Fine-Grained Sentiment Analysis: A Bag of Approaches

Giulio Starace
13010840 / University of Amsterdam
giulio.starace@student.uva.nl

Luuk Verheijen
11331704 / University of Amsterdam
luuk.verheijen@student.uva.nl

1 Introduction

In this work, we consider the process of sentiment analysis: extracting the positive or negative inclination expressed by the writer towards some object in a piece of text, in our case movie reviews. We investigate the performance of various algorithms with respect to a "fine-grained" analysis of sentiment: rather than classifying a review as positive or negative, we aim to classify on a scale from 1 (most negative) to 5 (most positive) as outlined by Pang and Lee (2005).

We make use of the Stanford Sentiment Treebank (SST) dataset (Socher et al., 2013), consisting of 11,855 sentences extracted from movie reviews, each augmented by a parse tree obtained from the Stanford Parser (Klein and Manning, 2003). Each sub-tree in the dataset is labeled with a fine-grained sentiment annotated by 3 human judges, allowing for the application of supervised sentiment analysis ML algorithms on the dataset.

Our work addresses a set of research questions. We approach our questions by comparing the performance of models that do and do not challenge the assumptions being investigated. We also perform further manipulations on the dataset, as explained in more detail in section 4.

We are curious to see whether word order is important for the task. Elman (1990) outlined how word order constrains "syntactic structure, selective restrictions, subcategorization, and discourse considerations", suggesting that it is likely for it to be impactful in most NLP tasks. Furthermore, most recent state-of-the-art (SoTA) results (Munikar et al., 2019; Brahma, 2018; Sun et al., 2020) on fine-grained classifications of SST make use use of models that are designed to exploit word order. However, Pham et al. (2021) recently found that word order is not necessarily always relevant, leaving the question open. We expected word order to have a non-negligible impact simply based on our intuition but ultimately found that the performance does not significantly change when enabling or disabling word-order exploitation, changing at most by 2 % points.

We also investigate whether the tree structure of the data helps. After all, It would be surprising if the additional information contained in the parse tree led to little improvement. We probe this question both by making use of models (presented in section 3.5) that can exploit tree-like data as well as by supervising these model at each node in the tree, vastly compounding the amount of available data. We expected these experiments to improve performance due to the greater use of the principle of compositionality (outlined in the next section). Our experiments did indeed show improved performance on the task. This was most apparent when supervising at each node, with a peak accuracy of 51.2%, 3.5% points higher than our next best model.

We also consider how performance depends on sentence length. Our reasoning is that longer sentences will have space for more contrasting statements (such as the reviewer liking one aspect, but disliking another), which could confuse the models. We also reason that longer sentences could lead to issues with vanishing gradients for some of our models (Hochreiter, 1998) and therefore generally expected performance to be inversely related to sentence length. We confirm this result across all models as can be seen in Fig. 2.

Finally, since we found performance to not vary vastly across some models, we were curious to see whether the trained models could distinguish themselves when relaxing the task to coarser sentiment labels. We found that the general trend remained the same, with increased accuracy across the board and some models improving more than others.

2 Background

Many traditional NLP approaches treat words as individual units: independent entries in a vocabulary with no notion of similarity. This treatment is simple and robust but can be limited. One alternative is considering the distributional hypothesis: the meaning of a word is defined by the way it is used and the context it is in (Harris, 1954). This leads to a vector-based representation of a word. We refer to these representations as “word embeddings” and many deep learning libraries such as PyTorch (Paszke et al., 2019) provide APIs for mapping indices (words) to relevant learnable vectors (embeddings).

Treating words in this manner allows for the application of the principle of compositionality (Mitchell and Lapata, 2010): the meaning of a phrase can be derived from the meaning of its parts. The SST takes inspiration precisely from this principle, providing a dataset with sentiment labels at each component (both at word and parse level) of a sentence.

