Market trends prediction using central bank speeches
Machine Learning for Natural Language Processing 2022

Antoine Palazzolo
ENSAE Paris
antoine.palazzolo@ensae.fr

Robin Hesse
ENSAE Paris
robin.hesse@ensae.fr

Abstract

In this report are presented methods aiming to predict market trends using only central bank speeches. The topic is particularly challenging and led us to use innovative methods to extract key features within long texts. In particular, to extract the features, we summarised the text and analysed the sentiment in them and tried several models (LSTM, BERT, Transformers, ...) but our best performing models were linear models where we achieved 70% of accuracy. Our goal was to predict if the value of the stock market one week after the last day of the time series would be higher than a weighted average of the values of the stock series. The code is to find on Github [1] or on Google Colab [2]

1 Problem Framing

Data: We have at our disposal the speeches of the American and European Central Banks (the Fed and ECB) which are published on a daily basis and are followed by all market players as they provide market insights for the time being and for the following days. Our goal was to predict the trend of two indexes measuring volatility: *VIX* (an Equity Stocks Volatility Index of the American markets) and *EURUSDV1M* (the Exchange rate Volatility of the USD/EUR market). In other words we aim to predict if the stock value will decrease or increase (binary classification).

The data was sparse. Indeed, for a given index to predict only one index value there were only a few speeches that were made, extracted and that we could use to strengthen our predictions. Another difficulty was that the speeches we had were quite long (there were 20 176 characters and 3 127 words on average) so we had to identify and extract all relevant information within those speeches. We also checked the languages within these speeches and only kept the speeches that were in English.

2 Experiments Protocol

Summarization: As the speeches were really long, we implemented a method to summarise them by selecting the top k sentences which are containing the words which appear the most (excluding basic English stop words). The summaries we computed were 12 sentences long, which is more or less equivalent to 500 tokens.

Embedding: We implemented several embedding techniques, including statistics methods such as TF-IDF, and more advanced techniques such as SentenceTransformers, a framework for state-of-the-art sentence embeddings. The initial work is described in our paper Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.

Sentiment analysis on summaries: The first way we decided to use our textual data was by applying some sentiment analysis pipeline on a summarised version of each speech. The basic idea behind this is that a speech being classified as negative may induce a higher probability for the stock market value to go down. We computed a sentiment label on each summary among negative, positive or neutral (-1, 0, 1), using a pre-trained model, fine-tuned on some financial data (Sanh et al., 2019).

Sentiment analysis on sentences, asymmetric search Another more original way to generate features we implemented is using asymmetric semantic search, in other words we are computing embeddings of all sentences within a speech and are trying to find which ones are answering the best to a specific query. In practice, we have computed embeddings for all sentences in one speech using the msmarco model, that one can find in the sentence-transformers library (Reimers and

[1] github
[2] colab

Gurevych, 2019). Then we have retrieved the top 5 sentences for which the cosine similarity is highest compared to the following query :

Will the index stock value will go up or down, what is the trend ?

We have finally run a sentiment analysis on these 5 sentences using the previously mentioned pipeline. As a complementary feature we have also retrieved the cosine similarity score as a weight for each sentence measuring their ability to answer the query.

Classification Model: Using the Embedding and/or the sentiment analysis result obtained, we implemented several classification models including: a logistic Regression, SVM, Random Forest, XGBoost, KNN Classifier, Naive Bayes, Gradient Boosting, Volting Classifier an a Sequential Model (LSTM, GRU).

3 Results

The naive model (that predicts both label with probability 0.5) and our logistic regression are used here as the baselines.

What is particularly striking in the results of Table 1 is the fact that the accuracy that we achieved has a very low variance on each dataset, which implies that the additional computational cost and complexity that are implied by a more complex model (for example the Sequential models) do not enable a better performance. So in this case the simpler models should be preferred to the more complex one.

The results on the dataset with embedded test are always lower than the one obtained with the sentiment analysis pipeline. Indeed the information fed with the sentiment analysis pipeline is already well analysed by very powerful pre-trained models and since we only want to extract from speeches a binary information this pipeline is particularly fitted for the task.

4 Discussion/Conclusion

Once implemented, we realized the models were highly sensitive to overfitting and were not performing good at all on VIX, although it had the higher validation accuracy. Tuning the network, adding dropout layers or regularization terms did not really improve anything and we were forced to admit that a better understanding of the features we used was needed before jumping into those kind of complex models. Still, our LSTM model achievew one of the best classification accuracies for EURO index, ie nearly 70%, which led us to think that maybe the model could behave better with a carefully chosen set of features and also maybe reducing the data sparsity, which we guess could represent some issue as well.

Adding the sentiment analysis of one or multiple sentences which achieve the best scores to the given query did not improve the classification scores, everything else kept equal compared to the model described above. Even though the selected sentences seemed like decent choices, maybe we should have test more architectures, other approaches to make a good use of this kind of features.

	Sentiment analysis		Embedded text	
	Accuracy	Time	Accuracy	Time
Logistic regression	59	0,01	58	0,5
SVM	71	0,1	57	0,2
Random Forest	61	0,17	57	0,5
XGBoost	60	0,8	58	1.9
KNN Classifier	71	0,001	56	0,007
Naive Bayes	61	0.002	48	0,006
Gradient Boosting	69	1,8	57	19.3
Volting Classifier	70	0,27	58	0,77
Sequential Model	70	18	61	22

Table 1: Accuracy Score (in %) and Fit time (in seconds) of several models

References

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.