Search Engine and Text Classifier

Buse Felamur  
*Faculty of Engineering, Environment and Computing*  
*Coventry University*Coventry, England  
[felamurb@uni.coventry.ac.uk](mailto:felamurb@uni.coventry.ac.uk)

This report is about the implementation of a search engine and its text classifier. Python 3 was used for coding and all of the code were written on jupyter notebook. Report consist of two major parts. First part talks about search engine (crawler, indexer and query processor), second part is about text classification and lastly, in appendix source code and dataset which was used for machine learning can be seen.

1. Search Engine

Type of the search engine made for this project is a vertical search engine. It can only search for news articles.

* 1. Crawler

Crawler, also known as a web crawler, browses the internet for indexing contents of everything or just specific websites. For this project, five different sources were used. These source are all RSS feeds and our crawler reads them. These five feeds are;

* http://feeds.bbci.co.uk/news/world/rss.xml
* http://feeds.bbci.co.uk/news/uk/rss.xml
* http://www.independent.co.uk/news/uk/rss
* https://rss.nytimes.com/services/xml/rss/nyt/World.xml
* https://rss.nytimes.com/services/xml/rss/nyt/US.xml
* feeds.washingtonpost.com/rss/world?tid=lk\_inline\_manual\_13

Crawler start with loading data from our selected feeds by using feedparser. If it is the first-time code is running and csv file is empty, then code puts column names, first ever feed and its data as first row to csv file. This part is very similar to how rest of the feeds were handled too. If our file is not empty then we load it into a dataframe. Crawler part of FirstTime function and a non-empty csv file can be seen in the following figures;



Figure 1 - Crawler part of firsttime function

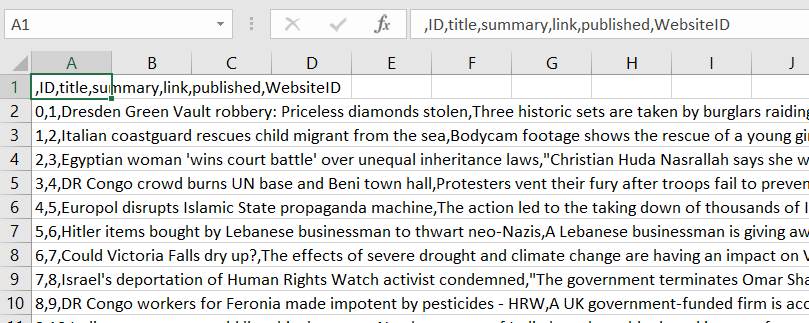


Figure 2 - Filled csv file

After that, we look all of the data that were loaded from feeds to see if any of them already included in the csv file. We compare every loaded feed’s link to the ones in the dataframe. New news is changed into a readable format with feedcompiler function and then added to a newer list with a new ID given by us. This ID is separate from index of dataframe and unique to every single news. If there are new news then this new list is converted into a dataframe and added to our main dataframe. After that we write this to our csv file and return the new news dataframe. Code can be seen in the following figure.



Figure 3 - crawler

* + 1. feedcompiler Function

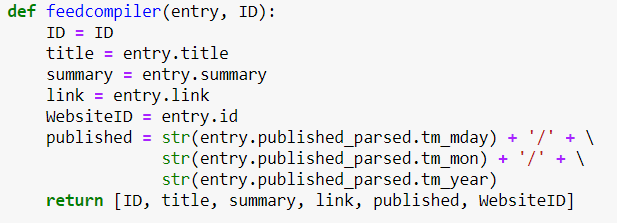


Figure 4 – feedcompiler

This function takes one entry and our assigned ID as inputs. From the entry, it extracts title, summary, link, data it is published and websites own id. It transforms data to day, month and year format. In the end, returns all data as a list.

In the main endless while loop, we use this crawler function to get all new news as a dataframe from the feed list. This dataframe edited and used for indexing which will be explained at the next part.

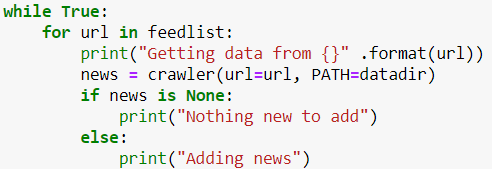


Figure 5 - endless while loop

* 1. Indexer

For this part, an inverted index is picked. Inverted index stores a mapping from the data it is given. It stores these data and its locations to a document or a table. Our index can be seen in the next figure.

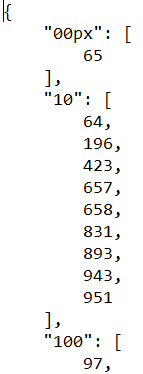


Figure 6 – Index

Our index stays in a json file in the json format. This picture means number “10” appears in these 9 news. Numbers representing the news are the IDs we gave them before storing the news.

