Stock Market Prediction
Predicting stock market movement from news text.

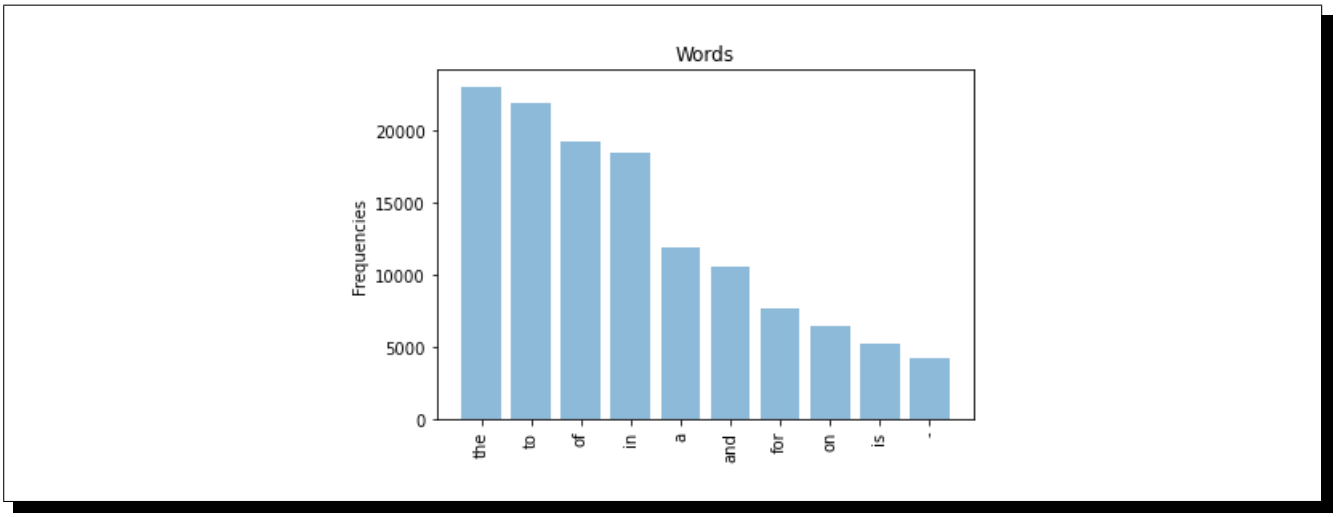
Omar Jarir m20201378
Chung-Ting Huang m20210437

Part I

Data Exploration:

In this data set, our target variable is the closing status. 1 means the index closing value rose or stayed the same, whereas 0 means the index closing value went down.

To understand our data, we start by analyzing the most common words in our corpus.



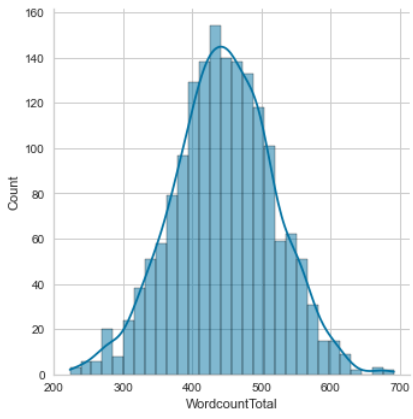
From the figure above, we see that the most common words are stop words. They do not provide any useful information for the machine learning models to be developed and therefore should be dropped from the corpus.

Next, we analyze the word counts for each headline. The most meaning full statistics are summarized in the following table.

