CSCE 636: Final Project Report

Sarra Bounouh, Pei Chen

Abstract

Sentiment analysis, or opinion mining, is the process of inferring the emotional tone behind a stream of text. It is used to understand affective states and subjective information expressed on online social media platforms. In this paper, we use the IMDb dataset to build several deep learning models that predict positive and negative reviews. These models include Long Short Term Memory (LSTM) and attention-based Transformers. Our research findings indicate that combining LSTMs with a Convolutional Neural Network (CNN) accelerates convergence to the best accuracy. We also show that Transformers converge even faster. Additionally, we find that adding dropout layers in our models is an effective method in preventing overfitting. Our best performing LSTM model yields an accuracy of 85.58% and the best attention model results in a test accuracy of 86.12% on the dataset.

1 Introduction

Sentiment analysis, or opinion mining, is the process of inferring the emotional tone behind a stream of text. It is used to understand affective states and subjective information expressed on online social media platforms. In this project, we propose to predict positive and negative reviews using several deep learning models, including Long Short Term Memory (LSTM) (Hochreiter, 1997 [11]) models and relatively recent attention-based Transformer (Vaswani, 2017 [1]) models. We propose to use the IMDb dataset (Andrew, 2011 [3]) to build our models and draw useful insights. Such models could be generalized to predict the sentiment of any social media post for various purposes.

In this paper, we start by describing our motivations then we present our methodology, including details about the dataset used as well as the models settings. Later, we showcase our results then discuss our findings. Lastly, we conclude with a global view on the work achieved so far and the main challenges we encountered.

2 Motivation

The goal of our models is to predict a sentiment from an input sequence of text. In this sense, our project falls at the crossroads between natural language processing, text analysis, and computational linguistics. The main motivation for us working on this problem is that sentiment analysis is especially useful to monitor public opinions widely spread on social media platforms. Moreover, its potential applications are broad and many organizations nowadays tend to feel the need to extract insights from social media. That is, in order to help them adapt their strategies or to predict stock market effects, as shifts in public opinions have proven to correlate with changes in the stock market. For instance, the Obama administration used insights from forum posts and news articles before the elections in 2012 to understand citizens' opinions and adapt their political campaign to public views. Another example is Expedia Canada, who managed to adjust their media advertisements when they observed an increase in negative reactions about their content.

3 Methodology

3.1 Dataset

We use part of the Large-scale Movie Review Dataset, a.k.a IMDb dataset (Andrew[3], 2011). This dataset contains 5k movie reviews (2.5k for training and 2.5k for testing). We use 1/5 of the training data as the development set for hyperparameters tuning and then retrain the model on the whole training dataset for testing.

As described in the introduction, next we will explore and compare the performances of two categories of Natural Language Processing (NLP) deep learning models: LSTMs and Transformers. But first, let's answer the question: why not use a basic Recurrent Neural Network (RNN) model to solve this classification problem?

3.2 Why Not RNN?

Theoretically, for sequential classification problems, we could use a basic RNN (cf. Figure 1) to get the representation of the sequence. However, in practice, RNNs suffer from either exploding gradients or vanishing gradients which makes the training process ineffective after few epochs for long data sequences.

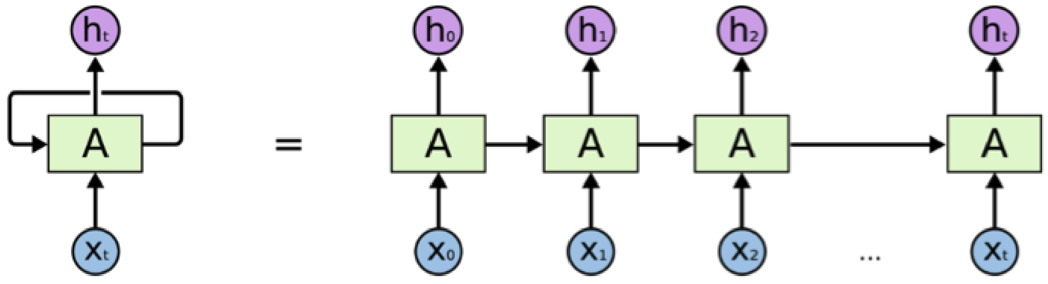

Figure 1: RNN representations - Folded representation (left) and unfolded representation (right)

On the other hand, LSTM models have proven to be resistant to such issues, as explained mathematically in [5]. The building block of LSTM is represented in Figure 2 (modified from [7]). Indeed, thanks to its forget gate activation, LSTMs solve the problem of gradients instability. LSTM create a connection between the forget valve and the gradients computation, that enables information that should not be forgotten, to flow through the forget gate to the next LSTM building block.

