STOCK PRICE PREDICTION USING CENTRAL BANKS SPEECHES

Thomas Kessous **Xinyu Liu** **Manan Gupta** **François Schmerber** **Wanqi Hong**

Abstract

Central bank speeches often provide information on the economic health of countries, trends and the near future of the global economy. These same trends are usually linked to fluctuations in market volatility indexes.

The aim of this work is to predict the evolution of the VIX and EURUSDV1M indices using speeches from the US and European central banks.

By using NLP models, decision trees and neural networks, we performed a regression task on the value of these indexes and a classification task on the bearish or bullish trend of these indices.

1 Introduction

Every few days, central bankers publish a speech supposedly explaining the financial situation in the world. These speeches are closely followed at the global level by all financial actors, and therefore have a strong influence on the evolution of financial markets and the global economy. Indeed, central bank communications can have an impact on various key economic factors, from interest rates to monetary policy, inflation expectations, credit, debt and overall financial leverage, for both the private and public sectors. As these speeches can have an impact on key macroeconomic factors and move financial markets, the ability to decipher and correctly interpret "central bank jargon" has become a key area of interest for all kinds of financial analysts and economic actors.

Based on speeches that have been delivered over the last 10 years by the FED (Federal Reserve) and the ECB (European Central Bank), our work consists in **predicting the evolution of two economic indexes**. These indexes are the VIX (Equity Stocks Volatility for the American markets) and EURUSDV1M (Exchange rate Volatility of the USD/EUR market). We want to perform two tasks:

- Task 1 (Regression): Participating systems will be evaluated using the Root Mean Square Error (RMSE) against the true stock prices ;
- Task 2 (Binary Classification): Participating systems will be evaluated using Accuracy (ACC) against the true labels.

This work takes on the challenge of dealing with both textual (bank reports) and numerical (stock value) data. In order to process the textual data, NLP techniques like summarization, words embedding , and sentiment analysis were implemented. Then, we created and optimised regression and classification models based on decision trees and neural networks, whose performance is compared.

2 Related Work

Before the emergence of machine learning and deep learning, the stock price data prediction is based on the previous sequence data, which would be modeled as a time series. (Ariyo et al., 2014) modeled the published stock data with the ARIMA model, which has a strong potential for short-term prediction.

Later, for the same data, (Selvin et al., 2017) adopted neural network to model the sequences using sequence models like RNN and LSTM. With more and more data is available, people start to use text data from speeches or reports to predict the stock price as well. Emotions from the speech are extracted by (Buechel et al., 2019) from the central bank speeches. Sentiments analysis is applied to speeches and social media for market prediction by (Pagolu et al., 2016). Besides extracting information from the text, pretrained language models like FinBERT (Araci, 2019) based on financial related text are used for text representation.

3 Model

The whole system consists of two main parts: textual representation and prediction architecture. We

first extracted all the useful information from the speech, including summarization, sentiment and emotion analysis. In the second part, we included all the information from the speech and also the stock price data, and then we applied different prediction models to finish the tasks.

3.1 Textual Representation

3.1.1 Named Entities Recognition

We considered doing a named entity recognition for our work. Indeed, economic changes are often due to the action of companies or financial institutions that have specific names. The BERT model allows us to extract all the names of organisations, and to build a new column in our dataset. We realised that most of the organisations mentioned in the document are state financial institutions or banks. However, we have not been able to exploit this effectively for our prediction work.

3.1.2 Text Summarization and Embedding

We've tried 3 different methods for summarize the speeches, which will all be explained in this part. After the summarization, we did some feature engineering and then we got the embedding of the summaries.

