Sentiment Analysis as a Supervised Learning Problem with Neural Models

Ioannis Gatopoulos (12141666) ioannis.gatopoulos@student.uva.nl
Manasa J Bhat (12184306) jbmanasa@hotmail.com

1 Introduction

Natural Language Processing(NLP) uses models like Bag-of-Words(BOW) which ignore the structure of sentences and models like Recurrent Neural Networks(RNN) which are highly sensitive to the sequence of words. In this paper, we study the performance of such models in sentiment analysis as a supervised learning problem for classifying movie reviews. One of the the most difficult tasks in language processing problems is, to train a model to understand the underlying semantic and syntactic structures of language, to capture the context and meaning. To do this, we explored different methods to exploit the information we can extract from each word/sentence. Therefore, we worked with various neural models to encode sentences and used these sentence representations to classify sentiments.

One hypothesis on sentiment analysis was that, since word order is an indistinguishable component of any language, order-less models would be dominated by sequence models. Hence we started the study by exploring three variants of the Bag-of-Words(BoW) model, all which disregard word-order information. We compared performance of these models against Long Short-term Memory(LSTM) models and Tree LSTM both which comply with word-order. Tree LSTMs performed the best proving the hypothesis true. We also analyzed how using tree structures of reviews improve performance.

We observed that the lengths of review sentences in our training data varied substantially. This motivated us to study the correlation of the sentence lengths with respect to performance. LSTMs, especially Tree LSTM again performed better than BOW models for all sentence lengths.

All the above experiments ignored fine-grained sentiment of each review. A natural question then was, what happens if we supervise sentiment of each word and phrase of a review? The accuracy of training data improved for all models, but the model performed poorly on validation and test data indicating over-fitting.

Tree LSTMs are the prime choice to capture language structure. But they still did not outperform other models significantly. We investigated the type of language usage where this model failed and found it to be common in reviews that were sarcastic in nature.

There is a huge difference between BOW models and LSTM models with respect to time taken to converge. We wanted to know which model is most time efficient without compromising on the performance. We found that Mini-batch LSTMs balanced between these two criteria the best.

2 Background

Bag-of-Words(BoW) models with word embedding feature-learning technique can be characterized as simple and flexible, and can be used in a myriad of ways for extracting features from documents. They take advantage of fully connected multi-layer Neural Networks (MLN) to learn features of words. Popular variances of BoW are the Continuous BoW (CBoW) (Imran Sheikh), a more flexible model in terms of the word embedding size, that allows learning more feature of words, and Deep CBoW, which is a deep learning method that employs multiple processing layers in order to learn more complex functions and capture more aspects of data.

However, it is obvious that as the vocabulary size increases, so does the vector representation of documents, leading to sparse vectors. A sophisticated way to control the size of the word representation is to limit it to a fixed size vector, an embedding. Word embedings exhibit some desired properties like: i) Words with similar meaning tend to be closer in the embedding space than words that are different ii) If two pairs of words have similar differences in their meanings, they should be approximately equally separated in the embedding space. Thus, these embeddings have proven to be efficient in capturing context similarity, analogies and due to its smaller dimensionality, are fast and efficient in processing core NLP tasks. One very well known method for pre-trained word-embeddings is GloVE. (Pennington and Manning, 2014).

Although semantic word spaces are very useful, they cannot express the meaning of longer phrases correctly. This issue can be addressed using LSTMs (Hochreiter and Schmidhuber, 1997), which are a special kind of RNN, capable of learning long-term dependencies. The main idea of the LSTM architecture is to maintain a memory of all inputs that the hidden layer received over time. This is done by adding up all (gated) inputs to the hidden layer through time to a memory cell. This way, errors propagated back through time do not vanish and even inputs received at previous time steps can play a role in computing the output of the network. Mathematically, the LSTM transition functions are as follows:

