Big Data Learning and Technologies (COMP4124 UNUK)

A Classification Model for Sentiment Analysis of Tweets

Guneek Deol, Zhuangzhou Feng, Amit Kumar, Evangelos Vagianos, Tingjian Yu

May 13, 2024

Abstract

Twitter is one of the largest social media in the world, and politicians use it to earn popularity, where voters, in return, show support for their favoured politicians. This document investigates the US 2020 Election [3] to determine the potential correlation between what voters post only and the winning candidate. Initial pre-processing mainly focused on Natural Language Processing for a digest of the tweets, together with formatting and shaping for later pipeline fitting. Visualisation, though mostly as the result of pipelines, is also included to show attributes of tweets. The team then used a local approach based on manually labelled tweet sentiment and decision trees for classification. Generally, the team tried multiple approaches and various pipelines using Spark NLP annotators on the tweet analysis. We came up with an evident conclusion as to why the president won the election despite all the arguments online.

1 Introduction

Twitter is one of, if not the most important, social media apps today. Celebrities, opinion leaders, movie stars, and great athletes seek support through various means on Twitter. Not surprisingly, politicians also use Twitter as a platform to share their ideas and gain popularity. According to the U.S. House's official website, all House of Representatives members have an official Twitter account, [2], and the same applies to the U.S. senators [4].

But one may wonder, when we already know politicians influence Twitter, does Twitter influence politicians in return with a vengeance? With this curiosity, the team investigated the US 2020 Election Twitter data to figure out the potential correlation between favouring tweets and the final president. The team used various Spark NLP annotators to build pipelines, and our research found despite many voters leaning against one side during the election, the candidate with wider popularity unsurprisingly won the election.

2 NLP Pre-processing

The team used the Spark NLP as the Natural Language Processing (NLP) tool for Natural Language Processing. According to the official Spark NLP website [9], the library provides various annotators to build pipelines.

- Detect languages

  Regarding language detection, a blog from the developers of Spark NLP [10] instructed Spark NLP users to build a fixed pipeline:

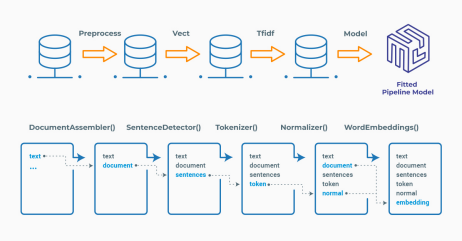

Figure 1: The example language detection pipeline

In our project, this process is combined with feature selection. Instead of the classical tokeniser and normaliser annotator-based detection, we used the newly supported `LanguageDetectorDL` [8] to detect languages. Then, the team used a linear pipeline for the feature selection to produce the bow, known as a bag of words, for pre-processing:

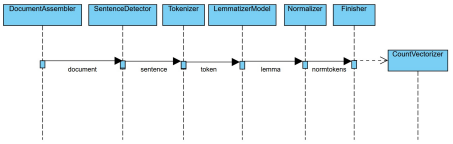

Figure 2: The bow processing linear pipeline

- Hot keywords

  The team originally filtered hashtags and @s but later decided to remove them. We found #Trump and #Biden the two dominant ones, but actually we expect them to be the most popular ones anyway. Instead, the hashtag #covid is an important one, along with sharp views against Donald Trump. Also, #MAGA, used as a slogan, is quite important during the 2020 election; many of the voters are either a passionate supporter or a complete hater. So we found instead of manually filtering everything, we should focus on the tricky and representative ones.

  The team further developed this model by dividing the detection into two sections:

  – The team first filtered retweets, as we are only concerned about tweets that were retweeted more than once. During the removal, the team used regex to define a udf:

  – To calculate the hot keywords, the team removed the filter for hashtags & mentions as this is an important context. We also saved them in the parquet files, especially given that running them again in future research will be too time-consuming.