Summary 1. Sentiment Word List

In order to establish a first, simplified summmarization method, we used different sentiment word lists provided by Tim Loughran and Bill McDonald in their paper Loughran and McDonald (2011). They provide a dictionary containing words divided into different sentiment categories: negative, positive, uncertainty, litigious, strong modal, weak modal, and constraining. In order to perform summarization, we used only the negative and positive categories. The idea of this first summarization method was to divide each speech into sentences and to rank those sentences using Loughran and McDonald's word lists. The more sentiment words a sentence would contain, the higher it would be ranked. In order to obtain a summary that contains as much sentiment as possible, we would simply select the highest ranked sentences in order to build the summary. The downside of this method is that there is essentially no continuity between the different sentences that compose the summary. However, on the other hand, this summarization method provides two significant advantages: first, the summarization process is relatively fast and secondly, we don't have to worry about the summary misrepresenting the original speech since the sentences are directly taken from it.

Summary 2. Text Rank Summarization

The second summarization method is an extractive summarization method which works by selecting the 5 most important sentences from the original speech text. This extracts the most important information from the speeches to save as much relevant information as possible. Since this method does not create any new text, the summary will have parts of the original text in it.

The main idea in this method is to find the similarities among the sentences in the speeches and then return only those sentences with the maximum similarity for the summary. The Cosine Similarity matrix is used for finding the similarities and then a TextRank algorithm is used to rank the sentences according to their importances.

In this method, the sentences are first extracted from the speeches and then vectors based on the words are created for each sentence. Then, the cosine similarity is calculated for the sentences by using the cosine distance between them. The Cosine distance between 2 vectors A and B can be written as: Cosine Distance (A,B) = 1 - Cosine(Angle between A and B). If 2 vectors are similar, the cosine distance between them will be low and the Cosine Similarity will be high. The sentences are then ranked based on similarity and the top 5 most similar sentences are chosen to make the summary.

Summary 3. GPT-2 Model

Different from the two summarization methods above which are extractive ones, GPT-2 model generates abstractive text summaries. Abstrative text summarization is considered as an advanced method, with the approach to identify the important sections, interpret the context and reproduce the text in a new way. In this way, the sentences in the summary are generated by the model instead of being extracted from the original text. This method needs a language generative model, which in our case is a pre-trained language model GPT-2.

GPT-2 is a large transformer-based language model with 1.5 billion parameters, and it's one of the state-of-art pre-trained language model. It's trained on a dataset with 8 million web pages for a simple objective: predicting the next word given all of the previous words within some text, and that's why the summaries it generated are natural and logical.

There are also some problems using pre-trained models for summarization. Firstly, the input size

is usually limited. There are less choices available due to this reason, and we have to adjust our data to fit the input size limit. Secondly, it's hard for us to evaluate if the generated summary contains the important information we want for the next step of prediction. In this way, we can not use fine-tune techniques to make the pre-trained model adapted to our own dataset.

Feature engineering

The two main flaws of the textual data were the holes between the different speeches and the small but nonetheless noticeable amount of foreign language speeches. First, we used only the last speech for predictions (the database is built such as there is always a last speech). When facing the rare cases of samples containing only one speech (the last one) in a foreign language, we simply replaced it by a short, neutral string. For that purpose, we built a function detecting if a speech is in a foreign language using Spacy's Language Detector. After that we decided to use the same amount of five speeches for each sample. In order to do so, we first grouped all the speeches of each sample in a list, then deleted all the foreign speeches in that list and then gathered the five last speeches, replicating the last speech when less than five speeches remained in the list and using a neutral string if no speech remained at all.

Summary Embedding

For the embedding of summarized speeches, we used 2 HuggingFace transformer models: "mrm8488/distilroberta-finetuned-financial-news-sentiment-analysis" and "distilbert-base-uncased-finetuned-sst-2-english". The first one is a distilled roberta model trained on financial news and it's used with the tree-based models. The idea was to tokenize each summary and then pad them all to the same size before recovering the last hidden state values of the model when trained on each tokenized summary. The second one is a distilled bert model that fits with the neural network decoder that we used.

3.1.3 Sentiment Extraction

We extracted some other features from the speeches other than the summaries, such as sentiments and emotions.

