Introduction This blog describes how I analysed central bank policy by means of NLP techniques in a past project. The source code is available in github repo.

**Business Context**

FOMC has eight regular meetings to determine the monetary policy. At each meeting, it publishes press conference minutes, statements as well as scripts in the website. In addition to this regular meetings, the members’ speeches and testimonies are also scripted on the website. At a meeting, the policy makers discuss, vote and decide the monetary policy and publish the decision along with their view on current economic situation and forecast, including Forward Guidance since 2012. The central banks intend to indicate their potential future monetary policy in their publications as a measure of market communication.

The objective of this project is to find latent features in those texts published by FOMC. First, I applied machine learning to economic indices to see the performance of prediction on those numerical data. Then, added pre-processed text data as additional feature in traditional machine learning technique to see if it contains the meaningful information. Finally, apply Deep Learning technique such as LSTM/RNN and BERT to see if these can better predict the rate hike/lower at each FOMC meeting.

**1. Retrieving Market**

Data Daily FED Rate and major economic indices can be obtained from Economic Research in FRB of St. Louis website called FRED: FED Rate GDP CPI / PCE Employment and Unemployment Retail Sales and Home Sales.

***Manufacturing PMI and Service PMI (***formerly known as “Non-Manufacturing Index or NMI) are published by ISM (Institute for Supply Management) website. Daily Treasury yield rates can be downloaded from US Treasury website in xml. Good to explore the details of the data on each website but it’s much more convenient to use Quandl, which provides Web APIs and Libraries to retrieve all the data in the same manner. All the data above are publicly available and free for personal use but you should always check the license terms in the original source in accordance to your objective. Once you create an Quandl Account, API Key is provided. For example, you can download data in python after pip install quandl like this:

Download the following texts from FOMC Meeting Calendar: Statements — available right after each FOMC meeting Meeting Minutes — available three weeks after each FOMC meeting, so may not be available for the latest Meeting Press Conference Transcripts — available at each FOMC meeting but only started in 2011 Meeting Transcripts — available five years after the meeting, so this cannot be used as input for the prediction while still good source to see the detail background for the old meetings Speeches — transcripts are published in this page and I used chair’s speech published between two meetings Testimony — a various testimony texts are also published in this page and I used Semiannual Monetary Policy Report to the Congress.

**3. Preliminary Analysis-**

In order to see if the texts may contain some useful insight to predict FED rate, I used Loughran and McDonald Sentiment Word List to measure the sentiment of statement. This dictionary contains several thousands words appearing in financial documents such as 10K, 10Q and earnings calls categorised to positive, negative, etc. It includes words in different forms, so stemming or lemmatising should not be applied. I applied a simply technique to flip the sentiment for negation (e.g. can’t, isn’t, no). Note that you need to obtain necessary licence for commercial use.

Also check the moving average of net sentiment with actual FED Rate decisions at each FOMC meeting. There’s a certain correlation with FED target rate, but it will not be easy to see during the Financial Crisis where the rate was at Effective Lower Boundary and quantitative easing was taken place. I treated QE announcement as a lowering rate event.

**4. Pre-processing Economic Index**

When FOMC decides the monetary policy, the difference from previous figure is also important. For each index, take difference from the previous period and the same period of the previous year for all the indices.

As a part of feature engineering, calculate taylor rules and see whether the first derivatives and difference from FED rate could be used. FED has released how policymakers use economic indices data on their website. Here, calculate Taylor Rule, Balanced-approach Rule, and Inertial Rule from raw data. The result looks to match with their publication and the correlation between these theoretical rates and actual FED Rates is quite high.

**5. Pre-processing Text Data**

There are around 200 decisions over the last two decades and a half. Depending on the models some inputs cannot be used due to missing data or available timing.

One of common issues you may face during text processing is how to handle long text in machine learning. Most of the neural net based algorithms are not capable to analyse such long texts like 10,000 words — 500 at maximum. Most of our input texts are too long to analyse as a whole document.

Typical solutions to this problem is either to use other algorithms such as jaccard/cosine similarity on the document vectors or to find a way to split the long text or to use some techniques to shorten such as text summarisation.

One simple solution used here is the text split technique to split the text by the number of words (e.g. 200 words with overlapping of 50 words) as shown below. This is a simple automated way but easily lose the context because the extracted 200 words may or may not contain relevant text and even off the topic.

Another issues is data imbalance — in this example, rate decision is “hold” for more than 60% chance and available decisions are only ~200 as the meeting is taken place eight times an year. Without having enough data, machine learning can easily overfit to the training data.

