Evaluating Neural Architectures for Sentence Representation and Sentiment Classification

Luc Buijs
14047144
lucbuijs123@gmail.com

Konstantinos Zafeirakis
15831256
konstantinos.zafeirakis@gmail.com

1 Introduction

Sentiment analysis is a popular task within the field of natural language processing (NLP). This task essentially boils down to using computational power to understand the semantic meaning implied in a text (Gao et al., 2019). This is essential to understand user-generated text, such as in social discussions or product reviews (Kanakaraj and Guddeti, 2015). Sentiment analysis can be extremely resourceful for recommender systems or marketing algorithms. According to Kanakaraj and Guddeti, the Bag-of-Words (BOW) model is a commonly used approach for sentiment analysis. This approach only considers individual words and builds the feature vectors based on the word count. The main problem of this approach is that the model does not take word-order into account and is unable to capture the relationship between words as the model considers each word individually. A model that tackles this problem is the Long Short-Term Memory (LSTM) model. This model is a recurrent neural network architecture that is capable of capturing long-term relationships in input sequences (Prasad et al., 2023). The model accomplishes this by using a set of memory cells that can selectively retain and discard information as needed.

In this report we will train and evaluate multiple variations of BOW and LSTM models on the Stanford Sentiment Treebank (Link). We aim to create models which are able to classify the sentiment of unseen movie reviews into one of five different classes; very negative, negative, neutral, positive and very positive. We will perform multiple experiments to gain more insight on the performance of different models on this specific task. We will start with investigating how important word order is for this task. This is done by evaluating the performance of the BOW models, which do not take word-order into account and compare this performance with the LSTM models which are capable of incorporating word-order. Barry performed a similar research, evaluating the performance of BOW models versus LSTM models that were trained for sentiment classification on a dataset containing food reviews. The LSTM models outperformed the BOW models, implying the importance of word order. These results align with the expectations for our own experiment.

Secondly, we will investigate whether a tree structure enhances the accuracy in sentiment classification by comparing the performance of a Tree-LSTM with a regular LSTM model. Tai et al. introduced introduced the Tree-LSTM in 2015. Their research revealed that the Tree-LSTM model performs significantly better than the baseline LSTM models, achieving higher accuracy for semantic relatedness prediction as well as for sentiment classification. Their research additionally showed that the LSTM models perform significantly better on shorter sentences. We will also investigate this by comparing the accuracy of our models on short versus long sentences.

Finally, we will compare the performance of the Tree-LSTM introduced by Tai et al. with the LSTM formulated by Le and Zuidema. The key of Le and Zuidema's approach is that their LSTM contains direct connections from the memory cells to the gates, potentially allowing more complex memory dynamics.

The highest accuracy on the classification task is achieved by the Tree-LSTM model. Le and Zuidema's model followed, only achieving a slightly less accuracy. The LSTM models perform significantly better than the BOW models overall, implying the importance of word order. The sentence length also has a significant impact on the accuracy. Models show to achieve greater accuracy on the shorter sentences, possibly due to these sentences being less complex than the longer ones.

2 Background

2.1 Word Embeddings

Word embeddings represent words as dense, low-dimensional vectors, capturing semantic and syntactic relationships. Unlike one-hot encodings, embeddings allow similar words to have similar vector representations. Common models like Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) learn embeddings by predicting words based on context. These embeddings are fine-tuned during model training to improve performance in tasks like sentiment analysis. In Tree-LSTM models, word embeddings encode sentences into vector representations for effective classification.

2.2 Bag Of Words

The Bag of Words (BoW) model (Qader et al., 2019) represents text as fixed-length vectors based on word occurrence, ignoring grammar and word order. While BoW is efficient and works well for tasks where word order is not crucial, it has limitations, such as producing sparse vectors and failing to capture word relationships. Nonetheless, BoW serves as a baseline for more advanced techniques like word embeddings.

2.3 LSTM

Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) address the vanishing gradient problem in traditional RNNs by using input, forget, and output gates to capture long-term dependencies. In natural language processing, LSTMs process sentences word by word, maintaining a hidden state that encodes contextual information. However, they are less effective at modeling hierarchical structures compared to Tree-LSTMs.

2.4 Tree LSTM

Tree-LSTMs (Tai et al., 2015) extend LSTMs to tree-structured data, capturing hierarchical relationships in tasks like sentiment analysis. Instead of processing data sequentially, Tree-LSTMs compute hidden states at each node based on its children's states. There are two variants: the Child-Sum Tree-LSTM, which aggregates information from multiple children, and the N-ary Tree-LSTM, which handles a fixed number of children. Tree-LSTMs are particularly effective for syntactically complex sentences, providing fine-grained sentiment supervision compared to flat models like BoW or sequential LSTMs.

