Practial Assignment 2: Representing Sentences using Neural Models

Piyush Bagad*
University of Amsterdam
piyush.bagad@student.uva.nl

Tadija Radusinović*
University of Amsterdam
tadija.radusinović@student.uva.nl

1 Introduction

Sentence representation is the process of embedding the important information within a sentence as a high-dimensional vector. Sentence representation using neural models involves training a neural network to produce representations which perform well when used on a given downstream task(s).

There are multiple ways of representing sentence input, which differ in what kind of structure the sentence is endowed with. In this report, we aim to answer how structural modelling choices influence performance of sentence representations on a downstream task. Specifically, we investigate the advantages of word order sensitive models which process data sequentially to bag-of-word (BOW) models, which regard the sentence as an unordered collection of tokens. We further investigate tree-based models, which construct sentence representations by recursively operating on the syntactic structure of the sentence (Socher et al., 2013).

Sentences vary in length, yet sentence representations are most often of a predefined size determined by modelling choices. We seek to find out how model performance varies across sentences of different size.

To answer these questions, we train our models on the sentiment classification problem for movie reviews. We make use of the Stanford Sentiment Treebank (SST) dataset (Socher et al., 2013). Parse tree annotations in SST are fine-grained: every node is annotated, along with a global sentiment classification. To evaluate the usefulness of fine-grained supervision, we compare models trained on whole sentences to models trained on whole sentences and sub-phrases at all depths. Moreover, we conduct an investigation of misclassified sentences by tracking down the node(s) in the parse tree at which a model makes a mistake.

* Equal contribution

Since word-order plays a big part in the semantic content of a sentence (compare *I loved X but hated Y* with *I loved Y but hated X*), we expect that order-awareness is crucial to achieving good sentence representations. Likewise, we would expect the tree structure of a sentence to be another useful source of information, allowing for more expressive representations.

Work which introduced the TreeLSTM by (Tai et al., 2015a) compares the performance of TreeLSTM representations with those gained by models including convolutional and various sequential RNN models. They also consider the impact of per-node supervision on accuracy and accuracy across sentence length, making their work closely aligned to ours. We further their results with our experiment by analyzing failure modes of tree-based models.

Our results show that incorporating tree-based structure into neural models leads to better-performing representations than those gained through sequential models, which in turn outperform bag-of-word models by a significant margin. In addition, we find that there is benefit to using more fine-grained supervision when training sequential and tree-based models. Finally, we find no significant difference between the two proposed variants of Tree LSTMs.

2 Background

2.1 Word Representations

Representing words in a (continuous) vector space has had a long history with impactful advances. The simplest word representations is the standard one-hot representation over the space of vocabulary. (Mikolov et al., 2013a) introduced the skip-gram model to efficiently represent words in a high-dimensional space such that semantically similar words occur closer together (also called as

word2vec model). (Mikolov et al., 2013b) proposed several efficient extensions to the skip-gram model. (Pennington et al., 2014) proposed GloVe which uses matrix factorization on a word-word co-occurrence matrix to obtain word representations. A sentence can be represented as a BOW as a sum of individual word embeddings $\mathbf{z} = \sum_t \mathbb{E}(x_t)$ where $\mathbb{E}$ maps one-hot to embedding space.

2.2 Recurrent Neural Networks

RNNs are a popular choice to model sequences. LSTMs (Hochreiter and Schmidhuber, 1997) are a special class of RNNs that are known to model long-term dependencies through a memory cell that is able to preserve information over long sequences. LSTMs take a sequence of vectors (e.g. word embeddings), and recursively process the sequence to obtain hidden-state vector h_t and the cell state c_t at each time step. In general, the sentence representation using LSTM can be stated as:

$$\mathbf{z} = f_\theta(\mathbf{x}) = h_T(\mathbb{E}(x_T), h_{t-1}, c_{t-1}) \quad (1)$$

where h_T denotes the last hidden state, $\mathbf{x}$ denotes the input sentence, z denotes the sentence representation (neural) model learnt using network f_θ. The transition equations at time t are given by:

$$\begin{pmatrix} i_t \\ f_t \\ g_t \\ o_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \tanh \\ \sigma \end{pmatrix} \begin{pmatrix} \mathbf{W}_i & \mathbf{W}_h \end{pmatrix} \begin{pmatrix} \mathbb{E}(x_t) \\ h_{t-1} \end{pmatrix} \quad (2)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g; \;\; h_t = o_t \odot \tanh(c_t)$$

We have ignored biases for brevity. Here, $\mathbf{W}_i \in \mathbb{R}^{4\times D}$ are the joint weights applying on input and $\mathbf{W}_h$ likewise for the previous hidden state.

