Final Project

IMDB Review Classification

Pranay Bindela, Sneha Sharma Satheesha, Nilay Anand

College of Professional Studies

EAI 6080: Advance Analytical Utilization

Professor Mimoza Dimodugno

Abstract

The report that follows is about a neural network application. We use what we've learned so far to build a neural network-based model that does sentiment analysis on IMDb reviews. The model is created using the transformer neural network, a unique approach to LSTM and GRU-like models that rely on attention or context. Although LSTM-based models have shown excellent results when used with NLP, our research found that transformers are the current state of the art for natural language processing due to their capacity to preserve infinite context given sufficient memory and compute resources. To show the capabilities of each approach, we compare them in our final project by doing sentiment analysis on IMDb reviews.

Introduction

Deep learning is a discipline of research that has emerged because of neural networks' enormous potential in machine learning. One form of neural network used for natural language processing is recurrent neural networks (RNN). In our project, we focus on LSTM, an RNN extension, and transformers, a unique architecture. LSTM was developed to address the issue of disappearing gradients by incorporating the ability to discard data as needed via "gates." Despite their excellent achievements in NLP, they have a limited awareness of context, which transformers address. They have an almost unlimited ability to absorb and preserve context, making them the current natural language retrieval champions. They have seen already been applied for sentiment analysis, machine translation, chat bots and more with phenomenal results. For our project, we compare both LSTM and Transformers by using them for sentiment analysis on IMDb reviews. The goal of this to test the next generation of neural networks, discussing their architecture, and the essential aspects of transformers that make them so powerful at understanding context.

Review Classification using Neural Networks

The goal of this project is to compare two types of neural networks for the task of sentiment analysis, both of which have shown great potential in natural language processing. Let us look at the architecture and implementation of each.

LSTM Architecture and Implementation

The architecture of Long Short-Term Memory (LSTM) was designed the overcome the problems with RNNs like vanishing gradient. The part of the network can be summarized as (Singh, 2021):

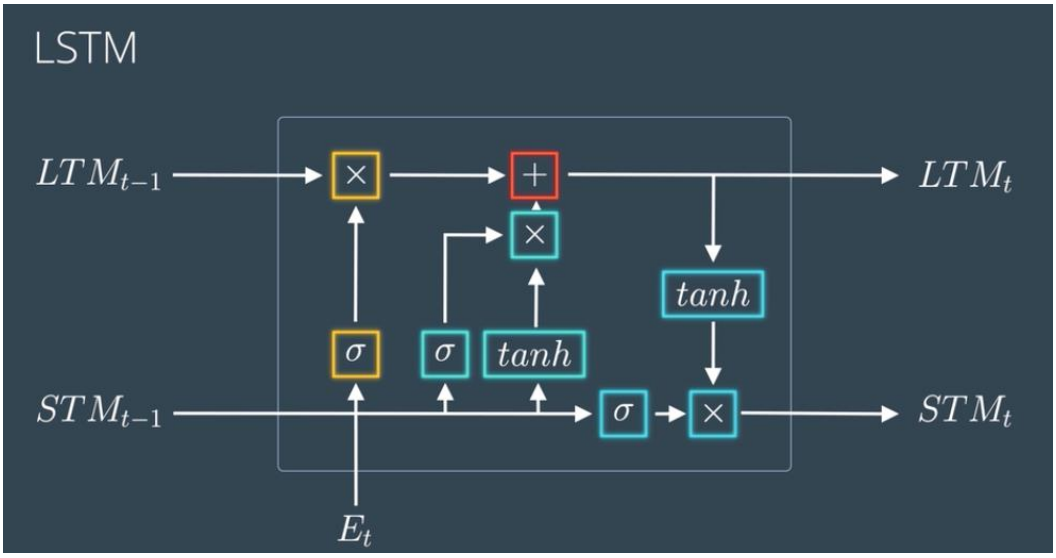
1. **Forget Gate:** LTM goes to forget gate and it forgets information that is not useful.
2. **Learn Gate:** Event (current input) and STM are combined so that necessary information that we have recently learned from STM can be applied to the current input.
3. **Remember Gate:** LTM information that we haven't forgotten and STM and Event are combined in Remember gate which works as updated LTM.
4. **Use Gate:** This gate also uses LTM, STM, and Event to predict the output of the current event which works as an updated STM.

This architecture allows the model to forget irrelevant data and remember only what is important for learning. Although, this architecture does still suffer from the problem of exploding gradient, it can be managed by tuning the learning rate of the model.

The complete architecture of the model can be summarized in the Figure 1.

Figure 1

LSTM Architecture



Source: Singh, G. (2021, January 21). LSTM Architecture | Understanding the LSTM Architecture. Analytics Vidhya

Our implementation of the model went through multiple stages:

- The initial training was done with a singular LSTM with linear layers, using the sigmoid activation function and training for 8 epochs, we got an accuracy of 80%
- Changing the activation function 84% accuracy on changing activation to SoftMax
- Further, using batch normalization after and before each linear layer improved the accuracy to 85% with just 2 training epochs sufficient to achieve this result
- Adding dropout layers to the model which randomly drop a few neurons form the model, the accuracy took a huge jump to 89% in 5 epochs. There were no improvements in the model so the approach needed modifications

- The previous model was trained as a language model so that it may understand the context of a language giving us better results for our application
- The language model was then trained on the IMDb review data, and this provided the best results yet. The final model resulted in an accuracy of **93% with validation loss of 0.23.**
- The parameters used for training were-
    - Loss function used – Negative log likelihood
    - Optimization - Adam optimizer

