Bags of Words are (Not) All You Need (NLP1 Practical 2 Submission)

Matey Krastev
University of Amsterdam
matey.krastev@student.uva.nl
#15258130

Lisann Becker
University of Amsterdam
lisann.becker@student.uva.nl
#12558141

1 Introduction

A major problem within NLP is capturing complex attitudes conveyed through text – the study of sentiment analysis. Various approaches have been proposed to solve this task, from traditional statistically-motivated ones, such as the classic Bag-of-Words model, to established deep learning techniques, with recurrent neural networks (RNNs) – more specifically, Long-short Term Memory (LSTM) neural networks, as well as the emerged state-of-the-art Transformer architecture.

As each possesses inherent strengths and weaknesses, we set out to address the following research questions with regards to sentiment analysis:

1. Impact of pre-trained embeddings. How do embeddings pre-trained on large datasets affect classification accuracy in sentiment? Recent research highlights the benefits of using embeddings pre-trained on large corpora for downstream tasks such as sentiment classification (Aydoğan and Karci, 2020; Ding et al., 2022; Krouska et al., 2020). We suspect that utilizing and finetuning pre-trained embeddings will improve general model performance across all model categories.

2. Structural efficacy. Does syntactic structure help to get a better accuracy? We set out to compare traditional LSTMs to the Tree LSTMs proposed by (Tai et al., 2015), which aim to incorporate structural information within the LSTM architecture. We expect the Tree LSTM models to outperform, as they are designed to be more syntactically sensitive and do not only incorporate context from previous, but also upcoming words.

3. Word order relevance. How important is word order for this task? One major limitation of BoW models that LSTM architectures have overcome: understanding linguistic context by accounting for word order. We expect that word order understanding is the most pivotal achievement in sentiment prediction, as it disambiguates context and hence, improves understanding of the sentiment behind a phrase. We therefore expect models that account for word order to perform superiorly.

4. Impact of sentence length. How does performance depend on the sentence length? For BoW-inspired models, we expect sentence length to not affect the results significantly, whilst specifically for the LSTM models, designed to account for within-sentence context, we expect more pronounced effect, in line with prior research (Zhao et al., 2020; Wang et al., 2017).

5. Node-level supervision. Does performance improve if sentiment is supervised at each node in the syntactic tree and does introduction of additional data lead to increased performance? We specifically target Tree LSTMs, where we treat each tree node as individual tree, attempting to increase training data size and thereby model performance, and through doing that expect to see an increase in classification accuracy. (Tai et al., 2015; Sinha et al., 2018).

6. Comparison with Transformer architectures. Transformer architectures have shown to be promising in a myriad of NLP tasks. Various recent studies (Qasim et al., 2022; Rietzler et al., 2019) have demonstrated that such models trained on large corpora can be adapted for a large variety of downstream tasks through the use of transfer learning techniques. We aim to further analyze how those state-of-the-art models perform against the statistical and RNN-based models.

Our findings show the evolution of simple to deep CBoW models lead to slight increases in performance, exept for extremely long (50 words) sentences. LSTM models overall outperformed BoW models and Tree LSTMs achieved the highest accuracies we have obtained (around 80 percent) for moderately long 14 or 20 word sentences.

2 Background

2.1 Bag of Words

The classic BoW model represents sentences as the frequency of words, disregarding their order or syntactic structure and classifies sentiments based on word count.

The model M consists of an embedding layer of size d with output dimension equalling the number of classes $|C|$ and an additional learnable bias parameter b_0. No linear layer is applied, thus $d = |C|$.

2.2 Continuous Bag of Words (CBoW)

CBoW builds on the original BoW method, but utilizes an arbitrary embedding dimension d. The output of the BoW sum is passed to a linear layer $\mathbf{W} \in \mathbb{R}^{d \times |C|}$ and a bias term $\mathbf{b} \in \mathbb{R}^{|C|}$ outputting logits for the class probabilities.

