Twitter Sentiment Analysis on 2024 US Presidential Election Candidates

Zhan Shu, Yuan Dou, Yingqi Ma

Introduction

- Our purpose is to build an application that reflects the real-time sentiments of 2024 United States Presidential Election from twitter users.

- Current forums are not precise on comparing candidates real-time rating.

- Our goal is to implement a web application which display the sentiment results of the top candidates numerically and graphically.

- The real-time trend of sentiments will also be shown on the web application.

Candidate

We choose top 6 popular candidates:

Asa Hutchinson

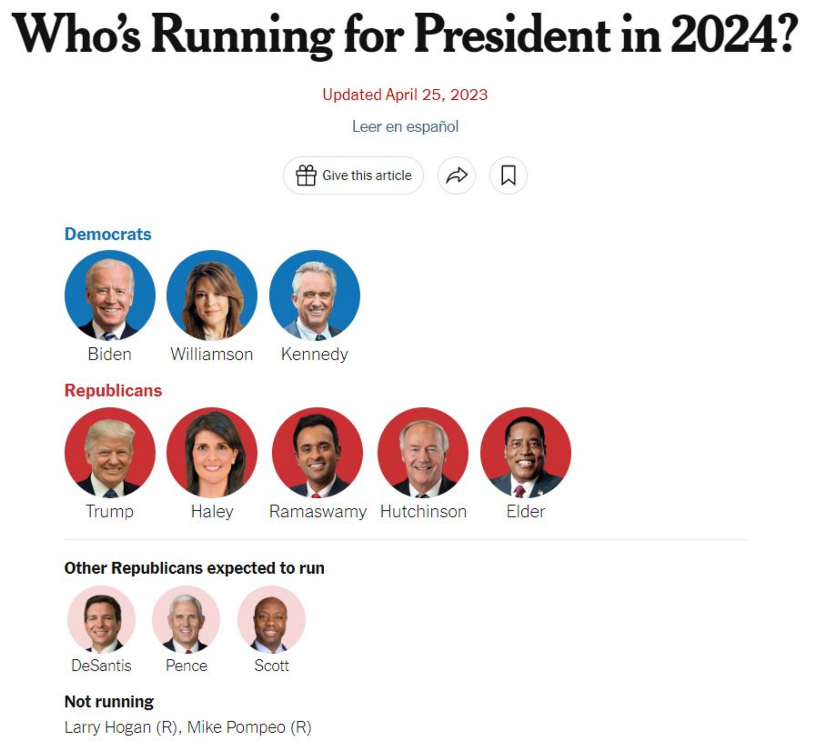
Joe Biden

Marianne Williamson

Nikki Haley

Vivek Ramaswamy

Donald Trump



Who's Running for President in 2024?

Updated April 25, 2023

Leer en español

Democrats

Biden Williamson Kennedy

Republicans

Trump Haley Ramaswamy Hutchinson Elder

Other Republicans expected to run

DeSantis Pence Scott

Not running

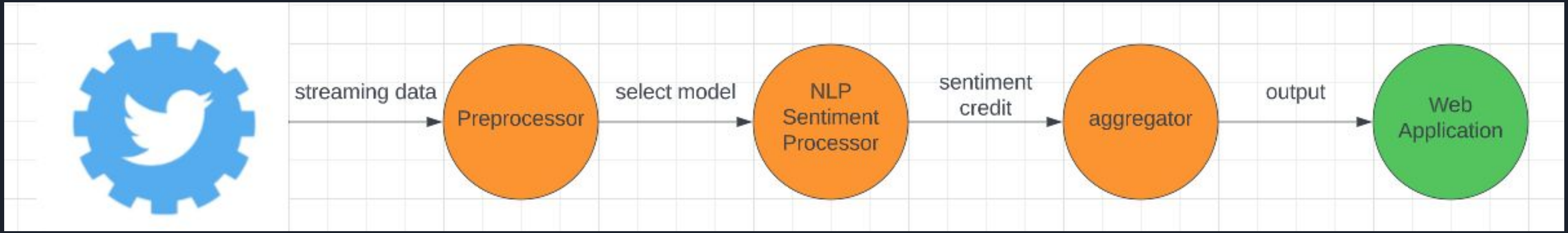
Larry Hogan (R), Mike Pompeo (R)

