Representing Sentences with Neural Models

Maarten Burger
University of Amsterdam

Kaya ter Burg
University of Amsterdam

1 Introduction

The task of sentiment analysis concerns itself with the automatic determination of the sentiment of a text. This can take the form of a classification task, either binary or multi-class. Thus, the main goal of sentiment analysis is to correctly predict the sentiment of a given text. In order to accomplish this, many subtleties need to be solved. We will address several of these in this report:

- Does taking word order into account improve the performance?
- Does tree structure improve the performance?
- How does sentence length affect the performance?
- Does supervising the label at each sub-tree improve the performance?
- Does performing lemmatization on the words improve performance?

The answers to these smaller questions would each help identify what factors can increase performance and can lead to the development a better sentiment classification model.

In order to answer the posed questions, we will implement several types of models, which can be roughly divided into two groups. The first group consists of models making use of the Bag-of-Words (BOW) (Harris, 1954) approach: a neural BOW model (Toutanova and Wu, 2014; Iyyer et al., 2015), a continuous BOW model (CBOW) and a deep continuous BOW model (Deep CBOW). The second group consists of Long Short-Term Memory (LSTM) models: a vanilla LSTM (Hochreiter and Schmidhuber, 1997; Gers et al., 2000) and an N-ary Tree-LSTM (Tai et al., 2015). We will compare the performance of an LSTM trained on the dataset with the words shuffled to the performance of an LSTM with the original dataset to see if word order affects performance. In a previous work it has been shown that accounting for word order in a CNN does show an improvement in performance for sentiment classification tasks (Johnson and Zhang, 2014). We analyze whether a tree structure alters the performance by comparing the performance of the vanilla LSTM with that of the Tree-LSTM. It has been shown in an earlier work that exploiting the inherent tree-like structure found in language by using a Tree-LSTM does improve performance for sentiment classification tasks (Tai et al., 2015). We test if sentence length has any effect by comparing the performances of the models on different sentence lengths. In (Giménez et al., 2020) it has been demonstrated that padding sentence length for sentences does improve performance in a sentiment classification task when using a CNN, which could indicate an increase in performance for sentences with higher sentence lengths. We utilize the structure of the Tree-LSTM to test if performance increases when we train on the labels of each sub-tree of the sentence parse trees. Previous work has shown that accounting for labels of sub-trees does increase performance for sentiment analysis tasks (Nakagawa et al., 2010). Another work has illustrated that utilizing supervised learning for the sub-tree labels increases performance for sentiment analysis tasks (Socher et al., 2013). Finally, we perform lemmatization on the words in the dataset and test if this leads to an increase in performance. In other works lemmatization of a dataset has increased performance (Bao et al., 2014; Symeonidis et al., 2018), but it should be noted that these works used a noisy dataset, whereas we did not.

As previous works have already answered similar questions to ours, we set our expectations to be in line with this literature.

The dataset we use for all our experiments is the Stanford Sentiment Treebank (SST) (Socher et al., 2013). This dataset contains sentences and their binary parse trees, where each node in the tree is annotated with its own sentiment score (ranged from 0 to 5). This allows for very fine-grained multi-class classification.

Our findings through the various experiments show that considering word order improves the performance, as does using the tree structures of the sentences. Supervising the label at each subtree does not make a difference in performance, but it does lead to more overfitting on the train data. The effect on sentence length differs among models. Lemmatization of the words in the data does not lead to a substantial difference in performance.

2 Background

To build our models we have made use of a plethora of techniques, in this section we will give a brief overview.

Bag-of-Words refers to the technique of representing a text as a multiset of its words (Harris, 1954). As such, a BOW model does not account for word order. We make use of a neural Bag-of-Words model, which learns word vectors specialised for the classification task (Toutanova and Wu, 2014; Iyyer et al., 2015). The size of the word vector is equal to the amount of classes.

Despite sharing a name, the Continuous Bag-of-Words (CBOW) model we use is different from the standard CBOW model reported in (Mikolov et al., 2013). The one we will use is very similar to the previously described NBOW model, but accepts any embedding size.