How the embeddings are composed is a design choice: summation and multiplication are simple to implement but due to commutativity will not capture the order in which words appear, which may carry additional semantic information for sentiment analysis. To be able to capture word order, recurrent neural networks (RNNs) (Hopfield, 1982) can be used, in which cells have an additional output that is recursively used as input in successive steps, allowing the model to “remember” previous steps. One of the most widely used and successful RNN cell variants is the Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), which augments the RNN cell with specialized “gates” for helping the model discriminate which information to “forget” and “remember”.

The examples in the SST, however, are more than sequences of words (sentences), with each example consisting of a full parse tree complete with sentiment labels at each node. Vanilla LSTMs cannot make use of tree-structured data, which led to the work done by Le and Zuidema (2015); Zhu et al. (2015) and Tai et al. (2015) presenting TreeLSTMs, which as the name suggest generalize their inputs to tree-like data structures.

3 Models

The work made use of the following models, each of which will be provided with a brief overview in this section. When training, cross-entropy loss was utilized for every model. Each model had an O-dimensional vector as output, where O here is the number of classes to predict (5 in this sentiment classification task). The argmax of this vector was used as the predicted label for the given sentence.

3.1 Bag of Words (BOW)

This model takes the words of a given sentence and embeds them each into an O-dimensional vector. The embedded vectors are then summed and a bias term is added. The summation causes word order to be discarded, which is why we refer to this architecture as a bag of words.

3.2 Continuous Bag of Words (CBOW)

This is an extension of our BOW, with the difference being that the embedding now is of an arbitrary dimension, E. A linear layer is then added to map the embeddings to the required output dimensionality of O. We set $E = 300$, which we justify in the next section.

3.3 Deep CBoW (DeepCBOW)

Here we extend our CBOW to achieve non-linearity by adding an additional two linear hidden layers between the embedding and the final linear layer, each followed by a non-linear activation function, the hyperbolic tangent (tanh). The linear layers respectively map from E to D, from D to D and from D to O, where we set $D = 100$ to encourage some latent variable modeling.

We consider two variants, a “default” variant where the embeddings are randomly initialized and learned as the model trains and a “pre-trained” variant, where GloVe pre-trained embeddings (Pennington et al., 2014) are used instead, keeping these fixed throughout training. We refer to the pre-trained variant as “PTDeepCBOW”. GloVe embeddings have a dimensionality of 300, which dictated the choice of $E = 300$ throughout the work, so to ease comparison between models with and without pre-trained embeddings. All our following models make use of the GloVe embeddings in the same way as was just described.

3.4 Long Short-Term Memory (LSTM)

Instead of summing the embeddings, the LSTM classifier passes them iteratively to the LSTM cell from section 2, using a hidden size of $D = 168$. This allows the model to capture word order, allowing further analysis in this direction. The output of the LSTM cell is then mapped from D to O with a final linear layer, after applying dropout (Srivastava et al., 2014) at a rate of 0.5 to achieve some regularization.

3.5 TreeLSTM

To exploit the tree-like structure of the data-set, we implement two variants of the TreeLSTM cells introduced by Tai et al. (2015), namely the N-ary TreeLSTM and the Child-sum TreeLSTM (CSTreeLSTM). The cells are then used in a similar fashion as was done for the LSTM cell, with a TreeLSTM classifier mapping the output of the cells to O using a linear layer preceded by a dropout of 0.5. For both models, we set the hidden dimensionality $D = 150$. For the N-ary TreeLSTM, we also experimented with supervising the sentiment at each node in the tree, and we refer to this model as "SubTreeLSTM".

4 Experiments

4.1 Training

To obtain mean and standard deviation statistics, each model was trained for 10 runs with different random seeds, barring the TreeLSTM variants which were trained for 3 runs due to time constraints. For optimization, Adam (Kingma and Ba, 2017) was used, with mini-batches of size 25. The learning rates and varying training duration expressed in number of epochs are shown in Table 1. The standard number of epochs used was 50. However, the BOW and DeepCBOW models took more epochs to converge and were therefore run for more, whereas the CSTreeLSTM converged in less (like the LSTM and TreeLSTM) and was run for only 30 epochs. The SubTreeLSTM was run for 7 epochs but because of the augmented training set there were 37 more batches per epoch than the other models, so it can be seen as equivalent to 259 epochs for the other models. This augmented set was created by using each sub-tree from the original training set as its own data point with the respective node's label.