Data source

- Twitter API
- Streaming data, fetched real-time
- Using Tweepy to connect to twitter and fetch data

Techniques

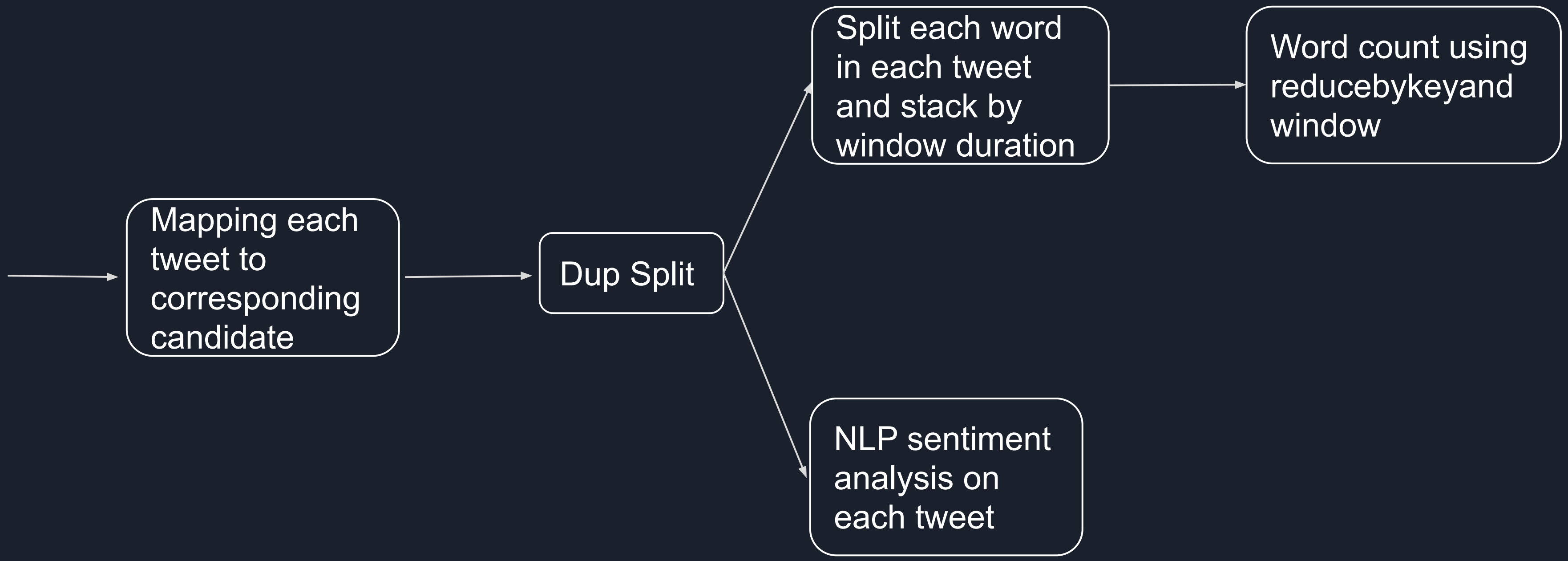
- Fetch streaming data using Twitter API
- Filter the data and keep those tweets that are relevant to the candidates
- Apply NLP sentiment analysis to each tweet to calculate the sentiment score of the respective tweet
- Aggregate the sentiment score and calculate the average sentiment for each candidate over each 6 minute tumbling window
- Count the word which appears most frequently for each candidate over each 6 minute tumbling window



Spark streaming

- Using tweepy and create a class that inherits from tweepy.StreamingClient to fetch data from twitter API in desire format.
- The class will wait for TCP connection and client socket once client class is running.
- Senddata function filters data and sends to the client socket with port and IP defined.
- The socketTextStream() function of StreamingContext() in PySpark streaming is used to create a DStream that represents streaming data from a TCP source, specified with IP and port.
- Then doing preprocessing to the DStream data.

