Phrase Based Sentiment Classification Using Tree-LSTMs: Does Attention Help?

Anhad Mohananey
Department of Computer Science
New York University
New York, NY 10003
anhad@nyu.edu

Daniel Rivera Ruiz
Department of Computer Science
New York University
New York, NY 10003
daniel.rivera@nyu.edu

Abstract

Sentiment Analysis is a classic problem in NLP. Tree Structured LSTMs are the current state of the art for this task. Although they effectively encode the recursive nature of language syntax through combining phrases, they fail to capture the dependencies between phrases which are not on the same path from root to leaf in an effective manner for the task of phrase sentiment classification. Our proposal introduces an Attention-based encoder-decoder mechanism to effectively capture these dependencies in the tree structure, and not just the ones limited to a bottom-up relation. Attention has proven to be extremely successful in neural machine translation, and we wish to apply it to improve the state of the art for phrase based fine-grained sentiment analysis.

1 Introduction

Traditionally, and even in deep learning literature, sentiment analysis is often treated as sequential task. Thus, LSTMs are a suitable approach to this problem. However, it is believed that linguistic knowledge is represented better in the form of tree structures. This has given way to a series of well performing models in recent years that predict the sentiment of the sentence using tree structures. Although these Tree Structured LSTM models do a fantastic job at predicting the sentiment of the entire sentence, they give moderate performance for prediction of sentiment label for individual phrases. For example, the tree structured models may do a good job at predicting the label for "A B C", where A, B and C are phrases of the larger sentence. But may do a poor job at predicting the label for "A B" or "C"(assuming the tree structure is ((A B) (C))).

Our hypothesis is that this happens because the dependence between two phrases X and Y is not modelled unless one is a sub-phrase of the other. In other words, only the nodes that are on the same path from the root to the leaf show dependency. To solve this problem, we add an Attention-based layer that takes the inputs from the tree as an encoder, and churns the output into an LSTM, which acts as a decoder. The final predictions are the outputs from the sequential LSTM. The main objective

of this paper is to gain some insight on whether or not attention improves the accuracy of fine-grained sentiment analysis. In other words, what we really want to know is: "does attention help?".

2 Related Work and Baselines

The explosion of deep learning in natural language processing has had a significant influence on the task of sentiment analysis. In recent years, certain variants of neural networks have been applied to this problem with varying success. In Kim (2014), Convolutional Neural Networks are introduced for sentiment analysis. This model experimented with different word embeddings, primarily random and pre-trained embeddings obtaining state of the art results when it was published. That same year, Quoc and Mikolov (2014) implemented a paragraph level vectorization model combined with word vectors for this task. A traditional neural network approach is to use recurrent neural networks (RNN) as they encode sequential information in a very efficient manner, and even though they have yielded good results, they fail to capture the syntax of the sentences.

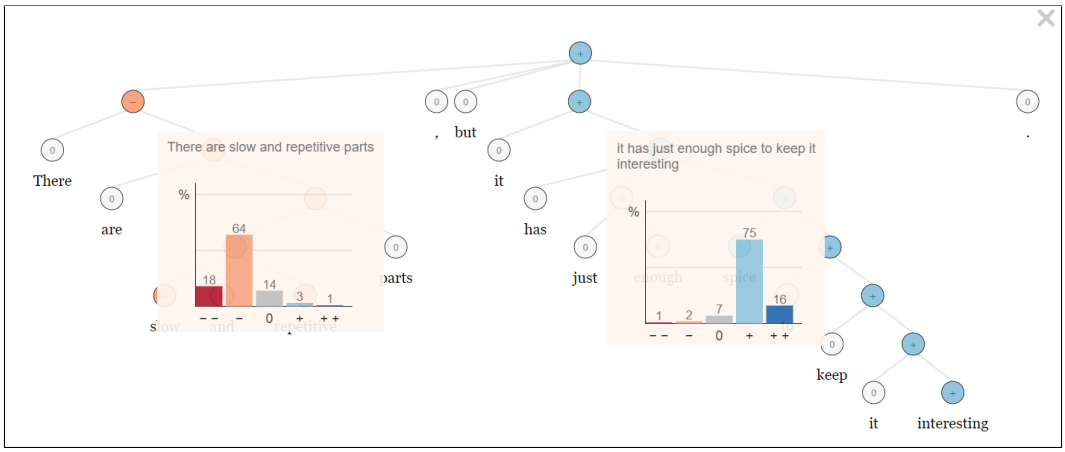

Figure 1: Classification results for the sentence *"There are slow and repetitive parts, but it has just enough spice to keep it interesting."* using the Stanford Sentiment Treebank.

