Practical 2 NLP 1: Sentiment Classification Leveraging Language Structures

Jose Luis Garcia
Student ID: 15388867

Carlos Miguel Patiño
Student ID: 15485250

1 Introduction

Sentiment analysis is a common task in natural language processing to determine the sentiment of a given text. Among the many datasets used for this task, the Stanford Sentiment Treebank (SST) dataset (Socher et al., 2013) stands out as a benchmark for fine-grained sentiment analysis of movie reviews, offering five sentiment classes: very negative, negative, neutral, positive, and very positive.

The main goal of this work is to explore how different model architectures—ranging from bag-of-words (BOW) approaches to recurrent and tree-structured models perform in sentiment classification, particularly when considering the role of word order, sentence length and syntactic structure. Specifically, we aim to address the following research questions:

- How does incorporating word order impact the performance of sentiment classification models?
- How does the length of reviews impact model performance?
- Related to the previous point, is it helpful to use the sentiment at each node on the tree to achieve better sentiment prediction for the entire test?
- Does leveraging the tree syntactic structures improve model robustness and performance?
- How does the N-ary Tree LSTM architecture compare to the Child-Sum Tree LSTM?

These questions are motivated by gaps in the literature regarding direct comparison of models that leverage different structures of language—i.e., word order, sentence length and syntax. Previous works, such as (Mikolov et al., 2013), have demonstrated the effectiveness of distributed word representations in capturing semantic relationships, while (Tai et al., 2015) showed the advantages of tree-structured models in encoding syntactic information. However, comparisons between these approaches for sentiment classification remain underexplored, especially in the context of varying text lengths and the robustness of models to longer sequences.

To answer these questions, we trained and evaluated ten different architectures on the SST dataset: Neural Bag-of-Words (BOW), Continuous Bag-of-Words (CBOW), Deep CBOW, Deep CBOW with pre-trained GloVe word embeddings (Pennington et al., 2014) (DeepCBOW (PT)), Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997), including variations that incorporated mini-batches during training (LSTM MB) and fine-tuning the pre-trained GloVe word embeddings (LSTM MB FT), and Tree-LSTMs (Tai et al., 2015), using two architecture variations: N-ary Tree LSTM and Child-Sum Tree LSTM (Tree LSTM CS). For the N-ary Tree LSTM, we also evaluated two cases: one where the entire tree was parsed only once as an input (Tree LSTM Bi) and another where each node was treated as a separate tree for additional examples (Tree LSTM BiN). These models vary in their ability to capture word order and syntactic structure. The BOW-based models treat sentences as unordered collections of words, while the LSTM models explicitly encode word order, and the Tree-LSTM models encode hierarchical structures.

The main findings of our report are:

- Using word order and the syntax structure of the text results in better accuracy when predicting sentiment.
- Models that do not incorporate word order cannot capture the review sentiment as effectively as models that do. However, we also show that the LSTM struggles more with longer sequences than BOW-based mod-

els. This finding aligns with the known issue (Hochreiter et al., 2001) of LSTMs learning long-term dependencies.

- Longer sentences become increasingly difficult to predict, resulting in a gradual decrease in the overall accuracy of all models as sentence length increases.

- Pretrained embeddings help compensate for the lack of word order in the BOW methods. We hypothesize that GloVe embeddings incorporate a notion of word order during training by using a window to determine co-occurrence.

- Supervising sentiment at the node level for the training data of the Binary Tree LSTM slightly improves the ability to learn sentiment representations, highlighting the importance of finer-grained supervision for hierarchical models.

- For cases where each node of the tree structure has a maximum of two children, the Child-Sum Tree LSTM architecture demonstrated slightly better generalization compared to the Binary Tree LSTM.

2 Background

Our objective is to predict the sentiment of a text sequence, so we need an approach that takes raw text to a representation we use throughout different model architectures. The first step in this process is transforming words (or tokens) into their vector representations called word embeddings. Word embeddings are dense vector representations of words that capture semantic relationships in a lower-dimensional space. Unlike simple one-hot encoding, where a sparse vector represents each word, word embeddings create dense numerical representations where words with similar meanings have similar vector representations. This semantic similarity is reflected in the vectors' distance and direction, making them particularly effective for sentiment analysis tasks. This report uses word embeddings as the first step of all the models we trained.