Figure 3: With regex

The final results are provided as word clouds:

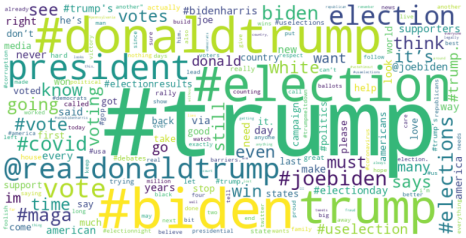

Figure 4: Trump word cloud

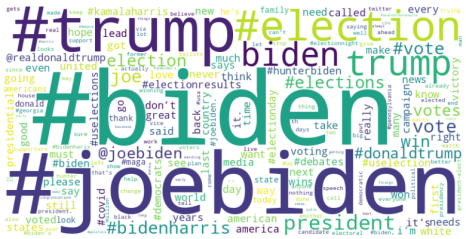

Figure 5: Biden word cloud

3 Model and pipeline fitting

3.1 Annotators

The team used a series of annotators provided by Spark NLP [5], and some of them were chosen after careful consideration:

- SentenceDetector is a RegEx-based processor for sentence boundaries. Notably, we found that the classical SentenceDetector suffices for our project, and there is no need for the upgraded CNN version SentenceDetectorDL.

- LanguageDetectorDL is a pre-trained CNN model for automatic language detection. Currently, we are using the Wiki-Tatoeba version [7], which can detect 375 languages. As one of the largest

social media used by billions worldwide, we expect many Twitter users to come from different countries and potentially speak all languages worldwide. As a result, it is essential to use a detector with great language coverage.

- Lemmatizer generates the lammas of words, i.e., run from running and ran. We used the model lemma_antbnc, the English lemmatizer [6], so we can have consistent hot keywords instead of treating vote and voting differently.

- Universal Sentence Encoder [1] is used for embedding vectors. Specifically, in our project, the USE is used in the sentiment analysis for sentence embeddings, which the SentimentDLModel later processes.

- CountVectorizer is, despite not being part of the Spark NLP library, another powerful tool used during the preprocessing. We used it to generate a bow as word vectors from our language detection result.

3.2 Pipelines

We employed a series of Language Processing (NLP) pipelines to analyze Twitter data related to the US 2020 Election. They were integral in transforming raw data into insightful analytics that could predict sentiment correlations with tweets regarding the election outcome.

- NLP Pre-processing Pipeline: The preprocessing pipeline was designed to cleanse and prepare the data for the required analysis. This was achieved by utilizing Spark NLP to handle this due to its robust capabilities in processing large datasets, which was ours efficiently. We ensured the data quality for subsequent analysis stages. This was achieved by integrating several annotators, including a tokenizer, lemmatizer, and language detector. The LanguageDetectorDL was crucial for identifying the languages of the tweets, ensuring that subsequent NLP tasks were performed on a dataset that contained the same type of data.

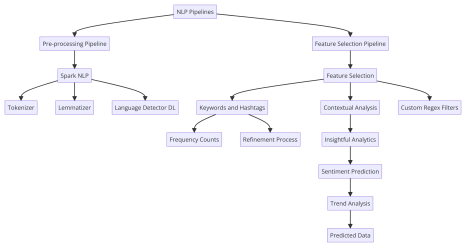

Figure 6: Pipelines Flowchart

- Feature Selection Pipeline: This pipeline focused on extracting meaningful features from the tweets, such as keywords and hashtags. We aimed to identify features that could indicate sentiment trends related to the election candidates based on tweets. Initially based on simple frequency counts of hashtags and mentions, this was later refined to focus on contextually significant terms using more sophisticated NLP techniques. This helped in focusing the analysis on more impactful data points. We filtered out noise in the data by implementing custom regex-based filters and focusing on semantically rich keywords.

The pipelines successfully processed and analyzed thousands of tweets to predict sentiment trends. This process demonstrated the effectiveness of carefully designed NLP pipelines in extracting and analyzing sentiment from Twitter data.

4 Additional Visuallisation

Besides the above results, we have reviewed the datasets and observed certain interesting attributes. Below are some top-tens:

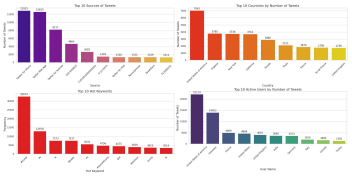

Figure 7: Top 10 for tweets

We used logarithms for like and retweet counting to compensate for the long tail. Tweets with thousands of likes are, despite being relatively rare in terms of frequency, actually the most dominant and viewed by many users. By adopting such a method, we could reduce the skewness of our data for a comprehensive overview.