The Deep Continuous Bag-of-Words (Deep-CBOW) model we use is an extension of the CBOW model where we add more linear layers with activation functions in between for more depth and non-linearity. We opted for one of the standard functions: the hyperbolic tangent (Sharma et al., 2017).

Word embedding refers to techniques used to map words to a mathematical representation, typically in the form of a vector of real numbers. For this report we used the Word2Vec algorithm (Mikolov et al., 2013) for word embeddings. We used the available pre-trained Word2Vec 300D Google News vectors which were truncated to only contain words in our dataset.

Long-Short-Term Memory units or LSTM units are typically found within a recurrent neural network architecture. The LSTM unit we used, follows the structure proposed by (Gers et al., 2000). This LSTM unit consists of a cell, an input gate, an output gate and a forget gate. Through these gates and the updating of the cell state the LSTM unit can control the flow of information over an arbitrary amount of time-steps. The unique aspect of such an LSTM unit is its ability to selectively remember previous cell states and what information is passed through.

N-ary Tree Long Short Term Memory networks or Tree-LSTMs are LSTM networks that are specialized for tree-structured network topologies (Tai et al., 2015; Le and Zuidema, 2015; Zhu et al., 2015). Standard LSTM structures follow a linear structure, whereas Tree-LSTM structures follow a tree structure. Tree-LSTM networks work well on tree-structured data, exploiting the structure to gain more relevant information and more accurate results.

Lemmatization refers to the process of turning inflected forms of words to their base form, allowing them to be more easily grouped together and understood by algorithms and reducing the vocabulary size. We made use of the WordNet lemmatizer for this work (Miller, 1995; Fellbaum, 1998).

3 Models

The standard BOW model consists of an embedding layer with a fixed size (the amount of classes) and a sum layer, where the embeddings for all words in the sentence are summed plus a bias value. The CBOW model consists of an embedding layer with an arbitrary size, a sum layer (without bias) and a final linear layer that projects the summed embedding back to the labels. The Deep CBOW also consists of an embedding of arbitrary size, then three linear layers with Tanh activation in between, where the last linear layer projects to the labels, and finally a sum layer. The pre-trained Deep CBOW (PT Deep CBOW) model follows the same architecture, but for this model we use the Word2Vec embeddings instead of letting the model learn its own embeddings. This means we also do not adjust the embedding layer during training for this model.

Both LSTM models consist of the same architecture: an embedding layer to embed the character it receives, followed by an LSTM cell (either a standard LSTM cell or a Tree-LSTM cell), a dropout layer (Srivastava et al., 2014) to reduce overfitting and finally a linear layer for the classification. The embedding layer is not trained, but rather the pre-trained Word2Vec embeddings are used.

For the LSTM models, we feed the sentences word by word or node by node to the model. The embedding layer converts them to an appropriate representation. The LSTM cell then makes a representation for the entire sentence, which is, after performing dropout, fed to a final linear layer, which maps this representation to the five sentiment labels.

All models are trained using cross-entropy loss, which is appropriate for multi-class classification tasks. For the optimisation of the model parameters we always use the Adam optimiser (Kingma and Ba, 2014).

4 Experiments

Model type	Iterations	Learning rate	Hidden dim
BOW	100,000	0.0005	-
CBOW	40,000	0.0005	-
Deep CBOW	35,000	0.0005	100
PT Deep CBOW	40,000	0.0005	100
LSTM	10,000	0.0003	168
Tree-LSTM	5,000	0.0002	150

Table 1: The hyperparameter settings for all the models we trained. "Hidden dim" refers to the size of the hidden layer.

All models and experiments were implemented using the PyTorch library (Paszke et al., 2019) and all evaluations were done using accuracy, i.e. the fraction of correctly classified examples. Each model was trained until the validation set accuracy had converged to a somewhat stable value (i.e. not much improvement between subsequent iterations). The hyperparameter settings for all models can be found in Table 1. Furthermore, the BOW model used an embedding size of 5, all the other models used an embedding size 300. The LSTM and Tree-LSTM models both used a batch size of 25 and a dropout value of 0.5.

For all experiments, except for the sub-tree one, we only used the label of the complete sentence (the sub-tree experiment uses the labels at each sub-tree). The train set contains 8544 sentences, the validation set 1101 sentences and the test set 2210 sentences.

