Representing Sentences with Neural Models

Serwan Jassim
ID: 12880647

Qiao Ren
ID: 11828668

1 Introduction

With the sheer amount of data in the information age it has become increasingly important to automate classification tasks. One of such is sentiment classification, which deals with the assignment of positive and negative labels to documents. Previously, methods based on lexical methods (Wilson et al., 2005) and naive Bayes (Pang et al., 2002) have been proposed for this task, however in this work we will focus on neural models. We examine the performance of several neural models on the Stanford sentiment treebank (SST) (Socher et al., 2013) dataset, which is made up of movie reviews.

The models examined in this work can be divided into two categories. The first consists of bag of word architectures, which don't take word order into account. These models include vanilla bag of words (BOW), continuous BOW (CBOW) and deep CBOW. Furthermore we investigate the more elaborate Long Short Term Memory (LSTM) model (Hochreiter and Schmidhuber, 1997) and Tree-LSTM (Le and Zuidema, 2015; Tai et al., 2015; Zhu et al., 2015) architectures. These don't only consider the word order, but can even process parsed tree in case of the Tree-LSTM and can therefore take syntactic information into account.

By comparing these models, we aim to solve the following research questions:

1. How important is word order for sentiment classification?
2. Does the tree structure help with the classification?
3. How does the performance depend on the sentence length?
4. Does supervision at each node in a tree improve the performance?
5. How do the Tree-LSTM formulations by (Le and Zuidema, 2015) and (Tai et al., 2015) compare?

Our goal by investigating these questions is to find the most robust and and confident classification method. We expected the LSTM models to outperform the BOW models due to the additional information that they incorporate. Additionally, we assumed that including the tree structure would increase the performance of the classifier. While we could confirm our first assumption, the results showed that the second one is actually not the case.

2 Background

2.1 Word Embeddings

The input to neural models are word tokens which can be represented as indices within a vocabulary. However, this representation doesn't incorporate any information about the actual meaning of the word. In order to be able to use more expressive representations of words, learnable word embeddings are used. The embedding of a word is simply a mapping to a vector of arbitrary dimension that encodes its meaning.

2.2 Feed-Forward Neural Networks

Feed-Forward Neural Networks (FFNN) can be interpreted as parametrized function approximators. They apply layers of affine transformations followed by non-linearities iteratively on an input and can learn optimal parameters through optimization. In this work they are used to obtain classification scores from different input feature vectors.

2.3 Bag of Word Models

The main property of BOW models is the fact that they drop any information about the word order. They do this by summing up the feature vectors of

all words in a sentence, which results in a single feature vector for the whole sentence. Subsequently, an arbitrary classifier can be used to classify this sentence representation.

2.4 Long Short Term Memory

The LSTM is similar to a FFNN, but instead of taking a single input, it can process a sequence of inputs recursively. It assigns a state to each input of the sequence, which can be subsequently used to perform classification. By using a memory state, it is capable of storing information about past inputs. Furthermore, it uses so-called forget, input and output gates to modulate the memory state. Opposed to BOW models, this model considers word orders since the words are processed sequentially.

2.5 Tree-LSTM

Tree-LSTMs are a generalization of LSTMs to tree-structured network topologies. Instead of processing word tokens recursively, Tree-LSTMs are applied to syntactic trees. They compute a state for each node in a tree, which is determined by all the children of a node. The state of the root node can be used to classify the whole tree.

3 Models

In this work, we use several variants of the BOW model. The first one is the vanilla BOW, which represents each word in a sentence by a feature vector with same dimensionality as there are number of classes. Each entry in the sentence feature vector corresponds to a prediction score for a class.

The second BOW model we use is the Continuous BOW (CBOW). This model drops the constraint to keep the dimensionality of the word representations equal to the number of classes. Instead, we use arbitrary number of dimensions and map the sentence feature vector with a one-layer FFNN to prediction scores.

The deep CBOW is very similar to the CBOW model. It drops a further constraint of using a one-layer FFNN and replaces it by a multi-layer one. In this work, we use a three-layer FFNN for the deep CBOW model.

Furthermore, we use the vanilla LSTM model in this work. We apply a one-layer FFNN on the state of the final word in the sentence to obtain the prediction scores for the sentiments. In addition to the vanilla LSTM, we also use the Tree-LSTM models as proposed by (Le and Zuidema, 2015) and (Tai et al., 2015). The main difference between these models is that the formulation by (Le and Zuidema, 2015) computes separate input and forget gates for each child of a node to produce an output state of a parent. The formulation by (Tai et al., 2015) on the other hand just uses separate forget gates per child. To compute the prediction scores from the treeLSTMs, we apply a FFNN on the root state of the syntax trees.