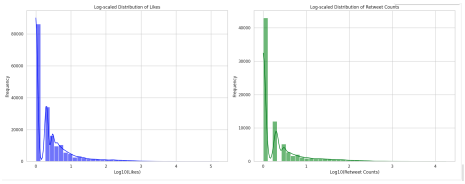

Figure 8: Logarithm transformed counting

5 ML Approaches

During the lectures, we learnt that the machine learning approaches applied to big data are subtly different from what we expect from normal machine learning techniques.

Namely, we trained our own sentiment analysis pipeline using the Global Approach. This is a standard approach where we have one pipeline and one unified model for all data. The implementation itself is not challenging, and the global accuracy of all data is 60%.

We have also tried the Local Approach. First, we divided the data into five blocks and assigned each to one of our teammates for manual labelling. Then, we used a DecisionTreeClassifier from PySpark for both a simple try and results for compassion. The result was not so convincing, with the accuracy as low as 41%.

After that, we divided the DataFrame on RDDs for parallelised storage and broadcasted it during the Spark session. Using a self-defined self-learn function, we mapped the decision tree classifier to each partition in the RDD for a final result based on the Local Approach.

After comparison, we decided to choose the Global Approach. The Local Approach is probably more advanced and technical, but hypertuning this model is more tricky and time-consuming than we thought. Besides, the global model actually run faster then the local one. Considering we already have the Global Approach running with a relatively convincing accuracy, we would say the Global Approach is the one to use.

6 Evaluation

6.1 Code evaluation

We evaluated the size-up for the training phase by taking the Biden hashtag dataset as sample data. We defined a testing function by taking a list of sampling sizes and then calculating the time interval between the beginning and the ending time of a pipeline fitting action.

```
[ ] scalability_test_pipeline(df,[0.1,0.25,0.5,1.0],lda_pipeline)

→ df sample:0.1 Time: 0.4481355529970176
  df sample:0.25 Time: 0.3841955610005243
  df sample:0.5 Time: 0.3809136480079144
  df sample:1.0 Time: 0.3203056399997877

[ ] scalability_test_pipeline(df,[0.1,0.25,0.5,1.0],nlpPipeline)

→ df sample:0.1 Time: 0.2523884760003057
  df sample:0.25 Time: 0.1710641010004103
  df sample:0.5 Time: 0.2497837390001223
  df sample:1.0 Time: 0.1905517039951749
```

Figure 9: Pipeline fitting intervals

6.2 Project evaluation

The team is proud to say the project has been well managed for weeks, and tasks have been evenly assigned to each of us.

Since before the Easter break, the team has started organising meetings to discuss potential project tasks and has kept the wheel running. We held weekly meetings to discuss and share what we learnt about Natural Language Processing. We sent useful blogs, documents, and links to our group chat and gave feedback on each other's work. We can say each of us contributed during this process, and more importantly, each of us did learn useful knowledge with the help of teamwork.

The team also used professional tools to keep track of tasks. We used the Task app provided by Teams to mark project milestones with explicit dates to track. We also used Microsoft Azure for shared codes and data, which served mostly as our repository during the project. Simply repeatedly having meetings is time-consuming and exhausting, but

combining both, we achieved fruitful results with on-track progress.

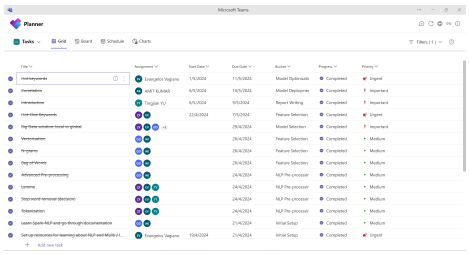

Figure 10: The Tasks app offered by Teams

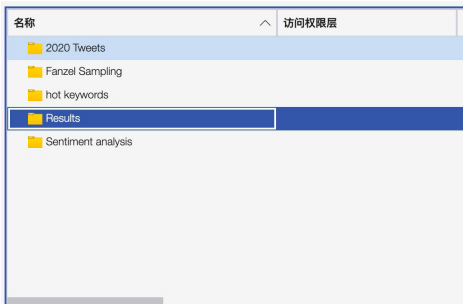

Figure 11: The Azure "repository"