4.2 Evaluation

We evaluate a model by calculating the accuracy of its sentiment predictions for each sentence in the test set. We calculate the accuracy by dividing the number of correct predictions by the total. During evaluation, the number of correct predictions is also stored per length of the input sentence for later analysis. Our coarse evaluation function added a half to the number of correct predictions for each prediction that was only removed from the ground truth by one. To check for the relevance of word order we also evaluated the LSTM after shuffling the words in the sentences of the test set.

5 Results and Analysis

Table 1 contains the test accuracies of each of our models from section 3. We note that models that make use of word order (LSTM and lower rows) tend to perform better, with the worst of these (LSTM) achieving an accuracy of 46.4% vs PT-DeepCBoW's accuracy of 45.2%. We find that word-order accounts for at least 2 % points of the LSTM's performance, as shuffling the words causes accuracy to drop by this amount. We notice that the CSTreeLSTM is only slightly better than the N-ary TreeLSTM, with differences in test accuracy that are arguably negligible. We also note that

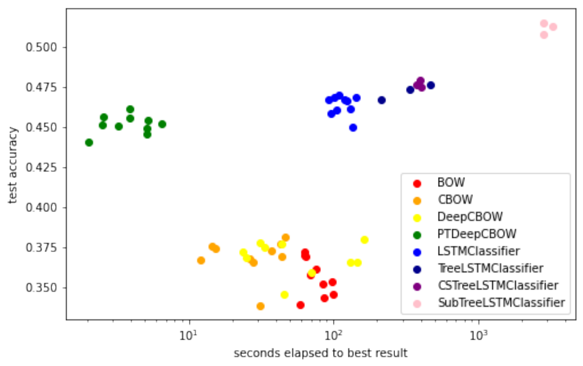

Figure 1: Test accuracy over time elapsed to best result. The x-axis is logarithmically scaled to aid with visualization.

tree structure indeed helps with achieving higher accuracy, although again this effect is only marginal. In fact, our TreeLSTM implementations score only a little over than 1 % points higher than our LSTM. With out TreeLSTM architecture, we also experimented with supervising the sentiment at each node. Here the resulting improvement is much more pronounced, with the SubTreeLSTM model scoring 3.5 % points more than the next best model.

This increase in performance is most likely due simply to the model being exposed to much more data and has little to do with sophistication in the architecture. Fig. 1 underlines this. As can be noted, while SubTreeLSTM achieved a higher accuracy, this took much longer than other models due to the amount of data that the model had to be exposed to.

Model	Test Set Acc. (Std.)	N. of Runs	N. of Epochs	learning rate
BOW	0.36 (0.01)	10	300	5e-4
CBOW	0.37 (0.01)	10	50	5e-4
DeepCBOW	0.368 (0.010)	10	150	5e-4
PTDeepCBOW	0.452 (0.005)	10	50	5e-4
LSTM	0.464 (0.006)	10	50	3e-4
LSTM (Permuted Sentences)	0.44 (0.01)	10	50	3e-4
TreeLSTM	0.472 (0.004)	3	50	2e-4
CSTreeLSTM	0.477 (0.002)	3	30	2e-4
SubTreeLSTM	0.512 (0.003)	3	7	2e-4

Table 1: Test set accuracies on the Stanford Sentiment Treebank. We report mean accuracy and standard deviation over a given number of runs. The number of epochs and learning rate used for training are also reported.

Interestingly we notice the remarkable efficiency of the PTDeepCBOW model, which performs almost identically to the LSTM models while taking order of magnitudes less time to reach this performance.

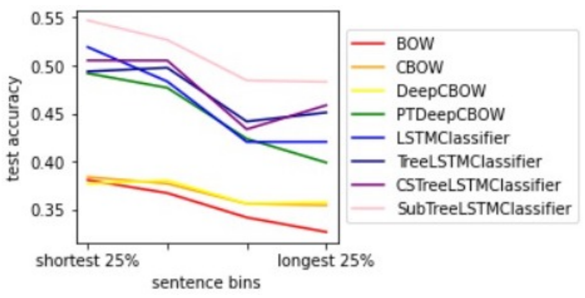

Figure 2: Test accuracy over quartiles of the dataset ordered by sentence length.