	count	mean	std	min	25%	50%	75%	max
Wordcount1	1690.0	16.831361	11.182562	2.0	9.0	12.0	23.00	56.0
Wordcount2	1690.0	16.856805	11.001193	3.0	9.0	12.0	23.00	55.0
Wordcount3	1690.0	16.500000	11.035239	2.0	9.0	12.0	22.00	58.0
Wordcount4	1690.0	17.403550	11.597727	3.0	9.0	12.0	24.00	59.0
Wordcount5	1690.0	17.588757	11.502289	3.0	9.0	13.0	24.00	56.0
Wordcount6	1690.0	18.186982	11.578571	3.0	9.0	14.0	25.00	56.0
Wordcount7	1690.0	17.005325	11.545338	3.0	9.0	12.0	23.00	59.0
Wordcount8	1690.0	16.689941	10.814150	3.0	9.0	13.0	22.00	56.0
Wordcount9	1690.0	18.956213	11.572634	3.0	10.0	15.0	26.00	54.0
Wordcount10	1690.0	17.674556	11.332872	3.0	9.0	13.0	24.00	57.0
Wordcount11	1690.0	16.466272	11.032290	2.0	8.0	12.0	22.00	55.0
Wordcount12	1690.0	18.110059	11.189088	3.0	10.0	14.0	24.00	54.0
Wordcount13	1690.0	18.874556	12.162055	3.0	9.0	14.0	26.00	64.0
Wordcount14	1690.0	18.633728	11.762206	3.0	9.0	14.0	26.00	53.0
Wordcount15	1690.0	19.365680	11.622899	3.0	10.0	16.0	27.00	59.0
Wordcount16	1690.0	18.288757	11.568600	2.0	9.0	14.0	25.00	58.0
Wordcount17	1690.0	17.923669	11.381618	2.0	9.0	14.0	25.00	55.0
Wordcount18	1690.0	17.292308	11.217891	2.0	9.0	13.0	24.00	55.0
Wordcount19	1690.0	18.646746	11.486757	3.0	10.0	15.0	26.00	57.0
Wordcount20	1690.0	17.752663	11.428326	3.0	9.0	13.0	25.00	61.0
Wordcount21	1690.0	16.496450	11.017949	3.0	9.0	12.0	22.00	57.0
Wordcount22	1690.0	19.701183	11.795728	2.0	10.0	17.0	27.00	55.0
Wordcount23	1690.0	16.959763	11.535957	2.0	9.0	12.0	23.00	58.0
Wordcount24	1690.0	18.023077	11.567141	3.0	9.0	14.0	25.00	55.0
Wordcount25	1690.0	19.109467	11.898096	3.0	10.0	15.0	26.75	56.0

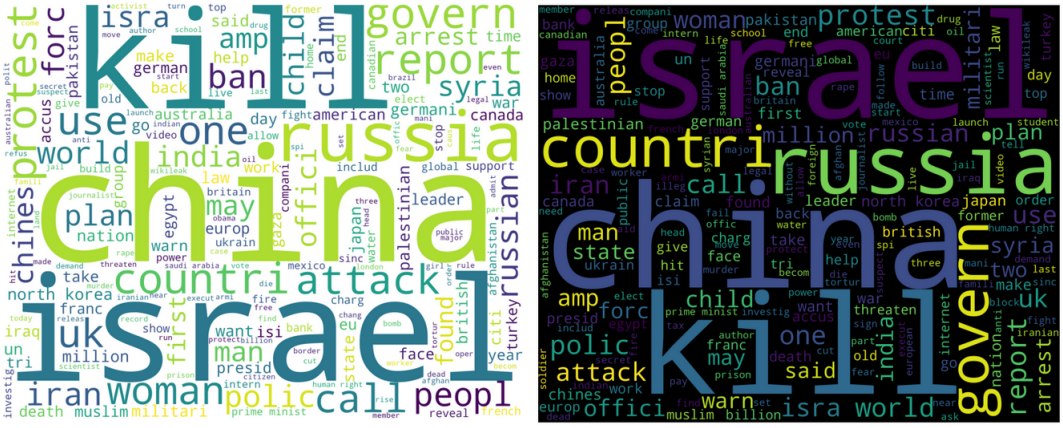
From the table above, we see that the headlines have very similar statistics. The maximum number of words in a headline is equal to 61, while the minimum number of words is equal to 2. Most headlines have few words (less than 25). Between the headline columns, the word count distribution is similar.

Then we count the total number of words per row in our training set. It has the following histogram.



The words count distribution seems to be normally distributed, with a mean value equal to 445.34 and a standard deviation equal to 72.44.

Before data preprocessing, we check the word cloud of the headlines excluding stop words. We get the two following words cloud. On the left is where the closing status was 1, whereas the one on the right had closing status 0.



As we can see both classes share common words such as “china”, “israel”, “kill” and “russia”.

