NLP 1 2023: Practical 2

Dylan Goode
University of Amsterdam / Address line 1
dylan.goode@student.uva.nl

Mina Janicijevic
University of Amsterdam / Address line 1
mina.janicijevic@student.uva.nl

1 Introduction

Recent NLP advancements have driven sentiment analysis, which seeks to understand emotional tones in text. This study explores various models for sentence-based sentiment classification. The models considered are a Bag-of-Words (BOW) model, a Continuous Bag-of-Words (CBOW) modeL, a Deep CBOW model, a Long Short-Term Memory (LSTM) models, and a Tree LSTM models (N-ary and ChildSum). For our investigation, we will be using the Stanford Sentiment Treebank (SST) dataset. Firstly, we investigate the significance of word order in sentiment analysis. Models like BOW disregard word order, potentially overlooking the syntactical nuances vital for sentiment interpretation. In contrast, models such as LSTM networks preserve word sequence, which we hypothesize to be advantageous for capturing sentiment accurately.

Secondly, we investigate if tree structures, compared to flat sequences, improve accuracy by capturing linguistic hierarchies (Tai et al., 2015). We expect Tree-LSTM models to excel in capturing sentence syntax, leading to better performance. The relationship between sentence length and model performance is also examined. We postulate that models like LSTMs would excel with longer sentences due to their capacity to remember long-range dependencies, whereas BOW models might falter.

Moreover, we explore the potential performance increase when supervising the sentiment at each node in the tree. By treating each node as a separate training example, we anticipate a richer learning experience for the models, thus enhancing performance (Socher et al., 2013). Lastly, we contrast N-ary Tree LSTM with Child-Sum Tree LSTM, suggesting that the former's structured approach better represents sentence composition than the latter, which aggregates child nodes indiscriminately.

In conducting these experiments, our study aims to contribute to NLP discussions, improving sentiment analysis models, especially in applications where capturing emotional subtleties is crucial Existing research acknowledges limitations in standard LSTM networks, especially their linear data processing approach, suggesting an incomplete understanding of word order in natural language (Mikolov et al., 2013). While Tree-Structured LSTM (Tree-LSTM) networks excel in handling tree-like structures, comprehensive comparative analyses with traditional LSTM models are lacking. Literature gaps persist in evaluating model performance across different sentence lengths, particularly for longer sentences. Exploring sentiment supervision at each tree node presents a valuable research opportunity. Additionally, detailed comparative analyses of N-ary and Child-Sum Tree LSTMs in sentiment analysis contexts are limited (Tai et al., 2015).

In the end, our experiments showed effectiveness of various models for sentence-based sentiment classification. LSTMs outperformed BOW models, highlighting the significance of word order. Tree-LSTM models excelled in capturing sentence syntax, especially in longer sentences. Node-level sentiment supervision had a limited impact on performance. Lastly, the structured approach of N-ary Tree LSTM showed promise in sentence composition representation. These findings contribute to the ongoing discourse in natural language processing, providing valuable insights for enhancing sentiment analysis models.

2 Background

2.1 BOW and CBOWs

The BOW model represents text as an unordered collection of words with their frequency of occurrence (Salton and Buckley, 1988). A document d is represented as a vector $\mathbf{v}_d \in \mathbb{R}^{|V|}$, where each element v_{di} corresponds to the count or the term frequency of word w_i from the vocabulary V. The CBOW model predicts a target word from a surrounding window of context words. The objective is to optimise the conditional probability $P(w_t|C_t)$, where C_t is the context of the target word w_t and t indexes the position in the text sequence. The model's parameters are learned by maximising the log likelihood: $L = \frac{1}{T}\sum_{t=1}^{T}\log P(w_t|C_t)$ where T is the length of the text sequence. The CBOW model thus captures word semantics by learning to predict words based on their contexts, leading to dense representations where semantically similar words have similar vectors.

2.2 Word Embeddings

Word embeddings (WE) transform discrete words into continuous vector spaces ($\mathbb{R}^n$), capturing semantic and syntactic nuances. Unlike traditional sparse representations like one-hot encoding, which fail to encapsulate word relationships, WE assigns each word w a dense vector $\mathbf{v}_w \in \mathbb{R}^n$. Techniques such as Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) learn these embeddings from large text corpora. They enable models to interpret words in context, significantly improving performance in tasks like text classification and sentiment analysis (Mikolov et al., 2013) & (Pennington et al., 2014).