To test whether word order has an effect on the performance, we compared the accuracy scores of an LSTM using the original dataset with an LSTM trained and evaluated on the dataset where the words in each sentence are shuffled. In order to verify if accounting for the tree structure in the sentences improves the performance, we compared the accuracy score of the vanilla LSTM with that of the Tree-LSTM. To see if sentence length affects performance, we split the test set on sentence length per ten (i.e. length 0 - 10, 11 - 20, etc.) and evaluated all models on these subsets. Furthermore, we compared the performance of the Tree-LSTM when only supervising the overall label of the sentence and when supervising the label at each sub-tree by extending the train set to contain all sub-trees. This increases the train set size to 318,582 samples. For the final experiment, we lemmatized each word in the dataset using the WordNet lemmatizer as implemented in the Natural Language Toolkit (NLTK) (Loper and Bird, 2002) and trained all models on this lemmatized dataset.

Model type	Validation	Test
BOW	0.308 ($\pm$0.014)	0.308 ($\pm$0.019)
CBOW	0.359 ($\pm$0.012)	0.362 ($\pm$0.019)
Deep CBOW	0.375 ($\pm$0.002)	0.368 ($\pm$0.010)
PT Deep CBOW	0.453 ($\pm$0.008)	0.442 ($\pm$0.003)
LSTM	**0.467** ($\pm$0.002)	0.460 ($\pm$0.006)
Tree-LSTM	0.450 ($\pm$0.005)	**0.467** ($\pm$0.002)

Table 2: The accuracy scores obtained from training the different models on the validation and test set. The reported scores are the average of three separate runs with different seeds. The standard deviation is given between parentheses.

5 Results and Analysis

In Table 2 the accuracy scores for the different models and tasks are summarised. All reported accuracy scores are the average of three separate runs with different seeds. The validation accuracy and train loss curves obtained during training in one of the runs can be found in Appendix A.

We can see that the more complex the BOW model becomes, the better it performs on both the validation and test set. The best BOW model is the Deep CBOW model with pre-trained Word2Vec word embeddings. The Deep CBOW is the most complex BOW model we introduced, as it has multiple hidden layers and can approximate non-linearities through the use of non-linear activation function between layers. Furthermore, the usage of pre-trained embeddings vastly improves the performance. Then the model does not need to learn its own embeddings on a relatively small dataset and can use tried and tested embeddings that work for a variety of NLP tasks. Moreover, it becomes clear that the LSTM models are better at classifying the sentiment of a sentence than all of the BOW models.

The LSTM trained on the original dataset achieves a validation accuracy of $0.467(\pm 0.002)$ and a test accuracy of $0.460(\pm 0.006)$, whereas the LSTM with a shuffled dataset only achieves a validation accuracy of $0.435(\pm 0.008)$ and a test ac-

Model type	≤10	≤20	≤30	≤40	>40
BOW	0.320	0.318	0.296	0.303	0.323
CBOW	0.395	0.361	0.332	0.312	0.363
Deep CBOW	0.379	0.351	0.363	0.353	0.303
PT Deep CBOW	0.504	0.458	0.403	0.379	0.434
LSTM	0.502	0.476	0.433	0.422	0.394
Tree-LSTM	0.480	0.494	0.419	0.4533	0.465

Table 3: Average accuracy of each model per sentence length. The columns denote the groups in which the sentences where divided according to their length and on those subsets the models were evaluated.

Model type	Validation	Test
BOW	0.314 (±0.003)	0.314 (±0.013)
CBOW	0.351 (±0.009)	0.369 (±0.007)
Deep CBOW	0.367 (±0.005)	0.366 (±0.013)
PT Deep CBOW	0.448 (±0.007)	0.446 (±0.014)
LSTM	0.462 (±0.002)	0.451 (±0.008)
Tree-LSTM	0.456 (±0.006)	0.463 (±0.007)

Table 4: The average accuracy scores obtained on the lemmatized validation and test set for each model trained on the lemmatized train set.

curacy of $0.447(\pm 0.009)$, both of which are lower. From this we can infer that word order does have an effect on the performance: taking it into account improves the accuracy score.