Finally, we use embedding layers at the start of all the models mentioned above and train them using the cross entropy loss layer.

4 Experiments

In this work, we evaluate the previously discussed BOW and LSTM models on the sentiment classification task. The dataset that we use for this examination is the Stanford sentiment treebank (SST) (Socher et al., 2013), which consists of constituency trees that are parsed from movie review sentences. Every node in a tree is labeled with one of the following sentiment scores: very negative, negative, neutral, positive and very positive. The goal of sentiment classification is to correctly predict the sentiment of the root node, which represents the sentiment of a whole sentence. The data is split into a training, development and test set, which are made up of 8544, 1101 and 2210 sentences respectively.

We train all of the models using Adam (Kingma and Ba, 2014) together with the cross-entropy loss. Table 1 provides a detailed list of all the models that we examine and their hyperparameter setups. Furthermore, we use an embedding size of 300 for all models. During training we save the model that performs best on the development set in terms of accuracy. We compare the models using the accuracy on the test set, averaged over three separate training runs with different seeds.

For the Deep CBOW, LSTM and Tree-LSTM models we utilize pre-trained GloVe word embeddings (Pennington et al., 2014) in order to start off with good word representations. We freeze the embedding layer in these model and continue to just train the weights.

In order to evaluate the impact of supervision at each tree node on the Tree-LSTM, we train it a second time using a dataset that is made up of all possible subtrees of the original training set. This results in a training set of 155022 examples, which is also the reason why we train this model for more

iterations than the first model that was only trained on entire trees (see Table 1).

Furthermore, we split the training set into four partitions and repeat the training on each of them to examine the impact of the sentence length on the performance of the models. These partitions include the following sentence lengths: two to 12, 13 to 18, 19 to 25 and 26 to 52. We choose these intervals to make sure that the partitions are of similar sizes (roughly 550 samples).

5 Results and Analysis

5.1 Word Order and Tree Structure

The second column in Table 2 shows the average accuracy and standard deviation for all models tested on the entire test set. In order to investigate the importance of word order for the sentiment classification task, we compare the group of BOW architectures to the LSTM models.

We find that all BOW models perform consistently worse than the LSTM models, with the best BOW model (pre-trained Deep CBOW) being 3 % off the accuracy for the worst LSTM model (vanilla LSTM). As opposed to LSTM models, BOW architectures aren't able to account for the word order of sentences, i.e. these results show that the word order is indeed an improving factor. This observation makes sense from an intuitive viewpoint. Consider for example the appearance of "not good" together with "awful" in a sentence. Without knowing that "good" is preceeded by "not" and instead just knowing that the individual words appear, one would be able to confidently classify the sentence.

When comparing the vanilla LSTM to the tree-LSTM variants, we observe that there is no real improvement in performance considering the standard deviation. This shows that the word order is a more decisive factor in sentiment classification. One reason why this might be the case is the fact that the five sentiments we discriminate are still very broad concepts and that the additional information that the tree structure introduces is redundant.

5.2 Sentence Length

We compare the accuracies that are computed on partitions of the test set with different sentence lengths and shown in Table 2 to investigate the impact of the sentence length on the performance of the models. Considering the standard deviation, we can't say with confidence how the BOW, CBOW and Deep CBOW models perform for different sentence lengths. The general trend for the other models though shows that it seems easier to classify shorter sentences than longer sentences. The LSTM models reach close to 50 % accuracy for sentences with a length between two and 12 words, while for a length between 26 and 56 words it's only around 45 %.

This result seems intuitively reasonable as well since short sentences are precise and to the point so that the general sentiment can be conceived easily. On the other hand, long sentences might discuss multiple viewpoints and might therefore contain multiple sentiments that are harder to understand. Even for humans it is much easier to grasp the content of short sentences than for longer ones.

5.3 Supervision at Each Node

When comparing the accuracies in Table 2 for the basic Tree-LSTM model to the Tree-LSTM that was supervised at each node during training, we can't observe any significant improvements considering the standard deviation. While beforehand, we expected an improvement of the performance due to more confident classification on the subtree levels, this behaviour seems to confirm our findings from Section 5.1. Compared to the word order, it seems like the tree structure doesn't include any additional information that is important for this task.