2.3 LSTM

RNNs process variable-length sequences by applying a transition function recursively on a hidden state vector h_t. At each timestep t, h_t depends on the current input vector x_t and the previous hidden state $h_{(t-1)}$ (cite elman). RNNs are known to struggle with learning long-distance correlations due to the exploding or vanishing gradient problem during training (Hochreiter, 1998). LSTMs (Hochreiter and Schmidhuber, 1997), address these challenges by introducing a memory cell to preserve state over long periods, effectively addressing the issue of long-term dependencies. An LSTM unit at time t comprises vectors in $\mathbb{R}^d$: input gate i_t, forget gate f_t, output gate o_t, memory cell c_t, and hidden state h_t. The gates, whose values range from 0 to 1, manage the memory cell's state (Hochreiter and Schmidhuber, 1997). See the full equations here:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (2)$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (3)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \quad (4)$$

$$h_t = o_t \tanh(c_t) \quad (5)$$

Tree-LSTMs handle data with hierarchical tree structures, addressing a limitation of standard LSTMs which traditionally process data in linear sequences (Tai et al., 2015). Each node in a Tree-LSTM is associated with a hidden state h_t and a memory cell c_t, paralleling the hidden state in traditional LSTMs. Transition dynamics in a Tree-LSTM are adapted to incorporate information from multiple child nodes. Hence the Tree-LSTM cell is designed to handle tree-structured input data. This cell differs from the standard LSTM by accommodating inputs from both left and right child nodes in a binary tree, thus requiring a modified gating mechanism. See the full equations in the Appendix A.

3 Models

We discuss the implementation of models for the sentiment classification task.

3.1 BOW, CBOW, & Deep CBOW

BOW model represents each word in a sentence as a d-dimensional vector, where $d = 5$ for sentiment classes. Sentiment prediction is based on the argmax of the summed word vectors, resulting in a sentence vector $\mathrm{s} \in \mathbb{R}^d$. CBOW model extends BOW by using a higher-dimensional embedding layer ($E = 300$) and a linear transformation to project the sum of word vectors to sentiment classes. This model captures more nuanced word semantics but ignores word order. Deep CBOW adds multiple layers with $\tanh$ activations to CBOW for enhanced text representations and improved predictions.

3.2 LSTM

Our LSTM classifier includes an embedding layer, LSTM cells, and an output layer for sentiment labels. It processes sequential input timestep by timestep and uses dropout to mitigate overfitting.

3.3 Tree LSTM

We employ Binary Tree-LSTM models, specifically the N-ary Tree LSTM and Child-Sum Tree LSTM, tailored for SST's syntactic trees. These models incorporate syntactic structures in sentiment classification, utilizing SHIFT and REDUCE operations. Labels $\hat{y}_j$ are predicted at each node j using a softmax classifier on hidden state h_j. Initial states for words are derived from embeddings through linear transformations, generating initial hidden states h_t and cell states c_t. The N-ary Tree LSTM combines the states of left and right children, reflecting the hierarchical tree structure, while the Child-Sum Tree LSTM aggregates hidden states across child nodes.

4 Experiments

In the initial phase, we partitioned the SST dataset into training (8,544 instances), development (1,101 instances), and test (2,210 instances) sets, focusing on sentence-level sentiment classification at the root node of each tree. Token extraction from tree strings resulted in an 18,256-token vocabulary with unique numeric IDs.

4.1 Data Processing and Model Parameters

Training included preprocessing the SST dataset, tokenizing sentences, and transforming them into word ID sequences. BOW model used 300-dimensional word embeddings and an initialized zero bias vector. CBOW model added a linear layer for output class projection. The Deep CBOW model had multiple layers with 100 units and Tanh activations. Optimization used the Adam algorithm (learning rate: 0.001) with a consistent batch size. Hyperparameters, such as dropout rates, were fine-tuned based on preliminary experiments for model complexity and overfitting risk. Cross-entropy loss was applied, and accuracy was the primary metric, assessed periodically on a validation set to ensure generalization. The training process involved forward and backward passes, with the Adam optimizer guiding weight updates. Techniques like gradient clipping and learning rate adjustment were employed as needed. Experiments were conducted in a controlled setting with a specific GPU configuration and a consistent software environment, including PyTorch version. Reproducibility was ensured by setting a fixed random seed, and early stopping based on validation loss was implemented to prevent overfitting.

4.2 Evaluation

Accuracy was the primary metric, monitored during training for early stopping. Evaluation included input processing, sentence-to-word ID conversion, and sentiment label prediction. Accuracy measured prediction precision, and models with the highest validation accuracies underwent final evaluation across different data subsets.

5 Results & Analysis

The models were tested using three distinct seeds (see Appendix B for seeds and extended results). The resulting mean accuracies and standard deviations are shown in Table 1.

Model	Mean	Std
BOW	0.2526	0.0163
CBOW	0.3406	0.0132
Deep CBOW	0.3578	0.0104
Deep CBOW with PT	0.4201	0.0005
LSTM	0.4466	0.0057
LSTM Mini-batch	0.4569	0.0071
LSTM Mini-batch,fine-tuning	0.4259	0.0080
N-ary Tree LSTM	0.4670	0.0079
Subtree	0.4611	0.0007
ChildSum Tree LSTM	0.4615	0.0006

Table 1: Mean and Standard Deviation of Model Performances

5.1 Experiment 1

To address the research question, we compare BOW and LSTM models. BOW models don't consider word order, while LSTMs are designed for sequence recognition. Our results in Table 1 confirm that LSTMs outperform BOW models, aligning with our hypothesis.

