Contractive Tree LSTM for Sentiment Classification

Jan Henrik Bertrand
University of Amsterdam
15683834
jan.henrik.bertrand@student.uva.nl

Louis Gehringer
University of Amsterdam
13322842
louis.gehringer@student.uva.nl

1 Introduction

Sentiment classification is a central task in natural language processing (NLP), yet key questions remain regarding the significance of word order, the role of syntactic tree structures, and the influence of sentence length on model performance. The potential advantages of supervising sentiment at every syntactic node, rather than solely at the sentence level, remain underexplored. Additionally, incorporating regularization techniques that have shown success in classical representation learning, such as contractive autoencoders (Rifai et al., 2011; Bertrand et al., 2024), could unlock further potential in enhancing TreeLSTMs. This study evaluates established text representation and sequence modeling approaches — Bag of Words (BOW), Continuous Bag of Words (CBOW) (Mikolov et al., 2013), Deep CBOW, Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997), and TreeLSTMs (Tai et al., 2015) — and extends TreeLSTMs with a contractive regularization as proposed in classical representation learning by Rifai et al. (2011). We refer to the proposed contractive variant of the Tree LSTM as CTREELSTM.

While BOW and CBOW are proven baselines for sentiment classification (Joulin et al., 2017), their disregard for word order raises questions about the importance of sequential information (Iyyer et al.). LSTMs model dependencies between words and have been extended to TreeLSTMs (Tai et al., 2015; Le and Zuidema; Zhu et al., 2015) to also model hierarchical relationships within the language. Understanding if tree-based models outperform sequence-based ones under the performance-runtime trade-off is crucial for guiding model selection. The varying complexity of real-world texts highlights the need to evaluate how models scale with longer sentences. Fine-grained supervision at syntactic nodes may better capture tonal shifts, enhancing TreeLSTM accuracy (Socher et al., 2013). Beyond that, adopting regularization methods that have proven to work well in classical representation learning (Rifai et al., 2011) could lead to additional performance improvements over conventional dropout used in existing literature (Tai et al., 2015).

TreeLSTMs are expected to outperform simpler models such as BOW and CBOW across all sequence lengths, with further gains in accuracy achieved through node-level sentiment supervision. Integrating the contractive regularization is expected to elevate TreeLSTM performance slightly, enhancing the model's ability to retain information more effectively as it is passed up the hierarchy. To test these hypotheses, we evaluate models on the Stanford Sentiment Treebank (SST) dataset across sentence lengths (1–5 to >30 tokens), assess subtree-level inputs for their impact on TreeLSTM performance, and implement a TreeLSTM variant that uses a contractive regularization term as in contractive autoencoders (Rifai et al., 2011) for the projection of left and right child node, to measure its effect on sentiment prediction accuracy.

The results show that the TreeLSTM yields the highest overall accuracy across all sequence lengths. The importance of word order is evident, as BOW models, which disregard it, deliver the lowest accuracy across all sequence lengths. Supervising sentiment at a node level slightly improves TreeLSTM performance, effectively capturing sentiment shifts at the phrase level. The inclusion of a contractive regularization did not lead to any significant changes in performance. Our results highlight the advantages of combining hierarchical modeling, fine-grained supervision.

2 Background

2.1 Bag-of-Words (BOW)

BOW represents sentences by summing embedding vectors, where each vector entry corresponds to one of the sentiment classes. For example, the sentence

"This movie is stupid" is represented as:

$$\mathbf{z} = \mathbf{v}_{\text{this}} + \mathbf{v}_{\text{movie}} + \mathbf{v}_{\text{is}} + \mathbf{v}_{\text{stupid}} + \text{bias}.$$

Vector $\mathbf{z} \in \mathbb{R}^C$ has dimensionality corresponding to C sentiment classes. The `argmax` of this vector predicts the sentiment.

2.2 Continuous Bag-of-Words (CBOW)

CBOW extends BOW by introducing dense word embeddings, representing words in a low-dimensional space. Embeddings encode semantic relationships between words, such that similar words are closer in the embedding space. Sentence representations are constructed by summing these embeddings into a single vector $\mathbf{x} \in \mathbb{R}^d$, where d is the embedding dimension. This vector is then projected into the output space of sentiment classes via a learnable matrix $W \in \mathbb{R}^{C \times d}$:

$$\mathbf{z} = W \cdot \mathbf{x} + \mathbf{b},\ \mathbf{z} \in \mathbb{R}^C$$

The `argmax` of $\mathbf{z}$ determines the sentiment class.

