DETERMINING TOPIC SENTIMENT ACROSS SOCIAL NETWORK PLATFORMS

Craig Dick

INTRODUCTION

- Motivation around project
- 2.8 million subreddit communities Reddit in 2021, Twitter 206 million daily users
- Misinformation is dangerous due to the health implications of COVID, can make the decision of a user getting the vaccine or not
- What does the project aim to do?
- Whether sentiment analysis can identify misinformation on social networks
- Code to collect data, then analyse sentiment to see if this highlights misinformation

BACKGROUND

- Natural Language Processing
  - Pre-processing of data, NLP tasks, sentiment analysis
- Collecting data on Twitter and Reddit
  - Streaming and REST API, Reddit and Pushshift API
- Sentiment Analysis
  - VADER, Afinn, TextBlob, Naïve Bayes, BERT
- Sentiment analysis with COVID
  - COVID brought new challenges, strong opinions on social media, label scales

REQUIREMENTS

Functional

- MH - System must be able to gather data off of Twitter and Reddit
- MH - The posts must be saved to a database
- MH - Search terms must be decided
- MH - Display the results of Twitter vs Reddit
- SH - Run the data collections for multiple weeks
- SH - Compare different sentiment analysis libraries
- SH - Analyse the sentiment in subreddits
- CH - Trained BERT model on labelled data
- WH – Website where user can search topic

Non-Functional

- MH – Graphs must be easily understood
- MH – Code should be commented
- SH – Code modularised where possible
- SH – The database should hold all of the data
- SH – Libraries should be scalable
- CH - The libraries should be optimised to get fast results
- WH - The APIs should be able to be used at all times

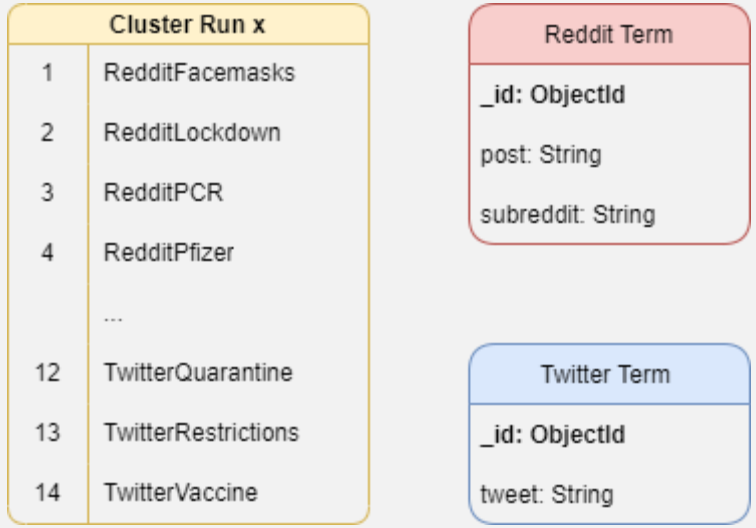
DESIGN

- Programming Language needed
- List of Search terms
- Data collection on social media
- Database storage
- Sentiment analysis libraries