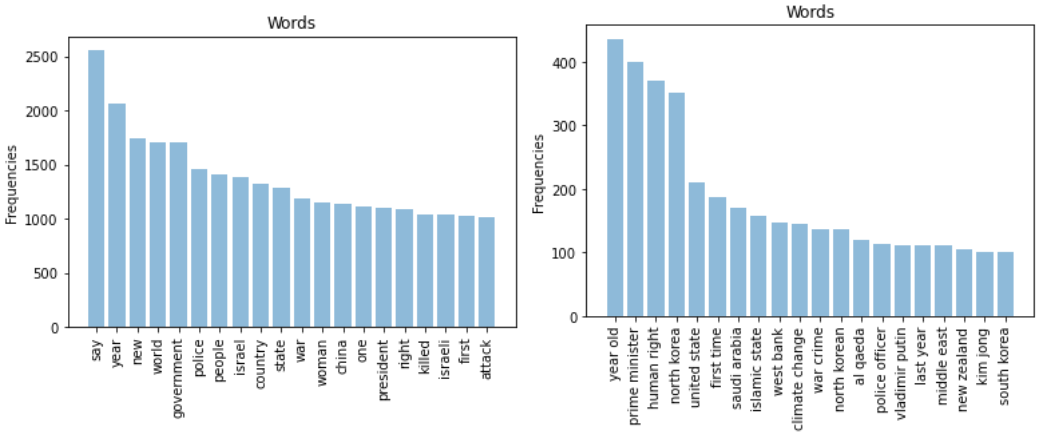
Part II

Data Preprocessing:

The following steps are taken to clean and preprocess the corpus:

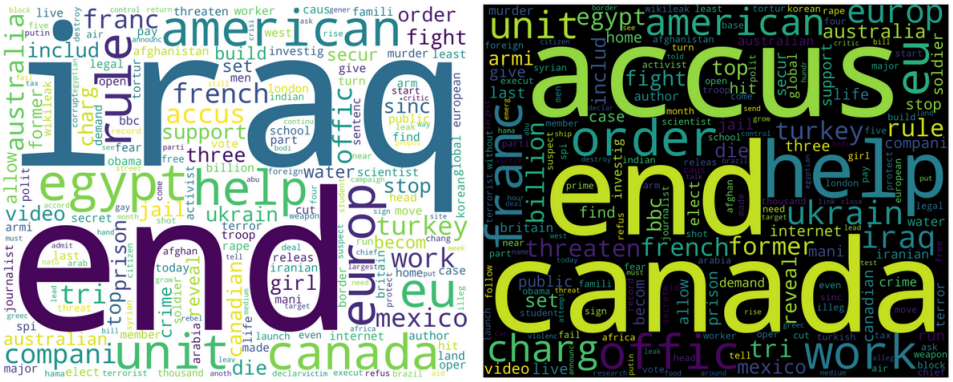
- Making sure all words are in lower case.
- Remove numbers, punctuation, tags, hashtags, links, abbreviations, and white spaces.
- Remove single character word.
- Remove stop words, which are the most common words in the corpus.
- Stemming.
- Lemmatize.

Next, we explore the cleaned data by investigating the most common n-grams.



As we can see the word “say” is the most common in the data set with over 2500 observations. In the case of the bi-grams “year old” is the most common with over 400 observations, followed by “prime minister”.

After removing the most common shared words in both categories of the target variable, we get the following word clouds.



As we can see from the images above the two categories do not share much common words now, this will help increase the predictive power of our models.

Part III

Modeling:

1 Feature engineering:

We use the following methods to transform our corpus into features that can be used as inputs to our models:

1. Bag of words: We use the CountVectorizer function from scikit-learn to create a bag of top 300 words that appear in at least 10 documents. We set the parameter ngram_range= (1,2) to generate unigrams and bi-grams.
2. TF-IDF: We use the TfidfVectorizer function to create the matrix of top 300 TF-IDF unigrams and bi-grams that appears in at least 10 documents and less than 80% of the total documents.
3. HashingVectorizer: This is similar to CountVectorizer, but instead of the actual words, the hashes of the words are stored to reduce memory footprint. It converts the text documents into a matrix of tokens occurrences. We use tokens of uni-grams and bi-grams.
4. Word embedding: We use the Google News data set that has 300-dimensional vectors for 3 million words. For each headline, we take the sum of the vector of each word then take the average.