As mentioned before crawler returns new news in a dataframe. In this part, title and summary columns of dataframe is fused and then edited with fixtext function which can be seen in the next figure.

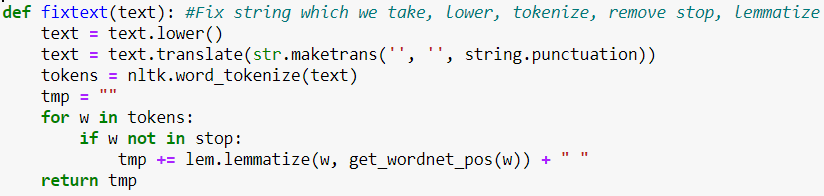


Figure 7 - fixtex function

This function lowers, tokenize and lemmatize the given text. Lemmatizing converts words into its base forms, similar to stemming but this time, code considers the context of the word. For example, when stemming the word “caring” might turn into “car” but with lemmatizing it turns into “care” instead. (Prabhakaran, 2019)

Later, we drop unused columns from our dataframe to prepare it for indexing. We assume there is already a json file exits. There can be data in it or it can be just empty. Empty file includes only closed curly brackets. We load this json file to use for later. Both the json file and the fixed dataframe used for the next step. We go all over each rows of the dataframe. Each row is checked word by word to see if word already exist in the json file. If word doesn’t exist then we create a new pair. If word is already here then it’s unique ID is checked and added to list if ID doesn’t appears. In the end we save these to the json file.



Figure 8 - index code

Next part of the code is for ranked retrieval. Purpose of this is displaying more relevant pages related to the given query. Search engine needs more information about our news for to do that. Method chosen for this is averaged word2vec. This model is a pre-trained single hidden-layered neural network. Shortly, with this model, we create a vector from each word and average them per news article.

We use our edited dataframe again. We go over all of its data again, word by word. Find its vector average, save into a dataframe and add new averages to csv file. While loop then goes into a sleep for an hour.

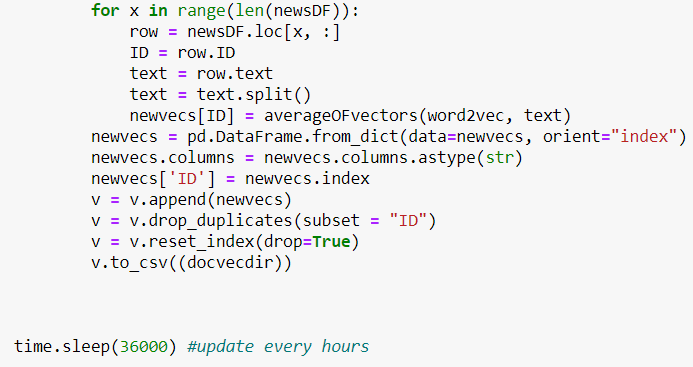


Figure 9 - word2vec part

* 1. Query Processor

Engine starts with asking the user to enter the query. Query can take any type of query. However, it seems to work better with shorter queries. When user enters it, code saves it. Loads our pre-trained classification model to find out which class this new query belongs to and shows it to user.

After that user is asked if they wish to enter a date. Depends on the answer, code might ask user to enter date. Results are shown after that.