Optimization (Redundancy Elimination)



Mapping each tweet to corresponding candidate

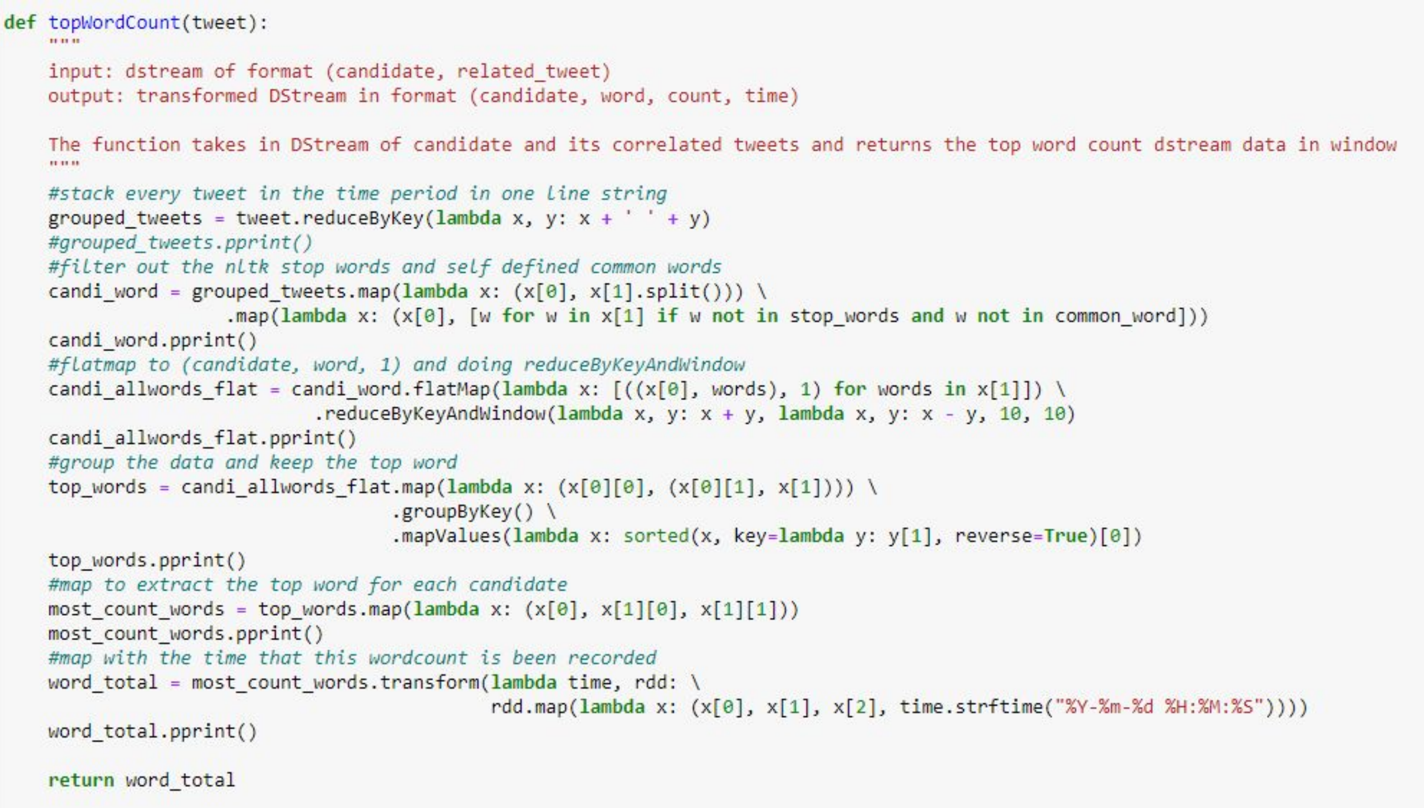
Dup Split

Split each word in each tweet and stack by window duration

Word count using reducebykeyand window

NLP sentiment analysis on each tweet

Word Count

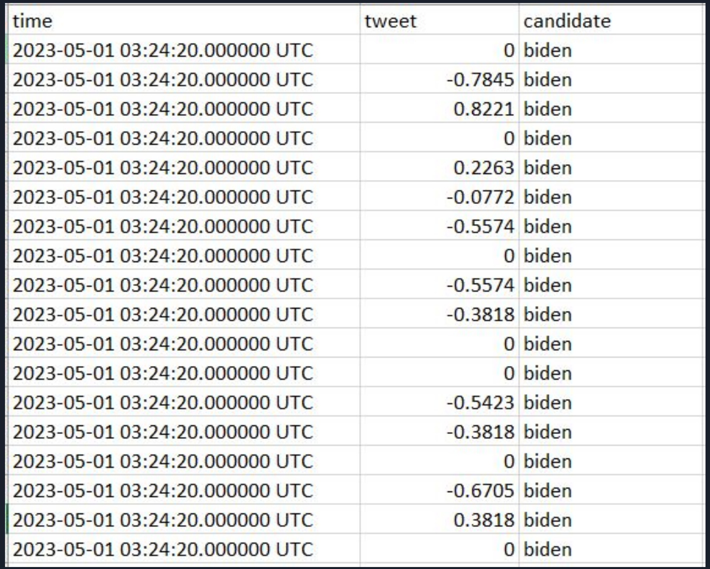


```python
def topWordCount(tweet):
    """
    input: dstream of format (candidate, related_tweet)
    output: transformed DStream in format (candidate, word, count, time)

    The function takes in DStream of candidate and its correlated tweets and returns the top word count dstream data in window
    """
    #stack every tweet in the time period in one line string
    grouped_tweets = tweet.reduceByKey(lambda x, y: x + ' ' + y)
    #grouped_tweets.pprint()
    #filter out the nltk stop words and self defined common words
    candi_word = grouped_tweets.map(lambda x: (x[0], x[1].split())) \
                    .map(lambda x: (x[0], [w for w in x[1] if w not in stop_words and w not in common_word]))
    candi_word.pprint()
    #flatmap to (candidate, word, 1) and doing reduceByKeyAndWindow
    candi_allwords_flat = candi_word.flatMap(lambda x: [((x[0], words), 1) for words in x[1]]) \
                            .reduceByKeyAndWindow(lambda x, y: x + y, lambda x, y: x - y, 10, 10)
    candi_allwords_flat.pprint()
    #group the data and keep the top word
    top_words = candi_allwords_flat.map(lambda x: (x[0][0], (x[0][1], x[1]))) \
                                   .groupByKey() \
                                   .mapValues(lambda x: sorted(x, key=lambda y: y[1], reverse=True)[0])
    top_words.pprint()
    #map to extract the top word for each candidate
    most_count_words = top_words.map(lambda x: (x[0], x[1][0], x[1][1]))
    most_count_words.pprint()
    #map with the time that this wordcount is been recorded
    word_total = most_count_words.transform(lambda time, rdd: \
                                            rdd.map(lambda x: (x[0], x[1], x[2], time.strftime("%Y-%m-%d %H:%M:%S"))))
    word_total.pprint()

    return word_total
```