Among the RNN models, Long-Short Term Memory (LSTM) usage has increased for sentence classification tasks over the last few years. Its main success resides on its ability to learn long term dependencies from text data. The most recent breakthrough work in sentiment analysis was by Tai et al. (2015) with the introduction of Tree Structured LSTMs, based on the inherent idea that language syntax definitions are recursive. They experimented with two different versions of Tree-LSTMs, namely the Child-Sum (Dependency) Tree-LSTM and the N-ary (Constituency) Tree-LSTM, to obtain state of the art performance on the Stanford Sentiment Treebank data set. They achieved an accuracy of 51% on the fine-grained 5 class sentence classification and 88% on the binary sentence classification. An example of the tree-based sentiment classification can be observed in figure 1.

The results reported in this paper were only at the sentence classification level, although phrase classification was also performed. Since we wish to run experiments for phrase classification, we build on top of their base code and add a method to run predictions at the phrase level.

3 Proposed Approach

RNNs work by processing inputs in a sequential manner and they are very efficient when working with input/output sequences of variable length. At each time step, a hidden output is created that depends on the current input as well as on the hidden output of the previous unit:

$$h_t = tanh(W(h_{t-1}) + U(x_t) + b) \tag{1}$$

Equation 1 is known as the RNN transition function. In this equation, h_t is the hidden output of the current cell, h_{t-1} is the hidden output of the previous cell and x_t is the input to the current cell. A non-linear function, like *tanh* in this case, is always at the top level of h. A major issue with RNNs is that during training components of the gradient can grow or decay exponentially. This makes it difficult for RNN models to capture long-term dependencies.

LSTM networks mitigate this problem by introducing additional gates that are able to capture dependencies over an extended period of time. The gates that form an LSTM are the following: i_t input gate, f_t forget gate, memory cell c_t and hidden state h_t. A simple model of an LSTM (where all the gates are not visible since they exist internally in each cell) can be observed in figure 2.

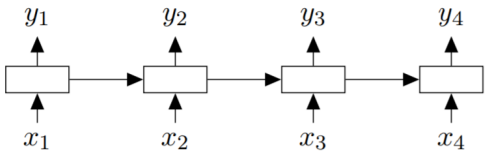

Figure 2: LSTM model.

In the original TreeLSTM paper (Tai et al., 2015), two variants of the Tree-LSTM were introduced: Child Sum and N-ary. For the purposes of this paper, the Child Sum Tree-LSTM was used as the encoder step of the model. The equations for these Tree-LSTM are the following:

$$\begin{aligned}
h_j &= \sum h_k \\
i_j &= \sigma(W^{(i)}x_j + U^{(i)}h_j + b^{(i)}) \\
f_{jk} &= \sigma(W^{(f)}x_j + U^{(f)}h_k + b^{(f)}) \\
o_j &= \sigma(W^{(o)}x_j + U^{(o)}h_j + b^{(o)}) \\
u_j &= \sigma(W^{(u)}x_j + U^{(u)}h_j + b^{(u)}) \\
c_j &= i_j \odot u_j + \sigma f_{jk} \odot c_k \\
h_j &= o_j \odot tanh(c_j)
\end{aligned}$$

In these equations, h_k represents the hidden state of the children and x_j represents the input, which in our case would be the word embedding of the phrase concatenated together. With this model, the gates can control how much importance is given to the current input and to the previous hidden state. Finally, the h_k values are summed together, hence the name Child Sum Tree-LSTM.

Tree-structured LSTMs perform well at the task of sentence classification because they consider tree-structures at the phrase level. Thus, they can perform phrase level classification as well as classification of full sentences. However, phrase level classification in tree-structured LSTMs fails to accurately capture dependencies between two phrases, even though they may have a common ancestor. A simple model of such a Tree-LSTM model can be observed in figure 3.

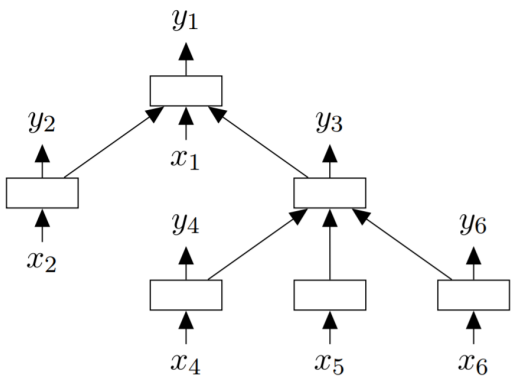

Figure 3: Tree-LSTM model.

With the limitations of the Tree-LSTM model in mind, we decided to develop an Attention-based encoder-decoder model like the one depicted in figure 4. In this model, the outputs from each node of the Tree-LSTM (encoder) serve as input for the second phase, implemented as a regular LSTM (decoder).