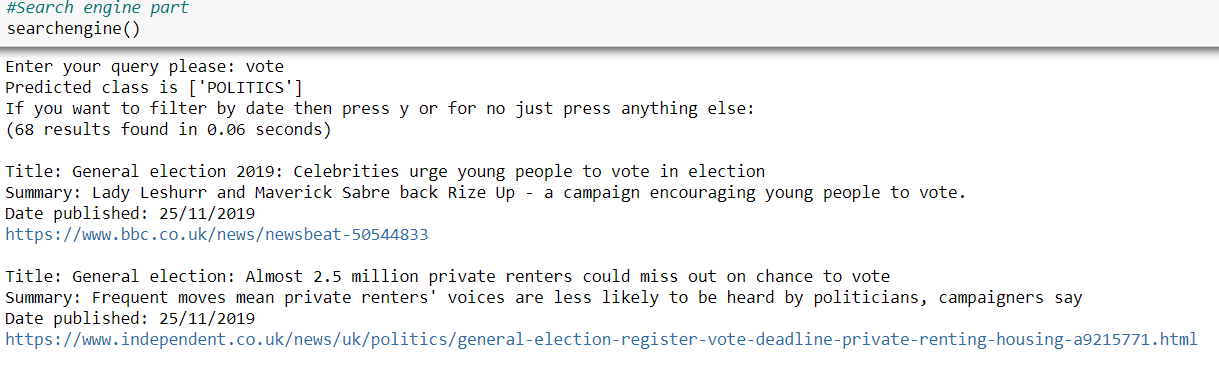


Figure 10 - search engine results

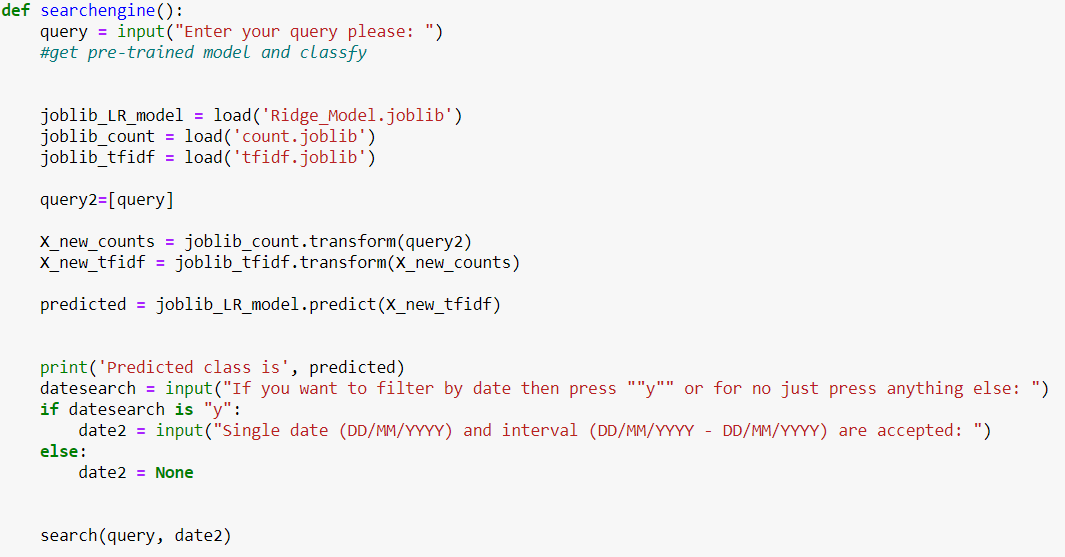


Figure 11 - search engine

In the search function, first we fix query similar to what we did in the index part of the code. Function we used, fixtext, is same as the one before. We go over the words in our query to see if they are in the index. If positive then we add it to our newly created list. If this list is not empty then we look for intersection of these list and return the rows of these intersections.

Now let’s look at the word2vec vectors we were keeping in a document. At first, we compute average vector of our query. Take the vectors of those were retrieved in the previous step. Cosine similarity (Karani, 2018) is used to compare and then order the results. We used cosine similarity because it measures the similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. There wasn’t any pre-made function for it so spatial.distance.cosine from SciPy was used. However, this returns the distance of vectors instead of similarity. It can be fixed by subtracting value from 1.

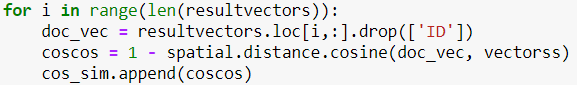


Figure 12 - cosine part

Next part of the code checks if user entered the date or not. If entered we save the date or the interval of dates. We take the articles with the same date. If not entered we carry on.

In the end we show the results by relevance.

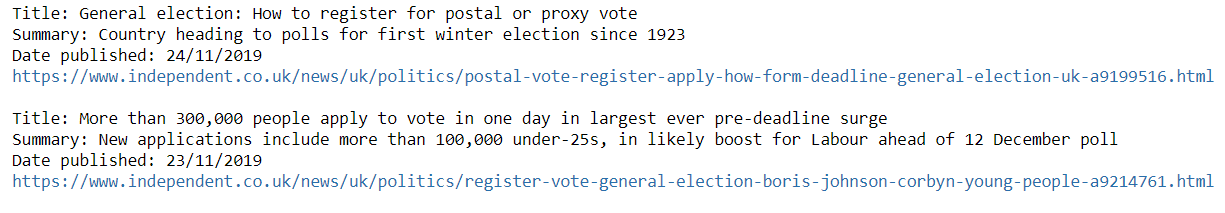


Figure 13 - example results

1. Text Classification

Dataset used for text classification is from Kaggle and it is called News Category Dataset. (Misra, 2018) This dataset contains around two hundred thousand news between the years 2012 and 2018. There are 41 different categories for news. All of the data stored in a json file.

First, we start with reading the json file and putting it into a dataframe which looks like in the figure.

