# NLP Final Project Report

Dongwook Lee (lee.dongw@northeastern.edu)
Nisarg Parikh (parikh.nis@northeastern.edu)

December 2023

## 1 Introduction

In our project, we compare the efficacy of Long Short Term Memory (LSTM) networks and Bidirectional Encoder Representations from Transformers (BERT) in the domain of sentiment analysis, particularly classifying financial articles into positive, negative, and neutral sentiments. The LSTM, a type of recurrent neural network, is adept at capturing long-range dependencies, enhanced by a custom-scaled dot-product attention mechanism implemented via PyTorch for numerical stability. In contrast, BERT leverages transformer architecture to manage long-range textual dependencies efficiently, which may give it an edge in deciphering the nuanced context of financial texts.

We will evaluate the models using standard metrics—accuracy, precision, recall, and F1-score—supported by a cross-validation approach to validate their robustness and generalizability. Ablation studies will further dissect the influence of individual model components on performance. Our aim is twofold: to determine the superior model for financial sentiment analysis and to deepen our comprehension of LSTM and BERT's capabilities within intricate NLP tasks, thereby enriching the field's practical applications.

### 1.1 Data Processing:

We use the Sentiment Analysis with Financial News from Kaggle. It has a total of 5322 entries with 2874 positive sentiment samples, 1703 negative sentiment samples and 798 neutral sentiment samples. We performed one-hot encoding for postivie, negative and neutral labels. We removed all links, stopwords, escape sequences and numbers. And we only consider the top 10k most happening words as our vocabulary which are larger than 2 characters. And then we also remove all words from the train and test data, all words which do not appear in the vocabulary. For our purposes as we are removing all non-vocabulary words from the train and test data we do not consider handling unknown tokens.

The maximum length of sentences in the dataset is 36 hence we pad each sentence smaller than 36 by a padding token. For LSTMs Each sentence begins with a Start of Sequence token and ends with an End of Sequence token. We consider a train:test split of 8:2 and similar for the train to validation split.

## 1.2 Long Short Term Memory:

The core building blocks of an LSTM unit involve three gates: the input gate ($i_t$), forget gate ($f_t$), and output gate ($o_t$). These gates are governed by mathematical operations that control the flow of information through the cell state ($c_t$).

Here, $x_t$ represents the input at time t, $h_{t-1}$ is the hidden state from the previous time step, W and b are weight matrices and bias vectors, and $\sigma$ is the sigmoid activation function.

The forget gate ($f_t$) determines what information from the previous cell state ($c_{t-1}$) should be discarded, while the input gate ($i_t$) decides what new information should be stored in the cell state. The update to the cell state ($c_t$) is a combination of the previous state and the new information. The output gate ($o_t$) then controls what information is passed to the next time step's hidden state ($h_t$).

For the Attention, by introducing a scaling factor, typically $\frac{1}{\sqrt{d_k}}$ where $d_k$ is the dimensionality of key vectors, the attention mechanism avoids issues like exploding gradients associated with unbounded dot products. This scaling ensures consistent behavior across varying dimensions and acts as an effective learning rate, allowing the model to attend to different parts of the input sequence effectively.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

where:

- $Q$, $K$, and $V$ are the query, key, and value matrices, respectively.
- $d_k$ is the dimensionality of the key vectors.
- softmax is the softmax function applied element-wise.

## 1.3 BERT:

BERT is built on the Transformer architecture, which was introduced in the paper "Attention Is All You Need" by Vaswani et al. The Transformer model is primarily based on attention mechanisms, especially self-attention, which allows the model to weigh the importance of different words in a sentence.

1. Self-Attention Mechanism:

   The self-attention mechanism allows each position in the encoder to attend to all positions in the previous layer of the encoder. Similarly, each position in the decoder can attend to all positions in the decoder up to and including that position.

2. Query, Key, and Value:

   Mathematically, if X is the input embedding, the Q, K, and V vectors are computed as follows:

$$Q = XW^Q, K = XW^K, V = XW^V \tag{2}$$

3. Attention Scores and Softmax:

   The self-attention scores are computed by taking the dot product of the Query vector with all Key vectors. This reflects how much focus each word should have on other words.

   The scores are then scaled down by the square root of the dimension of Key vectors (for stability), and a softmax function is applied to get the weights for the Value vectors.

4. Multi-Head Attention:

   In practice, the Transformer model uses what's called multi-head attention. This means that the Q, K, and V transformations are done multiple times with different, learned linear projections. This allows the model to jointly attend to information from different representation subspaces at different positions.

5. Output of Attention Layer:

   The outputs of each head are then concatenated and once again linearly transformed to form the final output of the multi-head attention layer.

