play as big of a role in sentiment classification as we previously thought.

In future research it is fundamental to further investigate how we could improve the efficiency of these respective models. Since we only trained on relatively small datasets, it already proved to be computationally expensive and time-consuming. For longer sequences and textual data, the computational complexity is bound to increase. We propose for future research to discover new methods for making the training more efficient.

References

P. Chauhan, N. Sharma, and G. Sikka. 2020. The emergence of social media data and sentiment analysis in election prediction. *Journal of Ambient Intell Human Compute*, 12(1):2601–2627.

Oscar B. Deho, William A. Agangiba, Felix L. Aryeh, and Jeffery A. Ansah. 2018. Sentiment analysis with word embedding. In *International Conference on Adaptive Science Technology (ICAST)*, pages 1–4.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jurgen Schmidhuber. 1997. *Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies*. John Wiley Sons.

Phong Le and Willem Zuidema. Compositional distributional semantics with long short term memory. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics: *SEM*.

T. Mikolov, I. Sutskever, K. Chen, G. Corado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. *Proceedings of the 26th International Conference on Neural Information Processing Systems*, 2(1):3111–3119.

B. Pang, L. Lee, and S. Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical Methods in Natural Language Processing- Volume 10*, pages 79–86.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

K. Saha and D. C. Munmun. 2017. Modeling stress with social media around incidents of gun violence on college campuses. *Proceedings of the ACM on Human-Computer Interaction*, 1:1–27.

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.

Greg Van Houdt, Carlos Mosquera, and Gonzalo Nápoles. 2020. A review on the long short-term memory model. *Artificial Intelligence Review*, 53.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. *Deep Learning for Sentiment Analysis*. John Wiley Sons, Ltd.

Xiaodan Zhu, Prinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures.

A Appendix A

In his appendix, we provide additional details about our model experiments and supporting information with respect to the results and analysis section.

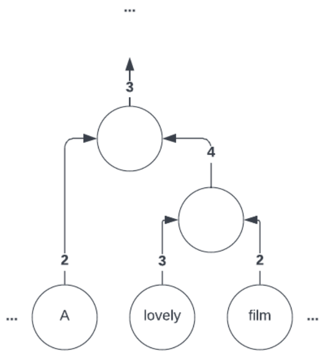

Figure 2: A visualisation of an arbitrary subtree in the SST dataset. The edge values represent the fine-grained sentiment scores, and the leaves are the sentence tokens.

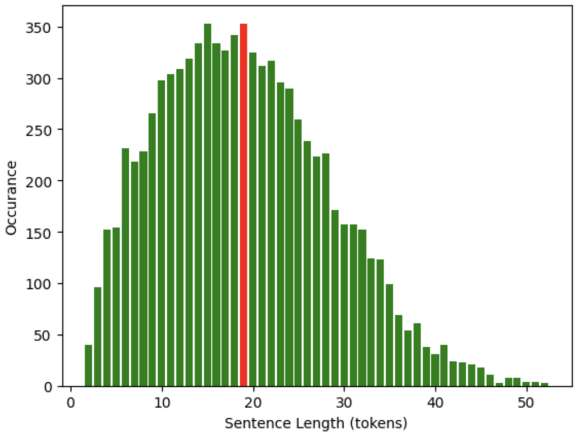

Figure 3: A histogram of the sentence length distribution across the training set. The occurrence of a given sentence length is thus plotted on the y-axis. The red bar in the histogram signifies the ’pivot’ through which we consider sentences short, or long (left of pivot: short, right of pivot: long). The pivot is the weighted mean, and for the SST dataset it is 19.