**6. Build and Train ML Models**

At high-level, the model takes textual inputs and meta inputs to predict three classes: Raise, Hold or Lower as follows. The point is how to combine textual input with numerical inputs and there are different ways to implement it.

Here built the following six models in addition to the baseline model. All the models are built and trained in pytorch and the source codes are available in Github repo, of which some parts are extracted in this post.

0. Baseline Model

This does not use textual inputs but just take meta inputs. 14 different classifiers with the default parameters are compared first to grab the baseline performance quickly .

Then, apply RandomizedSearchCV and GridSearchCV to find optimal hyper parameters and decided to use Random Forest as a base because it produces the best result with reasonable feature importance. StratifiedKFold is used for the cross validation.

Learning curve tells that the model overfits to training data and more data could potentially improve the performance. Looking at the confusion matrix, it is failing to predict “Lower” and “Raise” events.

**A. Cosine Similarity**

The texts are vectorised by Tfidf using Loughran-McDonald dictionary, which is used in preliminary analysis, and calculate the cosine similarity between two consecutive meetings. This value is the degree of change in the text direction (i.e. cosine of vectors), which may indicate the policy change. This is then combined with economic indices used in the baseline model.

**B. Tfidf**

Instead of cosine similarity, use the Tfidf vector itself as input. This would only work if the Tfidf vector directly holds meaningful information on the rate change. Using the tokenized text in the previous step, concatenate the Tfidf Vector with Non-textual inputs by FunctionTransfomer.

**C. LSTM**

LSTM (Long Short-Term Memory) is a popular RNN (Recurrent Neural Network) architecture that can hold long-term memory and short-term memory for sequence learning. There are a number of improved versions such as bidirectional LSTM while the one used here is a simple plain model. The output from the deep neural network is combined with economic indices after dropout and dense layer.

Then, separate the input to textual data and numeric data. The data loader is customised to yield the two types inputs. Training process is the same as usual case of processing text by LSTM.

**D. LSTM+GloVe**

The previous LSTM model creates own word embedding but there’s also pre-trained embedding. Here uses Global Vectors for Word Representation (GloVe) which was trained by wikipedia and gigaword (6B tokens). The idea is the pretrained word representation would boost the performance of the randomly initialised model.

The training steps are the same as the previous LSTM model.

**E. BERT**

BERT, or Bidirectional Encoder Representations from Transformers, is a transformer based language model as opposed to RNN, published by Google Research in 2018. The model used here is pretrained BERT\_BASE, which is a deep neural network with12 layers, 768 hidden units, 12 heads, resulting in 110M parameters and was trained on the Wikipedia and BooksCorpus. Apart from the model and BERT’s own tokenisation, the rest of architecture stays the same as the LSTM based model above. Usually it would be sufficient to use transformers.BertForSequenceClassification for the model but here needs to create own definition to concatenate text with non-text inputs.

**F. BERT + Pre-Sentiment Analysis**

Finally, I took another approach — instead of training the model directly on FOMC text, first train the model on other financial texts for sentiment analysis task. Then used the trained BERT model to analyse the sentiment of each sentence in FOMC text and calculate sentiment scores, which were then aggregated for each document and used as inputs to another ML model to predict the FOMC decision.

**Results**

The following shows the comparison of scores between the tested models. The deep neural network models performed poorly, which is basically due to lack of enough data to train these complex models. The last model with pre-sentiment analysis outperformed the other models but need to examine further if this improvement is with meaningful significance and consistency, and not due to just any additional inputs.

**Conclusion**

We examined whether FOMC text data contains useful insight to predict the FED target rate decision (i.e. Raise, Hold or Lower) at the next FOMC meeting. We could observe some useful information in the text to predict FOMC decision better. However, we could not improve the text based prediction performance by Neural Network. This is partly because there are small number of test data to train with each text very long. As a future work, there’re two main areas to improve:

1. Tackle the lack of enough training data — The models have clearly overfitted to train samples and failed to generalise well, especially boosting algorithms are prone to overfitting. Hyperparameter tuning and imputation was considered there. In addition, configuring the model and splitting data to augment the training data by synthetic approach could potentially beneficial.

2. Improve input text quality — The input texts contain a lot of irrelevant paragraphs, which have nothing to to with FED target rate decision. For example, there are information about regulations, organisation structure and infrastructures. Filtering out less relevant inputs will improve the accuracy of the model prediction as well as training efficiency.

Note that the code in this post is just some extracts. Please refer to Github repo for the complete source code.