Furthermore, it ought to be noticed that the Tree-LSTM model does not obtain a higher validation accuracy and it only has a slightly higher test accuracy score than the vanilla LSTM model. We thus reason that tree structures do not improve the performance, since a Tree-LSTM model uses the tree topology, whereas a standard LSTM model does not. Since there is barely an increase in performance, it does not seem the case that utilizing tree topology helps the model much.

In Table 3 the accuracy scores for different sentence lengths are reported. It should be noted that the distribution of sentence lengths in the dataset is not balanced and this can thus affect the performance. The distribution sentence length of the train, validation and test set can be found in Appendix B. From Table 3 we can see that sentence length has a different effect on each of the models. The BOW, Deep CBOW and vanilla LSTM models perform worse the longer the sentence becomes. The other models mostly follow the same trend, but have a peak at sentences longer than 40 words. We can thus not make a definite conclusion based on these results, but it seems like for most models, longer sentences decrease the performance. This could be due to the difficulty of modelling long-term dependencies. Perhaps a more balanced dataset with regard to sentence length would yield more concise results. The full table with standard deviations can be found in Appendix C.

We train a new Tree-LSTM model on the train set extended with each sub-tree and the corresponding label. This model is trained for 20,000 iterations. The validation accuracy and train loss curves can be found in Appendix A.8. This model obtains an average accuracy score of 0.464 (± 0.002) on the validation set and an average accuracy score of 0.459 (± 0.010) on the test set. Compared to the Tree-LSTM model trained only on the whole trees, this model performs slightly better on the validation set, but slightly worse on the test set. It should also be remarked that the subtree Tree-LSTM model suffers from overfitting, since the average training accuracy is 0.795 whereas the average train set accuracy of the standard Tree-LSTM is 0.4970. From this we infer that the subtree Tree-LSTM generalizes less well and that the Tree-LSTM trained only on complete trees should be preferred in this case.

The results on the lemmatized dataset can be found in Table 4. This shows that the results between the original and lemmatized dataset are very similar. Thus, lemmatization seems to have little effect on the performance across all models.

6 Conclusion

After performing the experiments, we can conclude that including word order improves the accuracy. Including the tree structure further improves the performance. The effect of sentence length is different across model types and remains somewhat unclear. Supervising each sub-tree label only leads to overfitting. Lastly, lemmatization of the data seems to have little to no effect.

Most results are in line with our expectations and previous works. Only supervising the sub-tree label leading to overfitting without improving overall accuracy was a surprising result, as previous work has shown an overall increase in accuracy when supervising sub-tree label.

The experiments regarding sentence length and sub-tree supervision could use more attention. Our results do not give a clear and complete picture as to their effect on the sentiment analysis task. The Tree-LSTM model using sub-tree labels suffered from overfitting, but this effect could be alleviated by more hyperparameter tuning or the introduction of more regularization or a bigger dataset. Also, the use of a more balanced dataset w.r.t. sentence length could yield a fairer comparison than what we have done here.

References

Yanwei Bao, Changqin Quan, Lijuan Wang, and Fuji Ren. 2014. The role of pre-processing in twitter sentiment analysis. In *International conference on intelligent computing*, pages 615–624. Springer.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to Forget: Continual Prediction with LSTM. *Neural Computation*, 12(10):2451–2471.

Maite Giménez, Javier Palanca, and Vicent Botti. 2020. Semantic-based padding in convolutional neural networks for improving the performance in natural language processing. a case of study in sentiment analysis. *Neurocomputing*, 378:315–323.

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

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. 2015. 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.

Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. *CoRR*, cs.CL/0205028.

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

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using crfs with hidden variables. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 786–794.

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.

Sagar Sharma, Simone Sharma, and Anidhya Athaiya. 2017. Activation functions in neural networks. *towards data science*, 6(12):310–316.

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.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Symeon Symeonidis, Dimitrios Effrosynidis, and Avi Arampatzis. 2018. A comparative evaluation of preprocessing techniques and their interactions for twitter sentiment analysis. *Expert Systems with Applications*, 110:298–310.