$$\begin{aligned}
i_t &= (W(i)x_t + U(i)h_{t1} + b(i), \\
f_t &= (W(f)x_t + U(f)h_{t1} + b(f), \\
o_t &= (W(o)x_t + U(o)h_{t1} + b(o), \\
u_t &= tanh(W(u)x_t + U(u)h_{t1} + b(u), \\
c_t &= i_t \odot u_t + f_t \odot c_{t1}, \\
h_t &= ut \odot tanh(c_t)
\end{aligned}$$

where i_t , f_t , o_t , c_t are the input gate, forget gate, output gate and the memory cell respectively, as the $\odot$ refers to element-wise multiplication.

Despite being known for capturing structural information implicitly, in practice LSTM sometimes lacks the claimed power in language processing. This is because of their linear chain type of structure which contrasts with the syntactic properties that the natural language exhibits; it naturally combines words to phrases. To solve this issue, the scientific community proposed Tree-LSTMs(Kai Sheng Tai, 2015a), a generalization of LSTMs to tree-structured network topology. At any given time step t, Tree LSTM is capable of composing its states from an input vector and hidden states of its child-nodes simultaneously. This is unlike the standard LSTM that assumes single child per unit. Moreover, it maintains a forget gate separately for each child node. This characteristic enables the Tree-LSTM to be able to aggregate information from each child node.

3 Models

In our implementation, the BoW model is a simple fully connected NN, where every word is represented as a five length vector. We sum up all the word-vectors of a review, resulting in a new vector of size five. This vector is mapped to the 5 sentiment classes and the maximum element corresponds to our prediction. We extend this approach by increasing word embedding size to 300. This is then linearly mapped to 5 values corresponding to each sentiment. In our efforts to make the model learn more complex functions, we made it deep (Deep CBoW) by adding 2 hidden layers of 100 nodes. The final layer is however a softmax yielding a probability over the sentiment classes. pt Deep CBoW model has the Deep CBoW model framework, but using GloVe embedding as a look-up table.

On LSTM model, GloVe word embedding of size 300 nodes are used, and an rnn layer that maps 300 nodes to 168. To prevent overfitting, we used dropout with probability 0.5 and to gain some speed, we used mini-batches of size 25.

The Tree LSTM architecture was recreated from Tai et als work on N-ary Tree-LSTMs. Every node in this model has two children, except for pre-terminal nodes. We defined a class(TreeLSTM) to encode a complete sentence and return the root and a second class(TreeLSTMCell) to return a new state when provided with two children.

4 Experiments

4.1 Dataset

We evaluate our models architectures on sentiment classification task of sentences sampled from movie reviews. We used the Stanford Sentiment Treebank (Richard Socher and Potts) which consists of 5-way fine-grained sentiment labels (very negative, negative, neutral, positive, very positive) for 215,154 phrases of 11,855 sentences. The standard splitting: 8544 sentences for training,

Model I	**Model II**	*p-values*
BoW	CBoW	0.028*
CBoW	Deep CBoW	0.307
Deep CBoW	pt D.CBoW	4.152e-7*
pt D.CBoW	LSTM	0.197
LSTM	mb LSTM	0.144
mb LSTM	Tree LSTM	0.759
Tree LSTM	Sub-Tree	0.906

Table 1: Various p-values, provided from sign test (the * indicates that the difference in performance is significant).

1101 for development, and 2210 for testing. The average sentence length is 19.1. The evaluation metric is the accuracy, given by $\frac{correct}{total} * 100$.

Standard binarized constituency parse trees are provided for each sentence in the dataset. Each node in a tree is annotated with a sentiment label.

4.2 Training Details

As mentioned before, we minimize the cross-entropy error. For all phrases, the error is calculated as a regularized sum:

$$E(\theta) = \sum_i \sum_j t_i^j \log y_j^{sen_i} + \lambda ||\theta||_2^2$$

where $y^{sen_i} \in R^{c\times1}$ is predicted distribution and $t_i \in R^{c\times1}$ the target distribution. c is the number of classes or categories, and $t_i \in c$ denotes the j-th element of the multinomial target distribution; i iterates over nodes, θ are model parameters, and λ is a regularization parameter.