3.3 Attention is All We Need!

Although RNNs and LSTMs are naturally compatible with sequential data, sequential computation property also makes them hard to train. Recently [1] proposed a Transformer model with merely attention mechanism to deal with sequences which is easy for parallel computation. As in Figure 3 (left)[1], one block of the Transformer consists of one multi-head attention layer, one fully connected layer, two layer-normalization after activations and two Resnet[8] connections. The details of the multi-head attention layer are on the right side of Figure 3, which concatenates multiple self-attention results as the output. In fact, people do not read texts sequentially, but globally and comprehensively. The non-locality property of Transformers can model this reading processing, hence we explore it in this work for textual sentiment analysis.

3.4 Model Settings: LSTM

After we explained the motivation behind using LSTMs instead of basic RNNs, now we implement this model on our dataset. We use a pre-trained word embedding Global Vectors for Word Representation (GloVe)[9]. We also use a maximum length of sequence of 300 words per review to truncate long texts, then we pad all short inputs with zero values. In this manner, all inputs are of the same size.

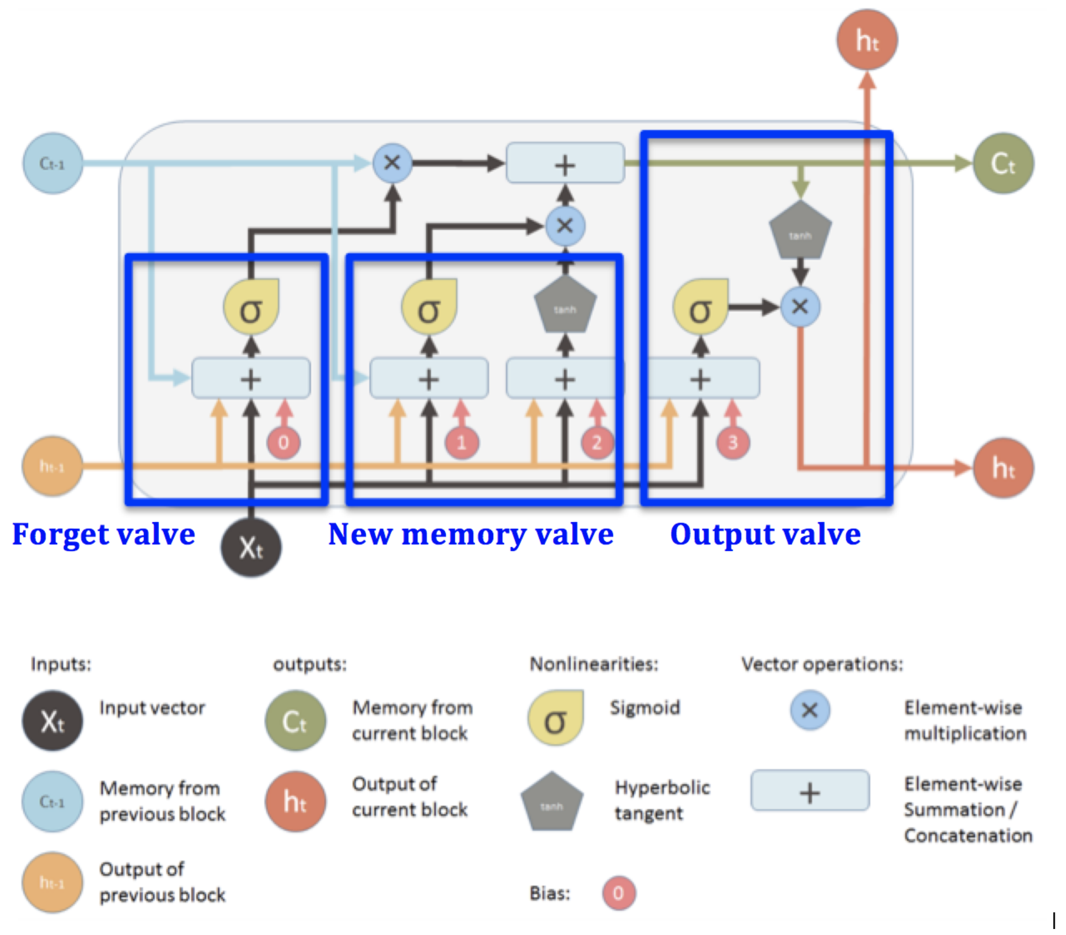

Figure 2: Diagram of LSTM's building bloc