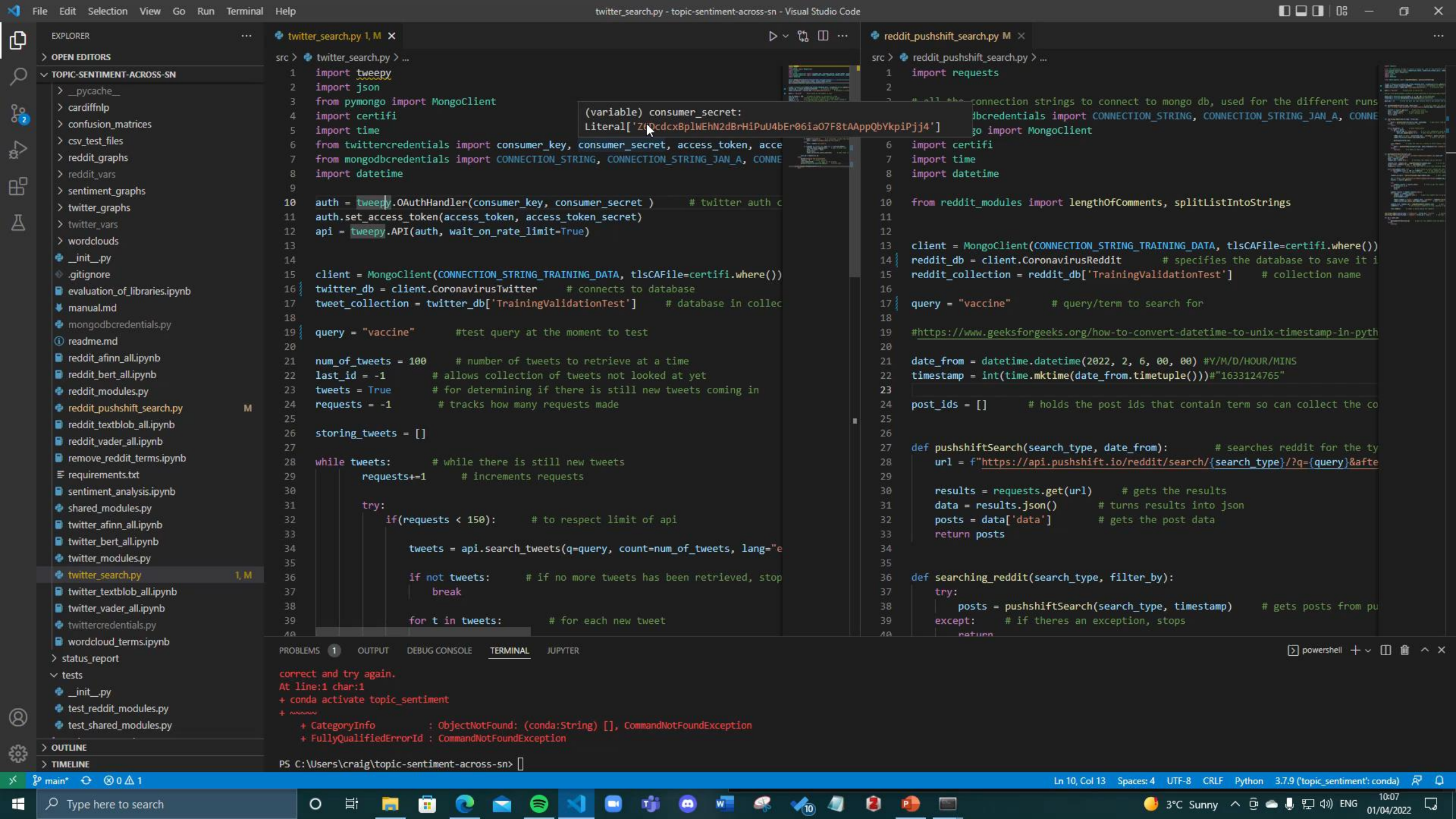
Layout of the design process to arrive at the end goal

IMPLEMENTATION

- Programming Language
- Software Engineering Practices
- Search Terms
- Database Storage
- Data collection APIs
- Sentiment Analysis – Labels, Libraries, Saving Results
- Graphs