5.2 Experiment 2

The investigation sought to evaluate tree structures in enhancing LSTM model accuracy. Results demonstrated a notable difference between the two models under consideration (see table 1 for the results. The N-ary Tree LSTM model exhibited slightly higher accuracy which aligns with the theoretical expectation that tree-structured models, due to their hierarchical data processing capabilities, would surpass standard LSTMs in tasks requiring complex relational data interpretation. The N-ary Tree LSTM's superior performance can be attributed to its enhanced ability to model hierarchical and nested data structures, potentially offering a more nuanced understanding of data relationships than a linear LSTM model.

5.3 Experiment 3

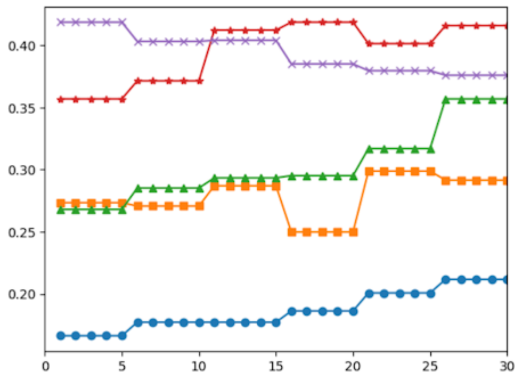

Figure 1: Performance accuracy of various language models across different sentence lengths. The x-axis denotes 'Sentence Length', and the y-axis 'Performance Accuracy'. Model markers: circles for BOW, squares for CBOW, upward triangles for Deep CBOW, stars for PT Deep CBOW, and crosses for LSTM.

Based on the plot provided, while the BOW model demonstrates improved performance with increased sentence lengths, it is still surpassed by other models. Our initial hypothesis posited that the LSTM network would exhibit superior performance, particularly for longer sentences, due to its ability to remember long-range dependencies. This is corroborated by the plot, which shows LSTMs outperforming BOW. Surprisingly, the Pre-Trained Deep CBOW (PT Deep CBOW) model exceeds the performance of the LSTM with extended sentence lengths, making it the top-performing model in this context. The likely reason for this could be the pre-training process, which equips PT Deep CBOW with a more nuanced understanding of language and context that proves advantageous in processing longer sentences.

5.4 Experiment 4

We investigated whether supervising sentiment at each node within a tree structure yields improved performance (see Table 1). Contrary to initial expectations, the results indicate that node-level supervision does not enhance performance. The N-ary Tree LSTM's slightly higher mean accuracy suggests superior efficacy in capturing comprehensive data relationships within the entire tree structure. The reduced performance of the Subtree approach may stem from the fragmentation of contextual information. When nodes are treated individually, the model may lose critical insights into the hierarchical dependencies and the overall context of the data, which are essential for a comprehensive understanding of the structure.

5.5 Experiment 5

The comparative study between N-ary Tree LSTM and Child-Sum Tree LSTM aimed to discern their respective efficiencies in sentence composition representation (see Table 1). The N-ary Tree LSTM's marginally higher mean accuracy suggests a slight superiority in handling structured sentence composition, as hypothesized. This outcome aligns with theoretical expectations, considering the N-ary Tree LSTM's design to distinctly process hierarchical sentence structures. These results, while confirming the hypothesis to a degree, also highlight the nuanced trade-offs between accuracy and consistency in different Tree LSTM architectures. The N-ary Tree LSTM's structured approach benefits sentence compositional representation but does not overwhelmingly outperform the Child-Sum Tree LSTM. This finding underlines the importance of considering both accuracy and consistency in model evaluation and selection, particularly in complex NLP tasks.

6 Conclusion

This study's evaluation of sentiment classification models using the SST dataset revealed the importance of word order and tree structures in sentiment analysis. Consistent with existing literature (Mikolov et al., 2013), LSTM models proved superior to BOW models, highlighting word order's significance. Tree-LSTM models, particularly the N-ary variant, outperformed standard LSTMs, confirming the advantage of tree structures in capturing complex linguistic hierarchies (Tai et al., 2015). Node-level sentiment supervision did not significantly improve performance, suggesting limitations in granular analysis at the node level. The marginal superiority of N-ary over Child-Sum Tree LSTMs indicates a balance in Tree LSTM architecture design. These findings, aligning with and expanding upon existing research, underscore the relevance of model architecture in sentiment analysis. Future work could explore hybrid models integrating the strengths of the examined architectures.

