News Classification

Natural Language Processing and Deep Learning

Hoyt Lui

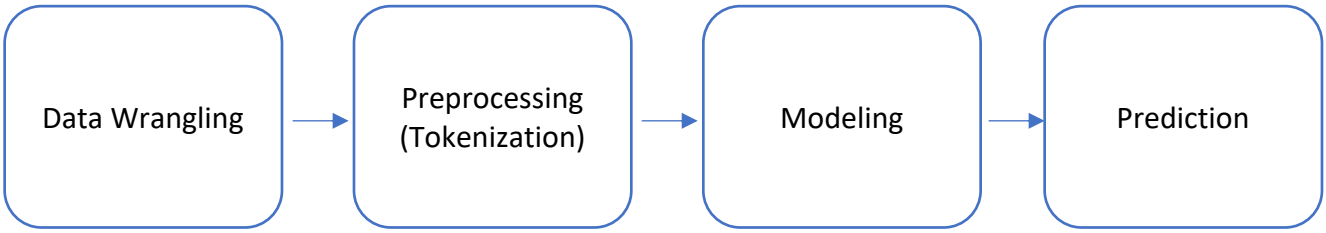
Problem Statement

Natural Language Processing (NLP) can be applied in many areas. In finance, particularly for investing and trading, it is widely used for sentiment analysis. While this approach is applicable to the investing world, we also need to understand that it will have a better result in long-term investing than in short-term trading, mainly because 95% of the traders lose money, and the top-tier traders who make profit consistently are those who set a correct trading mentality for the markets instead of relying on sentiment analysis. In other words, the reason that sentiment analysis does not work well in trading is because many tweets and threads are posted by traders or someone who shows interest in the stocks. And since majority of the traders lose money long-term, most textual data, which is also their opinions, are not too relevant to predict the stock price.

If we agree on NLP driving a better result in long-term investing, then we are also aware that NLP is not too useful for investing purposes. Part of the reason is that long-term investing focuses on company's performance, from products, revenue and cost, to debt, capital expenditure, and profit. It would be better to analyze the company's financial statements than their speeches on either the MD&A (Management Discussion and Analysis) session or earnings call presentation.

So, if NLP is not favorable in long-term investing either, then what is its use for investing and trading? This is how I apply NLP to my trading strategies. News is the trigger to drive a stock price either up or down, depending on the type of news. Some news creates a more longer-term effect, which may push the stock price upward for a few days up to a few weeks. It is also true for the opposite. Some news can plummet the stock price to close to nothing. On the other hand, some news has little effect, while some news is just noise as if there is no news. So, instead of classifying the news as "positive", "negative", or "neutral", which most people on the Internet would do, I would classify news in different terms. The more precise the better, but in the end, it should fit your own strategies to set up for success.

Workflow



Data Wrangling

I downloaded the financial news data from one of the Kaggle datasets. In fact, any news headline datasets will do the work after being categorized.

There are only two columns needed for this analysis – headline and type. At first, the type column contains only empty values because it is newly created. Our job is to categorize as much news in the headline as possible.

First, we will have to convert all text into lowercase, as shown below

	type	headline
0	NaN	stocks that hit 52-week highs on friday
1	NaN	stocks that hit 52-week highs on wednesday
2	NaN	71 biggest movers from friday
3	NaN	46 stocks moving in friday's mid-day session
4	NaN	b of a securities maintains neutral on agilent...
5	NaN	cfra maintains hold on agilent technologies, l...
6	NaN	ubs maintains neutral on agilent technologies,...
7	NaN	agilent technologies shares are trading higher...
8	NaN	wells fargo maintains overweight on agilent te...
9	NaN	10 biggest price target changes for friday

Then we will determine which categories are needed for classification. In this project, we classify news in 7 types.

0. Neutral. Any news that has little or no impact on stock price movement.
1. Analyst action. Initial coverage, ratings or price target adjustment by analysts.
2. Earnings. Earnings results or related news.
3. Corporate action. Dividend or share repurchase program announcement, new contract or partnership, settlement of litigation, etc.
4. Merger and acquisition. M&A with other companies.
5. Company guidance. Company providing estimates guidance to their future performance.
6. Options. Any news related to options activities with the company.

And these are the distributions of the news in each category.

```
neutral               246174
corporate_action      150656
analyst_action        123136
earnings               48158
options                17875
merger_acquisition     16471
company_guidance       14758
Name: type, dtype: int64
```

Close to 40% are categorized as Neutral, followed by 25% in Corporate Action, then 20% in Analyst Action. The rest are made up by Earnings, Options, Merger and Acquisition, and Company Guidance. Their relative percentages are stated below:

	count	%
0	246174	39.883803
3	150656	24.408484
1	123136	19.949840
2	48158	7.802303
6	17875	2.896012
4	16471	2.668544
5	14758	2.391013

Data Preprocessing

We split dataset into 80/20 training and test sets. Then we process the data using tokenization.