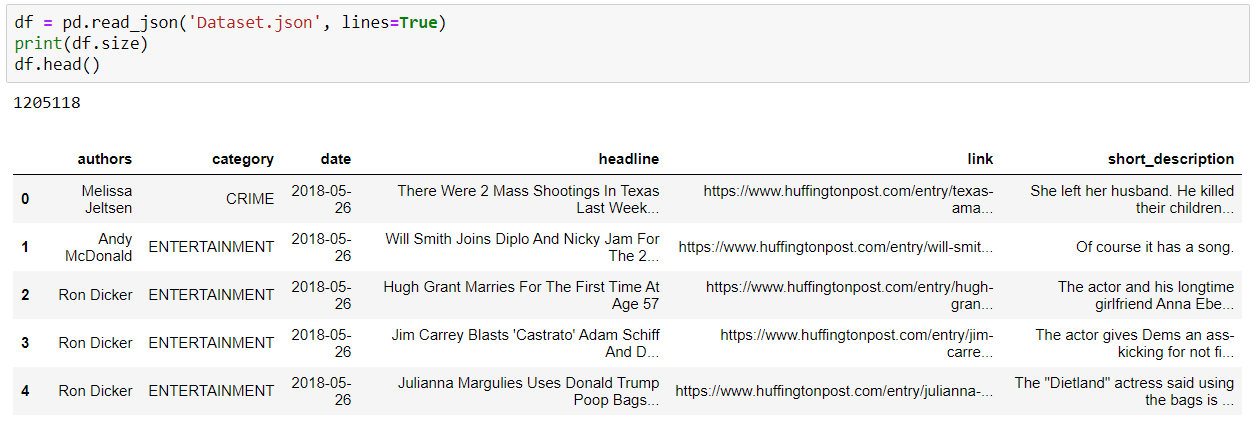


Figure 14 - json to dataframe

After that we remove the useless columns like link, date and authors. We fuse the headline and short\_description under text column so we can drop them.

Now we take a look at category column. There are categories we won’t use so we dropped rows with that categories. There are categories with similar content so we fused them together under a new category type. Last change we did with the data is balancing. Some news has more data than rest of them so we randomly dropped rows to make data more balanced. We don’t want our classifier to have bias to one category. Our new dataframe looks like in the following figure.

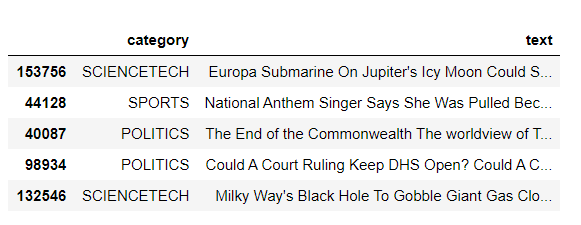


Figure 15 - edited dataframe

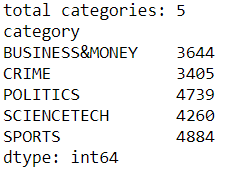


Figure 16 - new categories

After that it is decided that politics category needs to be updated to match with newer news so headlines and some part from the actual news were taken from websites and put into a txt file. We take these sentences and put into a dataframe and then add into our main dataframe.

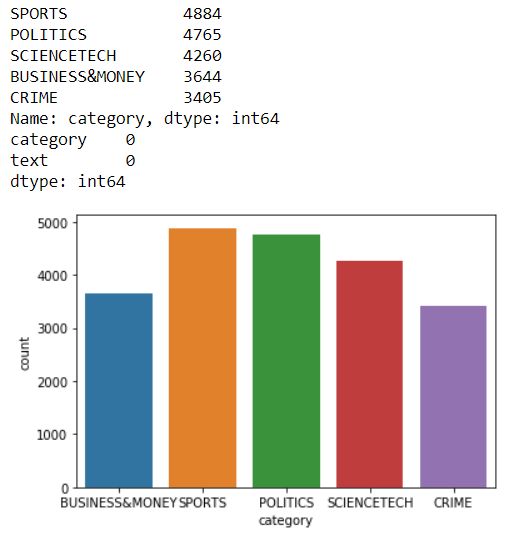


Figure 17 - Graphic for new category sizes

Now it is time for training the dataset. To do that first we split the data into test and training sets. After that we use CountVectorizer to text preprocessing, tokenizing and filtering the stopwords. It makes a dictionary of features and turns our data into feature vectors. (scikit-learn, 2019)

This causes a small problem if document sizes are too much different. Smaller files have lower average count value than longer ones even if their topics are the same. To solve this problem, we can use a feature called term frequencies. This feature divides the number of occurrences of every word in a document by the total number of words in that document. On top of that we can add one more feature. Downscaling the weights for words that appear in most of the files and therefore isn’t very much informative. This is called term frequency times inverse document frequency as known as tf-idf. We used TfidfTransformer to compute this. Code can be seen in the following figure. (scikit-learn, 2019)