All the models except Tree-LSTM and mini-batch LSTMs (with batch size of 25), implement Stochastic Gradient Descent (SGD). We used Adam for learning rate hyper-parameter optimization. An initial learning rate of 0.0005 is used for all BOW models and 0.0003 for vanilla LSTM and 0.0002 for other LSTM versions. We train every model until they start to overfit.

4.3 Statistics

In order to obtain statistical significance in our comparisons we used sign test on the results. Statistical significance is indicated using standard conventions: n.s.d. (non-significant): $p > 0.05$. Accuracy was determined as mean over 3 runs with different seeds. Table 1 shows the p-value for different model comparisons.

5 Results and Analysis

5.1 Word order

Looking at the performance results of models in Table 2, it is clear that the BoW classifiers were outperformed by the Sequential models, which indicates that word order definitely matters for sentiment analysis.

But we observed that increasing the word-embedding size using CBOW and then adding more layers to the model using Deep CBOW improved the performance by 4%. This improvement is because larger word-embedding size allows the neural network to capture more features

Model / Data	**Test Data**	**Iterations**	**Time**
Measure	%($\pm$ std)	$*10^3$	sec
BoW	33.3 (1.8)	174 (16)	432 (38)
CBoW	35.1 (1.7)	39 (6)	230 (61)
Deep CBoW	37.1 (0.2)	20 (3)	154 (26)
pt DCBoW	44.8 (1.3)	33 (3.4)	181 (19)
LSTM	42.9 (0.4)	5 (0.4)	700 (65)
mb LSTM	46.4 (1.1)	39 (0.5)	353 (54)
Tree LSTM	46.9 (0.9)	4 (0.5)	1132(139)
Sub-Trees	47.1 (0.1)	5 (0.1)	1633 (98)

Table 2: The mean and the standard deviation (in %)of accuracy from the best models in total 3 runs. Iterations and time show the number of iterations and seconds to find the best model respectively.

of a word in its embedding and adding layers to the network helps learn more complex function to better classify our data. This result suggests that order-less models can also perform well if we can tune the model to learn more aspects of data. This motivated us to use the GloVe pre-trained embeddings, where similar words are already trained to be close to each other, rather than training our embeddings solely based on the prediction error. This resulted in a significant performance boost, even better than vanilla LSTM even though the model still did not consider word-order.

The use of mini-batches to the LSTM model boosted performance by 3.5%. Even though Tree LSTMs performed the best among all the models, it proved to to be an insignificant leap from the mini-batch LSTM.

5.2 Fine-grain sentiment supervision

To supervise sentiment of every phrase and word of a review, we divided the review tree into sub-trees and used them to train the model. The size of train data set was now 318582. For BOW models, the accuracy of the models on training data increased exceptionally(70%), the accuracy on the validation and test set did not vary indicating overfitting. For Tree LSTM, the performance increased slightly. Supervising fine-grain sentiment therefore did not improve the performance very significantly. This maybe due to the fact that we are just adding data with different sentiments, but the sequence of the data is still the same for all the sub-trees. Hence the LSTMs do not actually have new substantial data to learn better.