2.3 Tree-LSTMs

Tree-LSTMs, concurrently proposed by (Tai et al., 2015b), (Le and Zuidema, 2015) and (Zhu et al., 2015), are a variation of LSTMs which operate on tree-structured data. The tree is processed recursively: hidden states of children nodes are combined to obtain the hidden state of the higher-level node. The sentence representation is given by the hidden state of the root node. In the N-ary Tree LSTM variant, the transition equations for node j with children indexed by $l \in \{1, .., N\}$ with hidden and cell states $\{h_{jl}, c_{jl}\}$ are given by (biases ignored for brevity):

$$\begin{pmatrix} i_j \\ f_{jk} \\ g_j \\ o_j \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \tanh \\ \sigma \end{pmatrix} \begin{pmatrix} \mathbf{W}^{(i)} & .. \mathbf{U}_l^{(i)} .. \\ \mathbf{W}^{(f)} & .. \mathbf{U}_{kl}^{(f)} .. \\ \mathbf{W}^{(g)} & .. \mathbf{U}_l^{(g)} .. \\ \mathbf{W}^{(o)} & .. \mathbf{U}_l^{(o)} .. \end{pmatrix} \begin{pmatrix} \mathbb{E}(x_j) \\ : \\ h_{jl} \\ : \end{pmatrix} \quad (3)$$

$$c_j = g_j \odot i_j + \sum_l f_{jl} \odot c_{jl}; \; h_j = o_j \odot \tanh(c_j)$$

Note that for each child, we have a separate weight matrix for each gate. In the child-sum variant of the TreeLSTM, child hidden states are summed before being passed to the gates (with the exception of the forget gate), which makes the model applicable to trees with nodes of varying order.

3 Models

Given a sentence representation $\mathbf{z} \in \mathbb{R}^D$, we want to learn a map $g_\phi : \mathbb{R}^D \to \mathcal{C}$, i.e. $\hat{y} = g_\phi(\mathbf{z})$. For BOW-based model variants, g_ϕ can be identity or a sequence of linear layers. For e.g. DeepCBOW has g_ϕ as 3 hidden layers etc. For LSTM variants, g_ϕ is a single linear layer with dropout for regularization. Each model is trained to jointly learn $\{f_\theta, g_\phi\}$ using cross-entropy loss. Note, softmax layer is assumed within the loss function. An exhaustive list of all variants can be found in Table 1.

4 Experiments

4.1 Task and Dataset

As detailed in Section 3, we consider the task of sentiment classification from sentences with labels from 0 (highly-negative) to 4 (highly-positive). For this, we use the Stanford Sentiment Treebank (SST) (Socher et al., 2013) dataset that has annotations at two levels: binary parse trees of sentences and sentiment label for each node (including the root node, i.e. overall sentence). We use the standard (provided) train/dev/test splits.

Prior to experimentation, we perform an exploratory analysis of the dataset to develop intuitions and identify obvious patterns, if any. As a concise summary, we highlight (a) the dataset is balanced across splits, (b) there are no obvious confounders (e.g., sentence length), (c) conditional distributions reaffirm common intuitions (e.g. the word *brilliant* appears primarily in highly positive labelled sentences). Please refer to the Supplementary section B.1 for more details.

4.2 Training and Evaluation

We train all the models using Adam optimizer with learning rates and iterations tuned based on the development set. Please refer to Appendix A.1 for an exhaustive list of hyperparameters. For evaluation, we use accuracy as the primary metric. For each experiment, we report average accuracy across three runs with different seeds (0, 42, 420).