This operation aims to tackle the challenge of using variable-length input data. This operation is also expected to be seamless for the model, as the latter will eventually learn that the zero values do not carry useful information. In this sense, the sequences remain of variable length in terms of content but are vectors of the same length for computational purposes.

In the previous implementation we have included in the midterm project report, we used a single LSTM layer with 100 hidden dimensions and 300 embedding dimensions. We also had used a limited number of epochs for training (max epoch of 20) as our model was quickly overfitting, and we ran the model using only batch sizes of 128 datapoints. This configuration enabled us to get a preliminary test accuracy of 80.54%. To tackle the overfitting issue and further enhance the LSTM model, we modified these configurations and performed more hyperparameters tuning. In the results we are presenting in this paper, we used four different models with the following common settings:

- One single LSTM layer
- Hidden dimensions: 128
- Embedding dimensions: 50
- Max epochs for training: 100

While varying several hyperparameters, we explored the below four models:

- *Model 1:* LSTM with recurrent dropout (i.e. dropout within LSTM hidden dimensions).
- *Model 2:* LSTM with input / output dropout layers (i.e. two dropout layers before and after LSTM hidden dimensions).

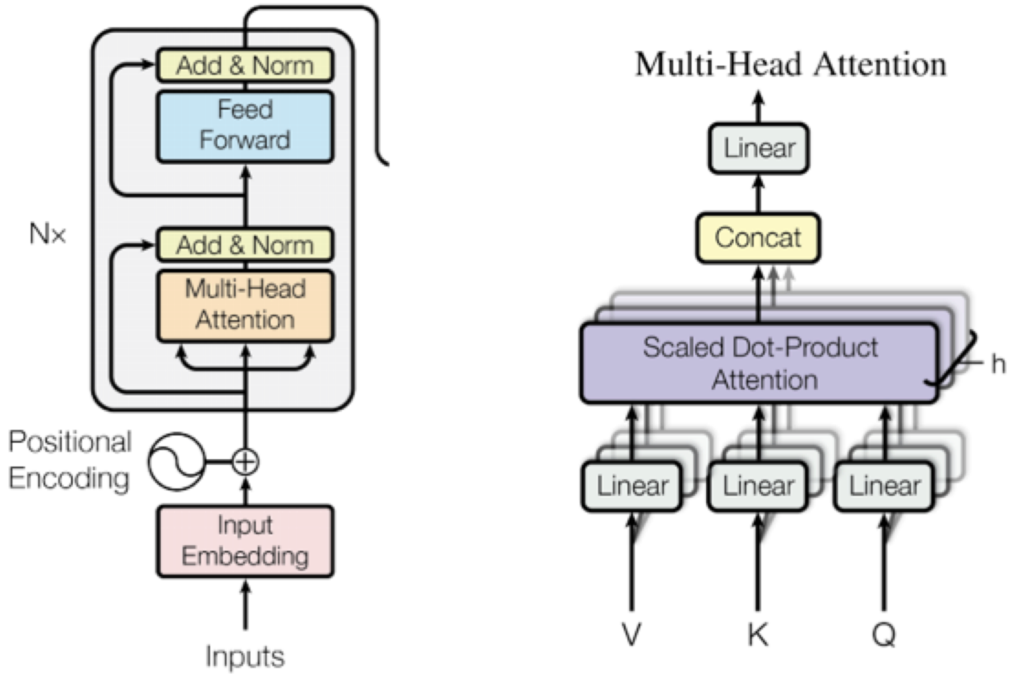

Figure 3: Overall Architecture of the Transformer Encoder (Left) and Multi-Head Attention Layers (Right)

- *Model 3:* LSTM with one-dimensional convolution.
- *Model 4:* LSTM with input / output dropout layers and one-dimensional convolution.