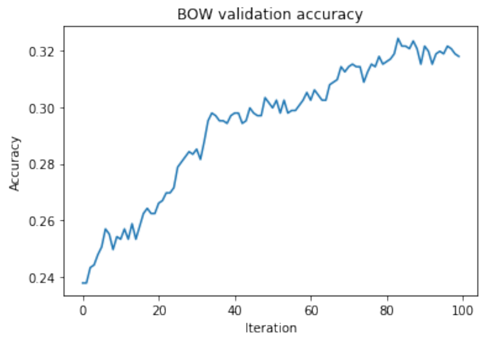
Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *ACL*.

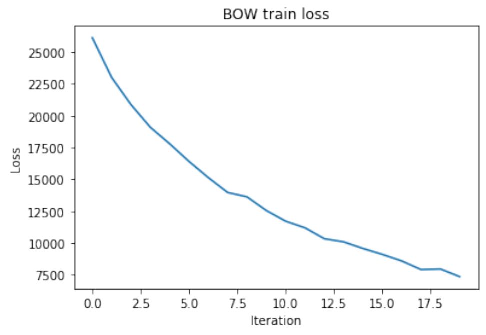
Kristina Toutanova and Hua Wu. 2014. Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 1: Long papers). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *International Conference on Machine Learning*, pages 1604–1612. PMLR.