Method	Accuracy	$\lvert\Theta\rvert$
BOW	0.329 ± 0.009	0.09
CBOW	0.348 ± 0.022	5.48
DeepCBOW	0.371 ± 0.007	5.52
PTDeepCBOW	0.442 ± 0.002	5.71
LSTM	0.460 ± 0.005	6.93
LSTM (subtrees)	0.476 ± 0.001	6.93
TreeLSTM (w2v)	0.462 ± 0.003	5.99
TreeLSTM (Glove)	0.467 ± 0.005	6.53
TreeLSTM (subtree)	0.496 ± 0.002	5.99
CSTreeLSTM (w2v)	0.461 ± 0.002	5.84
CSTreeLSTM (Glove)	0.467 ± 0.008	6.39

Table 1: Test set accuracies on the Stanford Sentiment Treebank (SST) dataset. For each experiment, we report the mean and standard deviation across 3 runs with different seeds. We also report the total number of parameters (in millions) in the model for each method. Here, w2v/Glove denote the word embedding initialization method, subtree denotes training with node-level supervision (see Section 5.4 for further details), prefix CS denotes the child-sum variant of TreeLSTMs. $\lvert\theta\rvert$ denotes the size of the model, i.e. number of parameters.

Method	Shuffle	Accuracy
LSTMClassifier	✗	0.460 ± 0.005
LSTMClassifier	✓	0.445 ± 0.002
TreeLSTM (w2v)	✗	0.462 ± 0.003
TreeLSTM (w2v)	✓	0.431 ± 0.006

Table 2: Evaluating the impact of word-order. We experiment with trained LSTMs by evaluating on two versions of test set: vanilla word-order (✗) vs one where words are randomly shuffled within sentences (✓).

5 Results and Analysis

5.1 Impact of Word-order

BOW models are word-order insensitive since we average the word embeddings in a sentence. Both LSTM-based models are order-sensitive and we observe an improvement of 1.8% with basic LSTM over the best bag-of-words model (PTDeepCBOW). Further, we probe the LSTM models by evaluating with and without random shuffling of words in the sentences from the test set. As reported in Table 2, we observe a relatively small drop of 1.5% for LSTM classifier whereas 3.1% for TreeLSTM model. Thus, for this task, word order does provide marginal improvements but it is not an inherently limiting factor for this task.

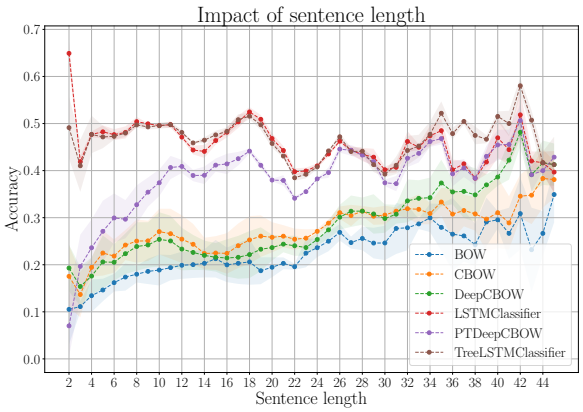

Figure 1: Evaluating the impact of sentence lengths on performance on the test set. For each l, we plot the accuracy (mean $\pm$ 1-standard deviation across the same 3 seeded runs) for each model averaging across all sentences with lengths in $[l-2, l+2]$. For the sentences with $l > 45$, we simply club them onto the set of $l = 45$.

5.2 Impact of Tree-structure

Vanilla LSTMs retain word-order but are insensitive to the syntactic structure of the sentence. TreeLSTMs consider the tree structure of the sentence and achieve marginally better performance than vanilla LSTMs while using sentence-level supervision only (Table 1). This gap further widens to over 2% while using additional node-level supervision. Thus, encoding tree-structure of sentences into the model improves performance.

5.3 Impact of Sentence Length

To evaluate our models against varying sentence lengths, we consider all the sentences in the test set and evaluate on each subset of given sentence length l. Our experiment design closely follows (Tai et al., 2015b) (see Figure 3 from the paper) and we report results in Figure 1. First, we notice that for all models, the performance improves as sentence length grows very large ($l > 20$). This may be attributed to having more (perhaps, enough) information per sentence to determine the sentiment irrespective of the model. Second, for short sen-

tences (2-8 words), both the LSTM models heavily outperform the BOW models. Likewise, for longer sentences (10-20 words), we observe a similar pattern. Third, among the LSTMs, tree-based model beats sequential for extremely long sentences. This is likely since TreeLSTMs, by design, can preserve more information since they operate on a tree-structure (appx. $\mathcal{O}(\log l)$) rather than sequential LSTMs (appx. $\mathcal{O}(l)$).