CBoW employs continuous word embeddings of arbitrary dimensions, enabling finer-grained understanding of word meanings and context but maintain the additive nature of BoW and still struggle to capture intricate semantics.

2.3 Deep CBoW

In order to enable learning more complex relationships within text, authors like Socher et al. (2012) employ deep neural networks for a variety of text-related tasks. The Deep CBoW architecture integrates multiple linear layers and non-linear activation functions into the basic CBoW structure, enabling more complex feature extraction. This enhances interpretive capabilities but introduces the chance of overfitting. Embeddings pre-trained on large corpora can improve deep CBoW performance, as they provide an informed prior instead of random initialization, and training over a restricted corpus.

Generally, BoW models progressed from incorporating simplistic word count to more sophisticated, layered, non-linear representations capturing word meanings. Next, statistical models of language shifted to more recent sequential RNN models, and inspired LSTMs which capture long-term dependencies.

2.4 LSTM Neural Networks

Long Short-Term Memory Networks (LSTMs) significantly surpass BoW models' performance as they account for word order and thereby context by maintaining distinct pathways for both long- and short-term information. It leverages a complex system of gates to regulate information flow, thus enhancing its capability to process sequential data like sentences, and can further overcome the vanishing gradient problem of Recurrent Neural Networks (RNNs). The following describes the standard LSTM architecture (Hochreiter and Schmidhuber, 1997):

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
c_t &= f * c_{t-1} + i * g \\
h_t &= o_t * \tanh(c_t)
\end{aligned}
$$

where x_t is the token at time t and h_{t-1} refers to the hidden state at the previous time step.

2.5 Tree LSTM Neural Networks

An advancement over traditional LSTMs is the N-ary Tree LSTM proposed by Tai et al. (2015), which adapts LSTM architecture to accommodate tree-structured network topologies. The preserved structural information of the hierarchical tree enables TreeLSTMs to base word meanings on syntax, not just sequence, and take future tokens into account. Learning these additional relationships can reportedly lead to better performance.

3 Models

3.1 Bag of Words Variants

All BoW models employ the base architecture of using an embedding input layer and a sum over the input tokens. The continuous models are represented as a classic BoW model with arbitrary embedding dimension and a final softmax layer with learnable weights. The Deep CBoW models are an extension of the CBoW model, with several fully-connected (FC) layers and non-linear activation function layers between the sum of the embedding and the final prediction layer described for CBoW.

For the models not utilizing pre-trained embeddings, we specify embedding dimension as a hyperparameter and fix it at 300. We further test the effects of pre-trained embeddings.

The number of fully connected layers is also a hyperparameter, but we focus more on the inherent capabilities of a deep architecture, so we fix those at two FC layers of 100 units each.

3.2 LSTM

In our implementation, we employ the standard LSTM architecture (Hochreiter and Schmidhuber, 1997) described in section 2.4. We employ a single LSTM cell with $d = 150$ hidden units, a dropout layer ($p = 0.5$) to reduce overfitting, and a final layer for output weights.

3.3 Tree LSTM

In our benchmarks, we employ the described by the original authors TreeLSTM cell with a Binary Constituency Tree architecture. We investigate the performance of two techniques the original authors described - concatenation and sum of children hidden states, as well as how they interact together.

With concatenation, like traditional LSTM cells, the hidden states for each leaf are concatenated and hidden representations are calculated for the concatenated vector. With summation, the hidden states for each leaf are summed together.

We employ a dropout layer ($p = 0.5$) to again reduce overfitting. Finally, the output of the root node of the tree is multiplied by a weights layer to compute the final logits for classification.

3.4 Transformer

For our reference Transformer, we utilize the base version of RoBERTa (Liu et al., 2019) – an optimized BERT-variant. We use the provided pretrained RoBERTa base weights and tokenizers. In order to adapt the model for sequence classification, we implement a linear layer on top of the output of the hidden state of the encoder, a ReLU activation, and a dropout layer ($p = 0.32$) to reduce overfitting, with a final classification softmax head.