MongoDB Schema to save the posts


File Edit Selection View Go Run Terminal Help
twitter_search.py - topic-sentiment-across-sn - Visual Studio Code
EXPLORER
OPEN EDITORS
TOPIC-SENTIMENT-ACROSS-SN
__pycache__
cardiffnlp
confusion_matrices
csv_test_files
reddit_graphs
reddit_vars
sentiment_graphs
twitter_graphs
twitter_vars
wordclouds
__init__.py
.gitignore
evaluation_of_libraries.ipynb
manual.md
mongodbcredentials.py
readme.md
reddit_afinn_all.ipynb
reddit_bert_all.ipynb
reddit_modules.py
reddit_pushshift_search.py M
reddit_textblob_all.ipynb
reddit_vader_all.ipynb
remove_reddit_terms.ipynb
requirements.txt
sentiment_analysis.ipynb
shared_modules.py
twitter_afinn_all.ipynb
twitter_bert_all.ipynb
twitter_modules.py
twitter_search.py 1, M
twitter_textblob_all.ipynb
twitter_vader_all.ipynb
twittercredentials.py
wordcloud_terms.ipynb
status_report
tests
__init__.py
test_reddit_modules.py
test_shared_modules.py
OUTLINE
TIMELINE
twitter_search.py 1, M
src > twitter_search.py > ...
import tweepy
import json
from pymongo import MongoClient
import certifi
import time
from twittercredentials import consumer_key, consumer_secret, access_token, acce
from mongodbcredentials import CONNECTION_STRING, CONNECTION_STRING_JAN_A, CONNE
import datetime
auth = tweepy.OAuthHandler(consumer_key, consumer_secret) # twitter auth c
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth, wait_on_rate_limit=True)
client = MongoClient(CONNECTION_STRING_TRAINING_DATA, tlsCAFile=certifi.where())
twitter_db = client.CoronavirusTwitter # connects to database
tweet_collection = twitter_db['TrainingValidationTest'] # database in collec
query = "vaccine" #test query at the moment to test
num_of_tweets = 100 # number of tweets to retrieve at a time
last_id = -1 # allows collection of tweets not looked at yet
tweets = True # for determining if there is still new tweets coming in
requests = -1 # tracks how many requests made
storing_tweets = []
while tweets: # while there is still new tweets
requests+=1 # increments requests
try:
if(requests < 150): # to respect limit of api
tweets = api.search_tweets(q=query, count=num_of_tweets, lang="e
if not tweets: # if no more tweets has been retrieved, stop
break
for t in tweets: # for each new tweet
(variable) consumer_secret:
Literal['ZdQcdcxBplWEhN2dBrHiPuU4bEr06iaO7F8tAAppQbYkpiPjj4']
reddit_pushshift_search.py M
src > reddit_pushshift_search.py > ...
import requests
bcredentials import CONNECTION_STRING, CONNECTION_STRING_JAN_A, CONNE
go import MongoClient
import certifi
import time
import datetime
from reddit_modules import lengthOfComments, splitListIntoStrings
client = MongoClient(CONNECTION_STRING_TRAINING_DATA, tlsCAFile=certifi.where())
reddit_db = client.CoronavirusReddit # specifies the database to save it i
reddit_collection = reddit_db['TrainingValidationTest'] # collection name
query = "vaccine" # query/term to search for
#https://www.geeksforgeeks.org/how-to-convert-datetime-to-unix-timestamp-in-pyth
date_from = datetime.datetime(2022, 2, 6, 00, 00) #Y/M/D/HOUR/MINS
timestamp = int(time.mktime(date_from.timetuple()))#"1633124765"
post_ids = [] # holds the post ids that contain term so can collect the co
def pushshiftSearch(search_type, date_from): # searches reddit for the ty
url = f"https://api.pushshift.io/reddit/search/{search_type}/?q={query}&afte
results = requests.get(url) # gets the results
data = results.json() # turns results into json
posts = data['data'] # gets the post data
return posts
def searching_reddit(search_type, filter_by):
try:
posts = pushshiftSearch(search_type, timestamp) # gets posts from pu
except: # if theres an exception, stops
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
powershell
correct and try again.
At line:1 char:1
+ conda activate topic_sentiment
+ ~~~~~
+ CategoryInfo : ObjectNotFound: (conda:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException
PS C:\Users\craig\topic-sentiment-across-sn>
main* 0 1
Ln 10, Col 13 Spaces: 4 UTF-8 CRLF Python 3.7.9 ('topic_sentiment': conda)
Type here to search
3°C Sunny
ENG
10:07
01/04/2022

SENTIMENT ANALYSIS CODE

```python
#---------- TEXTBLOB SENTIMENT ---------

def database_as_textblob(post):
    text = TextBlob(post)       # create
    polarity = text.sentiment.polarity

    if (polarity >= 0.05):      # if its
        sentiment = 'positive'

    elif (polarity <= -0.05):       # it
        sentiment = 'negative'

    else:
        sentiment = 'neutral'       # ot

    return sentiment
```

```python
#---------- AFINN SENTIMENT ----------#

def database_as_afinn(post):
    afinn_analyser = Afinn()        # crea
    polarity = afinn_analyser.score(post)

    if (polarity > 0):      # if its great
        sentiment = 'positive'

    elif (polarity < 0):        # if its l
        sentiment = 'negative'

    else:
        sentiment = 'neutral'       # othe

    return sentiment
```

```python
#---------- VADER SENTIMENT ----------#

def database_as_vader(post):
    vader_analyzer = SentimentIntensityAnalyzer()
    scores = vader_analyzer.polarity_scores(post)
    compound = scores['compound']       # gets the

    if (compound >= 0.05):      # if its >= 0.05,
        sentiment = 'positive'

    elif (compound <= -0.05):       # if its <= 0.
        sentiment= 'negative'

    else:
        sentiment = 'neutral'       # otherwise it

    return sentiment
```

```python
def database_as_bert(post):

    try:
        t = bert_preprocess(post)       # preprocesses post
        encoded_input = tokenizer(t, return_tensors='pt')       # puts post into tokenizer
        output = model(**encoded_input)     # puts input into the model

    except:     # exceptions were happening on unrecognised emojis, so it then removes this
        t = process_emoji(t)        # only removes emoji on exception, they can provide sentimental value to post
        encoded_input = tokenizer(t, max_length=512, truncation=True, return_tensors='pt')
        output = model(**encoded_input)

    finally:
        scores = output[0][0].detach().numpy()
        scores = softmax(scores)        # turns into probability
        ranking = np.argsort(scores)        # sorts scores in order
        ranking = ranking[::-1]     # reverses to get top ranking at first item in list
        sentiment = labels[ranking[0]]      # gets the sentiment of the top ranking from model

