Stock Return Prediction Using Transfer Learning on Textual Data

Final Project – AC295/CS115 – Milestone 3

TEAM: ABNORMAL DISTRIBUTION
ROHIT BERI, EDUARDO PEYNETTI, JESSICA WIJAYA, STUART NEILSON

Overview

We explore the use of state-of-the-art Natural Language Processing models to assign a sentiment score to financial documents and attempt to predict returns based on this information. Our work so-far has been summarized in this document below. For the details you can refer to the below mentioned notebooks.

Notebooks

- **10Ks_EDA_Model.ipynb:** Data processing, EDA, and transfer learning (tokenization and sentiment extraction) for the S&P500 companies' 10K documents
- **Tiingo_News_to_MongoDB_Atlas.ipynb:** News data extraction and transfer to MongoDB
- **Tingo_News_BERT_Feature_Extraction.ipynb:** Pipeline from MongoDB, Transfer learning using BERT for feature generation and sentiment score, Sentiment Aggregation and Feature extraction for modeling
- **Finnhub full to Mongo.ipynb:** News data exctraction using FinnHub API and transfer to MongoDB
- **Finnhub_article_EDA.ipynb**: EDA on news data from FinnHub
- **TFRecords_Finnhubnews_Pipeline.ipynb:** Pipeline from MongoDB, Transfer learning using BERT for feature generation and sentiment score, Sentiment Aggregation and Feature extraction for modeling
- **Baseline_Model.ipynb**: Outlines our two baseline models with data from both 10-Ks and News.
- **zipline.ipynb**: Pipeline to process stock data in Zipline from Quandl data.

Data Sources

We are using 5-different data sets for this project:

- News dataset from Tiingo comprising of over 27 million (9 million relevant) news summaries.

- New dataset from FinnHub comprising of over 3 million relevant news summaries
- 6000 plus 10Ks downloaded from SEC website
- Events announcements available from Capital IQ
- Fundamental Financial Data and Stock Prices using the Sharadar database from Quandl

Data Storage & Management

We are using 2-main storage mechanisms with multiple file storage systems. For data storage we rely on the following:

- **Google Drive:** Makes is easy to collaborate with team and can be mounted in Colab. Also keeps the costs lower as it is free.
- **MongoDB Atlas:** We use MongoDB Atlas for managing MongoDB Cluster to store text dataset and features.

For filing system, we rely on the following:

- **MongoDB (MongoDB Atlas):** for the raw text data with meta data as well as processed feature vectors.
- **SQL Lite (Google Drive):** for financial data and returns data we use SQL Lite as this format is required for efficient processing with Zipline
- **TFRecords (Google Drive):** for managing pipelines for training models
- Further, we make use of JSON, pkl and csv files as need to store intermediate results

Data Pre-processing

From data pre-processing perspective, our data can be divided into 2-categories:

- Textual Data
- Numeric (Financial) Data

Textual Dataset

The aim of textual data pre-processing is to summaries the textual data into numeric feature vectors. We follow mostly similar process for this step for all 4-textual datasets. The end-output of the pre-processing is:

- 768-dimentional pooled hidden layer representation of the text from the BERT Model
- Sentiment Score for the text from the BERT for Sequence Classification Model.

10K's

Pre-processing of 10K's requires an additional sept of parsing the text from the 10K and summarizing the text. We focus specifically only on item 1a (Risk Factors), 7 (Management's Discussion and Analysis of Financial Condition & Results of Operations), and 7a (Quantitative and Qualitative Disclosures About Market Risk). We used regex to match the item

headline/subtitle and extract the content out of these sections. The contents were saved (in the form of dictionary) for further processing and exploration.

Further, we realize that the extracted portion were long (up to 150,000 words), which create challenges for tokenization and modelling. Hence, we summarize each of the 3 sections' content (item 1a, 7, and 7a) using LSA technique. The final texts we're going to work with will primarily contain less than 512 words, which will work reasonably well with Bert tokenizer.