2.3 Deep CBOW

Deep CBOW furthers CBOW by introducing multiple non-linear layers, allowing for more complex representations. Instead of directly mapping the summed embedding vector $\mathbf{x}$ to the output, Deep CBOW applies a series of transformations with learnable weight matrices $W_1, W_2, \ldots$, biases $\mathbf{b}_1, \mathbf{b}_2, \ldots$, and non-linear activation functions f (e.g., ReLU):

$$\mathbf{h} = f(W_2 \cdot f(W_1 \cdot \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2).$$

The final transformed vector $\mathbf{h} \in \mathbb{R}^d$ is then mapped into the sentiment class space using an output weight matrix $W_{\text{out}} \in \mathbb{R}^{C \times d}$:

$$\mathbf{z} = W_{\text{out}} \cdot \mathbf{h}, \quad \mathbf{z} \in \mathbb{R}^C, \quad \text{Prediction} = \text{argmax}(\mathbf{z}).$$

2.4 LSTM

LSTMs model sequential dependencies using gating mechanisms to control the flow of information with key components: i) Forget gate: Filters information from previous cell state. ii) Input gate: Integrates new, relevant information to the cell state. iii) Output gate: Produces the hidden state for the current time step.

The hidden state at each time step is:

$$\mathbf{h}_t = o_t \odot \tanh(\mathbf{c}_t),$$

where o_t and $\mathbf{c}_t$ are the output gate and cell state, respectively. These mechanisms allow LSTMs to dynamically balance retaining and updating information, effectively capturing dependencies in sequential data.

2.5 Tree-LSTM

Tree-LSTMs extend LSTMs to hierarchical data. Instead of processing sequential inputs, Tree-LSTMs aggregate information from multiple child nodes. For a node j, the cell state aggregates information from its children $C(j)$:

$$\mathbf{c}_j = \sum_{k \in C(j)} f_{jk} \odot \mathbf{c}_k + i_j \odot \tilde{\mathbf{c}}_j,$$

where: i) f_{jk} is the forget gate for child k, ii) i_j is the input gate, iii) $\tilde{\mathbf{c}}_j$ is the candidate cell state for node j. The hidden state for node j is computed as:

$$\mathbf{h}_j = o_j \odot \tanh(\mathbf{c}_j),$$

where o_j is the output gate.

2.6 Contractive Autoencoders

Contractive autoencoders (CAE), proposed by Rifai et al. (2011), add the Frobenius norm of the encoder's Jacobian as a regularization term to the reconstruction loss. This encourages the model to learn to be invariant to noise in the input, which enhances generalization. It has shown to outperform other common representation learning techniques in classical representation learning settings (Rifai et al., 2011; Bertrand et al., 2024).

3 Models

3.1 Bag-of-Words (BOW)

Our BOW model is a simple neural implementation that maps each word to a trainable embedding vector of size 5, corresponding to the sentiment classes. Word embeddings are summed for each sentence during forward pass, with a trainable bias added, producing vector $\mathbf{z} \in \mathbb{R}^5$. The argmax of the vector determines the classification.

3.2 Continuous Bag-of-Words (CBOW)

Our CBOW model extends the BOW model using dense word embeddings with a dimensionality of 300, with an additional linear layer projecting it to the 5 sentiment classes.

3.3 Deep CBOW

Our Deep CBOW model has two linear hidden layers with a dimensionality of 100 followed by Tanh activation functions, introducing non-linearity over the regular CBOW model. Additionally, pre-trained word embeddings from GloVe are used to initialize the word embedding lookup table, leveraging the model's non-linear layers.

3.4 LSTM

Our LSTM model maps each word in the input sentence to a dense vector $\mathbf{v} \in \mathbb{R}^{300}$ ignoring padding tokens during embedding lookup. Each LSTM cell processes the input one timestep at a time, updating the hidden $\mathbf{h}_j$ and cell $\mathbf{c}_j$ states. The final hidden layer features a dropout layer with $p = 0.5$ to prevent overfitting. A further fully connected layer maps the final hidden state to the output dimension corresponding to the five sentiment classes.

3.5 TreeLSTM

Our TreeLSTM model encodes sentences via a hierarchical tree structure. Embeddings are processed based on a binary syntactic tree, updating hidden states $\mathbf{h}_j$ and cell states $\mathbf{c}_j$ for each node by combining the states of its child nodes $C(j)$. The root node's hidden state, representing the entire sentence, is passed through a dropout layer with $p = 0.5$ to prevent overfitting. A fully connected layer then maps the root hidden state to the output dimension corresponding to the five sentiment classes.