It is important to note that all these models include embedding layers, a linear transformation layer followed by a log softmax layer for binary classification. We also use negative log-likelihood as loss function and Adam optimizer for the backpropagation weights updates. In particular, the embedding layers are used to map each review into a real vector domain (i.e. word embedding). This technique helps encode each word of a movie review as a vector of real values where the meaning similarity between two words is translated by closeness in the real vector space. We present the results of these models in the results section.

3.5 Model Settings: Transformer

As mentioned in section 3.3, we use the Encoder part of the Transformer in the original paper to learn the representation of a text and then feed it into a softmax layer for binary sentiment classification. In our experiments with Transformer, we also use GloVe[9] word embedding as the initial word embedding and choose different hyperparameters such as number of blocks, number of multiple heads, dropout rate[10], batch size etc. to train the model. We then choose the best performance parameters on the validation dataset and retrain the model on the whole training dataset using the selected hyperparameters values . In order to compare with LSTMs, we also set a max epoch of 100, a learning of 0.001, a hidden size of 128, and we use Adam optimizer for all experiments. Our findings are reported in the following results section.

4 Results

These results were obtained by running the aforementioned models on four 2080ti GPUs in parallel to compare their performance. Furthermore, we run all the settings we presented in the previous section on 100 epochs to choose the best performing parameters before retraining and testing.

4.1 Findings: LSTM Model

We present in Table 1 the detailed results of our LSTM models as described in section 3.4.

Model	Batch	lr	hidden	Dropout	Kernel	Best Epoch	Valid Acc (%)	Retrained Test Acc (%)
1	128	0.001	128	0.2	-	80	74.8	82.92
1	64	0.001	128	0.2	-	98	79.1	83.82
2	64	0.001	128	0.2	-	90	80.6	**85.58**
3	64	0.001	128	0.2	7	45	79.1	83.7
3	128	0.001	128	0.2	7	36	74.5	81.24
4	64	0.001	128	0.2	7	84	79.4	83.66
4	64	0.001	128	0.6	7	68	80.6	**85.46**

Table 1: LSTM models results

Using the previously described architecture (cf. section 3.4) and by varying several hyperparameters, the best performing is *Model 2* (LSTM with input and output dropout layers) with a batch size of 64, a learning rate of 0.001 and a dropout rate of 0.2. This model achieves the best test accuracy of **85.58%** in epoch 90, which is better than the accuracy achieved in the midterm report implementation (80.54%). It is worth noting that the second best model follows closed the accuracy of the best performing model, with a test accuracy of **85.46%** at epoch 68. It is the *Model 4*, which combines LSTM with input / output dropout layers and a one-dimensional convolution. For this model, we selected the following hyperparameters: batch size of 64, learning rate of 0.001, dropout rate of 0.6 and kernel size of 7.

4.2 Findings: Transformer Model

We present in Table 2 the detailed results of our Transformer model as described in section 3.5. We achieve the best test performance of **86.12%**, which beats the best setting of LSTMs. We also note that Transformers converge much more easily: they all take approximately 10 epochs to reach their final accuracy, which is more efficient than LSTMs. Nevertheless, this model is also subject to overfitting and with a dropout rate of 0.6 it achieves a 100% accuracy on validation data but fails on the test set (83.78%). So we set a larger dropout rate for the models and find that 0.8 is the best setting. Moreover, we also find that a larger batch size is better for this model as well as for this dataset, thus we set it as 256. Also, we find that 2 blocks with 4 multiple self-attention heads are best for this dataset. In fact, more complex models with additional blocks and extra heads performed worse, and may be constrained by the data scale. Indeed, 2.5k samples may not be enough to train well larger models.

Batch	lr	Hidden	Blocks	Heads	Dropout	Best Epoch	Valid Acc (%)	Retrained Test Acc (%)
64	0.001	128	2	4	0.6	13	100	83.78
64	0.001	128	2	4	0.8	5	83.2	84.32
128	0.001	128	2	4	0.8	6	82.5	85.4
256	0.001	128	2	4	0.8	9	81.50	**86.12**
256	0.001	128	2	8	0.8	9	83.10	84.76
256	0.001	128	4	4	0.8	10	82.70	85.40
256	0.001	128	2	4	0.9	10	82.60	85.50
256	0.001	128	2	4	0.95	10	82.20	84.26

Table 2: Transformer models results

5 Discussion

In a sequence classification task, the challenges include dealing with: (a) sequences of inputs with varying lengths, (b) wide vocabulary, (c) different contexts conferring different meanings. We have discussed how RNNs are not an effective solution to perform this task on movie reviews data. Using truncation and padding we could overtake challenge (a) by making all inputs of the same size (yet varying size in terms of content). We also utilized GloVe word embedding to deal with a large vocabulary (challenge (b)). And finally, we employed LSTMs and attention-based models to take into account local or global contexts (challenge (c)).

