Exploring Tree-LSTM variations for classification of sentence sentiment

Vasileios Charatsidis
Amsterdam University (UvA)
Data Scientist @ Altran
Geldropseweg 9L, Eindhoven
charatsidisvasileios@gmail.com

Tycho Grouwstra
Amsterdam University (UvA)
Data Engineer @ Unilever Food Solutions
Maria van der Duinstraat 8, Poortugaal, Holland
tychogrouwstra@gmail.com

Abstract

In this paper we explore different versions of Tree-based Long Short-Term Memory (Tree-LSTM), which is a generalization of LSTMs to tree-structured network topologies.

We evaluate their efficacy on classification of sentence sentiment, a benchmark task in Natural Language Processing (NLP).

We compare them to several existing methods including (BOW), Continuous Bag of Words (CBOW), Deep CBOW, pre-trained word embeddings, LSTM, mini-batch LSTM, GRU, and binary Tree-LSTM.

1 Introduction

Problem Description

Traditional techniques like RNN and LSTM process information in a linear chain. However, natural language exhibits syntactic properties that would naturally combine words to phrases.

Tree-structured models are a linguistically attractive option due to their relation to syntactic interpretations of sentence structure, we build on these ideas by trying different architectures of Tree-LSTM.

As a way to test this, we have chosen to use a task in Natural Language Processing (NLP), as language is perhaps the structured data type for which data sets are most prevalent.

Specifically, we evaluate our algorithm on sentence-level sentiment classification, a common task in NLP used to gauge public opinion in e.g. reviews of consumer products and services or for stock evaluation. (Feldman, 2013)

The goal in sentiment classification is to learn for any given document whether it conveys positive or negative sentiment. This is challenging due to the complexity of natural language — while at the surface some words may carry an evident individual sentiment, in practice automating such analysis has to deal with issues such as negation, modification of meaning through e.g. adverbs (syntax), domain-specific meaning (disambiguation), sarcasm, context, as well as having to deal with vocabulary limitations (unseen words).

There have been various approaches to tackling such classification tasks over time, ranging from simple models (using 'dictionaries' ascribing a sentiment value to each word) to more complex ones combining syntactical data (binary trees) with semantic vectors such as GloVe (Pennington et al., 2014) and word2vec (Mikolov et al., 2013b). These involve trade-offs such as simplicity of implementation, availability of data, and computational requirements of model training (time, compute resources).

Research Questions and Goals

We will compare prediction quality and variance on the commonly used Stanford Sentiment Treebank (Socher et al., 2013), which holds sentences annotated on both sentence and node levels with a sentiment out of 'very negative', 'negative', 'neutral', 'positive', or 'very positive'. This means our baseline random prediction rate will be 20%.

To provide a broader overview, we discuss the following candidate features, each of which we hypothesize will contribute to a higher classification rate.

- sentiment associated with separate words using the Continuous Bag of Words model (Mikolov et al., 2013a);
- semantic vector models such as GloVe (Pennington et al., 2014) and word2vec (Mikolov et al., 2013b);

- word order using Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (Chung et al., 2014);
- sentence length, where longer sentences potentially seem easier to classify due to their higher amount of information, particularly for the models using sentence structure.
- sentence structure, using a syntactic binary tree model (Tai et al., 2015) (Le and Zuidema, 2015) (Zhu et al., 2015);

Summary of Findings

Our results turned out mixed, with 3 hypotheses accepted (bag of words, semantic embeddings, word order), and 3 yielding inconclusive results (sentence structure, sentence length).

2 Background

All of the approaches explored use word embeddings. Word embeddings (Firth, 1957) are a NLP technique from distributional semantics used to attribute features to words. The idea is that embeddings can be learned from the context a word is used in (notably what words tend to surround it), which can then be used in tasks including sentiment classification.

Popular embedding methods include:

- Google's word2vec model (Mikolov et al., 2013b), predicts words by context;
- Stanford's GloVe (Pennington et al., 2014) model, based on co-occurrence counts.