What is tokenization? And why is it important?
Tokenization in lexical analysis means that a text string is broken down into chunks of words, each word is called a token. These tokens will then be passed on to some other form of processing for further analysis.

Computers are different from human. They do not know the meaning of words. However, we can assign meaning of the words to the computers. Then they will categorize the assigned words into that meaning the next time they encounter the same word, which is the essence of tokenization – breaking down the words for the computer to understand.

We declare the variables for tokenization purposes as followed:

- Input dimension. We assign computer to memorize 1000 vocabulary words as the input dimension.
- Output dimension, also dense embedding dimension. Here, we set 16 as the shape of the output dimension.
- Input length. It is the length of input sequences. Without it, the shape of dense outputs cannot be computed. We set the max input length to be 142.
- OOV token. We assign "OOV" (out-of-vocabulary) so that it will be added to the word index as a complement to the existing vocabulary words.

What is turning texts into sequences? Why is the concept important?
After creating a tokenizer object, we will convert words into sequences of integers that is indexed with the vocabulary words we set earlier, i.e., 0-999 in this case. This process will have the program returned a list of words (or tokens).

For example, we have 4 words:
'elon'
'musk'
'loves'
'dogecoin'

After we turn text into sequence, we now have ['elon', 'musk', 'loves', 'dogecoin'], or in computer vision, e.g., [586, 587, 23, 748].

What is sequence padding? And why is it important?

Next, we pad sequences into all same-length sequences. The process is mandatory because all neural networks require inputs to have the same shape and size, as it is reading an array as an input. Therefore, we ensure all sequences having the same length by truncating and/or filling 0 in them.

- Max length. We set the max length to be the same input length we set for tokenizer.
- Truncating type. Since I think key ideas are usually presented early in the sentence, I set "post" as the truncating type, so any sequence longer than the max length will be truncated on the end. You can set "pre" alternatively to have it cut on the beginning otherwise.
- Padding type. I set "post" as the padding type, which means adding 0's on the end of any shortened sequences. Again, you can set "pre" alternative to add 0's on the beginning of sequences if you prefer to.

For example,
[['amd', 'will', 'supersede', intel', 'in', 'sales'],
['in', 'the', 'hunt', 'for', 'life', 'outside', 'earth']]

These 6-word and 7-word sequences will look like below in computer vision:
[[96, 34, 452, 194, 22, 48],
[22, 13, 56, 25, 65, 235, 385]]

If we set the max length at "8" words, and pad the sequences with "post". Then the computer will receive:
[[96, 34, 452, 194, 22, 48, 0, 0],
[22, 13, 56, 25, 65, 235, 385, 0]]

If we set the max length at "7" words, and pad the sequences with "pre". Then the computer will receive:
[[0, 96, 34, 452, 194, 22, 48],
[22, 13, 56, 25, 65, 235, 385]]

If we set the max length at "6" words, and truncate the sequences with "post". Then the computer will receive:
[[96, 34, 452, 194, 22, 48],
[22, 13, 56, 25, 65, 235]]

If we set the max length at "5" words, and truncate the sequences with "pre". Then the computer will receive:
[[34, 452, 194, 22, 48],
[56, 25, 65, 235, 385]]

Modeling

What is Sequential model? How does it differ from Functional model?
In short, sequential model is a linear stack of layers that creates layer-by-layer for problems. It is straightforward and easy to set up compared to functional model, which is more flexible and capable of creating complex networks that specifically connect different input and output layers. For this project, we can build a simple model using sequential model to group a linear stack of layers into it.

Layer selection – why pooling layers?
Pooling layers use pooling operation to calculate the value in each patch of each feature map. (Max pooling will calculate the maximum value, while average pooling will calculate the average value.) The problem with feature maps is that they record precise position of the input features, meaning small movements in the position of the input feature will result in a different feature map. By using pooling layers, the results are down sampled (or pooled) feature maps that summarize the presence of features in the patches of the feature map.

This can be the most activated or the average presence of a feature in the patch, depending if you use max pooling or average pooling methods.

I used average pooling layers because I wanted to find the average, or more common, value in the news headlines. However, you can try max pooling layers to return the maximum value. You can also try the other layers such as recurrent layers (especially LSTM) or activation layers, which are also the popular alternatives.