3.6 CTreeLSTM

Our CTREELSTM varies the TreeLSTM model by replacing the dropout following the projection of the left and right child into the parent node's hidden state with a contractive regularization term. This regularization encourages robustness in the learned representations by constraining the transformations, reducing sensitivity to small perturbations in the child states and thereby improving information retainment. The remaining architecture and processing follow the standard TreeLSTM framework, with the root node's hidden state used for final classification.

4 Experiments

We run four different experiments to answer the posed research questions: (1) a comparison of all models, except for the proposed CTREELSTM, also referred to as the main comparison, (2) a comparison of the previously mentioned models for different sentence lengths based on the checkpoints from the first experiment, (3) a comparison of a N-ary Tree-LSTM as of Tai et al. (2015) to our CTREELSTM and (4) a comparison of the Tai et al. TreeLSTM trained with node-level supervision versus trained on the entire sentences only. We run all model trainings thrice with different random weight initializations and report the mean test performance on the Stanford Sentence Treebank (SST)[1]. We chose hyperparameters such as the learning rate, number of iterations and the weighting factor for the contractive regularization term based on initial explorative experiments and intuition due to the fact that proper hyperparameter optimization was not possible under our resource constraints. We provide an overview of all hyperparameters used in Appendix A.

5 Results & Analysis

Model	Test Accuracy	Runtime (sec)
BOW	0.2789 (0.0010)	9.83 (0.11)
CBOW	0.3130 (0.0095)	75.47 (0.72)
Deep CBOW	0.4226 (0.0060)	81.97 (3.08)
LSTM	0.4609 (0.0139)	230.96 (5.20)
TreeLSTM	**0.4676 (0.0041)**	720.16 (16.36)

Table 1: **Main Comparison:** Sentiment classification performance comparison on the SST dataset along-side its runtime. Results are based on three runs with different model initializations. Standard deviations are in parentheses. **Higher is better for Accuracy. Lower is better for runtime.**

5.1 Importance of Word Order

Table 1 illustrates the importance of incorporating word order to capture sentiment. All BOW variants, which disregard word order, achieve lower test accuracy relative to the LSTM models. It is noteworthy that Deep CBOW narrows the gap significantly, likely due to its highly non-linear and deep architecture that might implicitly capture sequential information.

5.2 Comparison of LSTM and TreeLSTM

TreeLSTM marginally outperforms the standard LSTM in test accuracy (cf. Table 1), showcasing the benefit of incorporating syntactic tree structures. However, this improvement comes with sig-

[1]The dataset is publicly available on their website.

nificantly higher runtime and greater variability in performance, making TreeLSTM less favorable in its current form. These limitations highlight the need for further enhancements, such as node-level supervision and regularization, to fully realize its potential.

5.3 Comparison of TreeLSTMs

As can be seen in Table 2, we do not obtain improved results with CTREELSTM. The gap in performance is statistically in-significant under a 95% confidence interval. However, using CTREELSTM comes at a cost of about 6.8% higher training time. The impact our the proposed regularization method is rather small, which is expected, given that the regularization term only affects 3.44% of the parameters in the model.

Model	Test Accuracy	Runtime (sec)
CTREELSTM	0.4703 (0.0111)	832.54 (14.32)
TreeLSTM	0.4753 (0.0063)	779.72 (13.21)

Table 2: **Comparison of TreeLSTM Variants:** Test accuracy and runtime for TreeLSTM and CTREELSTM on the SST dataset. Results are based on three runs with different model initializations. Standard deviations are in parentheses. **Higher is better for Accuracy. Lower is better for runtime.**

CAEs have shown to enhance reconstruction and downstream performance (Bertrand et al., 2024) and thus the amount of information that is retained in the embedding. Although this positive effect could not be observed in our work, a more exhaustive hyperparameter optimization could enable better outcomes with contractive regularization.

5.4 Data Augmentation

We observe a slight improvement in test accuracy (cf. Table 3) when training the TreeLSTM on a syntactic node-level. This way, the model is forced to generalize well also on a sub-sentence level, which we hypothesize fosters sentence understanding. By learning to classify the sentiment of all sub-parts of the sentence, additional understanding is used to inform the decision on a sentence level, ultimately leading to a slightly better performance.