Fig. 2 plots our model test accuracies over quartiles of the dataset ordered by sentence length. Throughout we note a slight negative trend, indicating that longer sentences generally impact performance negatively, supporting our hypothesis outlined in section 1. Finally, our experiment with

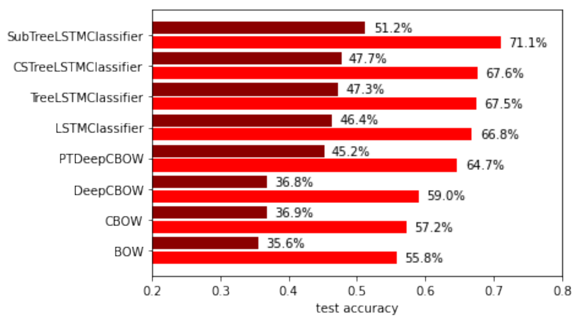

Figure 3: Model test accuracy when considering fine-grained (dark) and coarsened (light) evaluation.

coarsening the evaluation (with results reported in Fig. 3) showed that most models improve similarly when allowing for some leeway. One slight exception is the DeepCBOW, which performs worse than the CBOW with fine-grained evaluation, but performs better by a significant margin with coarse evaluation. This means that DeepCBOW does not get the exact correct label as much as CBOW, but is generally closer to the ground truth. This is arguably more important, as we hypothesize that humans are also likely to be off the ground truth by one.

6 Conclusion

In conclusion, we find that word-order is of marginal importance at best when performing fine-grained sentiment analysis. We also note that sentence length impacts performance negatively, with longer sentences causing more trouble across all models. The marginality of the word-order results is echoed by other experiments, such as when incorporating tree-structured data. The most pronounced improvements in accuracy are in fact only achieved when training on vastly greater amounts of data by supervising the sentiment at each node when using TreeLSTMs. This led to a brief investigation in model efficiency as captured in Fig. 1 which underlined the incredible efficiency of the PTDeepCBOW model compared to the rest of the models. Future work may therefore find it interesting to investigate further in the capabilities of simple models provided with pre-trained embeddings, such as for instance replacing static embeddings with contextualized language models such as BERT (Devlin et al., 2019) or GPT (Radford et al., 2018) or providing PTDeepCBOW with larger amounts of data, potentially by utilizing parts of sentences with their respective sub-tree labels.

References

Siddhartha Brahma. 2018. Improved Sentence Modeling using Suffix Bidirectional LSTM. *arXiv:1805.07340 [cs, stat]*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*.

Jeffrey L. Elman. 1990. Finding Structure in Time. *Cognitive Science*, 14(2):179–211.

Zellig S. Harris. 1954. Distributional Structure. *WORD*, 10(2-3):146–162.

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(2):107–116.

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

J. J. Hopfield. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan. Association for Computational Linguistics.

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.

Jeff Mitchell and Mirella Lapata. 2010. Composition in Distributional Models of Semantics. *Cognitive Science*, 34(8):1388–1429.

Manish Munikar, Sushil Shakya, and Aakash Shrestha. 2019. Fine-grained Sentiment Classification using BERT. *arXiv:1910.03474 [cs, stat]*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *arXiv:cs/0506075*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

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, Doha, Qatar. Association for Computational Linguistics.

Thang M. Pham, Trung Bui, Long Mai, and Anh Nguyen. 2021. Out of Order: How Important Is The Sequential Order of Words in a Sentence in Natural Language Understanding Tasks? *arXiv:2012.15180 [cs]*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training. *OpenAI Blog*.

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.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Zijun Sun, Chun Fan, Qinghong Han, Xiaofei Sun, Yuxian Meng, Fei Wu, and Jiwei Li. 2020. Self-Explaining Structures Improve NLP Models. *arXiv:2012.01786 [cs]*.

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.

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*, pages 1604–1612. PMLR.