5.3 Tree Strucuture

Natural language exhibits tree like syntactic structure. Tree LSTMs use this tree structure to learn

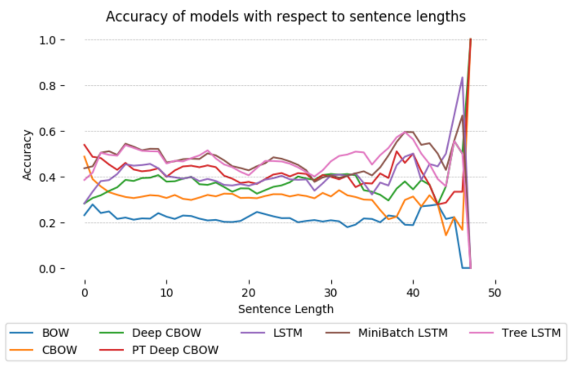

Figure 1: Accuracy of models with respect to sentence lengths

the review structure better. At any given point, tree LSTM composes its state from an input vector and hidden states of its child-nodes simultaneously. Therefore Tree LSTMs perform better than all other models.

5.4 Sentence length

To compare model performance based on sentence lengths, for every sentence length l in the test data, we evaluated accuracy for sentences with length in the window $[l-2, l+2]$ for each of our models(Kai Sheng Tai, 2015b). For this investigation, we found it best to compare performances only for sentences with length ranging from 10 to 40 because the test data was not uniformly distributed for all sentence lengths. We observed that LSTM models performed better on longer sentences than BOW as expected due to their ability to learn long-term dependencies in sentences as seen in Figure 1. For sentences above length 40, we can see from the graph that the accuracy is very inconsistent for every model due to performance skew resulting from very less test data.

5.5 When do the Tree models fail?

Below are some representative examples of reviews that that the tree models misclassified.

In its ragged, cheap and unassuming way, the movie works. (Type (i); Prediction: 1, Labeled: 3)

A gentle, compassionate drama about grief and healing. (Type (ii); Prediction: 4, Labeled: 3)

It's a great deal of sizzle and very little steak . (Type (i); Prediction: 3, Labeled: 1)

Categorizing our findings, we concluded that there are two prominent cases when Tree-Structure models misclassify reviews: i) When the reviews are written slackly, poetically or sarcastically. ii) When the model cannot classify between two very close sentiments like Negative and Very Negative. Former is a well known issue in language processing. The model needs to be trained to find the differentiating features for sarcastic and non-sarcastic text to handle this. In the second case, even humans find it hard to classify a review into two very similar classes. We believe, this type of misclassifications can be reduced by adding more reviews with closely related sentiment to the train data.

5.6 Convergence

Table 2 shows the mean iterations and the time that the models need to converge to the best performance on the validation data. After this time, the models tend to overfit. The BOW models converge faster than other models with respect to time and our analysis is that Deep CBOW with pre-trained word embeddings is the best time-efficient model. Though tree LSTM performs the best, it takes the longest time to converge. Mini-batch LSTMs on the other hand strike a balance between performance and convergence time and therefore we conclude it to be the ideal classifier among all the models we evaluated.

6 Conclusion

We performed sentiment analysis on movie reviews as a supervised learning problem. The result of the experiments confirmed the literature that sentiment classifier models which take word-order into account do perform better than models that ignore word sequence. Surprisingly, the outcome indicated that the LSTM performs almost equally well as the Deep CBoW model with pre-trained embeddings. We believe that the reason behind this is that the database we worked with was not diverse and convoluted enough to unveil the full potential of the LSTM models. Tree LSTMs proved to be the best performing model given the fact that they are capable of handling the underlying syntactic structure of natural language. We believe if the LSTM models were trained with a sarcasm detector method, the performance will be greatly improved and therefore we suggest looking into such methods.

References

Hochreiter and Schmidhuber. 1997. *Long short term memory*.

Dominique Fohr Georges Linare‘s Imran Sheikh, Irina Illina. Learning word importance with the neural bag-of-words model.

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

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

Socher Pennington and Manning. 2014. *GloVe: Global Vectors for Word Representation*.

Jean Y. Wu Jason Chuang Christopher D. Manning Andrew Y. Ng Richard Socher, Alex Perelygin and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank.

[1]

[1]https://colab.research.google.com/gist/jbmanasa/65a9dcdc9640b2c6649c0dd05b498c15/john-nlp1-practical-ii.ipynb