3.5 Cost Function

For all described models, we employ conventional multiple-class cross-entropy loss with logits.

4 Experimental Setup

4.1 Dataset

Throughout all of the stages, we employ the Stanford Sentiment Treebank (Socher et al., 2013), commonly known as SST, with its standard train, test, and validation splits. The SST comprises manually annotated sentences from movie reviews, structurally split into the nodes of a binary tree, where each level of the tree represents the actual sentiment of the child expression. For certain experiments, we tested additional data augmentation techniques by introducing subtrees to the training pipeline. The fine-grained classification task comprises predicting the sentiment of the sentence from a set of five labels: “very negative”, “negative”, “neutral”, “positive”, and “very positive”.

4.2 Training Specifics

All of the experiments are implemented with batch size 32, three different seeds, and across 30 epochs. For the BoW models we train with Adam optimizer and a learning rate of 0.003. For LSTM and TreeLSTM, we adopt a learning rate of 0.0003. Finally, for finetuning the RoBERTa model, we used a learning rate of 0.00003.

For the BoW methods word embeddings were learned from our training dataset’s vocabulary. For the DeepCBoW, as well as the LSTM models, we use pre-trained word embeddings. The GLoVe (Pennington et al., 2014) embeddings were selected in order to be as close as the results published by (Tai et al., 2015), but *Word2Vec* (Mikolov et al., 2013) embeddings were also considered and available as hyperparameters. The pre-trained embeddings can be further fine-tuned, and this comprises one of our experiments for the LSTM model. For RoBERTa we use the built-in tokenizer and its own trained embeddings that we finetune along with the entire model.

Hidden layer dimensionalities can be considered as hyperparameters, but in our experiments we have focused on the effectiveness of simple and relatively shallow models.

After each training epoch, the validation set loss and accuracy are calculated. The model with the highest validation set accuracy at the end of the training loop is saved and used on the test set.

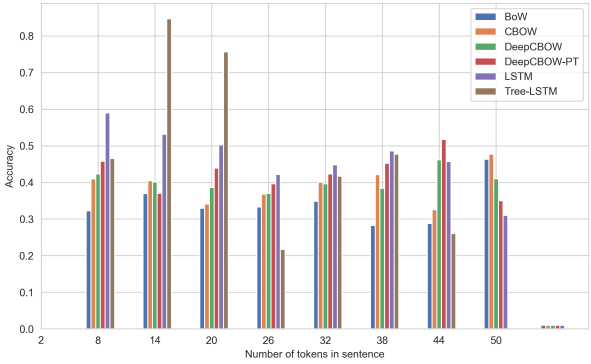

Figure 1: Model accuracy by sentence length.

Model	Accuracy	Parameters
BOW	33.6 (0.6)	0.08
CBOW	35.9 (1.3)	4.9
DeepCBOW	37.9 (1.4)	5.0
DeepCBOW-PT	42.4 (0.6)	5.1
LSTM	47.5 (0.3)	6.5
LSTM-FT	47.1 (0.8)	6.5
TreeLSTM CAT	47.0 (0.3)	6.8
TreeLSTM CAT (ST)	48.3 (0.2)	6.8
TreeLSTM SUM	46.6 (0.3)	6.5
TreeLSTM CAT (ST)	47.0 (0.3)	6.5
RoBERTa-base	55.3 (0.5)	124.6

Table 1: Test set accuracy on the SST dataset for the fine-grained 5-class sentiment classification task, along with total number of model parameters (in millions). Models trained on subtrees indicated with "(ST)".

5 Results and Analysis

The results indicate a clear improvement of neural network-based models compared to the simple statistical BoW approach. With each level of complexity, the performance of the models increased.

Addressing research question 1, pretrained embeddings likely affected the results the most (see Figure 2). All models which utilized the GloVe embeddings performed better overall – specifically the deep CBOW variant benefited the most. In fact, attempting to finetune the pretrained embeddings may have yielded the opposite result – a decrease in accuracy on unseen data and worse training, as exemplified by the results in Table 1 and Figure 2.