5.4 Alternative Tree-LSTM Formulation

In addition to the Tree-LSTM formulation by (Tai et al., 2015), we also trained a Tree-LSTM according to the formulation of (Le and Zuidema, 2015) and list its accuracy in Table 2. The performance of the two models is very similar and we can't say with confidence which one performs better. It is possible though, that the formulation by (Le and Zuidema, 2015) works better on a larger training set since this model has roughly twice as many parameters. While it should be more expressive due to this, it is also more prone to overfitting.

6 Conclusion

In this work, we examined several neural models on the sentiment classification task. The main difference between these models is the fact that the BOW models drop information about word order within a sentence, while LSTM models incorporate syntactic information in their computations. As we expected, the LSTM models performed consistently

Model	Learning Rate	Iterations	Batch Size	Hidden Size
BOW	$5 \cdot 10^{-4}$	$1.0 \cdot 10^{5}$	1	/
CBOW	$5 \cdot 10^{-4}$	$4.0 \cdot 10^{4}$	1	/
Deep CBOW	$5 \cdot 10^{-4}$	$2.5 \cdot 10^{4}$	1	100
Deep CBOW (pre-trained)	$5 \cdot 10^{-4}$	$3.0 \cdot 10^{4}$	1	100
LSTM	$2 \cdot 10^{-4}$	$5.0 \cdot 10^{3}$	25	150
Tree-LSTM	$2 \cdot 10^{-4}$	$5.0 \cdot 10^{3}$	25	150
Tree-LSTM (subtrees)	$2 \cdot 10^{-4}$	$2.0 \cdot 10^{4}$	25	150
Tree-LSTM (Zuidema & Le)	$2 \cdot 10^{-4}$	$5.0 \cdot 10^{3}$	25	150

Table 1: Hyperparameter setting for the training of the individual models. The "pre-trained" note refers to the use of GloVe embeddings, while "subtrees" refers to the supervision at each node in the tree-LSTM.

Model	all sentences	2 to 12 words	13 to 18 words	19 to 25 words	26 to 56 words
BOW	31.5 ± 1.3	30.7 ± 2.3	30.7 ± 2.2	32.2 ± 0.2	32.3 ± 2.9
CBOW	36.7 ± 0.7	38.5 ± 1.9	36.0 ± 0.5	34.4 ± 2.3	37.7 ± 1.5
Deep CBOW	37.4 ± 0.7	38.3 ± 2.5	37.2 ± 0.8	37.5 ± 1.9	36.5 ± 0.2
Deep CBOW (pre-trained)	44.0 ± 1.1	47.6 ± 2.2	45.0 ± 0.2	41.4 ± 2.1	41.9 ± 0.8
LSTM	47.2 ± 0.1	$\mathbf{52.7 \pm 0.3}$	48.7 ± 0.9	42.9 ± 0.9	44.2 ± 1.3
Tree-LSTM	47.4 ± 0.9	49.9 ± 0.5	49.3 ± 0.8	44.4 ± 1.2	$\mathbf{46.2 \pm 1.5}$
Tree-LSTM (subtrees)	$\mathbf{48.6 \pm 1.1}$	51.3 ± 1.5	$\mathbf{50.6 \pm 0.8}$	$\mathbf{46.2 \pm 1.5}$	$\mathbf{46.2 \pm 1.3}$
Tree-LSTM (Zuidema & Le)	47.8 ± 0.4	52.0 ± 0.9	49.1 ± 0.8	44.0 ± 1.4	$\mathbf{46.2 \pm 1.5}$

Table 2: Accuracies for all neural models averaged over three separate training runs with different seeds and their standard deviations. The column titled "all sentences" includes accuracies for the entire test set, while the remaining columns show the results on partitions of the test set. The titles of these columns denote the sentence lengths that are included in these partitions. The "pre-trained" note refers to the use of GloVe embeddings, while "subtrees" refers to the supervision at each node in the tree-LSTM.

better than the BOW models due to this different behaviour. Unexpectedly though, we weren't able to improve the performance of the vanilla LSTM with Tree-LSTMs. Neither of the Tree-LSTM variants that we examined seemed to add further information that is relevant to the sentiment classification task. However, we suggest to repeat this experiment with the Tree-LSTM formulation by (Le and Zuidema, 2015) on a larger training dataset due to the overfitting on the relatively small SST dataset. Since it is more expressive than the formulation by (Tai et al., 2015), we expect it to perform better on a larger dataset.

References

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

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.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 79–86. Association for Computational Linguistics.

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.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

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

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 347–354, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 1604–1612. JMLR.org.