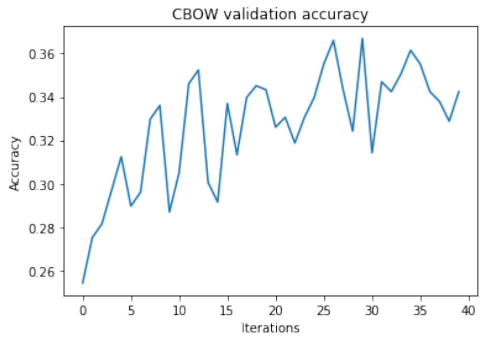
A Accuracy and loss curves

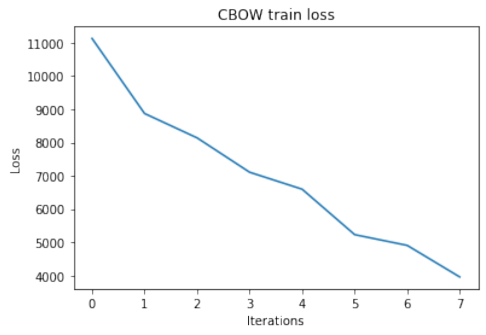
A.1 Bag-of-Words model



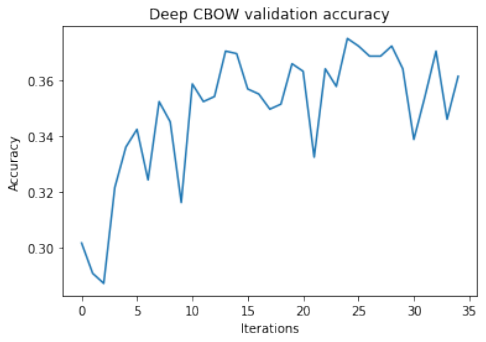


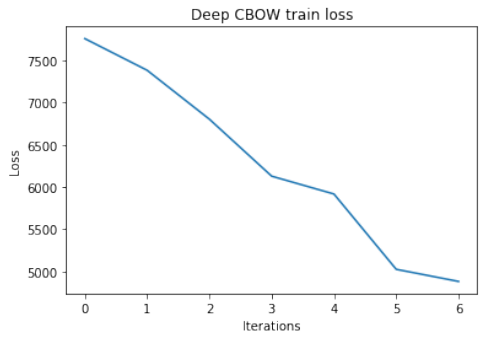
A.2 Continuous Bag of Words model



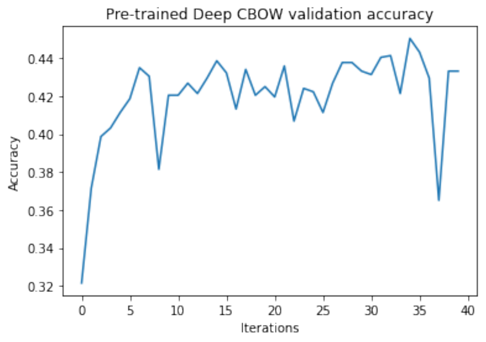


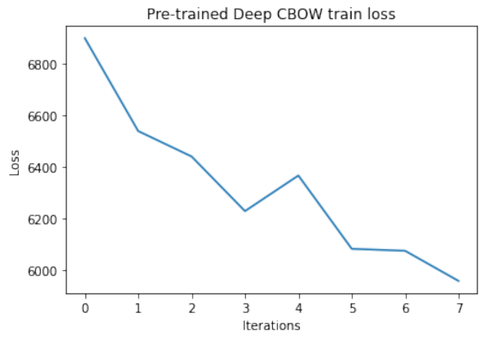
A.3 Deep Continuous Bag-of-Words model



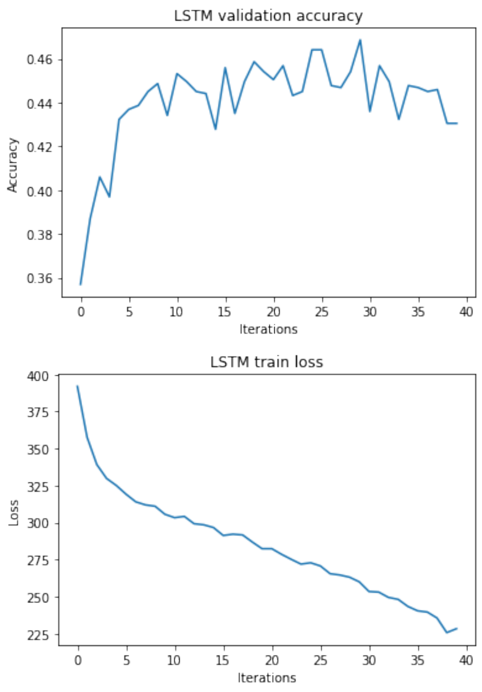


A.4 Pre-Trained Deep Continuous Bag-of-Words model

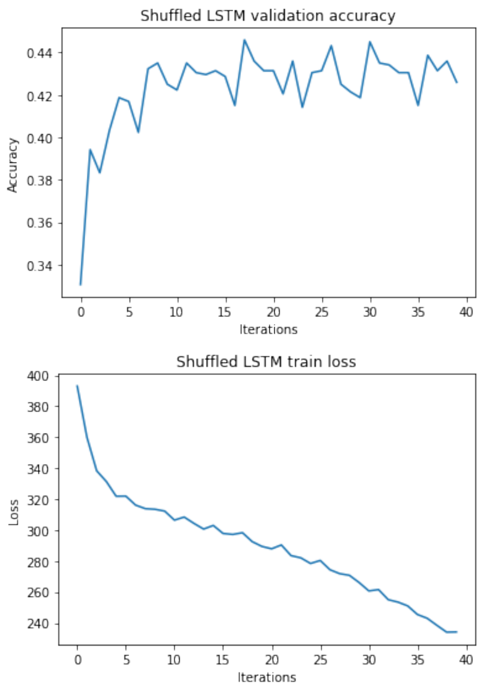




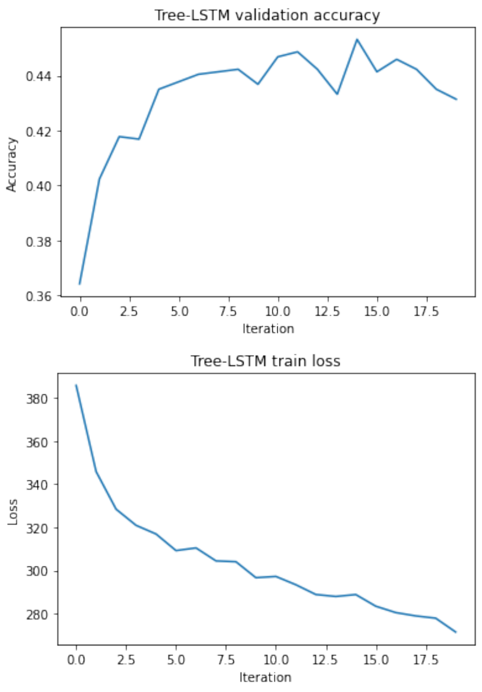
A.5 LSTM model



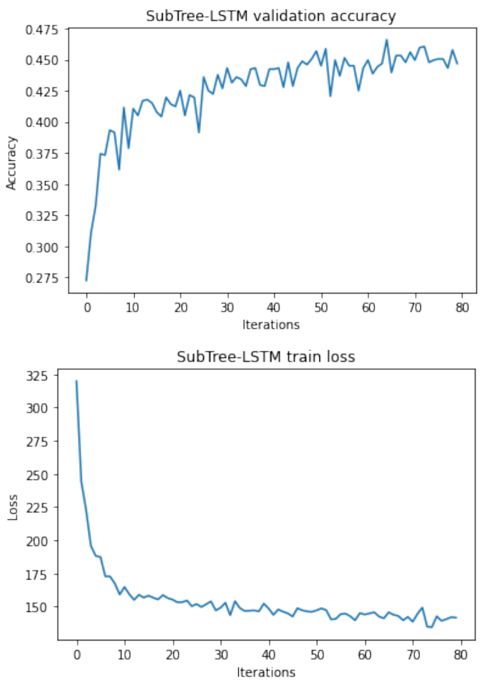
A.6 Shuffled LSTM model



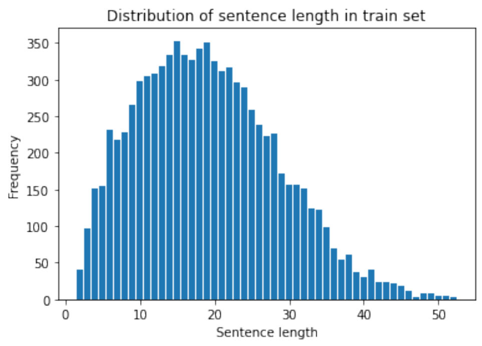
A.7 Tree-LSTM model

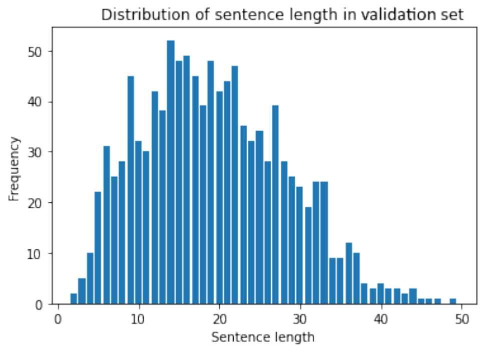