Transformer Architecture

This model was designed to overcome the pitfalls of LSTM model, to enhance its ability to learn on large datasets. The architecture of transformers is designed to retain infinite context of a language given enough processing resources and memory. The architecture of transformers consists of the following (Vaswani et al., 2017) –

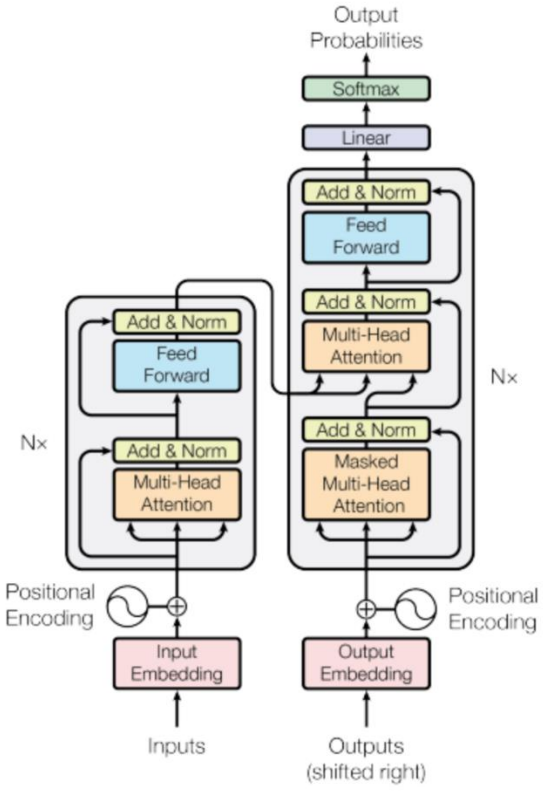
- Encoder - The encoder is made up of N = 6 identical layers. Each layer is divided into two sub-layers. The first is a multi-head self-attention mechanism, while the second is a basic, completely linked feed-forward network that is positionally coupled. Following layer normalization [1,] we use a residual connection [11] around each of the two sub-layers. In other words, the output of each sub-layer is LayerNorm(x + Sublayer(x)), where Sublayer(x) is the function that the sublayer itself implements. To support these residual connections, all sub-layers and embedding layers in the model give outputs with size dmodel = 512.
- Decoder - The decoder is similarly made up of N = 6 identical layers. The decoder inserts a third sub-layer, which conducts multi-head attention over the encoder stack's output, in

addition to the two sub-layers in each encoder layer. We use residual connections surrounding each sub-layer, like the encoder, followed by layer normalization. We additionally change the decoder stack's self-attention sub-layer to prevent positions from attending to following positions. This masking, along with the fact that the output embeddings are offset by one place, ensures that predictions for position *i* can only rely on known outputs at locations less than *i*.

The basic architecture can be seen in the Figure 2.

Figure 2

Transformer Architecture



Source: (Vaswani et al., 2017). Attention Is All You Need

Note - L2 Regularization does not function for any of the models.

For our implementation and use case we focused only on the encoder part of the transformer, the decoder is not applicable for use in this case as we are not generating any new text from the input. To create our transformer model, we used the same base code of LSTM with a few modifications:

- We replaced the LSTM initializer with a transformer initializer. This allowed us to keep the parameters similar between the two model for comparison.
- The number of layers were reduced as transformers and more efficient and calculate the complete context of the sentence in each layer.
- The dropout amount and the embedded dropout amounts were reduced to reduce the data loss which improved results.
- The learning rate was too high for transformers; thus, it was dropped from 10^{-3} to 10^{-4} and finally to 10^{-5}.
- We also tried a different learning rate that improved the accuracy of the model to get a final accuracy of 89%

Results

Both models performed well in the final iteration. Although research does indicate that transformers have more potential if supplied with more data. The accuracies that model each model achieved can be seen in Figure 3.

Figure 3

Model Accuracy for LSTM (Left) and Transformers (Right)

epoch	train_loss	valid_loss	accuracy	time
0	0.023620	0.226539	0.932520	03:11
1	0.034979	0.244642	0.927360	02:43
2	0.039197	0.318953	0.909880	02:58
3	0.029498	0.322379	0.917920	03:09
4	0.026992	0.331342	0.921000	02:57
5	0.011977	0.338810	0.926040	03:00
6	0.008789	0.335589	0.928000	03:02
7	0.005584	0.362975	0.927040	03:02
8	0.001221	0.363702	0.926960	02:58
9	0.001586	0.368670	0.926960	02:51

```
0.22919915616512299 0.925
0.18906308114528655 0.9390625
0.2785822212696075 0.89375
Valid Loss  0.3609338641166687 0.88203125
0.2257683753967285 0.9265625
0.23262809514999389 0.9234375
Valid Loss  0.40727649331092836 0.85546875
0.21252832412719727 0.9234375
0.22596641182899474 0.9234375
Valid Loss  0.341300331056118 0.8734375
```

Conclusion

In conclusion, we identified that transfer learnt LSTM is giving us higher accuracy in comparison with Transformers. We got an accuracy of 89% on LSTM, 93% on transfer learnt LSTM and 88% on Transformers. We have a better chance of higher accuracy on Transformers if the amount of data is increased. LSTM is a better model for classification and Transformers are better for long term contextual models like translation and language modelling.

References

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017, June 12). *Attention Is All You Need*. arXiv.org. Retrieved October 27, 2022, from https://arxiv.org/abs/1706.03762

Singh, G. (2021, January 21). *LSTM Architecture | Understanding the LSTM Architecture*. Analytics Vidhya. Retrieved October 28, 2022, from https://www.analyticsvidhya.com/blog/2021/01/understanding-architecture-of-lstm/