The first model group we trained is based on the neural Bag-of-Words (BOW) approach. The neural BOW, continuous BOW (CBOW) (Mikolov et al., 2013), and Deep CBOW aim to learn word embeddings that we then add to build a sentence representation that can be used in downstream tasks.

One of the shortcomings of BOW models is that the sum to generate the sentence representation does not encode the order of the words in the sentence. That loss of information can be problematic when we require the word order to classify a sentence's sentiment correctly.

We also trained LSTM (Hochreiter and Schmidhuber, 1997) and Tree LSTM (Tai et al., 2015; Le and Zuidema, 2015; Zhu et al., 2015) models to incorporate word order information. The LSTM is a recurrent neural network with three gates—input, forget, and output—that regulate information flow through a memory cell and a hidden state. The gates manage to discard, store, and output weighted information at each time step, allowing the model to maintain and update long-term dependencies.

The Tree LSTM architectures extend the standard LSTM by incorporating a tree structure into the computation, which enables the processing of hierarchical data instead of sequential data. Each node in the tree represents a computation unit that incorporates information from its child nodes and the current input, updating its hidden and memory states accordingly. These models adapt the forget, input, and output gates to handle contributions from multiple child nodes, allowing the model to selectively retain or discard information from specific child nodes as it needs.

3 Models

As mentioned in the previous section, CBOW models generate a sentence representation by summing the word embeddings of all words in the sentence. The embedding dimension depends on the model: for neural BOW, it matches the number of classes in the task (5 in our case), while for CBOW, Deep CBOW, and LSTM models, it is a hyperparameter set to 300. We then project the embedding dimension to the five classes using a linear layer in the CBOW and LSTMs models, and an MLP with three layers in the DeepCBOW case. The hidden layers in the MLP have a dimension of 500. Finally, we use the final output layer of the network to predict the class with the highest score. For LSTM models, the hidden state of the sequence's final element serves as input to a linear projection layer that outputs class scores. Cross entropy loss function was used to quantify the model's error and update the weights through the backpropagation algorithm

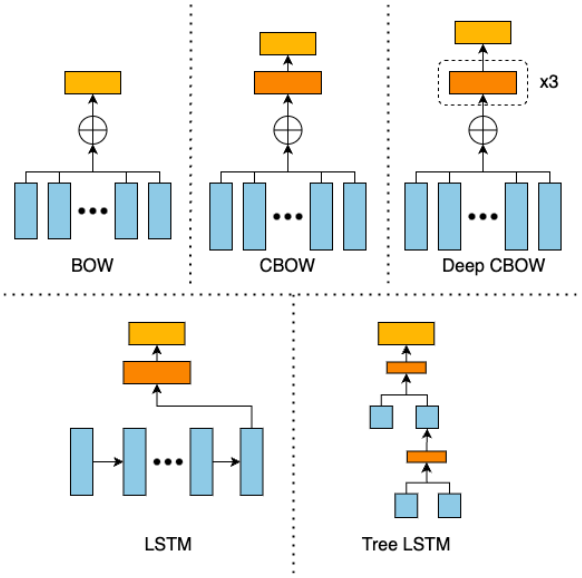

Figure 1: Diagrams for all the architecture we used to predict the sentiment of a text. The output of the models is in yellow, and the projection layers that take embeddings to intermediate representations are in orange. The word embeddings (or the node states for the Tree LSTM) are in blue.

(Rumelhart et al., 1986).

We summarized all the architectures in Figure 1, showing how we start from the token sequence to the final layer we use for classification.

4 Experiments

We trained ten architectures to compare different approaches to classify sentiment on the Stanford Sentiment Treebank (SST) (Socher et al., 2013) dataset. The classification task was to predict the sentiment of the whole text out of five classes: very negative, negative, neutral, positive, and very positive. Each review had only one class associated with the text, so our models were multiclass classifiers.

We used the same word-level vocabulary for the BOW and CBOW models, while switching to pre-trained word embeddings for the DeepCBOW and all LSTM models. Specifically, we used pre-trained GloVe embeddings (Pennington et al., 2014) and constructed a vocabulary based on the words with available embeddings. Additionally, we experimented by freezing the pre-trained embeddings during training and optimizing only the linear layers in the forward pass, except for one of the Binary Tree LSTM classifiers where the embeddings were fine-tuned.