Fundamental Financial Data and Stock Prices

We obtain daily price data, as well as quarterly financials from the Sharadar database from Quandl. This database contains daily open/close/high/low price and volume data for over 14,000 US based companies and 150 financial indicators.

We aim to simulate a portfolio of the 1500 largest stocks by market capitalization, a universe that varies through time. We also wish to analyze stock returns in different timeframes, such as daily/weekly/monthly/quarterly, while taking into account dividends, stock splits, and de-listings. Finally, we seek to analyze our portfolio by looking into different ways of aggregating its returns, such as by company size, stock price volatility, sector, and industry.

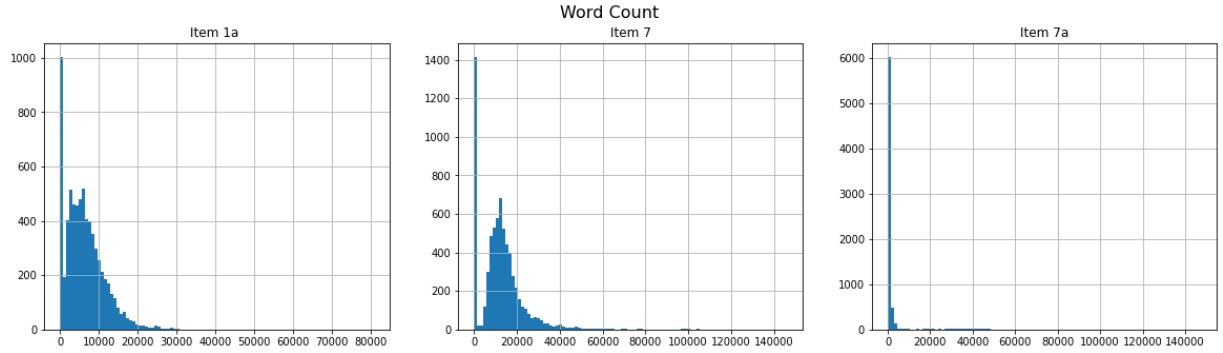
To achieve all these objectives, we use Zipline, which is an open-source library from Quantopian geared towards anlaysis of large portfolios of stocks. We process the raw price data from Quandl into a SQL database into a format that is highly optimized for required calculations such as sorting, moving averages, etc. This is done by a method known as ingestion in Zipline. We also process the fundamental indicator data into numpy sparse arrays that can be loaded into Zipline. Finally, we create a pipeline to load the data into Zipline and calculate any financial information that we require, such as stock returns, market capitalization, volatility, and ranking of sentiment from our different models.

Exploratory Data Analysis

10K's

The exploratory data analysis for the parsed 10K's documents involve the following steps:

- When exploring the data, we realize that a small subset of the documents were either not in standardized format (hence we can't find the contents for item 1a, 7, and 7a), or the output or the parsed texts do not give us meaningful information (e.g. only containing very short sentences, or short paragraphs that refer to other pages/sections instead, etc.). We figured out these issues by plotting the word count for each of those sections, and identify the problematic documents as the big spike with very low word count. These data were then filtered out.

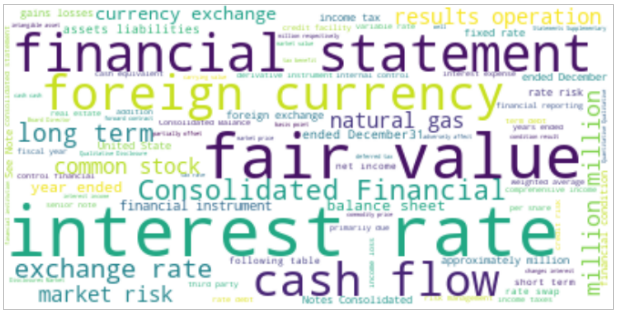


- As an attempt to visualize the data for further EDA, we created some word clouds to illustrate what these contents from item 1a, 7 and 7a mostly focus on. The most common words are displayed in the word cloud for each of item 1a, 7 and 7a. We think that these common words make sense under given section (item 1a: Risk Factors, item 7: Management's Discussion and Analysis of Financial Condition & Results of Operations, and item 7a: Quantitative and Qualitative Disclosures About Market Risk)