We applied 3 different pretrained models for the sentiment analysis and compared their efficiency and final results, which are *finBERT* Araci (2019)(a language model based on BERT for financial NLP tasks), a fine-tuned checkpoint of *RoBERTa-large* Heitmann et al. (2020) [1], and *DistilRoBERTa* based model[2], which is a distilled version of the RoBERTa-base model.

finBERT is a fine-tuned model specifically for financial domain, it is trained on the Financial Phrasebank dataset. Unlike *RoBERTa-large* and *DistilRoBERTa*, which are trained on different data sources such as tweets, reviews etc. Therefore *RoBERTa-large* and *DistilRoBERTa* have a better generalization while *finBERT* would be more suitable for finance related tasks. *finBERT* predicts positive, neutral and negative labels while the other 2 predict only positive and negative labels.

While applying the models, we found out that *RoBERTa-large* takes more time than *DistilRoBERTa*. But in the meanwhile, *RoBERTa-large* is said to outperform *DistilRoBERTa* as well. Both of the 2 models are case-sensitive. So necessary preprocessing steps are needed.

However, all these pretrained models are hard to deal with long sequences. The maximum number of tokens allowed for these models are 512. Figure shows how many tokens we have for each of the whole speech.

We used 2 different approaches to deal with this problem.

The first approach is to split all the tokens into several chunks which is of token size 512. As for the last chunk, to keep the chunk size the same for the model to train, we padded all the remaining tokens to be 0. For each chunk, we got a label and a score as the probability for that label. Then we took the average scores over all the chunks. The drawback of this approach is that it considers every chunk to have the save weight. So the result is the presence involves a lot of noise and the scores of the sentences may balance out in total.

The second approach is to apply the pretrained model to the summaries we got, which has less than 512 tokens. However, this approach depends largely on the quality of the summaries.

For each data, we get the average of sentiments of all the speeches as the sentiment feature of the data.

[1] https://huggingface.co/siebert/sentiment-roberta-large-english

[2] https://huggingface.co/mrm8488/distilroberta-finetuned-financial-news-sentiment-analysis

3.1.4 Emotions Extraction

To extract the emotion features from the speech, we used the method described in the paper Buechel et al. (2019). A popular theory in psychology uses real-valued vectors to represent universal basic emotions such as Joy, Anger, Sadness... In this work, we employed the VAD model because of its greater flexibility. Components of the emotions are Valence(pleasure vs displeasure), Arousal(calmness vs. excitement) and Dominance(being controlled vs. having control over). Figure 2 shows how the common emotions are represented in this 3-dimension space. To compute the VAD score, we adopted a comparably simple lexicon-based approach which models document emotion based on word frequency combined with empirical measurements of lexicalized word emotions. To achieve accuracy, we first preprocessed the speeches by removing titles, subtitles, and footnotes, and then used the open source tool *JEMAS* Buechel and Hahn (2016), which takes speeches as input and produces the VAD scores and standard deviation of each dimension respectively. For simplicity, we computed only the VAD scores of the last speech of each datapoint as the emotion features of that data. The results we got are shown partly in 3 Appendix. Figure 1 in the Appendix shows the 2-dimension plot of all the speeches.

3.2 Prediction Architecture

3.2.1 Tree-Based Model

As explained above, after recovering the last hidden state values of the Roberta model when trained on each tokenized summary, each set of last hidden state values was used as a representation of the corresponding summary. This provided us with a set of new features that could be fed into a classifier or regressor model (we started simply with ensemble methods relying on tree based models). However, the large amount of values in each last hidden state made it prone to overfitting for those models. That is why we decided to build, for each speech, a single feature obtained by taking the mean of the values of the last hidden layer. Furthermore, the emotions extracted from the speeches (Valence, Arousal and Dominance) and the associated VAD score were added to the model for training to better represent the emotion of the speeches. The data was then used to fit an ExtraTrees Classifier and Regressor for both the indices to obtain the predictions for the regression and classification tasks.

3.2.2 Neural Network Model for Regression Task

Considering the embedding we got from the pretrained language model as the output of an Encoder, we built the deep learning network as an Encoder-Decoder structure. Since we already had the encoder model, we only needed to design the structure of the decoder. In order to avoid overfitting, we tried to design a simple network with less parameters.