We tuned the optimal number of epochs by using early stopping with the patience of 10 epochs. In the Appendix, we included the number of iterations we used for training in Table 3. The optimizer we used to update the weights was Adam (Kingma and Ba, 2017) with a learning rate of 5×10^{-4}. The metrics we used to monitor model performance were the accuracy and cross-entropy loss, and we saved the checkpoint with the best accuracy to evaluate the final performance of each architecture. We used a development dataset to evaluate model performance during training and hyperparameter optimization and calculated the metrics in the report using a separate test set.

To account for randomness in the initialization, we trained each BOW architecture three times and reported the mean and standard deviation for the model metrics. However, we were unable to do this for the LSTMs due to the increased model complexity and longer training times.

5 Results and Analysis

The accuracy results for all the models show three distinct groups in terms of performance. The LSTMs, Tree LSTMs, and DeepCBOW models with pre-trained weights (DeepCBOW (PT)) have an average accuracy above 0.43. The other three models—BOW, CBOW, and DeepCBOW—have accuracies of 0.39 or lower. The group with worse performance has in common that they do not use word order to make the prediction, so we can conclude that word order is important for sentiment classification.

The observation about word order above is also supported by the DeepCBOW (PT), which performs similarly to the LSTM-based models despite being a CBOW-based model. The DeepCBOW (PT) has information on the co-occurrence of words because the GloVe embeddings were trained to predict whether two tokens co-occur in a window. Although co-occurrence within a window is not strictly word order, it does encode a certain distance between words—i.e., words that are far apart would not be in the same window. That is why the DeepCBOW (PT) model has an accuracy closer to the LSTM-based models than other CBOW-based ones.

The results show that the LSTM tree structure helps to get better accuracy, but the additional information obtained from these structures is not large. The gain, measured as the accuracy gain from the LSTM to the Tree LSTM, is only 1.3% in the best

case found. Furthermore, the variance between the two architectures shows an overlap in performance, so we cannot conclude that using the tree structure results is more accuracy.

The SST dataset contained texts of lengths ranging from 2 to 56 tokens. We grouped the lengths into four groups to test how each model performs on different sentence lengths. We summarize the results in Figure 3 and provide the bucket distribution in Table 2 in the Appendix. Figure 3 shows that we have a split in performance similar to the one we observed in Figure 2—i.e., we see a performance gap from the three best models compared to all bottom ones. Also, we see a downward trend for all cases where accuracy decreases as the examples are longer. A downward trend is expected because a longer text may contain more nuances and information that we need to consider to capture the text's sentiment accurately.

The results in Table 1 show that all CBOW-based models have the largest drop for examples with 30 or more tokens, while almost all the LSTM-based models have the highest drop for the examples with 20 to 29 tokens. Also, we show that the LSTM struggles with long texts more than CBOW-based models. This behavior showcases the known issue (Hochreiter et al., 2001) of the LSTM architecture to carry information through long sequences. In contrast, BOW-based models avoid this issue since all word positions contribute equally to the representation, making them less sensitive to sequence length.

The results also show that leveraging the tree structure available in the dataset makes the model robust to long sequences. Among the Tree LSTM models, the Binary Tree LSTM demonstrated a smaller performance drop across all sequence lengths, including long ones, while maintaining a relatively high accuracy compared to other models. This result highlights that incorporating hierarchical structural information is more effective for handling long texts than relying solely on word order. LSTM models struggle more with moderately long sequences, they maintain better overall accuracy across all sequence lengths compared to BOW-based models. Similarly, CBOW-based models show significant drops in accuracy for examples with 30 or more tokens, while LSTM models exhibit the largest drops for sequences of 20 to 29 tokens.

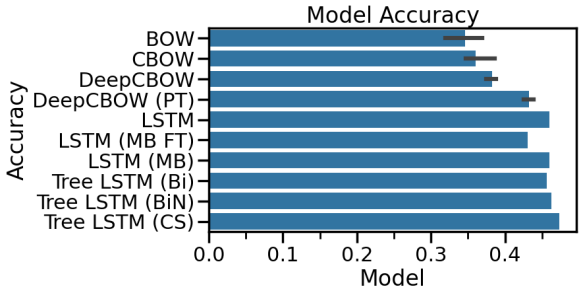

Figure 2: Accuracy of the models we trained for the sentiment classification task. The best model was the Tree LSTM, with a close second being the LSTM. The DeepCBOW with pre-trained GloVe embeddings (DeepCBOW (PT)).