VADER sentiment analysis

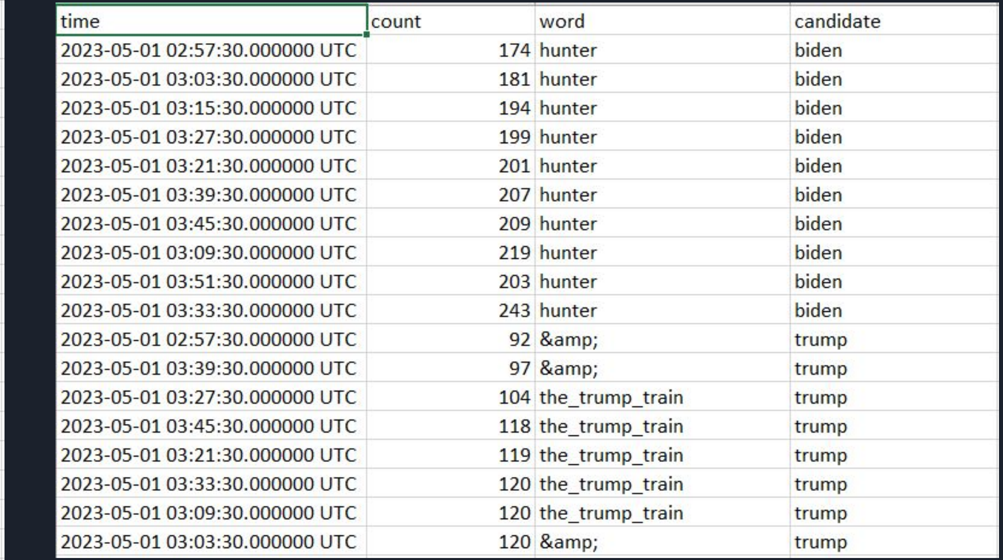
- VADER (Valence Aware Dictionary for Sentiment Reasoning) is a model used for text sentiment analysis that is sensitive to both polarity (positive/negative) and intensity (strength) of emotion.
- VADER’s SentimentIntensityAnalyzer() takes in a string and returns a dictionary of scores in each of four categories: negative, neutral, positive, compound (computed by normalizing the scores above).
- Compound >= 0.05: Positive
- Compound <= -0.05: Negative
- -0.05 < Compound < 0.05: Neutral

Process streaming results



time	tweet	candidate
2023-05-01 03:24:20.000000 UTC	0	biden
2023-05-01 03:24:20.000000 UTC	-0.7845	biden
2023-05-01 03:24:20.000000 UTC	0.8221	biden
2023-05-01 03:24:20.000000 UTC	0	biden
2023-05-01 03:24:20.000000 UTC	0.2263	biden
2023-05-01 03:24:20.000000 UTC	-0.0772	biden
2023-05-01 03:24:20.000000 UTC	-0.5574	biden
2023-05-01 03:24:20.000000 UTC	0	biden
2023-05-01 03:24:20.000000 UTC	-0.5574	biden
2023-05-01 03:24:20.000000 UTC	-0.3818	biden
2023-05-01 03:24:20.000000 UTC	0	biden
2023-05-01 03:24:20.000000 UTC	0	biden
2023-05-01 03:24:20.000000 UTC	-0.5423	biden
2023-05-01 03:24:20.000000 UTC	-0.3818	biden
2023-05-01 03:24:20.000000 UTC	0	biden
2023-05-01 03:24:20.000000 UTC	-0.6705	biden
2023-05-01 03:24:20.000000 UTC	0.3818	biden
2023-05-01 03:24:20.000000 UTC	0	biden

Sentiment Analysis result



time	count	word	candidate
2023-05-01 02:57:30.000000 UTC	174	hunter	biden
2023-05-01 03:03:30.000000 UTC	181	hunter	biden
2023-05-01 03:15:30.000000 UTC	194	hunter	biden
2023-05-01 03:27:30.000000 UTC	199	hunter	biden
2023-05-01 03:21:30.000000 UTC	201	hunter	biden
2023-05-01 03:39:30.000000 UTC	207	hunter	biden
2023-05-01 03:45:30.000000 UTC	209	hunter	biden
2023-05-01 03:09:30.000000 UTC	219	hunter	biden
2023-05-01 03:51:30.000000 UTC	203	hunter	biden
2023-05-01 03:33:30.000000 UTC	243	hunter	biden
2023-05-01 02:57:30.000000 UTC	92	&amp;	trump
2023-05-01 03:39:30.000000 UTC	97	&amp;	trump
2023-05-01 03:27:30.000000 UTC	104	the_trump_train	trump
2023-05-01 03:45:30.000000 UTC	118	the_trump_train	trump
2023-05-01 03:21:30.000000 UTC	119	the_trump_train	trump
2023-05-01 03:33:30.000000 UTC	120	the_trump_train	trump
2023-05-01 03:09:30.000000 UTC	120	the_trump_train	trump
2023-05-01 03:03:30.000000 UTC	120	&amp;	trump