The data we had for the decoder consisted of two parts: the speech text embedding and the stock price data. The output of the encoder for each speech text was a 768 dimensional vector, which is the speech text embedding that we use in the decoder. According to previous studies and experiments, we found that the previous stock prices are more useful in predicting the following stock price, so we decided to shorten the length of the speech vector when combining the two of them together in order to let the stock prices play a more important role.

The whole network structure is shown in the above figure. After shortening the vector of the speech, we concatenated it with the stock price data which is also a vector of dimension 20. We designed a hidden layer before the final output, and in practice we designed it as a linear layer with 64 neurons. To avoid overfitting, we also added some dropout layer between the linear layers.

4 Experiments

4.1 Data

We have 1254 data points in our training set, each composing a 20-day series of speeches, a 20-day series of stock, a target value for classification task and a target value for regression task. And the data are shuffled.

4.1.1 Statistical Analysis

- Speech Distribution.

  We calculated the number of speeches of each data point. The minimum number of speeches per each data point is 1, the maximum is 23 and the average is around 10. Figure 5 shows the distribution.

  And there are duplicate speeches in different data points. The average number of duplicates is around 8.

- Number of Words Distribution

Since the pretrained language models usually have a limited input size, we needed to find out if the number of words per speech is within the size limit. From the graph below, we can see that most of the speeches have around 3000 - 4000 words, which is definitely over the limit. That's why we needed to summarize the speech before embedding them.

4.1.2 Data Preprocessing

In order to get better precision, we did some data cleaning jobs such as removing the titles, authors and footnotes. Some pretrained models can use the raw data directly without preprocessing. However, more preprocessing steps such as stemming, lemmatization could be done for other tasks.

4.2 Results

The tree based model results on both the regression and classification tasks are given in the tables below. The result of the regression task are measured by the RMSE metric while the classification results are based on the accuracy metric.

From the results, we can see that in the regression task, the model performed very well on Index2 with an RMSE of 0.199. However, the model failed to perform similarly well on Index1 as it had an RMSE of 0.548 for combined RMSE of 0.373.

On the other hand, for the classification task, we can see that the model performed very well on Index1 with an accuracy of 0.655 which is high. However, the model failed to performed similary well on Index2 with an accuracy of 0.585 for a combined accuracy of 0.62.

From the results, it can be seen that the model performed slightly better in the classification task than the regression task.

All	Index1	Index2
0.373	0.548	0.199

Table 1: Model RMSE on Regression Task

All	Index1	Index2
0.620	0.655	0.585

Table 2: Model Accuracy on Classification Task.

4.3 Training Result of the Neural Network

Due to limited time, we did not submit the results of the neural network regression task, but we've trained the model using PyTorch. Training the model took time and computing power, since we don't have a GPU, it's not very practical to run experiments with different parameters. In the end, we trained the model with 16 as the batch size, 0.01 and MSE loss as the optimization goal. Also, due to limited computing power, we didn't fine-tune the pretrained model with the decoder, because the pretrained model has way too many parameters.

In the graphs above, we can see that the loss in the training set has not converged yet, but in the validation set it has almost converged. For further improvement, we need to unfreeze the layers in the pretrained encoder and fine-tune it.

5 Conclusion

In conclusion, we utilised multiple techniques to enable us to predict stock market trends using central bank speeches. We used abstractive summarization from GPT-2 to enable summary embedding from a pretrained distilroberta financial model. Furthermore, we also utilised sentiment and emotion extraction techniques to generate further inputs for the model training. Finally, for the predictions, we conducted a number of experiments to find the most suitable models, selecting from both tree-based models and a Neural Network model for the regression task.

Future work: Due to limited time, we haven't fully explored how neural network performs in the prediction tasks. The decoder structure could be more creative, which needs more experiments with different model structures. Also, we didn't have time to add the sentiment and emotion extraction results in the network, which could be explored further. Considering the sequence characteristic that the data has, sequence models like RNN could also be tested. Besides the decoder structures, fine-tuning the pretrained model of the encoder is also worth trying.