Answering research question 2, the constituency and dependency tree models did not demonstrate better performance than classic LSTMs. Further testing and hyperparameter tuning may be required, as the loss in TreeLSTMs seems to fluctuate wildly making the Tree model fail to learn more useful features compared to the LSTMs.

Our findings indicate that word order majorly impacts the quality of our predictions, answering research question 3. The BoW-variants are outperformed by the LSTM and Transformer architectures, which preserve some of the essence of the context within a given sequence. This result also answers research question 6 by finding that Transformer models were competitive state-of-the-art models for sentiment analysis.

Considering performance across varying sequence lengths in the context of research question 4, our findings in Figure 1 suggest that traditional BoW methods perform similarly and relatively robustly across all sequence lengths, with few outliers. In contrast, traditional LSTMs, performed consistently better than BoW models across samples from low to medium length, while struggling for longer sequences, suggesting that the model is prone to "forgetting" and thus misclassification. The Tree model in particular exemplifies that, as it has been able to perform significantly better on medium length sequences, but struggles for longer sequences. Not pictured is the RoBERTa model, but we surmise it should yield similar results as the BoW variants, as although it does by design consider word order, by its nature is able to attend to all parts of the sentence at one.

With regards to the Tree LSTMs, the models trained on the additional data provided by the subtrees, indicate improvement in accuracy – both for the concatenation, and sum of children cases – answering research question 5, likely due to the additional training data made available for the model.

On the other hand, summation expectedly did not provide a significant benefit in accuracy over concatenation. Very likely due to the fact that concatenation allows for more expressiveness in our model. This expressiveness allows the model to learn more complex relationships and therefore achieve higher performance, as shown here.

A major downside to the better performing models (LSTMs and BERT-based) is the required training and inference time. RoBERTa in particular, with more than 120 mil. parameters, restricts wider application of the model on low resource devices.

6 Conclusion

We investigated past and state-of-the-art approaches in fine-grained sentiment classification, along with their advantages and disadvantages. Modern deep neural network-based models have shown significant improvement over the more naive BoW-based models. The usage of pretrained embeddings can potentially lead to significant improvements over training embeddings from scratch on a restricted dataset. BoW and NN models can be combined for substantial improvements for minimal performance overhead, yet LSTMs and Transformers remain superior for sentiment prediction.

Further work and parameter-tuning may be required in investigating how neural tree models can be improved for sentiment analysis.

References

Murat Aydoğan and Ali Karci. 2020. Improving the accuracy using pre-trained word embeddings on deep neural networks for turkish text classification. *Physica A: Statistical Mechanics and its Applications*, 541:123288.

Zishuo Ding, Heng Li, Weiyi Shang, and Tse-Hsun Peter Chen. 2022. Can pre-trained code embeddings improve model performance? revisiting the use of code embeddings in software engineering tasks. *Empirical Software Engineering*, 27(3):63.

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

Akrivi Krouska, Christos Troussas, and Maria Virvou. 2020. Deep learning for twitter sentiment analysis: the effect of pre-trained word embedding. *Machine Learning Paradigms: Advances in Deep Learning-based Technological Applications*, pages 111–124.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Rukhma Qasim, Waqas Haider Bangyal, Mohammed A Alqarni, Abdulwahab Ali Almazroi, et al. 2022. A fine-tuned bert-based transfer learning approach for text classification. *Journal of healthcare engineering*, 2022.

Alexander Rietzler, Sebastian Stabinger, Paul Opitz, and Stefan Engl. 2019. Adapt or get left behind: Domain adaptation through bert language model finetuning for aspect-target sentiment classification. *arXiv preprint arXiv:1908.11860*.

Koustuv Sinha, Yue Dong, Jackie Chi Kit Cheung, and Derek Ruths. 2018. A hierarchical neural attention-based text classifier. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 817–823.

