STOCK VOLATILITY FORECASTING

Vivek Mittal
School of Computing and Electrical Engineering
Indian Institute of Technology , Mandi
b18153@students.iitmandi.ac.in

January 9, 2021

ABSTRACT

In this report, we describe a method for forecasting the stock volatility using earning calls. We used verbal and textual features along with previous stock information to forecast future volatility. We use SentenceTransformer for encoding every sentence and then we use a self attention based BLSTM model to accumulate the text features. Similarly, we use a self-attention based BLSTM model for encoding the verbal features. We simply concatenate the verbal and text features along with past stock trends to make final volatility prediction.

***K*eywords** Stock Forecasting · Neural Network · Multimodal Feature Fusion

1 Overview

Task modeling. We approach the task of stock volatility forecasting as a regression problem in which given an earning call e, comprising of an audio A, and aligned text T, and stock prices $p_{[0,n]}$, we aim to learn a predictive regression function $f(e_{\{T,A\}}) \rightarrow v_{[0,n]}$, where p_i is the closing price of a stock at day i and $v_{[0,n]}$ is

$$v_{[0,n]} = \ln\left(\sqrt{\frac{\sum_{i=1}^{n}(r_i - \overline{r})}{n}}\right) \tag{1}$$

where, r_i is the return price of day i, $\frac{p_i}{p_{i-1}} - 1$ and $\overline{r}$ is the mean return over the period of 0 to n days.

We further divide the audio A and transcripts T in k text-audio aligned sentences where, $k \in [1, N]$, N is the maximum number of sentences a earning call can contain.

1.1 Dataset & Preprocessing

We use a subset of MAEC dataset [1] for training our algorithm. We divide our subset into 1401 training samples, 351 validation samples and 438 testing samples. For every sample we have audio features $A = \{a_1, a_2, \ldots, a_k\}$ and text $T = \{t_1, t_2, \ldots, t_k\}$ where k is the number of sentences spoken during the earning call, t_i is the text for i$^{\text{th}}$ sentence. For every audio sentence we have 29 features i.e. $a_i \in \mathbb{R}^{29}$, some of these features were `NaN` in the dataset thus we replace them with zeros. We extracted the stock prices information from Yahoo Finance and after calculating the volatility we found for some samples `numpy.log` producing `-inf` values so we also discarded those samples from our dataset.

1.2 Model Details

Text Features We use a pretrained SentenceBERT [2] model for extracting sentence embedding. The SentenceBERT model generates a matrix $M \in \mathbb{R}^{k \times Q}$ where, k is the number of sentences in the transcript T and Q is the embedding size of SentenceBERT. After this, the sentence embeddings matrix M is passed to BLSTM that will produces a finer sentence embedding for every sentence with the context of other sentences as well. Before passing the matrix M to

Table 1: Results

Model	MSE Error (Test set)
Model 1	0.7635
Model 2	0.6685

BLSTM we pad the set of sentence embedding to make all of them of same size $N \times Q$. The BLSTM produces a matrix $M' \in \mathbb{R}^{N \times 2h_t}$ where h_t is the hidden size of the BLSTM layer.

Verbal Features For every earning call, we have an audio matrix $A \in \mathbb{R}^{k \times P}$ where k is the number of sentences spoken the earning call and P is the number of features calculated for every sentence. Similar to text features (Sect. 1.2), we use a BLSTM layer to learn the features of audio. Before passing the matrix A to BLSTM we first pad A to make it of size $N \times P$. The BLSTM produces a matrix $A' \in \mathbb{R}^{N \times 2h_a}$, where h_a is the hidden size of BLSTM.

1.3 Feature Fusion and Prediction

We train 2 linear layer networks with a softmax activation to predict the importance of every sentence embedding for M' and A' respectively. Each of the network gets $M'_i \in \mathbb{R}^{2h_t}$ and $A'_i \in \mathbb{R}^{2h_a}$ as input and produces a number m_i and a_i as output respectively where i indexes over the sentences spoken in the earning call.

$$M'_{s,i} = \frac{\exp(m_i)}{\sum_{i=1}^{N} \exp(m_i)}; A'_{s,i} = \frac{\exp(a_i)}{\sum_{i=1}^{N} \exp(a_i)} \tag{2}$$

After the softmax we get $M'_s \in \mathbb{R}^N$ and $A'_s \in \mathbb{R}^N$. The text features M' and audio features A' are then scaled with M'_s and A'_s respectively.

$$M^* = \sum_{i=1}^{N} M'_{s,i} M'_i \tag{3}$$

$$A^* = \sum_{i=1}^{N} A'_{s,i} A'_i \tag{4}$$

We get the final text features $M^* \in \mathbb{R}^{2h_t}$ and audio features $A^* \in \mathbb{R}^{2h_a}$. Note that we don't use masking while calculating the softmax as we pass the complete padded embeddings to the BLSTM.