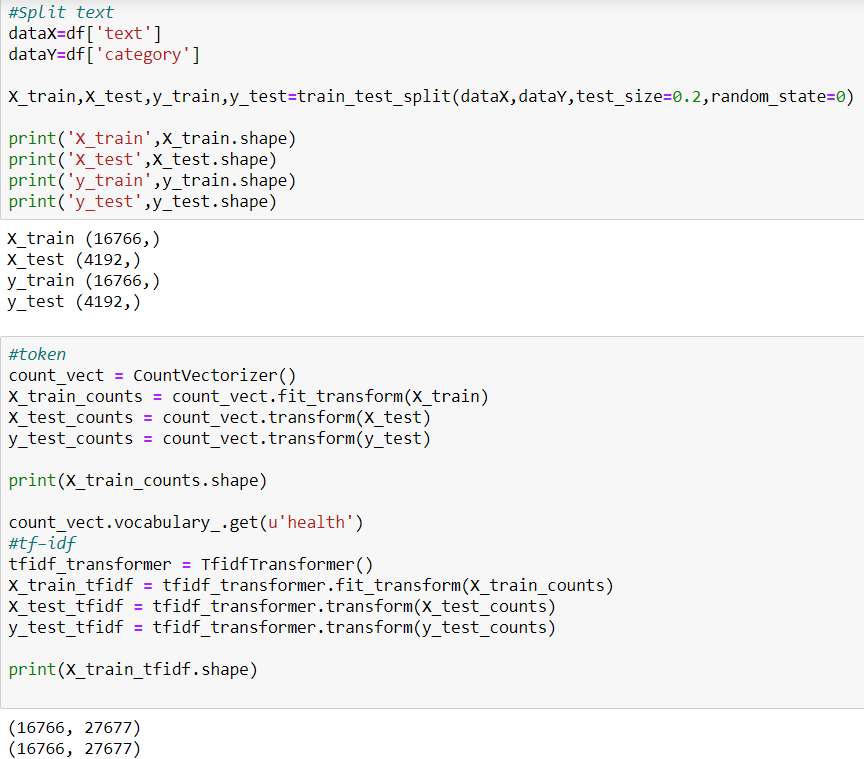


Figure 18 - Splitting, count and tf-idf

Finally, we can train our dataset. Naïve Bayes and Ridge Regression were used for this. Naïve gave us accuracy of 0.9088 and ridge gave us accuracy of 0.9788.

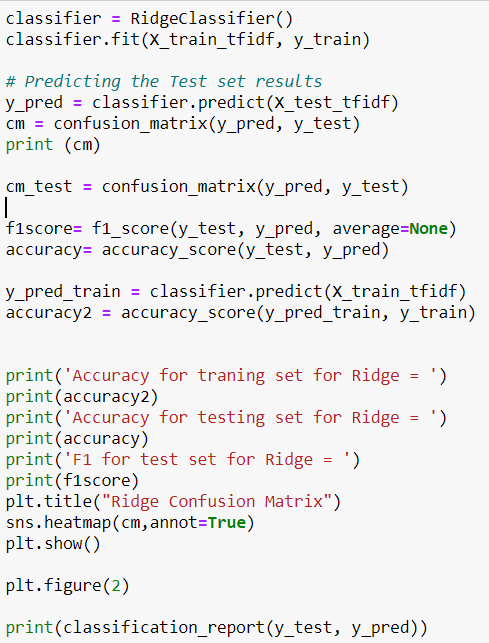


Figure 19 - code for training

According to these ridge gave us better accuracy so it is decided to use that one instead. Confusion matrix and overall report for both can be seen in the following figures.