3 Models

3.1 BoW

Moreover the BOW model uses an embedding layer to map input word IDs to dense vectors, which are summed across the sentence. A trainable bias term is added to this sum. The resulting logits are used for sentiment classification. The model is trained to predict the sentiment based on this aggregated representation. Cross-entropy loss was used during training

3.2 CBoW

The CBOW model leverages word embeddings of size 300. The input words are first embedded using an embedding layer, and their embeddings are summed across the sentence. The summed embeddings are passed through a linear layer, which projects the result to the sentiment classes. The model predicts the sentiment based on this projection. Cross-entropy loss was used during training

3.3 DeepCBoW

The DeepCBOW model extends the CBOW approach by incorporating non-linearity. It starts with an embedding layer of size 300, followed by two hidden layers with 100 units each, activated by the Tanh function. The final output layer projects the result to the sentiment classes. The input words are embedded and summed, then passed through the hidden layers before making the sentiment prediction. Cross-entropy loss was used during training

3.4 LSTM

The LSTM model is built with an embedding layer of size 300 and a hidden state of size 168. The model uses pre-trained word embeddings, which are frozen during training. The input words are passed through an LSTM network, where the hidden state incorporates information from the entire sentence. The final hidden state is used for sentiment prediction. Cross-entropy loss is used for training.

3.5 LSTM minibatched

The LSTM model with minibatch training uses pre-trained word embeddings, where the embedding layer is initialized with fixed word vectors (non-trainable). The training process updates the

model after each minibatch, and evaluation occurs periodically. In the fine-tuned version, the model undergoes additional training to refine its parameters, further enhancing accuracy and generalization by updating the embeddings and adjusting the optimizer's learning rate to improve performance. Cross-entropy loss is used for training.

3.6 Tree LSTM

The model uses pre-trained word embeddings, which are frozen during training. The sentences are processed using the TreeLSTM, which handles hierarchical structures by incorporating information from both left and right children. The root state of the tree is used for classification. Cross-entropy loss is employed for training

3.7 Le & Zuidema's LSTM

The key difference made for this model with respect to the LSTM model, is the addition of three linear projections for cell state connections. Another difference comes from the input and forget gate which include cell state contributions in this variant.

4 Experiments

Our research consists of four experiments as discussed in the introduction. For every experiment the results are the average of 3 iterations each with 3 different seeds namely seed 1, 2 and 3. As for the embedding type the GloVe embeddings were used. Finally when it came to choosing a specific threshold which will classify the sentences to either short or long, after experimentation with different thresholds it was decided that the threshold would be 18 with higher than 18 token sentences being classified as long and lower as short. This is because at 18 tokes the total short and small sentence number is the closest (1003 to 1104) making it close to the average token length.

4.1 Data

The dataset that we used was the Stanford Sentiment Treebank (SST) which is a benchmark dataset for sentiment analysis, designed to assess models' ability to understand compositional semantics. Unlike typical sentiment datasets, SST provides sentiment labels at both the sentence level and the phrase level, offering a tree-structured representation of sentences derived from their syntactic parse trees.

Each node in the tree represents a phrase, annotated with one of five sentiment categories: very negative, negative, neutral, positive, or very positive. This granularity enables training models that predict fine-grained sentiment for each constituent in a sentence, making SST a valuable resource for evaluating Tree-LSTM models and other techniques that leverage hierarchical information.

SST's rich annotations and syntactic structure make it ideal for studying the impact of sentence composition and tree-based architectures on sentiment prediction tasks.

4.2 Training parameters

For the training of all the Bow models namely the Bow, CboW and DeepCBoW the adam optimizer was used with a learning rate of 0.005, the Cbow model had an embedding dimension of 300 for 5 classes as did the DeepCBow model but with the addition of a hidden state of 100. For the LSTM and Zuidema's LSTM training the model is optimized using the Adam optimizer with a learning rate of 0.0003. As for the LSTM minibatched versions finetuned and not the models had a hidden state size of 168 and an embedding dimension of 300, and were trained using a batch size of 25 and optimized with Adam at a learning rate of 0.0004. Finally the Tree LSTM model is constructed with an embedding layer of size 300 and a hidden state of size 150 and was optimized using the Adam optimizer with a learning rate of 0.0004.

5 Results and Analysis

5.1 Word Order Importance

Word order is crucial for this task, as shown by the performance differences between models. Bag-of-Words (BOW), which ignores word order, achieves the lowest scores , reflecting its inability to capture sequential relationships. CBOW improves slightly by using embeddings but still falls short compared to models that incorporate order.

Sequential models like LSTM significantly outperform unordered approaches, with scores up to 0.487. The addition of hierarchical modeling in LSTM-TREE boosts performance further (0.496). These results highlight the importance of capturing both sequential and structural dependencies.

Therefore models that incorporate word order, especially LSTMs, demonstrate the strongest performance, emphasizing its critical role in understanding sentence structure for this task.