Word count result

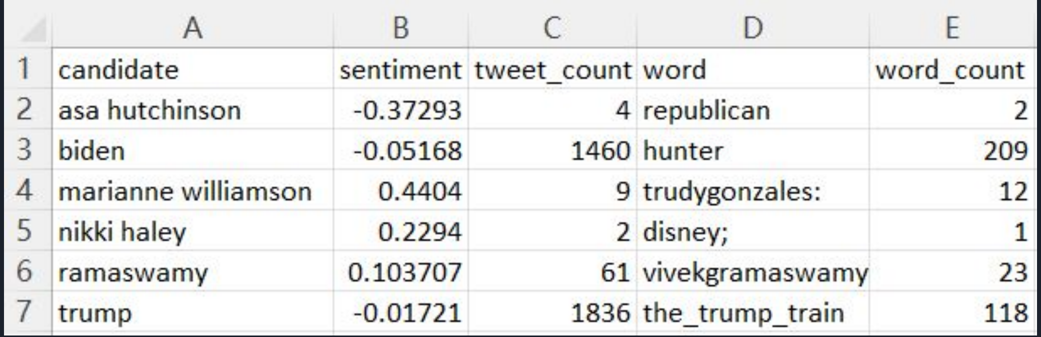
Process streaming results

- Use spark sql window function to create 6 minute windows depending on the time column.
- Aggregate sentiment score to calculate average sentiment score over each window for each candidate.
- If a candidate has no tweet over a window, assign a value of 0 to average sentiment score and tweet count.
- Add word count result to the dataframe
- Add word and word count to a row if the time is within the range of a window.
- Break the result dataframe into smaller dataframes, with each smaller dataframe showing the sentiment analysis and word count result for a 6 minute window.

Result

Run 1 hour (6 minutes per time window)