Richard Socher, Yoshua Bengio, and Christopher D Manning. 2012. Deep learning for nlp (without magic). In *Tutorial Abstracts of ACL 2012*, pages 5–5.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y 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.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. 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 1556–1566, Beijing, China. Association for Computational Linguistics.

Liangguo Wang, Jing Jiang, Hai Leong Chieu, Chen Hui Ong, Dandan Song, and Lejian Liao. 2017. Can syntax help? improving an lstm-based sentence compression model for new domains. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1385–1393.

Jingyu Zhao, Feiqing Huang, Jia Lv, Yanjie Duan, Zhen Qin, Guodong Li, and Guangjian Tian. 2020. Do rnn and lstm have long memory? In *International Conference on Machine Learning*, pages 11365–11375. PMLR.

A Training Hyperparameters

All models were run using three different seeds for reproducibility. The seeds were 0, 42, and 80084.

Model	$\|d\|$	h	θ
BOW	5	0	82,910
CBOW	300	0	4,944,500
DeepCBOW	300	[100, 100]	5,015,005
DeepCBOW-PT	300	[100, 100]	40,650
LSTM	300	150	270,910
LSTM-FT	300	150	6,490,055
TreeLSTM	300	150	6,534,905
RoBERTa-base	768	512	124,647,424

Table 2: Overview for each model: $|d|$ denotes the dimensionality of the word embeddings; h - the hidden dimensionality; and θ - the total number of *trainable* parameters.

Table 2 lists the hyperparameters used in the final models.

B Data Preprocessing

Whenever we are not working with pretrained embeddings, we build up our vocabulary based on the

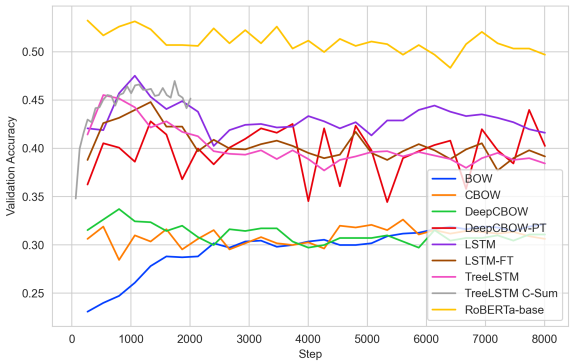

Figure 2: Validation set accuracy during training (batch size 32).

training set of SST. Special mappings are reserved for the <unk> and <pad> token, the former initialized to the sample Normal distribution wherever relevant, while the latter is represented as a vector of zeroes.

During batching, the data is padded out to the maximum sample length in the given batch. The padded values correspond to the "<pad>" token. Such values are not counted (they are zeroed-out) towards the calculation of the final result, nor are they trainable. Furthermore, any tokens not found in the training vocabulary are mapped to the "<unk>" token.

C Additional Questions

What is the size of the vocabulary? 18280.

What is the ID for "century"? 1973.

What are the first 10 words in the vocabulary (based on their IDs)? `<unk>, <pad>, ., <,>, the, and, a, of, to, 's`

What are the 10 most common tokens?

1. . ⇒ 8024
2. , ⇒ 7131
3. the ⇒ 6037
4. and ⇒ 4431
5. a ⇒ 4403
6. of ⇒ 4386
7. to ⇒ 2995
8. 's ⇒ 2544
9. is ⇒ 2536
10. that ⇒ 1915

How many hapax legomena are there in the dataset? 9543 (52.2%)

Sample of 20 words from the dataset: Scarpia comparing bag Bravo time-it-is embedded ick King overwhelmingly Theatre capture greatly Scarcely window limited delightful benevolent intensity welcomes not-very-funny

First four lines from the embeddings files:

Word2Vec (Google News 300D)

```
in 0.0703125 0.08691406 ... -0.0625
for -0.011779785 -0.04736328 ... 0.02416
that -0.01574707 -0.028320312 ... -0.1484
is 0.0070495605 -0.07324219 ... 0.1069
```

Source code

Available at: https://github.com/m-krastev/sst-pytorch/