Model 1 M^* and A^* are concatenated and normalized using $l_2 - norm$. These final features $F_1 \in \mathbb{R}^{2(h_t+h_a)}$ are passed though a two layer neural network to produce the final prediction. We calculate the MSE loss between the predicted and actual values for training the complete neural network via backpropagation.

Model 2 (using previous stock information) For every stock we also extracted the closing price and volume traded for last 10 days. These 20 values are also concatenated with A^* and M^* to produce final features $F_2 \in \mathbb{R}^{2h_t+2h_a+20}$. These features F_2 are then l_2 normalized and passed though a two layer neural network to produce the final prediction. We calculate the MSE loss between the predicted and actual values for training the complete neural network via backpropagation.

1.4 Implementation Details

We implemented our algorithm in PyTorch. We used Adam [3] optimization algorithm for training our neural network with an initial learning rate of 0.0003 and weight decay of 1e-5. We use a batch size of 32, the maximum number of sentences in a earning call i.e. N equal to 300 and SentenceBERT [2] produces a 768 dimension embedding i.e. Q. We train our model for 50 epochs. Further training details can be found in our code.

1.5 Experiments & Results

We experimented with two models: first (Model 1) that does not use the previous days stock data and second (Model 2) that uses previous days stock information. The results are summarized in table 1 and loss curves are shown in figure 1.

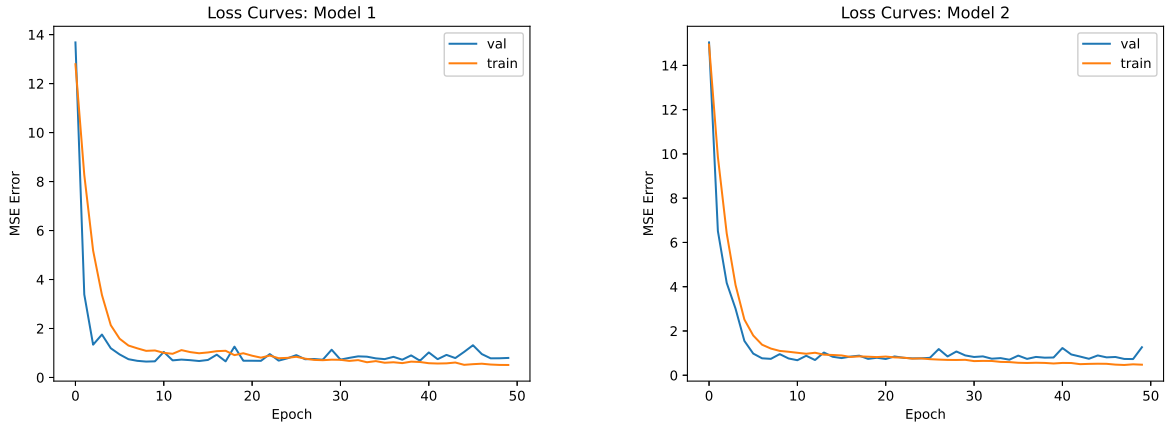

Figure 1: Loss Curves

Inference We found using `torch.log` causing exploding gradients, this is because most of the values (without using ln) lies between 0 to 1, and the log curve has a large gradients in that region. We don't see any problem of gradients in the LSTM cells. We also found that using ln in volatility equation 1 has a lot of effect on the error. We found if we remove the ln from the volatility equation the error can be reduced to 0.0052 but we can not do this as it will change the definition of volatility. The volatility has a importance of ln because for stock market we need to model a very small difference in the return and as we know ln changes rapidly between 0 and 1 so, it is a very good choice for this modelling this effect.

2 Conclusion

In this report, we described a method for forecasting stock volatility after an earning call. Our model predicts the volatility using the text transcript and audio of the earning call. We designed a BLSTM based model that encodes both the text and audio. We used a SentenceBERT model for generating the text embedding and used statistical features of audio data. We found that using previous days stock data like closing price and volume traded can improve the performance of our model for forecasting the future stock volatility. Future work can focus on better encoding the audio features using MFCC features or using a CNN based models, the previous stock information can be used in a better way by using BLSTM specially trained for predicting future stock volatility that can reinforce the prediction done using earning call data only.

References

[1] Jiazheng Li, Linyi Yang, Barry Smyth, and Ruihai Dong. Maec: A multimodal aligned earnings conference call dataset for financial risk prediction. In *Proceedings of the 29th ACM International Conference on Information amp; Knowledge Management*, CIKM '20, page 3063–3070, New York, NY, USA, 2020. Association for Computing Machinery.

[2] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.