2 Classification models

We decided to implement many models to compare their performances:

- k-nearest neighbors' algorithm.
- Logistic regression.
- Gaussian Naive Bayes.
- Random forest classifier.
- Artificial neural network.
- Ada boost.
- LSTM.

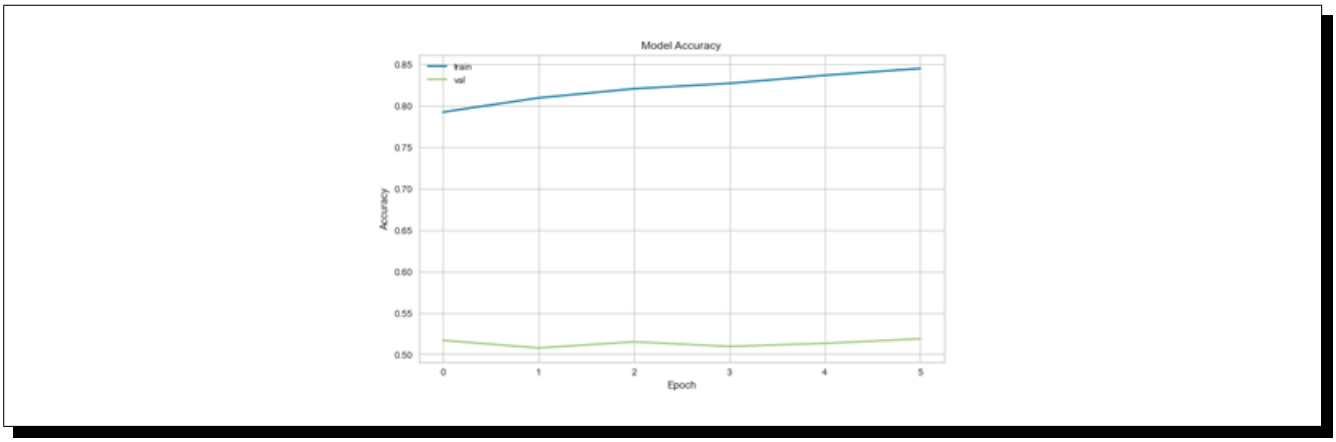
Part IV

Evaluation and Results

Using the scikit-learn RepeatedKFold function we evaluate the performance of our models using 3-fold cross validation repeating 20 times on the whole training data set.

	Recall	Precision	Accuracy	F1-Score	AUC	Best performing feature
KNN	0.768	0.713	0.710	0.709	0.773	TF-IDF
LR	0.807	0.780	0.775	0.775	0.860	Hashing
NB	0.702	0.780	0.733	0.732	0.786	TF-IDF
RF	0.832	0.622	0.637	0.618	0.708	Hashing
MLP	0.850	0.755	0.764	0.753	0.855	TF-IDF
ADA	1.00	0.534	0.535	0.374	0.833	TF-IDF

We also tried LSTM and got the following accuracy curve. The model over fits the training data.



From the table above we can see that Logistic Regression and Artificial Neural Network are the best two model, now we are going to fit these models on the training data set then assess the performances on the validation set obtained by splitting the original training set using the scikit-learn function train_test_split, the validation set size is equal to 20% the size of the original training data set. The split procedure uses stratify sampling to ensure training and testing set has the same ratio of closing status 1 / closing status 0.

The obtained results are as follow:

	Measure	Train	Test
0	ACCURACY	0.91	0.54
1	PRECISION	0.91	0.56
2	RECALL	0.91	0.54
3	F1 SCORE	0.91	0.53
4	AUC	0.91	0.53
5	AUC_Proba	0.97	0.51

	Measure	Train	Test
0	ACCURACY	0.84	0.51
1	PRECISION	0.84	0.54
2	RECALL	0.84	0.51
3	F1 SCORE	0.84	0.51
4	AUC	0.84	0.51
5	AUC_Proba	0.92	0.52

On the left-hand side, the results of the logistic regression model, on the right-hand side the results of the artificial neural network. These results show clear over fitting on the training data set, this might be due to the training set and the validation set not sharing the same tokens.

Our final prediction uses MLP since it is one of the best performing models in cross-validation. Although logistic regression has slightly better performance while using cross validation, the performance between train and test has a wider gap, which suggests it's more over fitted.