5.4 Additional Supervision via Node Labels

We also experiment with using node-level labels in the sentence tree with both LSTM models. As one would expect, in both cases, additional supervision helps improve the accuracy by 1.6%, 3.4% respectively (see Table 1). Apart from the benefits of more data, note that this approach also benefits by learning from fine-grained syntax of the sentence in contrast to overall label - this especially benefits TreeLSTM. For example, a highly positive sentence may have phrases that are either neutral or negatively oriented. This supervision enables the model to focus on the most relevant parts of the sentence that derives the overall label.

5.5 Child-sum Tree vs N-ary Tree LSTM

(Tai et al., 2015a) introduced two variants of tree LSTMs: (a) *N-ary* in which each node has a branch factor of atmost N and allows separate set of parameters for each child, (b) *child-sum* where the hidden state of a node is defined as the sum of the hidden states of the children. We experiment with both variants and observe no significant difference in accuracy between the two as reported in Table 1. The similar performance may be attributed to the fact that the dataset only has binary connections in the tree structures and it may suffice to model the left and right children with a single set of parameters instead of two.

5.6 Error Analysis of Tree-based Models

A natural question to ask about TreeLSTM models is whether we can characterize its success and failure modes and relate it to linguistic notions? We design an experiment to answer a specific question: how does the misprediction rate change with the height of node w.r.t. the leaves? As shown in Figure 2, we observe that among successfully classified sentences (orange), the misprediction rate increases initially and then decreases as the node height grows. This implies that for the leaf nodes and lower-level nodes, it may mispredict, however,

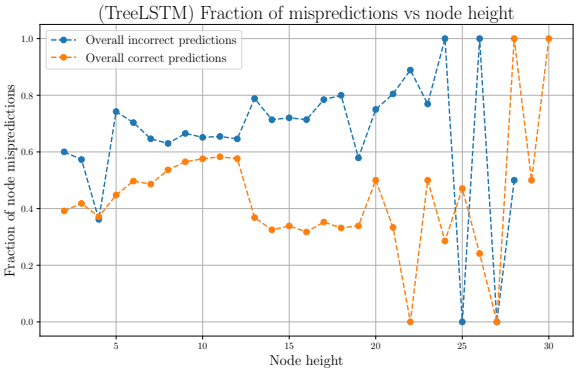

Figure 2: For TreeLSTM, for a given node height, we plot % of mispredicted phrases for overall successfully classified sentences and overall misclassified sentences.

as it sees more context, the predictions get better. This is almost inverted for sentences that were overall mispredicted[1].

6 Conclusion

We present an empirical comparison of various sentence representation models on the sentiment classification task. Our experiments analyze the robustness of these models to factors like word-order, sentence length etc. Our analysis reinforces the well-researched notion that sequential and tree-based models are robust to word-order and change in sentence length. Further, for tree-based LSTM models, we evaluate the impact of node-wise supervision and compare the N-ary and child-sum variants. Our results are consistent with (Tai et al., 2015b). We also highlight an interesting mode of failure of tree-based LSTMs pointing to their over-reliance on node-level supervision, without which the misprediction rate increases as one goes from leaf-level nodes to the root. Having node-level supervision also demands extremely fine-grained annotation which can be costly and/or error-prone. A possible future direction could be to improve tree-based models via using the semantics (e.g. POS tags) as a form of weak supervision along with the sentence-level supervision.

[1]We only report results for seed 0 since the pattern is similar for other seeds and predictions change across seeds making it hard to compare on the same plot.

Acknowledgments

We thank our TA Simone Astarita for useful feedback on the submission format etc,. We also thank the instructors, fellow students and TAs (esp. Mario Giulianelli) for fruitful discussions on Piazza.

References

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

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory.

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

Tomás Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013c. 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.

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. 2015a. 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.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015b. Improved semantic representations from tree-structured long short-term memory networks. *CoRR*, abs/1503.00075.