## 2 Experiments

Something that is common across all experiments is we perform 5 fold K-fold cross validation with our sentiment analysis dataset. We obtain our dataset from [Add reference]. We holdout 20% on all data to form the test dataset. And 20% of the rest to form the validation dataset. The remaining data is treated as the Training data.

### 2.1 LSTMs:

We performed multiple experiments across LSTMs of varying hidden dimensions $(16, 32, 64, 128, 256, 512)$, dropout rates $(0.0, 0.25, 0.5)$, for Bidrectional LSTMs with/out Scaled Dot attention.

We used Google Colab with a T4 gpu to train the LSTM models. We use CrossEntropy loss to perform back propogration with the Adam optimizer with the default parameters in Torch. We use the LSTM module that ships with the Torch library with Dense Neural networks for the sentiment analysis. All other variables are kept static across models.

We trained each LSTM model for 250 epochs where each 50 epochs we switch to a new fold of the dataset.

## 2.2 BERTs:

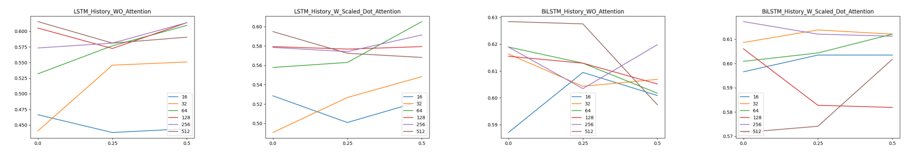
We evaluated three distinct models to ascertain the impact of varying BERT configurations on our text classification task, which aimed to categorize texts into three classes. The first model, BERT Base, served as our benchmark with its 12-layer architecture, offering a balance between computational efficiency and performance. The second model, BERT Large, extended the complexity with 24 layers and 16 attention heads, hypothesized to enhance performance through its augmented capacity for capturing more nuanced patterns in the data, albeit at the cost of increased computational demand. Lastly, we experimented with a custom-configured BERT model, retaining the hidden size of the base model (768) but adjusting the architecture to include 16 hidden layers and 12 attention heads. This custom model was designed to test the hypothesis that additional layers might yield more abstract text representations and possibly improve generalization, without escalating to the full scale of BERT Large.

## 3 Results

We show the results of the LSTM and Transformer separately and then compare the best performing models in each case.

### 3.1 LSTMs:

Here are the test accuracy plots for the 4 architectures we tried with the combinations of Bidirectional LSTM architecture(BiLSTM) and Scaled Dot-Attention(SDA):



(a) LSTM W/O Attention (b) LSTM W SDA (c) BiLSTM W/O Attention (d) BiLSTM W SDA

Figure 1: Test accuracy of different model architectures with different hidden dimensions across the dropout hyperparameter

We see that when we do not use the BiLSTM architecture design, lower dimensional hidden and cell states do not perform well. And when we that SDA adds a performance boost of 2-4% for either architecture. But the best performance between these architectures is seen with no SFA, no Dropout and the largest hidden dimension size(512). But SDA ensures that we can use dropout, which is desirable as it adds attributes to all models such as sparsity, model robustness and generalizability.

### 3.2 BERTs:

Based on the results presented in the table, we can deduce that the BERT Large model outperforms the BERT Base and the Custom BERT Configuration models across all evaluated metrics. Specifically, BERT Large achieves the highest accuracy (0.91), precision (0.88), recall (0.85), and F1-score (0.86), indicating a superior overall performance in classifying texts into the three categories. Although the Custom BERT Configuration exhibits a slight improvement in accuracy (0.84) over the BERT Base model (0.83), both models show identical precision, recall, and F1-scores (0.80, 0.77, and 0.78, respectively). These results suggest that while the Custom BERT Configuration does not significantly enhance performance compared to BERT Base, the increased complexity and capacity of BERT Large seem to have a notable positive effect on the model's ability to generalize and accurately classify the given text data.

| | BERT Base | BERT Large | Custom BERT Configuration |
|---|---|---|---|
| Accuracy | 0.83 | 0.91 | 0.84 |
| Precision | 0.80 | 0.88 | 0.80 |
| Recall | 0.77 | 0.85 | 0.77 |
| F1-Score | 0.78 | 0.86 | 0.78 |

## 4 Conclusion

In conclusion, our project's sentiment analysis of financial texts revealed that BERT models markedly outperform LSTMs, with BERT's accuracies surpassing the 0.60 benchmark set by LSTM models. BERT's transformer architecture, adept at handling the intricacies of language and context, proves essential in analyzing the nuanced sentiment of financial articles. This superiority in performance highlights the relevance of using advanced models like BERT for complex NLP tasks in the financial sector. Our findings endorse BERT's robustness and its potential for real-world applications, thereby offering valuable insights for future NLP endeavors in finance.