For all our models that use pre-trained embeddings we decided to choose word2vec.

We will now briefly outline the approaches evaluated.

3 Models

Bag-of-words (BOW)

The bag of words model (Harris, 1954) simplifies the problem of understanding natural language by focusing only on the separate words in a document — that is to say, ignoring word order and therefore sentence structure.

Under this approach, one ascribes values to each separate word, so as to evaluate a document based on the total values of its constituent words.

In the simple case, each word might be assigned just a scalar value summarizing e.g. its sentiment value, though other data sets like our Sentiment Treebank (used here) instead assign a vector containing one value for each predicted output class.

We implement the BOW model through a simple neural network (McCulloch and Pitts, 1943), using the Treebank annotated word vectors as inputs. Our implementation starts with randomly initialized weights, stochastic gradient descent (Robbins and Monro, 1951).

Continuous bag-of-words (CBOW)

Continuous bag-of-words (Mikolov et al., 2013a) is an extension to the above model that allows one to train on an intermediate representation of higher dimensionality before having to predict the probability for different output classes, allowing us to utilize a broader understanding of the words. Here we learn a linear mapping from the embedding to the output classes. For our CBOW implementation we used a word embedding vector with size 300.

Deep continuous bag-of-words (Deep CBOW)

Deep continuous bag-of-words (Deep CBOW) add to the above by using deep neural networks (Ivakhnenko and Lapa, 1973) that add hidden layers to allow learning non-linear mappings between the embeddings and output classes. For our Deep CBOW implementation we used a word embedding vector with size 300.Our Deep CBOW uses a single hidden layer consisting of 100 units, with tanh activation functions.

Long Short-Term Memory (LSTM)

Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) is a technique extending Recurrent Neural Networks (Elman, 1990) capable of learning long-term dependencies.

LSTMs are explicitly designed to avoid the long-term dependency problem with the ability to remove or add information to the cell state, carefully regulated by structures called gates. Gates are a way to optionally let information through.

We use these to expand on the BOW approaches above so as to take into account word order. This extra info makes them more powerful at the cost of longer training times.

Mini-batched LSTM

The mini-batch (Hinton et al., 2012) version of LSTM is similar, but updates based on small batches of data points, allowing it to average their gradients to gain a smoother convergence path.

The parallel nature of this makes the approach amenable to optimization using parallel linear algebra libraries, such as BLAS (HANSON et al., 1977) for CPU and CUBLAS (Nvidia, 2008) for GPU.

Gated Recurrent Unit (GRU)

Gated Recurrent Unit(Chung et al., 2014) combines the forget and input gates into a single 'update gate'. It also merges the cell state and hidden state. The resulting model is simpler than standard LSTM models, and as such it has been growing increasingly popular.

Tree-LSTM

N-ary Tree-LSTMs (Tai et al., 2015) (Le and Zuidema, 2015) (Zhu et al., 2015) generalize LSTMs from linear chain topologies to tree-structured network topologies, allowing to improve prediction for tasks involving tree-structured input or output. As our sentiment dataset uses binary syntactic trees, our case therefore requires binary Tree-LSTMs.

Tree-LSTM variations

The LSTM cell, that Tree-LSTM uses, has a sigmoid layer called the input gate layer which decides which values are going to be updated. Next, a tanh layer creates a vector of new candidate values, C, that could be added to the state. Inspired by the simplifications that work well in a GRU we explored 6 different variations. We changed the cell that receives x but also cell that receive children.