6 Appendix

References

Dogu Araci. 2019. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*.

Adebiyi A Ariyo, Adewumi O Adewumi, and Charles K Ayo. 2014. Stock price prediction using the arima

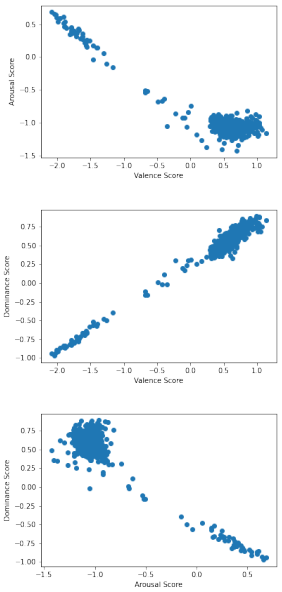

Figure 1: The common emotions represented in Valence, Arousal and Dominance dimensions

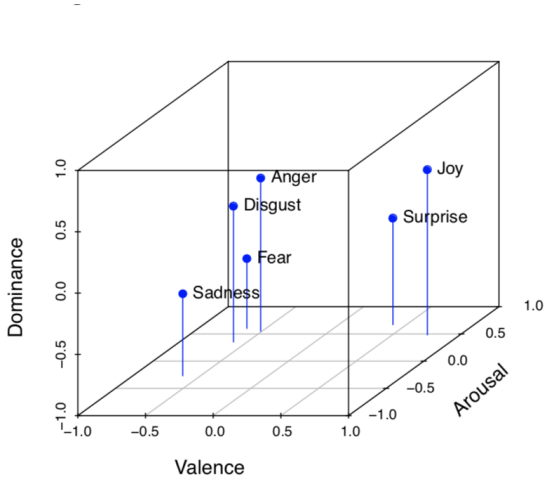

Figure 2: The common emotions represented in Valence, Arousal and Dominance dimensions

	speech	stock	target_classif	target_reg	valence	arousal	dominance
0	[{'ECB': [" Good morning. I very much appreci...	[-0.4729582071304321 3, -0.1724076122045517, -0...	0	-0.408630	0.57556	-1.07790	0.57139
1	[{'ECB': [], 'FED': []}, {'ECB': [], 'FED': []...	[-0.4529215097427368, -0.4202300310134887 7, -0...	0	-0.585796	0.46057	-1.03085	0.50224
2	[{'ECB': [], 'FED': []}, {'ECB': [" "A curren...	[-0.5910692811012268, -0.3601199090480804 4, -0...	1	0.469822	0.36477	-1.13755	0.44246
3	[{'ECB': [], 'FED': []}, {'ECB': [], 'FED': []...	[-0.6796526312828064, -0.7492538094520569, -0....	0	0.127088	0.50432	-1.06736	0.46534
4	[{'ECB': [' How to preserve and consolidate t...	[-0.0216050371527671 8, -0.0564056336879730 2, -...	1	0.441348	0.58990	-0.97332	0.59077

Figure 3: VAD scores example

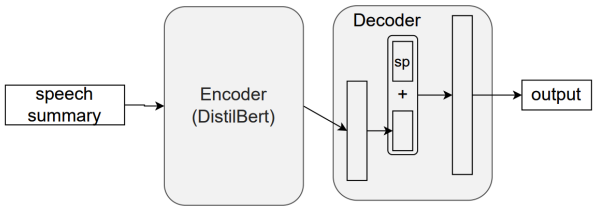

Figure 4: The Decoder network structure

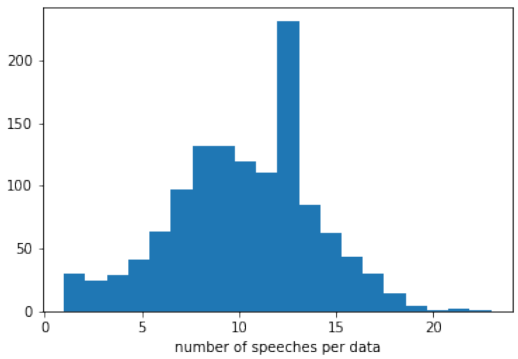

Figure 5: Distribution of number of speeches of each data point

model. In *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, pages 106–112. IEEE.