Model	Iterations	Accuracy	Short sent.	Long sent.
BOW	29666	0.267 (0.012)	0.255 (0.006)	0.277 (0.018)
CBOW	29333	0.353 (0.012)	0.366 (0.015)	0.341 (0.015)
DeepCBOW	17333	0.379 (0.011)	0.383 (0.013)	0.378 (0.01)
PTDeepCBOW	24666	0.434 (0.009)	0.455 (0.015)	0.412 (0.006)
LSTM	23666	0.465 (0.011)	0.487 (0.014)	0.444 (0.013)
LSTM Minibatch	3916	0.462 (0.05)	0.495 (0.01)	0.428 (0.006)
LSTM Minibatch Finetuned	2000	0.457 (0.003)	0.482 (0.007)	0.436 (0.012)
LSTM-TREE	2666	0.473 (0.004)	0.496 (0.003)	0.451 (0.01)
LSTM-ZUIDEMA	20666	0.472 (0.003)	0.495 (0.009)	0.451 (0.002)

Table 1: Results across three runs with different random seeds each. Table shows the amount of iterations until convergence, average accuracy (followed by the standard deviation) and the accuracy on short and long sentences.

5.1.1 Impact of Tree Structure on Accuracy

Incorporating a tree structure improves accuracy by capturing hierarchical relationships in the data. Comparing models, LSTM-TREE achieves higher accuracy than standard LSTM for all sentence lengths. This improvement reflects the ability of tree-based models to model compositional semantics, effectively understanding how smaller units combine into larger meanings.

The tree structure allows the model to go beyond linear word order, capturing dependencies and relationships across different sentence levels. This advantage is significant for tasks requiring nuanced understanding, such as in our case sentiment analysis or syntax-based predictions.

5.1.2 Impact of Sentence Length on Accuracy

The sentence length has a significant impact on the accuracy of the sentiment classification models. Table 1 shows that each model except the first BOW model achieves a higher accuracy on the short sentences than on the long sentences. A reason for this might be the fact that semantic information of shorter sentences can be learned more easily because they are simply less complex than longer sentences. Shorter sentences often have a simpler structure and fewer dependencies between tokens.

5.1.3 How do the Tai et al. Tree LSTMs compare to Le & Zuidema's formulation

The Tree LSTM model and the formulation by Le & Zuidema both perform similarly in terms of accuracy and handling short and long sentences. Both models achieve an accuracy around 0.472 to 0.473, with LSTM-TREE slightly outperforming LSTM-ZUIDEMA in short sentences but having almost identical performance in long sentences. The Tree LSTM leverages a hierarchical structure that potentially aids in better processing of complex sentence dependencies, while the LSTM-ZUIDEMA formulation focuses on more sophisticated gate computations but shows comparable performance in this setup.

6 Conclusion

The experiments revealed that word order plays a significant role in sentiment classification. This result aligns exactly with our expectations and is consistent with findings from other research. While models like BOW, which ignore word order, typically underperform, BOW surprisingly achieved higher accuracy on long sentences. This may be due to its ability to handle the broader context of longer sentences more effectively, despite its lack of sequential understanding. The other models showed notably better on shorter sentences, also aligning with the expectations. LSTM-based models, particularly Tree LSTM, performed the best overall by capturing sequential and hierarchical relationships. This proves that introducing a tree structure significantly enhances the accuarcy in sentiment classification. The LSTM-ZUIDEMA variation provided similar performance with more complex gating mechanisms.

For future research, it could be interesting to extend upon the Tree-LSTM experiment by investigating the impact of using different tree construction strategies (e.g. constituency vs dependency trees). This can help us to understand which structure contributes most to this specific task. Furthermore, hyperparameter tuning can be considered for future work, as this was not conducted in our research and might enhance the models' performance.

References

James Barry. 2017. Sentiment analysis of online reviews using bag-of-words and lstm approaches. In *AICS*, pages 272–274.

Zhengjie Gao, Ao Feng, Xinyu Song, and Xi Wu. 2019. Target-dependent sentiment classification with bert. *Ieee Access*, 7:154290–154299.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Monisha Kanakaraj and Ram Mohana Reddy Guddeti. 2015. Nlp based sentiment analysis on twitter data using ensemble classifiers. In *2015 3Rd international conference on signal processing, communication and networking (ICSCN)*, pages 1–5. IEEE.

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. *arXiv preprint arXiv:1503.02510*.

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

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. volume 14, pages 1532–1543.

Oruganti John Prasad, Subham Nandi, Varun Dogra, and Dammu Sai Diwakar. 2023. A systematic review of nlp methods for sentiment classification of online news articles. In *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–9. IEEE.

Wisam Qader, Musa M. Ameen, and Bilal Ahmed. 2019. An overview of bag of words;importance, implementation, applications, and challenges. pages 200–204.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks.

A Example Appendix

This is a section in the appendix.