- Tree-LSTM with the variations below in the cell that receives input x as shown in Figure 1.
  - original cell.
  - Tree-LSTM without the input gate.
  - Tree-LSTM without the tanh layer.
- simplified Tree-LSTM (Tree-sLSTM) where the Tree-sLSTM Cell has only 2 forget gates

$$f_{t-right} = \sigma(W_{fr} \cdot [hr_{t-1}] + b_r)$$
$$f_{t-left} = \sigma(W_{fl} \cdot [hl_{t-1}] + b_l)$$
$$c = (f_l * hl_{t-1} + b_l) + (f_r * hl_{t-1} + b_r)$$
$$h = c$$

Where $f_{t-right}$ and f_{t-left} are the forget gates of left and right children respectively, c is the cell state and h the hidden state (which in this case they are the same). σ is the sigmoid function and b_l and b_r are the biases.

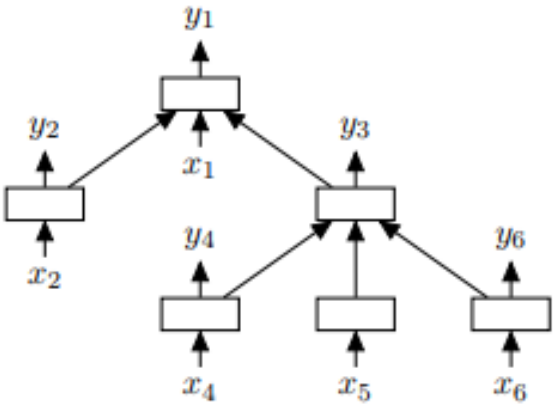

Figure 1: Tree-LSTM structure.

We explored three different variations on it, as the original:

- original cell.
- Tree-sLSTM without the input gate.
- Tree-sLSTM without the tanh layer.

4 Experiments

All our implementations were done in Python using PyTorch (Paszke et al., 2017), trained using cross-entropy loss on the Sentiment Treebank's sentence labels, using argmax to pick the output label for the prediction that has the highest likelihood.

Models used a learned bias vector plus the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.0005. The learning was disabled as it used the word2vec pre-trained model.

Models were normally trained for $30,000$ iterations, other than the slower non-batched LSTM, which was trained for $25,000$ iterations and the mini-batch 32 methods (mini-batch LSTM, GRU, Tree-LSTM) that are trained only for $12,000$ due to excessive training times. It is worth noticing though that the best working model, in all mini-batched methods, was produced before the 8000th iteration since the models after it were overfitting. We implemented our Tree-LSTM using dropout (Srivastava et al., 2014) with p=0.5. We parse the Treebank syntactic trees to binary shift-reduce vectors.

In this experiment we compare the average accuracy over 3 runs of all methods in the task of classification of sentence sentiment.

5 Results and Analysis

The results are visualized in the graph of Figure 2. The table below displays the average accuracy in the test set of the methods over 3 runs in ascending order.

Models	Avg accuracy
BOW	$0.265 \pm 4.31e{-6}$
CBOW	$0.317 \pm 6.15e{-6}$
Deep CBOW	$0.376 \pm 8.65e{-5}$
Pretrained Deep CBOW	$0.430 \pm 2.50e{-5}$
LSTM	$0.442 \pm 5.36e{-6}$
mini-batch 32 GRU	$0.444 \pm 4.12e{-6}$
mini-batch 32 LSTM	$0.451 \pm 4.28e{-5}$
Tree-sLSTM, no tanh layer	$0.452 \pm 7.63e{-6}$
GRU	$0.453 \pm 7.04e{-5}$
Tree-sLSTM	$0.453 \pm 1.71e{-4}$
Tree-LSTM, no input gate	$0.458 \pm 1.32e{-5}$
Tree-LSTM, no tanh layer	$0.465 \pm 2.61e{-5}$
Tree-LSTM	$0.465 \pm 2.78e{-5}$

- Tree-LSTM performed best as expected.
- Remarkably tanh layer seems to not be so important. In both main variations of Tree-LSTM, original and simplified, their performance is almost the same with their corresponding variation without the tanh layer, 0.465 and 0.453 respectively.
- For Tree-LSTM it seems that the existence of input gate is more important since the Tree-LSTM without input gate performs 0.7% worse than the original, 0.458 and 0.465 respectively.
- Surprisingly, the GRU performed very well surpassed only from the Tree-LSTM versions. Counter-intuitively, it surpasses also it's mini-batch version which does not make sense and might be the product of variance.
- Finally, as expected the mini-batch LSTM outperforms the non batched LSTM.