7 Conclusion

After combining the sentiment analysis and the precinct voting data, we found a positive correlation between the votes Trump received and his popularity, and the same applies to negative sentiments against Biden:

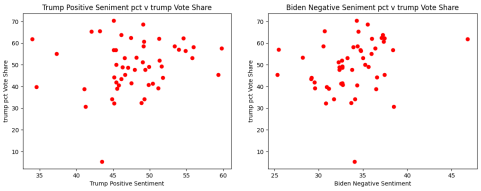

Figure 12: Trump positive sentiment

Figure 13: Biden negative sentiment

However, despite the fact that Biden didn't receive as much PCT vote share as Trump does, he genuinely gained more popularity, indicated by the higher positive sentiment share and lower negative sentiment share:

Generally, it is safe to say Biden is more popular among the public, as represented in

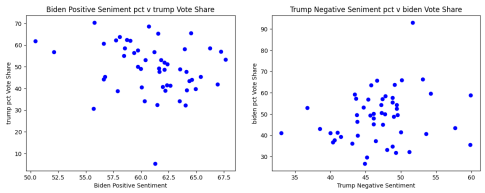

Figure 14: Biden positive sentiment

Figure 15: Trump negative sentiment

the joint diagram of Biden and Trump's positive sentiment by the blue dots in Fig 16:

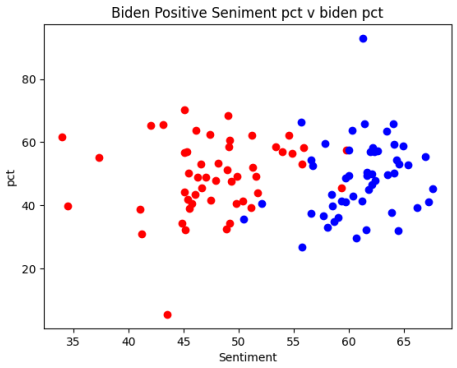

Figure 16: Positive sentiment on both candidates

We have also researched mean values: the positive sentiments versus vote share that Biden got is significantly larger than that of Trump, representing that overall, even given the same positive sentiment, Biden would have received more votes, not to mention he received more popularity against Trump.

```
[ ] mean_values = merged_df.drop(columns = ['State']).mean()
    mean_values

Trump+ vs Vote%          9.133938
Biden+ v Vote%          14.503593
Trump-  v Biden Vote%    9.282963
Biden- v Trump Vote%    17.164242
dtype: object
```

Figure 17: Mean values of sentiment vs vote

References

[1] Daniel Cer et al. *Universal Sentence Encoder*. 2018. arXiv: 1803.11175 [cs.CL].

[2] House Press Gallery. *House Twitter Handles Members' Official Twitter Handles House Press Gallery*. https://pressgallery.house.gov/member-data/members-official-twitter-handles. Accessed: 2024-05-04.

[3] Manch Hui. *Kaggle US Election 2020 Tweets*. https://www.kaggle.com/datasets/manchunhui/us-election-2020-tweets. Accessed: 2024-05-13.

[4] IAVM. *Senator Twitter Handles Senate Twitter Accounts*. https://www.iavm.org/sites/default/files/documents/senate_twitter_accounts_0.pdf. Accessed: 2024-05-07.

[5] John Snow Labs. *Annotators*. https://sparknlp.org/docs/en/annotators. Accessed: 2024-05-10.

[6] John Snow Labs. *English Lemmatizer lemma_antbnc*. https://sparknlp.org/2021/11/22/lemma_antbnc_en.html. Accessed: 2024-05-10.

[7] John Snow Labs. *Language Detection and Identification Pipeline 375 Languages*. https://sparknlp.org/2020/12/05/detect_language_375_xx.html. Accessed: 2024-05-10.

[8] John Snow Labs. *Spark NLP Annotators LanguageDetectorDL*. https://sparknlp.org/docs/en/annotators#languagedetectordl. Accessed: 2024-05-08.

[9] John Snow Labs. *Spark NLP State of the Art NLP Library for Large Language Models (LLMs)*. https://sparknlp.org/. Accessed: 2024-05-08.

[10] Gursev Pirge. *How to Detect Languages with Python: A Comprehensive Guide*. https://www.johnsnowlabs.com/how-to-detect-languages-with-python-a-comprehensive-guide/. Accessed: 2024-05-08.