5.5 Sensitivity to Sentence Length

As can be seen in Figure 1 in Appendix B, LSTM-based methods are mostly invariant to changes in sentence length, while the Bag-of-Words methods profit from a higher sentence length. We argue

Model	Test Accuracy	Runtime (sec)
TreeLSTM	0.4676 (0.0115)	778.20 (13.15)
SubTreeTreeLSTM	**0.4804 (0.0066)**	405.74 (29.14)

Table 3: **Comparison with SubTree Augmentation:** Test accuracy and runtime for TreeLSTM and SubTree-TreeLSTM on the SST dataset. The SubTreeTreeLSTM was trained with augmented data using all sub-trees of each sentence tree. Results are based on three runs with different model initializations. Standard deviations are in parentheses. **Higher is better for Accuracy. Lower is better for runtime.**

that while LSTMs can infer additional meaning from shorter sentences (e.g. "The movie is not good") by parsing semantic information, such as the "not" that negates a positive word, while the Bag-of-Words methods only sum up each word representation naively without taking relationships into account. For the aforementioned example, a Bag-of-Words model might predict a more neutral score as "not" and "good" cancel each other out, while a well-trained LSTM will confidently and correctly classify as negative. For longer sentences on the other hand, a Bag-of-Words method is somewhat effective as semantics become less crucial for sentiment understanding, especially when the sentence is composed of multiple sub-sentences with the same sentiment. This suggests that the performance gap between LSTMs and BOW-based methods might be smaller when it comes to document-level sentiment prediction instead of sentence-level prediction. Overall, the LSTMs still outperform the BOW-based models with a wide margin in our sentence-level experiments.

6 Conclusion

To conclude, our experiments highlight the importance of word order in sentiment classification, with LSTMs and TreeLSTMs outperforming BOW approaches. TreeLSTMs offer marginal accuracy gains over LSTMs due to hierarchical modeling but at a high computational cost. Node-level supervision enhances accuracy, while the proposed contractive regularization (CTREELSTM) has minimal impact. These findings align with prior research, though the limited effect of regularization and TreeLSTM scalability issues were unexpected. Future work should focus on improving TreeLSTM efficiency, exploring advanced regularization, and extending these models to document-level tasks to further test their generalizability.

References

Jan Henrik Bertrand, Jacopo Pio Gargano, Laurent Mombaerts, and Jonathan Taws. 2024. Autoencoder-based general purpose representation learning for customer embedding.

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

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé Iii. Deep unordered composition rivals syntactic methods for text classification. In *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)*, pages 1681–1691. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

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*, pages 10–19. Association for Computational Linguistics.

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

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. 2011. Contractive auto-encoders: explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 833–840, Madison, WI, USA. Omnipress.

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. *arXiv preprint arXiv:1503.00075*.

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

A Hyperparameters & Loss functions

We used the following hyperparameter configurations based on insights from initial experiments, intuition and based on recommendations from the notebook given for the assignment. Models are optimized based on the Cross Entropy Loss:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c})$$

Parameter	BOW	CBOW	Deep CBOW
Vocab. Size	18280	18280	18280
Embedding Dim.	5	300	300
Hidden Dim.	–	–	100
Layers	1 Emb	1 Emb, 1 Lin	1 Emb, 2 Lin
Activation Func.	–	–	Tanh
Optimizer	Adam	Adam	Adam
Learning Rate	0.0005	0.0005	0.0005
No. Iterations	60000	10000	10000
Batch Size	1	1	1

Table 4: Hyperparameters of BOW models

Parameter	LSTM	TreeLSTM	CTreeLSTM
Vocab. Size	18280	18280	18280
Input Dim.	300	300	300
Hidden Dim	168	150	150
Optimizer	Adam	Adam	Adam
Learning Rate	2e-4	2e-4	2e-4
No. Iterations	30000	30000	30000
Batch Size	25	25	25
Contract. Weight (λ)	–	–	0.003
Dropout (p)	0.5	0.5	0.5

Table 5: Hyperparameters of LSTM models

B Sentence length analysis

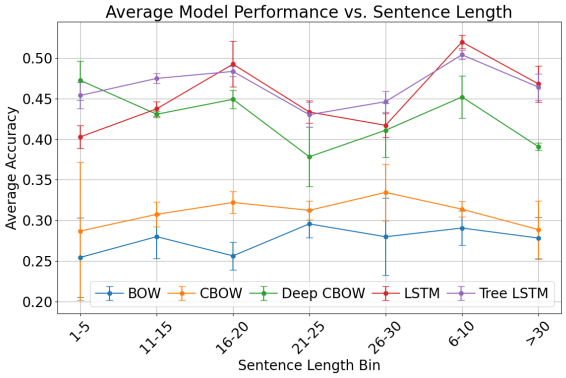

Figure 1: Comparison of all models with regards to sentence length on the SST dataset. Error bars show the standard deviation of the three training runs. **Higher is better.**