Parameters are set below for building the neural network:

```
# build neural network
model = Sequential()
model.add(Embedding(vocab_size, embedding_dim, input_length=max_length))
model.add(GlobalAveragePooling1D())
model.add(Dense(32, activation='relu'))
model.add(Dense(7, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

First off, note that this model consists of one 1000-neuron input layer, one average pooling layer, one 32-neuron hidden layer, and one 7-neuron output layer.

- Activation. I set “relu” (or rectified linear activation function) so that it returns all positive values in the hidden layers. In the output layer, I set “softmax” as the function that normalizes the output to a probability distribution over 7 predicted classes in this case.

- Loss. “Categorical cross-entropy” is used because the output can only belong to 1 out of 7 possible categories. The lower the loss score, the lower the variance, the more accurate the model.
- Optimizer. Adam optimization is used, which applies stochastic gradient descent (SGD) to allow the function to jump to better local minima.
- Metrics. Accuracy is used to calculate how often predictions equal labels.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 142, 16)           16000
_________________________________________________________________
global_average_pooling1d (Gl (None, 16)                0
_________________________________________________________________
dense (Dense)                (None, 32)                544
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 231
=================================================================
Total params: 16,775
Trainable params: 16,775
Non-trainable params: 0
_________________________________________________________________
```

We set epoch, which is the hyperparameter defining the number of times that the learning algorithm will work through the entire training dataset, to be 6. As shown below, the optimal model should be built when epoch reaches 4-6 times, where we start to see losses between training and validation sets narrow and almost touch at the 6th time – 0.0393 vs 0.0392. Therefore, we should stop further training to avoid over-fitting the model.

In this model, the model accuracy is around 98.6-98.7%.

```
Epoch 1/6
15431/15431 [==============================] - 14s 881us/step - loss: 0.5934 - accuracy: 0.7945 - val_loss: 0.0948
- val_accuracy: 0.9748
Epoch 2/6
15431/15431 [==============================] - 14s 875us/step - loss: 0.0812 - accuracy: 0.9780 - val_loss: 0.0595
- val_accuracy: 0.9824
Epoch 3/6
15431/15431 [==============================] - 13s 862us/step - loss: 0.0544 - accuracy: 0.9841 - val_loss: 0.0480
- val_accuracy: 0.9854
Epoch 4/6
15431/15431 [==============================] - 13s 851us/step - loss: 0.0458 - accuracy: 0.9860 - val_loss: 0.0438
- val_accuracy: 0.9858
Epoch 5/6
15431/15431 [==============================] - 13s 857us/step - loss: 0.0414 - accuracy: 0.9869 - val_loss: 0.0411
- val_accuracy: 0.9867
Epoch 6/6
15431/15431 [==============================] - 13s 847us/step - loss: 0.0393 - accuracy: 0.9871 - val_loss: 0.0392
- val_accuracy: 0.9872
```

Testing

The sample phrases below are used to test the validity of the model. And it results in getting 9 out of 9 correct. This result is amazing but also expected as the model has a 98.7% accuracy.

```
('Pfizer receives FDA approval for its COVID-19 vaccination',): corporate_action
('Tesla expects to deliver 350,000 cars in the next quarter, estimates revenue increased by 5%',): company_guidance
('Intel plans to acquire Disney to expand its consumer sector',): merger_acquisition
('Apple announces a share repurchase program for up to $25 million',): corporate_action
('Microsoft ex-CEO Bill Gates says he loves China',): neutral
('Elon Musk will go to the moon with his dogecoin',): neutral
('Morgan Stanley analyst upgrades Tesla to Buy, raises price target to $4,000',): analyst_action
('AMD Q4-2025 Adj. EPS $1.89 beats $0.85 estimate, sales $3.37B beat $1.33B estimate',): earnings
('Notable put options activity in Gamestop',): options
```

Predicting

We further used the model to predict the 100 most recent news of Tesla. And we found that 50 out of 100 are neutral, which means the news are meaningless. Logically speaking, it is true because most news is not significant enough to move the stock price. 36 of the news are related to earnings. It looks reasonable as well, because they just announced their next earnings call for this month few hours prior to the time I ran the code, creating a lot of hype and rumors around their earnings presentation.

```
0    50
2    36
4     6
3     5
5     2
6     1
Name: sent, dtype: int64
```

```
{0: 'neutral', 1: 'analyst_action', 2: 'earnings', 3: 'corporate_action', 4: 'merger_acquisition', 5: 'company_guid
ance', 6: 'options'}
```

Takeaway

This model tells us few key takeaways:

- It reaches optimal after running epoch around 6 times
- It produces a 98.7% accuracy of news classification
- It predicts 9 out of 9 sample news correctly
- It classifies half of the most recent news of Tesla in neutral, which reinforces most news causes little or no impact to the stock price movement

Further work

3 approaches can be considered for further improvement of the model:

- Make the dataset more balanced by adding more news in the categories other than 'neutral'.
- Try other neural network models and add more layers to further optimize the model.
- Hand-label the headlines. It will be time-consuming but turns out to be more accurate because some news may have subtle or alternative meaning with the same words. Accuracy figure may decline, but this will be a better way to apply artificial intelligence based on human intelligence.