Let the output of each node of the Tree-LSTM be represented by z_i. As in traditional Attention-based models, we apply a linear layer on top of these z values coming from the encoder. The linear layer is applied in such a way so that the i^{th} input to the decoder can be expressed as follows:

$$d_i = \sum w_{ij} z_j$$

The decoder is a standard sequential LSTM which works by taking the output of the previous hidden layer, as well as the current input d_i, which comes from the encoder. The output at each cell is passed through a simple *logsoftmax* layer to give us the log probability of each class, which in this case is associated to phrase-level sentiment prediction.

To sum up, the decoder has one output cell for each phrase in order to perform phrase level classification. At each decoder unit, an additional attention factor is taken into consideration. The attention factor is represented by the weighted average over all the output stages (phrases) of the encoder, with the weights being trained by a single final cost function.

Apart from the attention model, a simple encoder-decoder model was used to benchmark the results for phrase classification. The main difference between the two models is that the simple encoder-decoder uses the same context vector for all inputs to the decoder phase. These kind of models usually perform worse than Attention-based models due to the fact that the context is of fixed length, so variable length sequences that are particularly long cannot be appropriately encoded. Moreover, there is no way of associating different levels of importance to certain phrases when making the prediction for a particular phrase. The hypothesis is that the Attention-based model should perform better than the simple encoder-decoder, since the latter considers only a subset of the possibilities of the former.

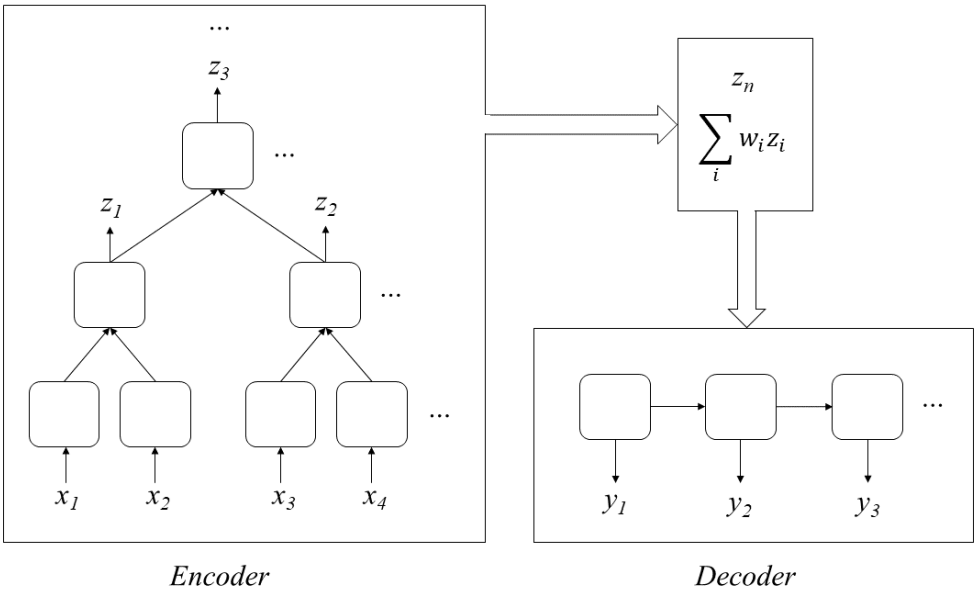

Figure 4: Proposed Attention-based Model. The output of the decoder $y_1, y_2, ...y_n$ represent the phrase-level scores of the classifier.

4 Experiments

In the original experiments from Tai et al. (2015), the Tree-LSTM model was trained for five epochs and evaluated on the dev set (1101 examples) after each epoch. The best-scoring model on the dev set was then used to obtain the final accuracy of the model on the test set (2210 examples).

Due to time and computing power limitations, we decided to run our models for only one epoch to skip evaluation on the dev set and calculate accuracy directly on a sample of 100 examples from the test set. To establish a fair baseline for all the models that were to be evaluated, the original Tree-LSTM model was also run under these conditions.

4.1 Configuration and Parameters

Table 1 shows the configurations and hyper parameters used for each one of the three models that we considered for this paper. The resulting number of model parameters, which is also included in this table, depends largely in the complexity of the model: it can be seen, for example, that the Attention-based model has around 500 times more parameters than the original Tree-LSTM model.

5 Results

Table 2 shows the results of the experiments detailed in section 4. It can be observed that the original Tree-LSTM model yields better results by about 4 points. Comparing the two new proposals, the one that captures attention scores slightly better than the one that doesn't.