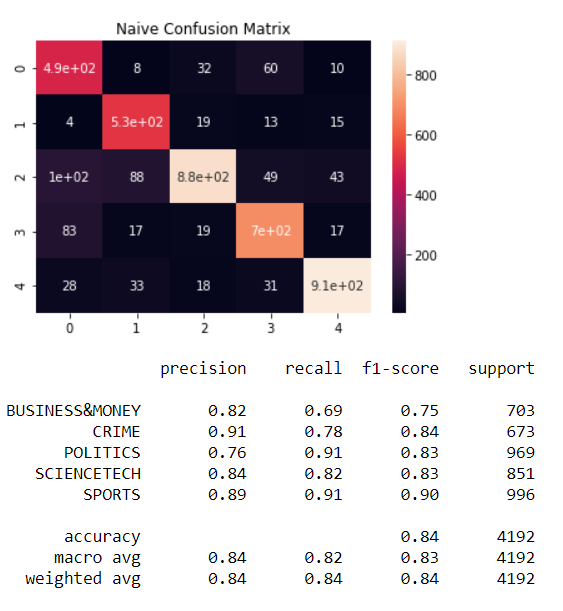


Figure 20 - Naive matrix and score per category

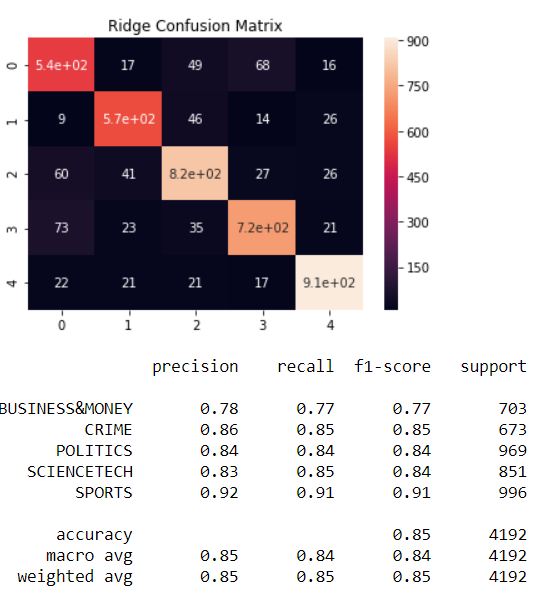


Figure 21 - Ridge matrix and score per category

Before saving the classifier into a file, we do the training again but with full dataset instead.

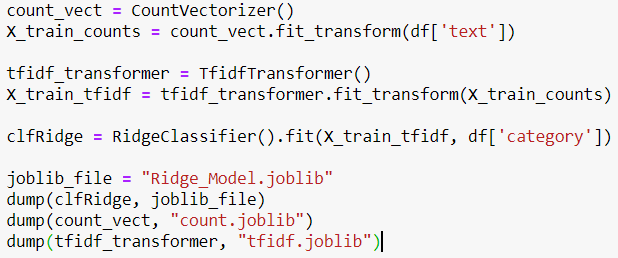


Figure 22 - Full file training

# References

Karani, D. (2018). *Introduction to Word Embedding and Word2Vec*. Retrieved from https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa

Misra, R. (2018). *News Category Dataset*. Retrieved from Kaggle: https://www.kaggle.com/rmisra/news-category-dataset

Prabhakaran, S. (2019). *Lemmatization Approaches with Examples in Python*. Retrieved from machinelearningplus: https://www.machinelearningplus.com/nlp/lemmatization-examples-python/

scikit-learn. (2019). *Working With Text Data*. Retrieved from scikit-learn: https://scikit-learn.org/stable/tutorial/text\_analytics/working\_with\_text\_data.html

Appendix

Contents of politics.txt;

General election 2019: Leaders clash over Brexit in TV debate

Leading figures from the UK's political parties have clashed on Brexit, the NHS and terror legislation in the latest televised general election debate.

The UK goes to the polls on 12 December.

Tory Rishi Sunak was pushed to rule out a no-deal Brexit if the Conservatives won, but did not give a direct answer.

Boris Johnson widens lead in polls over Jeremy Corbyn to 10 points, amid fall in support for Brexit Party

Boris Johnson has widened his lead over Jeremy Corbyn to 10 points, amid a further fall in support for the Brexit Party, according to a poll.

Labour and the Brexit Party have each dropped by one point to 33 per cent and 4 per cent respectively, while the Liberal Democrats remained on 13 per cent.

General election poll tracker: Labour vote firms up with two weeks to go

Brexit

Lib Dems suspend campaigner after apparent email forgery

Brexit: Johnson ‘will have to call second referendum if he fails to win majority’

With Opinium poll showing his lead narrowing, academics warn PM could need support of minority parties for Brexit deal

Boris Johnson could be forced into holding a second referendum on Brexit next summer if he fails to win a majority in the House of Commons but remains as prime minister, according to a new report by academics at University College London.

The Guardian view on Boris Johnson’s fact-free claims: dodging responsibility on terror attack

General election 2019

ITV election debate: the winners and the losers

Nicola Sturgeon, Richard Burgon and Adam Price score points in seven-way debate while Tories and Nigel Farage struggle

Election debate: the night’s winners and losers

Rebecca Long-Bailey and Caroline Lucas shine as Tories and Brexit party struggle

UK parliament's prorogation: all your questions answered

Government shuts down parliament for total of weeks as Brexit deadline nears

Prorogation is a formal mechanism to end a session of parliament. It means parliament’s sitting is suspended and it ends all current legislation under discussion.