Both best performing models (*Model 2* and *Model 4*) tackle the previously observed overfitting phenomenon by including regularization steps in the model. This enabled us to also increase the number of training epochs without degrading the model's performance. Moreover, we noticed that the regularization method that worked best with our data is the input / output dropout. The recurrent dropout on the other hand yielded a slightly inferior performance. Since we observed such result, we chose to use the input / output dropout layers in *Model 4*, rather than the recurrent dropout, to obtain a model that combines both a regularization method (input / output dropout) and a spatial feature extractor (CNN). Additionally, although *Model 2* performs better than *Model 4*, the latter might be actually better in practice than the former. That is because we see an advantage in adding the convolution as it accelerates the convergence of the model (*Model 2* achieves its best performance in epoch 90, while *Model 4* achieves a similar performance in epoch 68). Hence, exploiting the locality of data by augmenting LSTM with a CNN, helps the model learn invariant features for positive and negative sentiment. Then these spatial features catalyze the training process when they are fed into the LSTM recurrent layer.

Nonetheless, we observed during the initial LSTMs training that the model might be quickly overfitting the training data, which is a general limitation of LSTMs according to the literature. In addition, LSTMs might be removing some of the vanishing and exploding gradients issues encountered in basic RNNs, but these issues are still not completely solved with LSTMs either. Moreover, RNNs and LSTMs in general require a lot of computational resources due to the several recurrences and gates they utilize. That is why in our work we also explored Transformers which are not only more robust to overfitting but also have less number of parameters, no need for recurrent layers and are hence much more "hardware-friendly".

Lastly, we find that when giving up all the recurrent inductive bias, only attention-based Transformers can converge much faster and give us a better performance. Within this sentiment identification task, Transformers successfully capture the non-local semantic sentiment of the whole document and produce the best performance. Although attention models do suffer from overfitting, the dropout setting can largely alleviate this problem. We also find that larger batch sizes are more efficient in this model compared to LSTMs. We see this as an additional advantage of Transformers as larger batch sizes in the training steps help reduce the time to run each epoch, making Transformers even more effective.

6 Conclusion

In this work, we used the IMDb dataset (Andrew[3], 2011) to classify a sequence of comments into positive or negative sentiments. During this process, we learnt more about the advantages and disadvantages of RNNs, LSTMs and models using attention mechanism. We dug deeper into the vanishing and exploding gradients' mathematical proofs and we explored several LSTM and Transformer deep neural networks. This exploration process includes not only adding regularization but also combining our models with other types of neural networks, such as CNNs, and venture into interpreting why such combinations work better. We were also concerned by verifying experimentally the differences between LSTMs and attention-based models results.

Finally, some of the challenges and learnings that we encountered in this project include: (i) achieving a good understanding of Long Short Term Memory building blocs and attention mechanisms before we tackle Natural Language Processing models in class, (ii) working on preprocessing the data was not an easy procedure and required a good amount of research, and lastly (iii) classifying contextual sentiment, such as implicit and sarcastic comments, is a hard task.

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin. (2017). Attention Is All You Need. NIPS.

[2] Ambartsoumian, A., Popowich, F. (2018). Self-attention: A better building block for sentiment analysis neural network classifiers. arXiv preprint arXiv:1812.07860.

[3] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).

[4] https://github.com/jensjepsen/imdb-transformer

[5] Arbel, N. (Dec. 2018). How LSTM networks solve the problem of vanishing gradients: A simple, straightforward mathematical explanation: https://medium.com/datadriveninvestor/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577

[6] Olah, C. (Aug. 2015), Understanding LSTM Networks: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[7] Yan, S. (Mar. 2016). Understanding LSTM and its diagrams: https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714

[8] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[9] Pennington, J., Socher, R., Manning, C. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[10] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

[11] Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. Neural computation, 9(8), pp.1735-1780.