Item 1a

Item 7

Item 7a



Transfer Learning

We use *BertTokenizer* (pretrained with '*bert-base-uncased*') to first tokenize the textual datasets. The tokenized texts were then fed in to the *TFBertModel* to obtain the pooled hidden states/feature space, and to *TFBertForSequenceClassification* to get a sentiment from these texts (both were pretrained with '*bert-base-uncased*'). In addition, we also explore other pretrained model FinBert

(Bert that is trained/fine-tuned using financial datasets – "*ipuneetrathore/bert-base-cased-finetuned-finBERT"*).

Finally, we have the following as the feature space using Transfer Learning on the textual dataset:

- BERT Hidden layer representation of text data using TFBertTModel with 'bert-base-uncased'
- BERT Hidden layer representation of text data using TFBertModel with 'bert-base-cased-finetuned-finBERT'
- BERT Sentiment score for the test data using *TFBertForSequenceClassification model with* 'bert-base-uncased'
- BERT Sentiment score for the text data using *TFBertForSequenceClassification model with* 'bert-base-cased-finetuned-finBERT'

The above vectors and sentiment score serves as feature space for our baseline and advanced models for predicting stock returns.

Baseline Models

We have two baseline models:

Loughran McDonald Sentiment:

This is a dictionary-based sentiment model. It is constructed by analyzing word frequencies in 10-Ks and 10-Qs. More information can be found in "https://sraf.nd.edu/textual-analysis/resources/".

A number of studies have shown that dictionary-based sentiment models built rom financial texts show statistical significance in predicting stock returns. For a summary of different methods, please see:
Loughran, Tim, and Bill Mcdonald, 2016, Textual Analysis in Accounting and Finance

Following the suggestions by Loughran and McDonald, we use the positive and negative vocabulary in their dictionary, we calculte text frequency/inverse document frequency (tf-idf). We sum the positive "tf-idf" scores and subtract the negative" tf-idf" scores.

Sentiment from a HuggingFace

We use the BertForSequenceClassification BERT, with a binary softmax layer as a final output layer, without anyfine tuning as a second benchmark. This model model was originally trained with data from the GLUE dataset, as well as reviews such as Rotten Tomatoes for movies. The output for any news or 10-k summary provides a probability of classification which we take as positive if above 0.5 and negative if below 0.5.

We use this model as a baseline as it wasn't trained with financial data, to be able to compare results vs a BERT model trained with financial data.

Next Steps

- Process Finnhub data and add it to all instances of our news dataset for final model training.
- Finetune a model similar to the sentiment BERT baseline model with our data and with stock returns as a noisy classification target.
- Construct a more complex sentiment model using BERT feature vectors and train with our data and with stock returns as a noisy classification target.
- Portfolio construction and profit/loss attribution model. This step will let us know if we can find statistical significance in predicting stock returns using our sentiment scores. We will form daily portfolios comprised of the 1500 highest market cap in our stock universe using sentiment as a ranking tool.
- Profit/loss attribution to common factors such as market capitalization, and stock volatility, sector and industry. Doing this analysis will let us have a better understanding where the statistical significance is coming from.
- Sequential profit/loss analysis predicting returns a day before news came out, the day news came out, and a few days into the future. We will use this as evidence towards our model achieving causality or not. We expect the predictability to decrease sequentially.
- Visualization of BERT and FINBERT layers with methods seen in class.
- Finally, time permitting, we will look into building ensemble using the sentiment scores and the hidden layer features from BERT to build a stock return prediction model. We are currently evaluating between deep-learning based models, decision-tree based models and reinforcement-learning based models.