Table 1: Experiments configuration and parameters

	Tree-LSTM	Encoder-Decoder	Attention
Epochs	1	1	1
Batch size	25	25	25
Train set size	8544	8544	8544
Test set size	100	100	100
Word vectors dimension	300	300	300
Tree states dimension	150	100	100
Maximum nodes per tree	N/A	100	100
Regularization strength	10^{-4}	10^{-4}	10^{-4}
Learning rate	0.05	0.05	0.05
Word vectors learning rate	0.1	0.1	0.1
Model parameters	200k	250k	100M

Table 2: Test set accuracies.

Model	Accuracy
Tree-LSTM	0.81
Encoder-Decoder	0.76
Attention	0.77

6 Conclusions and Future Work

The experiments suggest that the results for the Attention-based model were slightly lower than the state of the art results obtained with the Tree-LSTM. They also show that the Attention-based model performed better than the context-based encoder-decoder model. The fact that the attention model performed better than the simple encoder decoder was quite predictable, since it is designed to work with sequences of varying length and it represents the temporal nature in a better manner.

Understanding why the results for Attention were slightly less than a plain tree structure is far more interesting and is the primary purpose of this paper. This can be attributed to multiple factors:

1. Attention-based models typically take more time to train, and with our limited amount of resources we were able to run all models for just one epoch. Perhaps with more training time the Attention-based model could give better results. These difference in training time can be quite substantial given that the Attention-based model incorporates a lot more parameters that need to be trained. In comparison to the original Tree-LSTM model, the extra parameters are created by the decoder and the linear attention layer.

2. The Attention-based Model might not be yielding state of the art results because there might be little or no dependence between phrases adjacent to each other that are not on the same path while traversing the tree. For example, lets consider the sentence *Marie had a good time*, and lets assume that the phrases *Marie had* and *a good time* are at the same level in the tree. In this case, the sentiment that the first phrase conveys would have little influence or dependence on the sentiment of the second.

3. It might be the case that the dependency parse trees used in this paper are not ideal for the task at hand. It would be interesting to explore other type of trees that could help the Attention-based model perform better.

4. Finally, due to time limitations we were not able to tune the hyperparameters of the model appropriately. Ideally, more sophisticated techniques like grid search or random search could be used to better tune the parameters of the Attention-based model.

In terms of future work, a better evaluation must be done. This includes a framework for searching over suitable hyperparameters, as well as training all models with appropriate early stopping. A major assumption of this entire exercise was that the parse trees were available for the model to use, but in a real life situation this is very unlikely to occur. Hence, it would be interesting to approach this problem by using latent tree learning models such as SPINN (Samuel R. Bowman, 2016), Reinforce (Dani Yogatama, 2016) and Chart Parsing to learn how to parse with a supervised sentiment objective in mind. This approach would help to determine if learning the tree parses using the sentiment objective improves the tree structure.

7 Our Code Contributions

To implement this paper, we built upon the existing code base from Tai (2017). The following were our code contributions:

- Evaluation code for the Tree-LSTM in the original paper for phrase level classification. The code base as well as the results for the original paper only talked about the results at the sentence level, although the model provided the parameters to output sentiments at each phrase node.

- Lua code written using the Torch framework for implementation of the encoder-decoder and Attention-based models. Both forward propagation and backward propagation were coded from scratch. (The original Tree-LSTM code was used as an initial approximation, but significant changes had to be made).

- Evaluation code for the encoder-decoder and Attention-based models. Since these models presented substantial differences from the original Tree-LSTM, the code for evaluation had to be different as well.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014. arXiv preprint arXiv:1409.0473.

Chris Dyer Edward Grefenstette Wang Ling Dani Yogatama, Phil Blunsom. Learning to compose words into sentences with reinforcement learning, 2016. URL https://arxiv.org/pdf/1611.09100.pdf.

Yoon Kim. Convolutional neural networks for sentence classification, 2014. arXiv preprint arXiv:1408.5882.

Bo Pang and Lillian Lee. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, 2:1–135, 2008. URL http://www.cs.cornell.edu/home/llee/omsa/omsa-published.pdf.

Le Quoc and Tomas Mikolov. Distributed representations of sentences and documents. *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014. URL https://cs.stanford.edu/~quocle/paragraph_vector.pdf.

Abhinav Rastogi Raghav Gupta Christopher D. Manning Christopher Potts Samuel R. Bowman, Jon Gauthier. A Fast Unified Model for Parsing and Sentence Understanding, 2016. URL https://arxiv.org/pdf/1603.06021.pdf.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank, 2013. URL https://nlp.stanford.edu/~socherr/EMNLP2013_RNTN.pdf.

Kai Sheng Tai. Tree-Structured Long Short-Term Memory Networks, 2017. URL https://github.com/stanfordnlp/treelstm.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks, 2015. arXiv preprint arXiv:1503.00075.