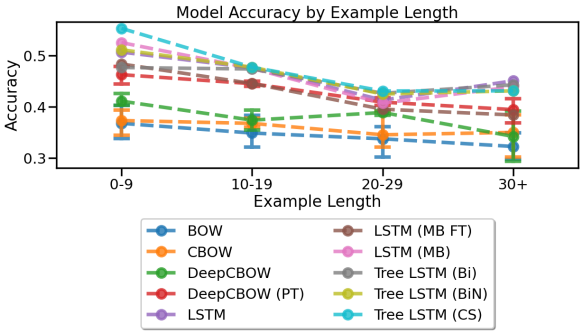

Figure 3: Accuracy of the model as a function of the length of the example. We group the examples in buckets to provide an easier trend visualization and report the mean and variance calculated from three runs per model. All models present a similar behavior of decreasing accuracy as the examples are longer.

Model	0-9	10-19	20-29	30+
BOW	0.0	-5.2	-8.2	**-12.3**
CBOW	0.0	-1.5	-7.4	**-6.2**
DeepCBOW	0.0	-9.0	-5.4	**-16.7**
DeepCBOW (PT)	0.0	-3.8	-11.6	**-14.7**
LSTM	0.0	-6.0	**-18.5**	-11.0
LSTM (MB FT)	0.0	-7.9	-18.2	**-20.6**
LSTM (MB)	0.0	-9.6	**-22.5**	-16.4
Tree LSTM (BiN)	0.0	-6.6	**-16.9**	-15.7
Tree LSTM (Bi)	0.0	-0.6	**-10.5**	-7.1
Tree LSTM (CS)	0.0	-13.7	**-22.1**	-22.0

Table 1: Percentage drop of accuracy using the bucket of best performance as a baseline. All the reported numbers in the table are percentages, and the numbers in bold correspond to the largest drop in performance for each model.

6 Conclusion

Our experiments demonstrate that incorporating word order and syntax structure improves sentiment classification accuracy and robustness to review length. Models like Tree-LSTMs, which leverage hierarchical syntax, outperform simpler approaches, particularly for longer reviews. However, we observed that LSTMs struggle with very long sequences despite their sequential modeling capabilities, aligning with known limitations in learning long-term dependencies. Interestingly, while less sophisticated, BOW-based models showed better robustness to sequence length than LSTMs, due to their equal weighting of all word positions in the representation.

Finally, pre-trained GloVe embeddings effectively enhanced all the models that used it, specially DeepCBOW (PT), compensating for the lack of explicit word order by encoding contextual information through co-occurrence windows. This result compensates for the lack of explicit word order modeling in CBOW-based models. LSTM managed to obtained better overall results but supervising sentiment at the node level for the Binary Tree-LSTM improved performance by enabling finer-grained learning of sentiment representations, emphasizing the importance of localized supervision in hierarchical models. Furthermore, for this dataset, our comparison between the N-ary Tree-LSTM and Child-Sum Tree-LSTM showed that the Child-Sum approach provided slightly better generalization, likely due to its efficient aggregation of information across child nodes.

Future work could explore attention mechanisms over the LSTM's hidden states to address the long-term dependency problems or use the tree structure with transformers to compare the approach with the Tree LSTM model.

References

S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. C. Kremer and J. F. Kolen, editors, *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press.

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

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 10–19, Denver, Colorado. Association for Computational Linguistics.

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

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.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

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.

Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1604–1612, Lille, France. PMLR.

A Appendix

Length Bucket	Example Count	Example Share (%)
0-9	432	38
10-19	848	30
20-29	675	20
30+	255	12

Table 2: Table showing the distribution of samples for each length bucket.

Model	Number of Iterations
BOW	250,000
CBOW	150,000
DeepCBOW	50,000
DeepCBOW (PT)	100,000
LSTMs	25,000
Tree LSTMs	30,000

Table 3: Number of iterations to reach convergence for each model.