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*, volume 37 of *Proceedings of Machine Learning Research*, pages 1604–1612, Lille, France. PMLR.

Model	Iters	Batch	LR
BOW	250k	1	5e-4
CBOW	100k	1	1e-4
DeepCBOW	30k	1	5e-4
PTDeepCBOW	60k	1	1e-5
LSTM	2k	25	2e-4
LSTM (subtree)	30k	25	2e-4
TreeLSTM	5k	25	2e-4
TreeLSTM (subtree)	50k	25	2e-4
CSTreeLSTM	50k	25	2e-4

Table 3: Hyperparameters for each of the models.

A Appendices

A.1 Hyperparameters

We tuned all hyperparameters (where needed) on the dev set. For TreeLSTM and PTDeepCBOW, we initialized with 300-dimensional word embeddings. We experimented with `word2vec` (Mikolov et al., 2013c) and Glove (Pennington et al., 2014). We found Glove fractionally better than `word2vec` on TreeLSTM. For consistency, we retained `word2vec` throughout all our experiments unless otherwise stated. All models were trained on a Apple Mac M1 machine with no GPUs.

We use Adam optimizer with learning rates tabulated in Table 3. For LSTMs, we use a mini-batch of 25 sentences. We use dropout regularizer with a rate of 0.5 in the output layers of LSTM and TreeLSTM. For the experiments with subtrees, we only modify the train set and consider all possible subtrees (and its label) within each sentence. The training set size grows from 8544 to 318582. The varying number of iterations for various models can be attributed to the different model capacity and is also constrained by computational resources and time. Note that BOW-models are trained with single-sample batches unlike LSTMs. For example, BOW is a tiny model and needs a large number of iterations. On the contrary, a pretrained embedding CBOW model already has a good initialization and thus requires less number of iterations.

B Supplemental Material

B.1 Digging deep into the dataset

Basic exploration of a dataset helps in identifying pitfalls and/or patterns that invariably benefit model development or motivate experiment design. For the SST dataset, we divide our analysis between the following aspects.

Identifying imbalance and confounders. First, we plot the label distribution across the train and dev splits and observe that the data is balanced (Figure 4). Next, we ensure that sentence length is not a *confounder* i.e., a variable that influences both the dependent variable and independent variable, causing a spurious association. It can happen that all positive sentences are quite short and all negative sentences are very long. Thus, the model may not learning anything semantic and may exploit the sentence length for prediction. There can be several such confounders but we restrict to checking sentence length. As shown in Figure 4, sentence length is not a confounder in this dataset.

Conditional distributions. We also look at distributions such as $p(w|y)$ (Figure 4) where w is a token (word) and y is the class label. We also plot $p(y|w)$ denoting distributions over labels of certain special tokens (Figure 3). Notice that occurrence of highly positive words like `brilliant` and `amazing` correlates with positive sentiment labels. Likewise, for negative words like `trash`, `terrible`. Neutral words like `film` occur with positive and negatively labelled sentences and hence will not play significant predictive role.