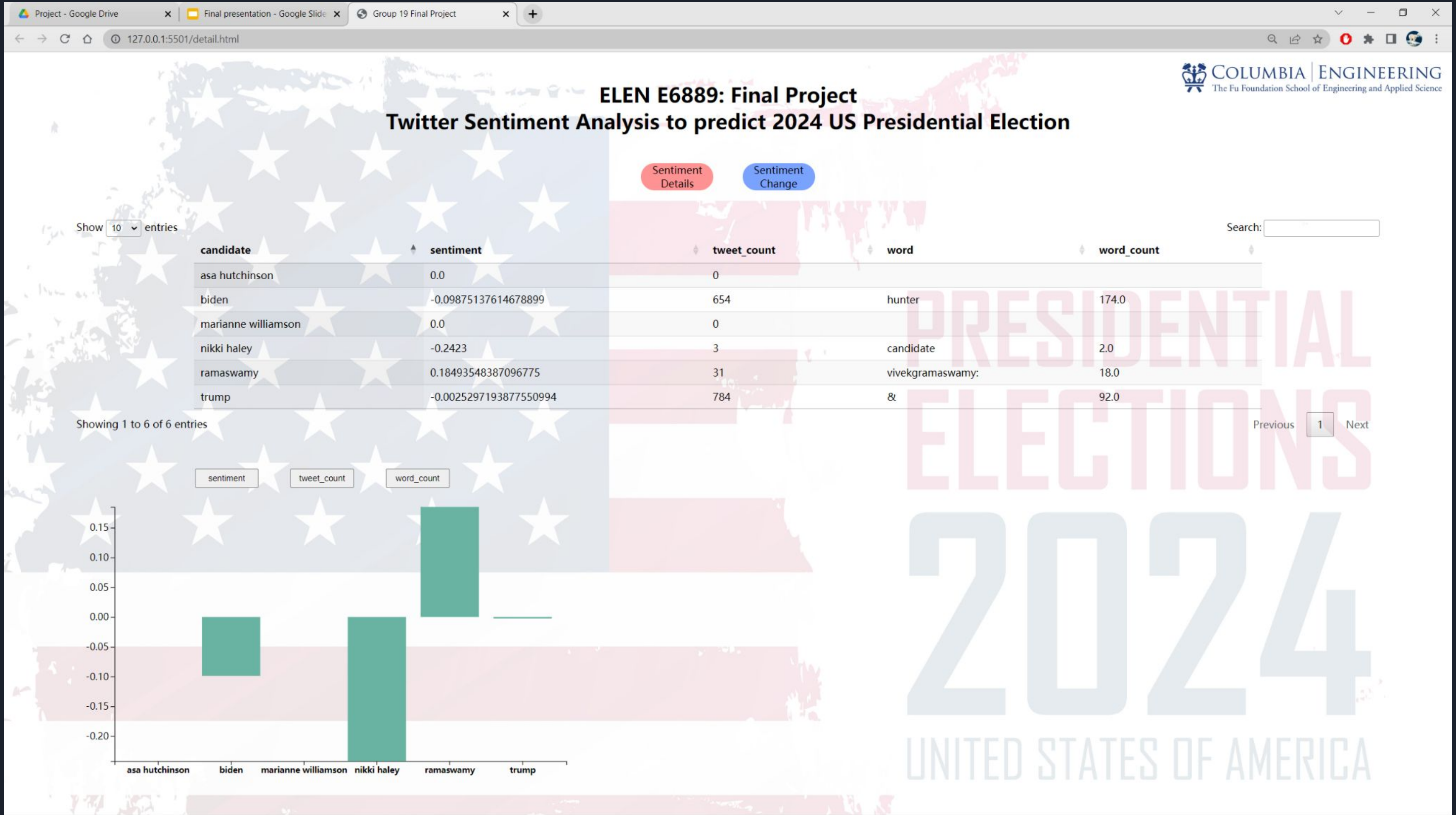
10 csv files of sentiment details for 10 time windows



	A	B	C	D	E
1	candidate	sentiment	tweet_count	word	word_count
2	asa hutchinson	-0.37293	4	republican	2
3	biden	-0.05168	1460	hunter	209
4	marianne williamson	0.4404	9	trudygonzales:	12
5	nikki haley	0.2294	2	disney;	1
6	ramaswamy	0.103707	61	vivekgramaswamy	23
7	trump	-0.01721	1836	the_trump_train	118