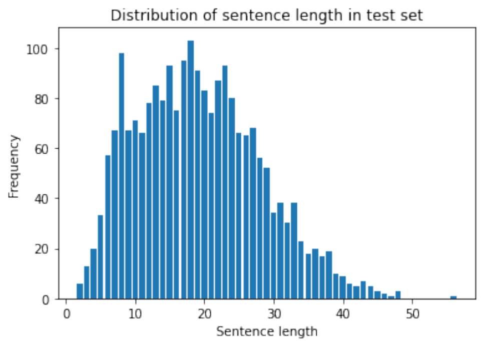
A.8 Subtree-LSTM model



B Distribution of sentence lengths in the train, validation and test sets







C Complete sentence length table with standard deviations

Model name	**10**	**20**	**30**	**40**	**50**
BOW	0.320 (±0.014)	0.318 (±0.023)	0.296 (±0.026)	0.303 (±0.038)	0.323 (±0.062)
CBOW	0.395 (±0.018)	0.361 (±0.006)	0.332 (±0.021)	0.312 (±0.032)	0.363 (±0.065)
Deep CBOW	0.379 (±0.018)	0.351 (±0.014)	0.363 (±0.014)	0.353 (±0.021)	0.303 (±0)
PT Deep CBOW	0.504 (±0.003)	0.458 (±0.003)	0.403 (±0.018)	0.379 (±0.019)	0.434 (±0.037)
LSTM	0.502 (±0.011)	0.476 (±0.004)	0.433 (±0.008)	0.422 (±0.015)	0.394 (±0.066)
Tree-LSTM	0.489 (±0.009)	0.494 (±0.008)	0.419 (±0.011)	0.4533 (±0.017)	0.465 (±0.014)