14m UK voters live in areas held by same party since second world war

Electoral reform campaign group says lack of change shows voting system is broken

Brexit news

Brexit debate



Figure 23 - Crawler#Index part 1

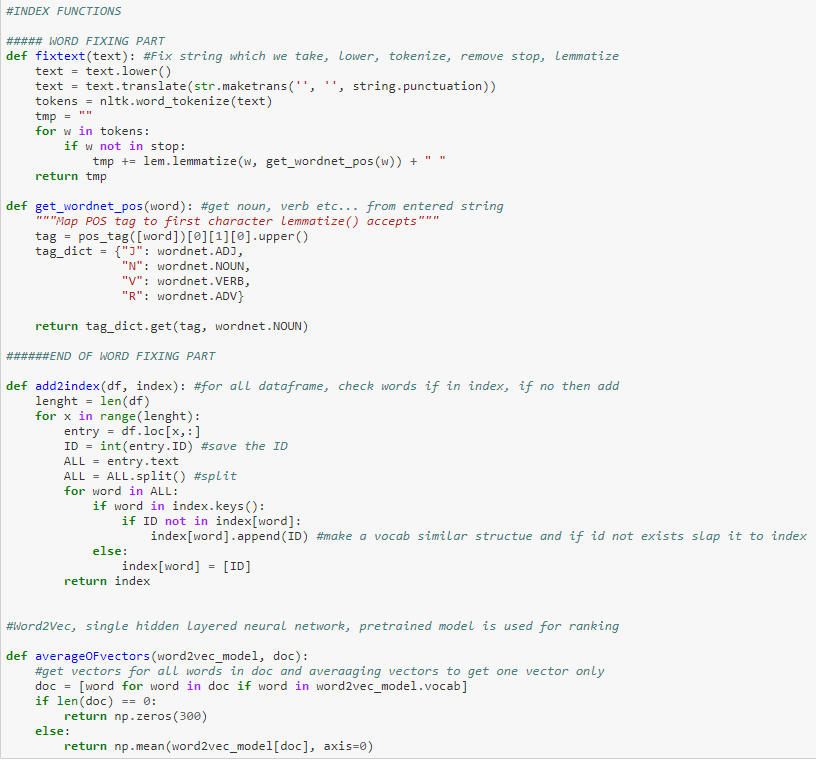


Figure 24 - Crawler#Index part 2

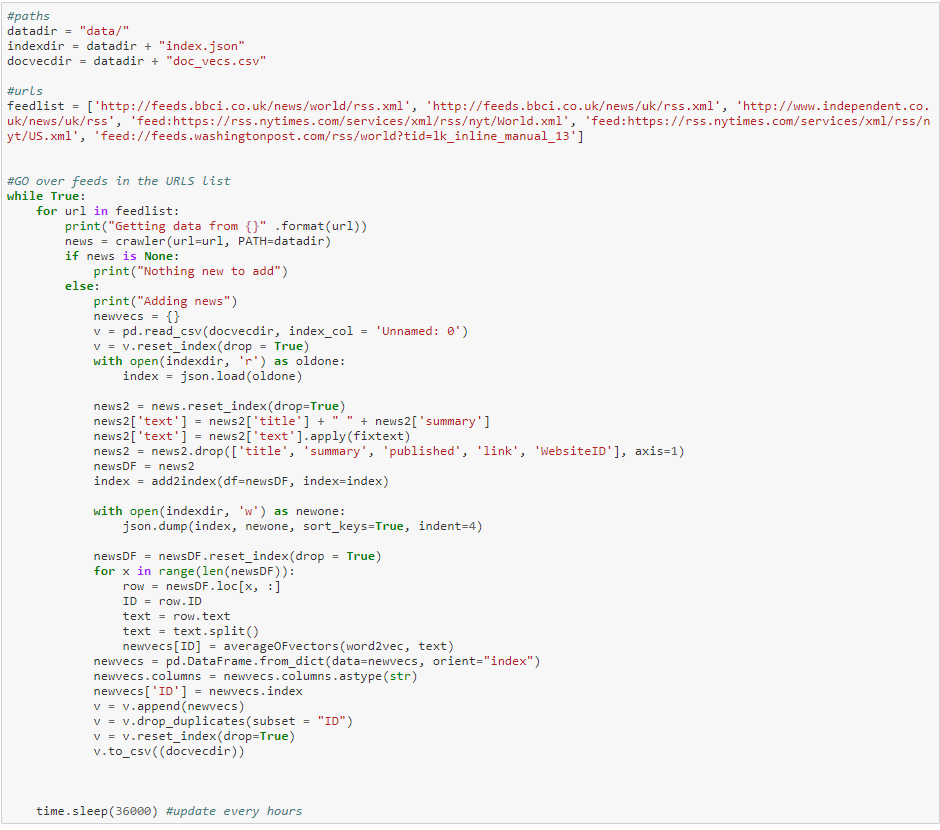


Figure 25 - - Crawler#Index part 3