Sven Buechel and Udo Hahn. 2016. Emotion analysis as a regression problem–dimensional models and their implications on emotion representation and metrical evaluation. In *ECAI 2016*, pages 1114–1122. IOS Press.

Sven Buechel, Simon Junker, Thore Schlaak, Claus Michelsen, and Udo Hahn. 2019. A time series analysis of emotional loading in central bank statements. *arXiv preprint arXiv:1911.11522*.

Mark Heitmann, Christian Siebert, Jochen Hartmann, and Christina Schamp. 2020. More than a feeling: Benchmarks for sentiment analysis accuracy. *Available at SSRN 3489963*.

Tim Loughran and Bill McDonald. 2011. When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *Journal of Finance, 66:1*, pages 35–65.

Venkata Sasank Pagolu, Kamal Nayan Reddy, Ganapati Panda, and Babita Majhi. 2016. Sentiment analysis of twitter data for predicting stock market movements. In *2016 international conference on signal processing, communication, power and embedded system (SCOPES)*, pages 1345–1350. IEEE.

Sreelekshmy Selvin, R Vinayakumar, EA Gopalakrishnan, Vijay Krishna Menon, and KP Soman. 2017. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)*, pages 1643–1647. IEEE.

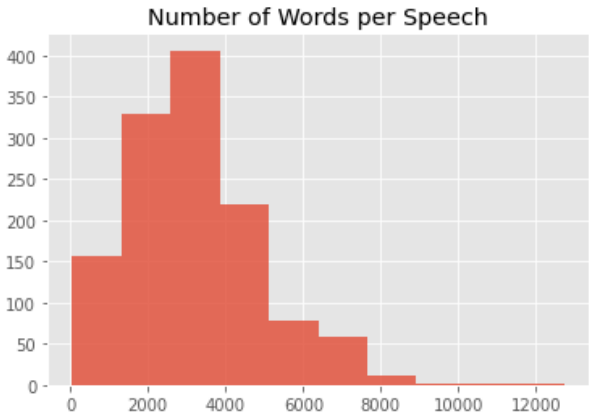

Figure 6: Number of Words Distribution per speech

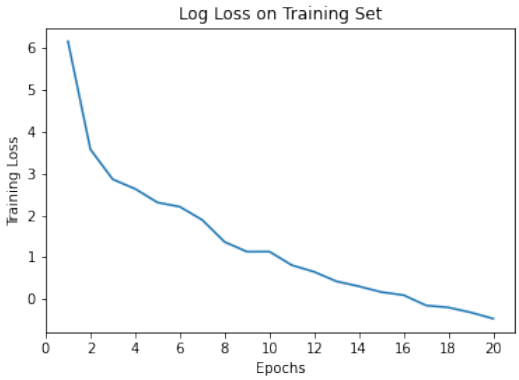

Figure 7: Log training loss with epochs

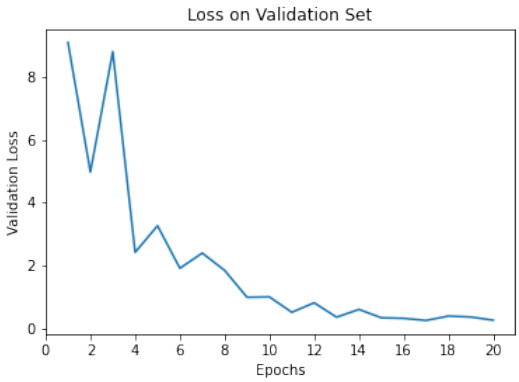

Figure 8: Validation loss with epochs