References

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 6(2):107–116.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

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

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing Management*, 24(5):513–523.

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. *CoRR*, abs/1503.00075.

A Example Appendix

A.1 Background Appendix

A.1.1 Tree-LSTM

Tree-LSTMs, a sophisticated extension of the conventional LSTM framework, are engineered to handle data with hierarchical tree structures, addressing a limitation of standard LSTMs which traditionally process data in linear sequences (Tai et al., 2015). Each node in a Tree-LSTM is associated with a hidden state h_t and a memory cell c_t, paralleling the hidden state in traditional LSTMs. Transition dynamics in a Tree-LSTM are adapted to incorporate information from multiple child nodes. Specifically, in this paper we are considering the N-ary Tree-LSTM variation, as introduced in (Tai et al., 2015) which incorporates input from multiple child nodes, each contributing uniquely to the parent's state. The equations for the model are as follows:

$$i_j = \sigma\left(W^{(i)}x_j + \sum_{\ell=1}^{N} U^{(i)}h_{j\ell} + b^{(i)}\right), \quad (6)$$

$$f_{jk} = \sigma\left(W^{(f)}x_j + \sum_{\ell=1}^{N} U^{(f)}_{k\ell}h_{j\ell} + b^{(f)}\right), \quad (7)$$

$$o_j = \sigma\left(W^{(o)}x_j + \sum_{\ell=1}^{N} U^{(o)}h_{j\ell} + b^{(o)}\right), \quad (8)$$

$$u_j = \tanh\left(W^{(u)}x_j + \sum_{\ell=1}^{N} U^{(u)}h_{j\ell} + b^{(u)}\right), \quad (9)$$

$$c_j = i_j \odot u_j + \sum_{\ell=1}^{N} f_{j\ell} \odot c_{j\ell}, \quad (10)$$

$$h_j = o_j \odot \tanh(c_j), \quad (11)$$

where x_j represents the input vector at node j, W and U denote weight matrices, b are bias vectors, σ is the sigmoid function, $\tanh$ is the hyperbolic tangent function, and $\odot$ denotes element-wise multiplication. The N-ary Tree-LSTM is noted for its 'off-diagonal' matrices $U^{(f)}_{k\ell}$, which provide nuanced control over the information propagated from child nodes.

We note that this model reduces to the conventional LSTM when the tree structure is a linear sequence, evidencing its adaptability and generality. For further details, readers can refer to the original paper introducing Tree-LSTMs (Tai et al., 2015).

B Model Performance

The seeds we used in our experiments (42, 1764, and 74088).

The following tables summarize the performance of various models used for sentiment classification on the SST dataset. The models are evaluated based on their accuracy at different training iterations for the training, development, and test datasets.

Table 2: Performance Metrics of Sentiment Classification Models

Model	**Iteration**	**Train Accuracy**	**Dev Accuracy**	**Test Accuracy**
BOW	30000	0.3221	0.2906	0.2742
	30000 (Best)	0.3161	0.2216	0.2348
	29000 (Best)	0.2983	0.2516	0.2489
CBOW	22000 (Best)	0.6728	0.3560	0.3389
	18000 (Best)	0.5721	0.3388	0.3253
	30000 (Best)	0.7533	0.3524	0.3575
Deep CBOW	24000 (Best)	0.5714	0.3751	0.3719
	27000 (Best)	0.6313	0.3760	0.3543
	26000 (Best)	0.4201	0.3551	0.3471
PT Deep CBOW	7000 (Best)	0.4388	0.4269	0.4208
	2000 (Best)	0.4251	0.4114	0.4199
	7000 (Best)	0.4356	0.4196	0.4195
LSTM	23000 (Best)	0.4655	0.4351	0.4543
	21000 (Best)	0.4837	0.4487	0.4448
	12000 (Best)	0.4621	0.4541	0.4407
Mini-Batched LSTM	6000 (Best)	0.5437	0.4605	0.4489
	5000 (Best)	0.5242	0.4623	0.4661
	6750 (Best)	0.5666	0.4650	0.4557
Mini-Batched LSTM with Fine Tuning	2000 (Best)	0.7894	0.4160	0.4353
	1000 (Best)	0.5531	0.4133	0.4267
	1000 (Best)	0.5259	0.4096	0.4158
Tree LSTM	5250 (Best)	0.5455	0.4469	0.4638
	2500 (Best)	0.4924	0.4487	0.4597
	4250 (Best)	0.5186	0.4469	0.4778
CTREE	4250 (Best)	0.5020	0.4569	0.4611
	3900 (Best)	0.5232	0.4598	0.4624
	3200 (Best)	0.5094	0.4583	0.4610
SubTree	2800 (Best)	0.4823	0.4497	0.4605
	3100 (Best)	0.4756	0.4502	0.4621
	3000 (Best)	0.4841	0.4509	0.4608