Figure 26 - Query part 1

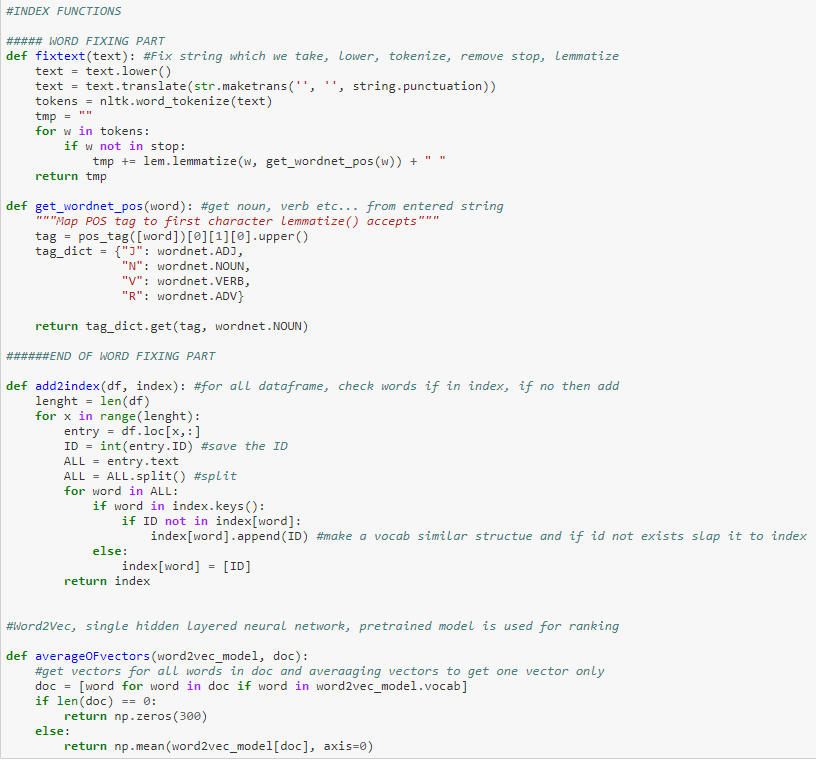


Figure 27 - Query part 2



Figure 28 - Query part 3



Figure 29 - Query part 4



Figure 30 - Classifier part 1



Figure 31 - Classifier part 2

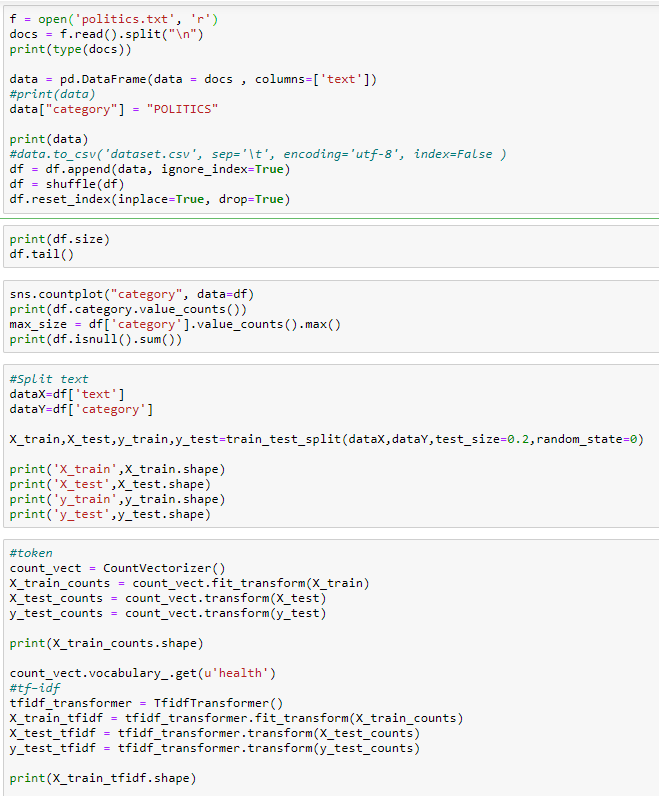


Figure 32 - Classifier part 3

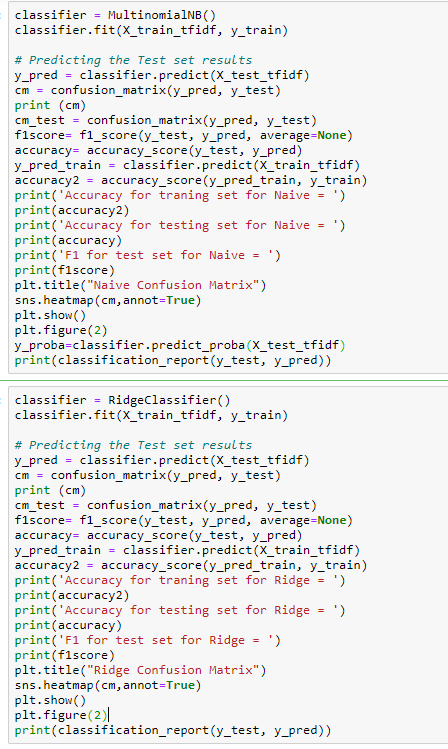


Figure 33 - Classifier part 4



Figure 34 - Classifier part 5