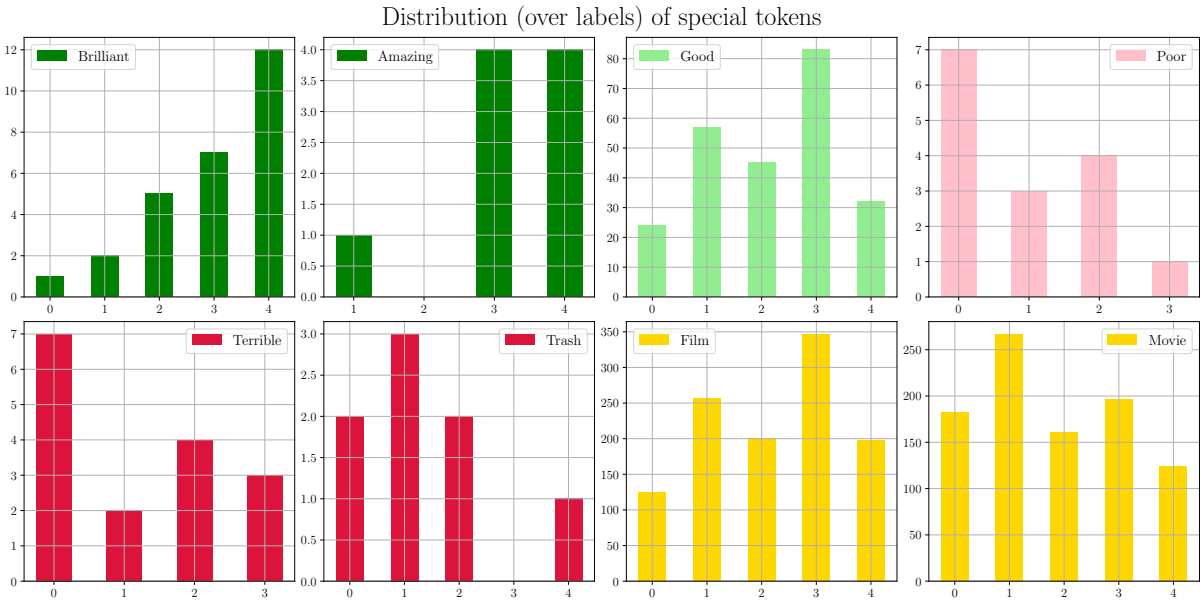

Figure 3: Distribution over labels of occurrence of specially chosen tokens that denote highly positive or negative sentiment within the sentence.

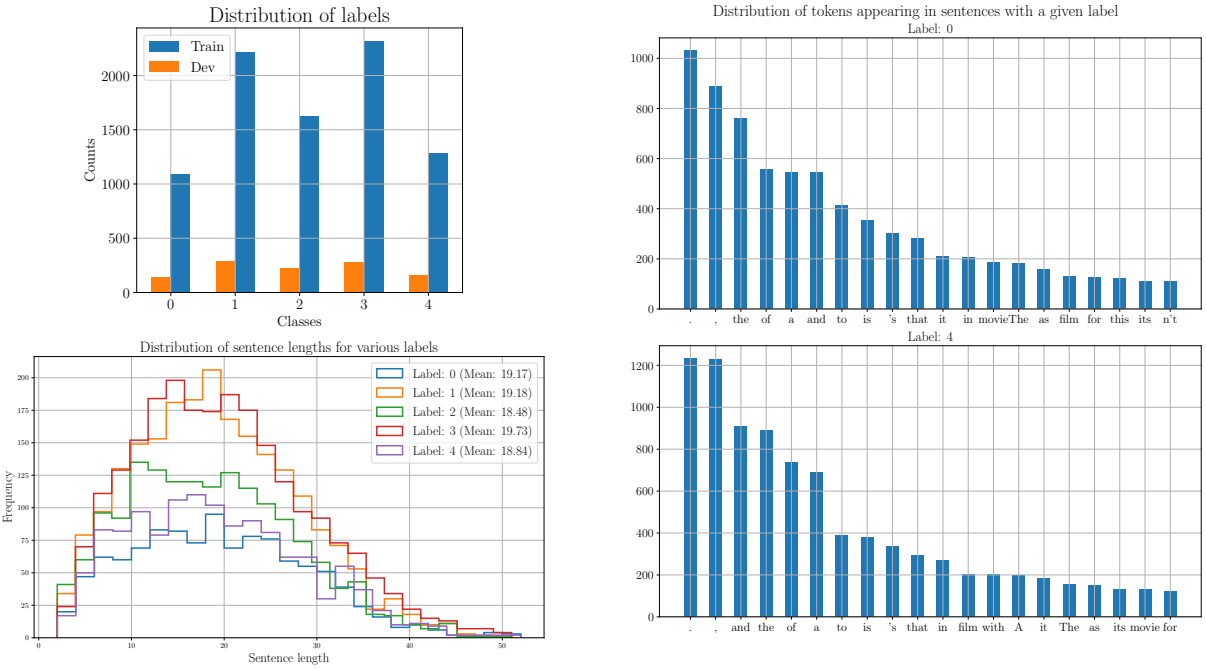

Figure 4: (Left top) Distribution of labels in train and dev splits in the SST dataset. (Left bottom) Distribution of lengths of sentences conditioned on label. (Right) Most frequently occurring tokens for labels $\{0, 4\}$