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{1}$$

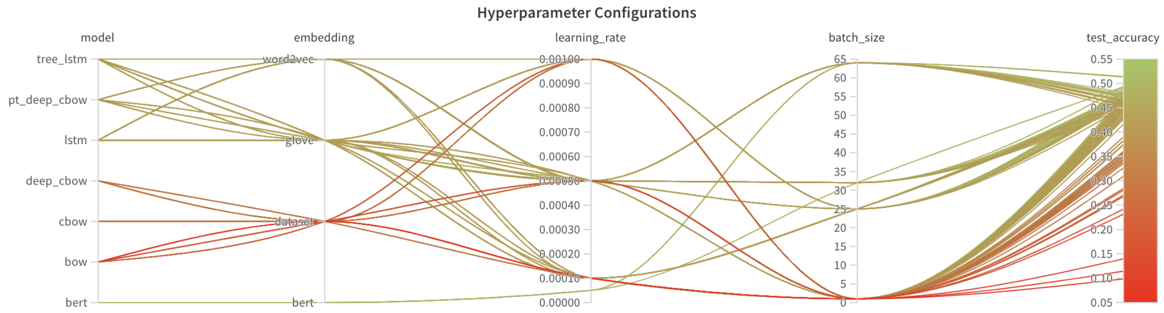

Figure 4: A parallel coordinate plot showcasing the hyperparameter configurations experimented on for each model during hyperparameter tuning. The lines with a greener colour indicate a higher test set accuracy, while the red suggests a poorer generalisation performance.

Hyperparameter Tuning					
Model	Batch size	Embedding	Learn. Rate	Acc	*Acc*∗*
BOW	1	dataset	0.001	0.279 (+ 12.9%)	0.247
CBOW	1	dataset	0.001	0.354 (+ 1.4%)	0.349
DeepCBOW	1	dataset	0.001	0.385 (+ 5.2%)	0.366
DeepCBOW^p	1	Word2Vec	0.00001	0.452 (+ 2.5%)	0.441
LSTM	1	Word2Vec	0.001	0.462 (+ 1%)	0.459
LSTM Mb	25	GloVe	0.0005	0.47 (no improv.)	0.47
TreeLSTM	25	GloVe	0.001	0.479 (+ 1.2%)	0.473
BERT	64	BERT	0.00005	0.514 (+ 4.9%)	0.49

Table 3: Optimal model hyperparameters found during tuning. We showcase here only the hyperparameter configurations which achieved the highest test accuracy (note: averaged over 3 runs with different random seeds) in our experiments. The last column *Acc*∗ resembles the model baseline accuracy. Furthermore, the found relative improvement over the baseline is displayed next to the best attained test accuracy.

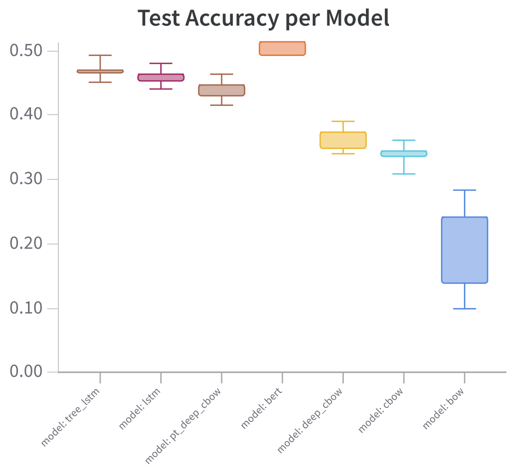

Figure 5: Boxplots for each model's test accuracy, for 3 varying seeds.

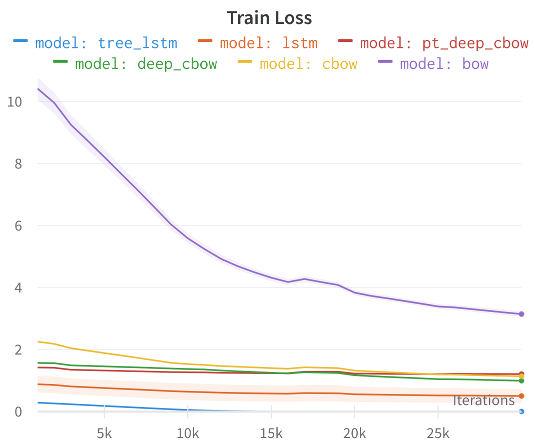

Figure 6: The training loss across interactions for the baseline models.