Considering the training times, GRU was roughly 17% faster to train than LSTM and the Tree-sLSTM was 50% faster to train than the original Tree-LSTM.

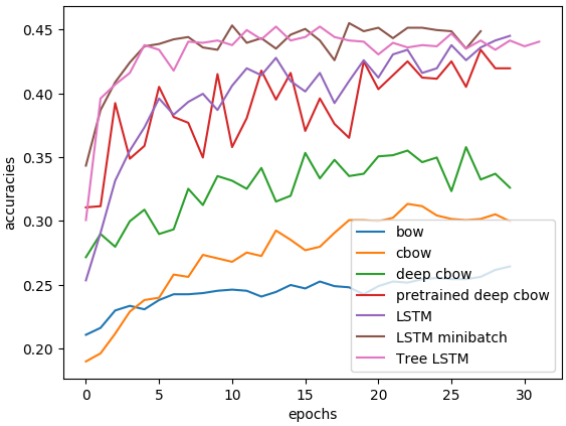

Figure 2: accuracies across models

To facilitate reproducible research we have published our source code on Github [1]

6 Conclusion

Our results turned out mixed, with 3 hypotheses accepted (bag of words, semantic embeddings, word order), and 2 yielding inconclusive results (sentence structure, sentence length).

Bag-of-words and semantic embeddings unsurprisingly both gave accuracies higher than their baselines (random for BOW).

Word order turned out quite important as well. Models utilizing order outperformed those without it in almost all cases — only our vanilla LSTM took a few epochs to catch up to the performance of the pre-trained bag-of-words model.

Sentence structure started out as a useful predictor, with an accuracy in early rounds of already 38%, but mini-batch LSTM quickly caught up to TreeLSTM, after which their scores essentially converged.

For sentence length it was hard to discern any clear pattern, as models generally had a variance that appeared higher than any trends in performance. The more sophisticated models generally performed well on both shorter and longer sentences. As a result, our premise of longer sentences yielding higher accuracy (particularly on models using sentence structure) remains inconclusive.

In conclusion, we make a few suggestions for follow-up research:

[1] https://github.com/VCharatsidis/BOW-LSTM-TreeLSTM/blob/master/BOW-LSTM-TreeLSTM.ipynb

- further investigate the effects of sentence length on sentiment classification accuracy on a larger dataset, so as to limit the effects of variance.
- ...

References

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science* 14(2):179–211.

Ronen Feldman. 2013. Techniques and applications for sentiment analysis. *Communications of the ACM* 56(4):82–89.

John R Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis* .

RJ HANSON, CL LAWSON, DR Kincaid, and FT Krogh. 1977. Basic linear algebra subprograms for fortran usage-an extended report. In *Sandia Technical Report SAND 77-0898*, Sandia Laboratories Albuquerque, NM.

Zellig S. Harris. 1954. Distributional structure. *¡i¿WORD¡/i¿* 10(2-3):146–162. https://doi.org/10.1080/00437956.1954.11659520.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. 2012. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on* page 14.

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

A.G. Ivakhnenko and V.G. Lapa. 1973. *Cybernetic Predicting Devices*. Jprs report. CCM Information Corporation. https://books.google.be/books?id=FhwVNQAACAAJ.

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* .

Warren S. McCulloch and Walter Pitts. 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* 5(4):115–133. https://doi.org/10.1007/BF02478259.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

CUDA Nvidia. 2008. Cublas library. *NVIDIA Corporation, Santa Clara, California* 15(27):31.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

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.

Herbert Robbins and S Monro. 1951. [a]a stochastic approximation method, [o] annals math. *Statistics* 22:400–407.

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.

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.

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

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