Sentiment details of latest time window

Demo

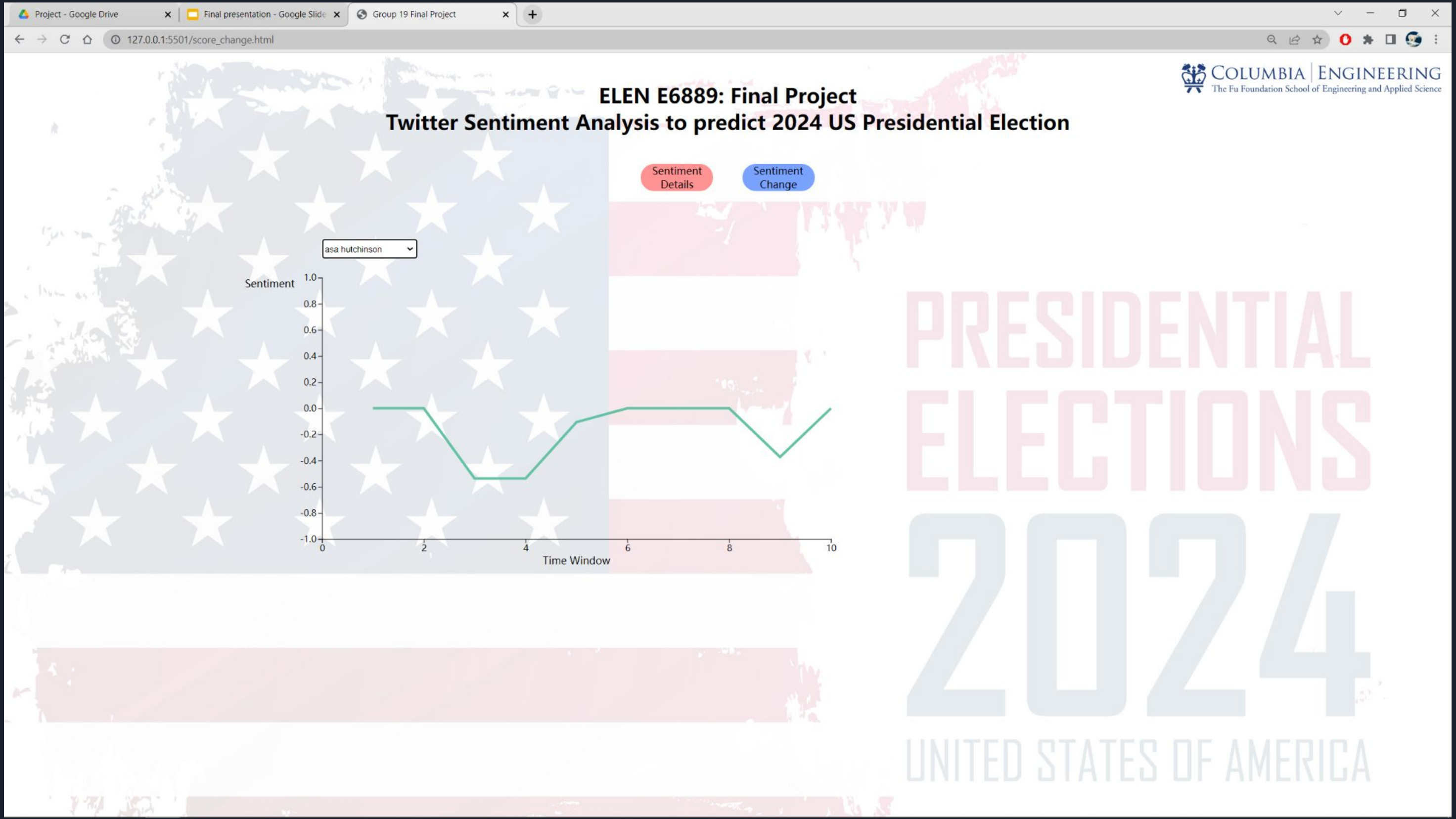

Project - Google Drive
Final presentation - Google Slide
Group 19 Final Project
127.0.0.1:5501/detail.html
COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

ELEN E6889: Final Project
Twitter Sentiment Analysis to predict 2024 US Presidential Election

candidate	sentiment	tweet_count	word	word_count
asa hutchinson	0.0	0		
biden	-0.09875137614678899	654	hunter	174.0
marianne williamson	0.0	0		
nikki haley	-0.2423	3	candidate	2.0
ramaswamy	0.18493548387096775	31	vivekgramaswamy:	18.0
trump	-0.0025297193877550994	784	&	92.0

Showing 1 to 6 of 6 entries

Demo



Project - Google Drive
Final presentation - Google Slide
Group 19 Final Project
127.0.0.1:5501/score_change.html
COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

ELEN E6889: Final Project
Twitter Sentiment Analysis to predict 2024 US Presidential Election

Future Works

1. Save wordcount and tweet to bucket and use directly by the window aggregation function.
2. Establish connector between dstream and web application
3. Implement twitter streaming with the window aggregation.

Thank you!