    return sentiment
```

EVALUATION - SYSTEM

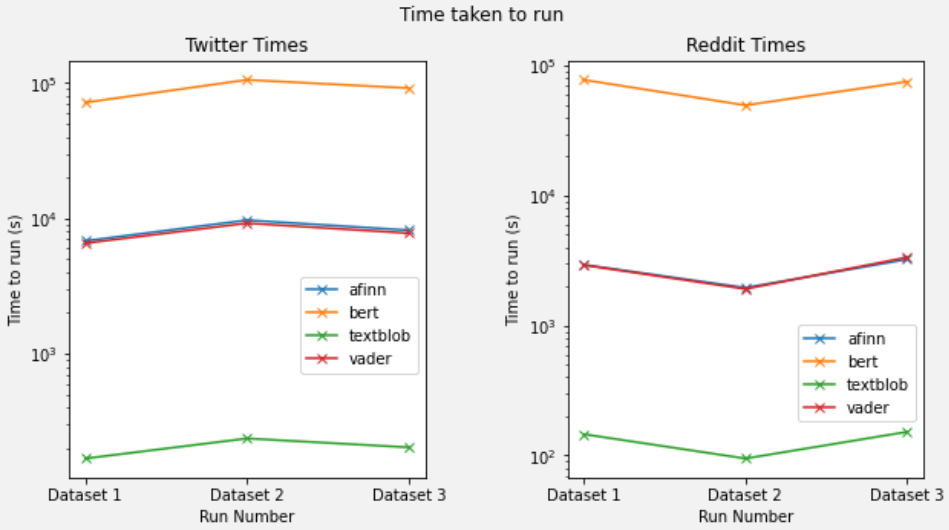
- Feature testing with Pytest

Test name	Stmts	Miss	Cover
tests/__init__.py	0	0	100%
tests/test_reddit_modules.py	41	0	100%
tests/tests_shared_modules.py	44	0	100%
TOTAL	85	0	100%

Term	Twitter Run 1	Twitter Run 2	Twitter Run 3
Facemasks	8460	73825	37780
Lockdown	137365	173641	167871
PCR	44595	135507	71512
Pfizer	83197	116621	79829
Quarantine	71657	131953	73405
Restrictions	108885	126852	151074
Vaccine	234068	238990	265256
TOTAL	688227	997389	846727

Term	Reddit Run 1	Reddit Run 2	Reddit Run 3
Facemasks	1465	1197	4054
Lockdown	31243	1421	26687
PCR	8792	3025	17500
Pfizer	16591	4590	18683
Quarantine	16589	7116	19004
Restrictions	37651	24298	40161
Vaccine	171180	149126	170466
TOTAL	283511	190773	296555

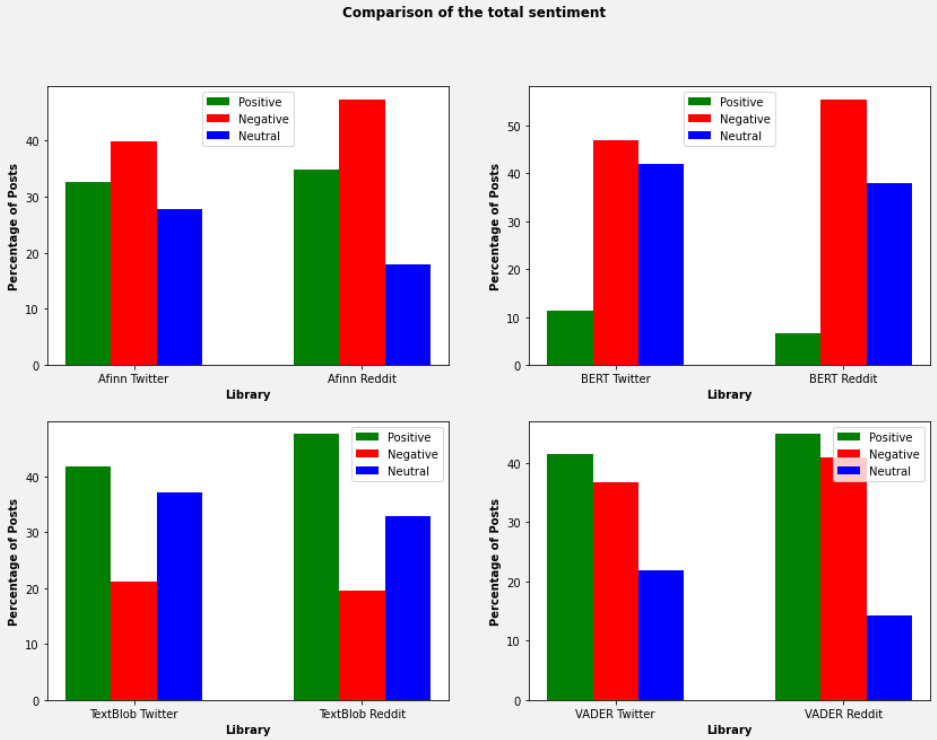
EVALUATION - MODELS

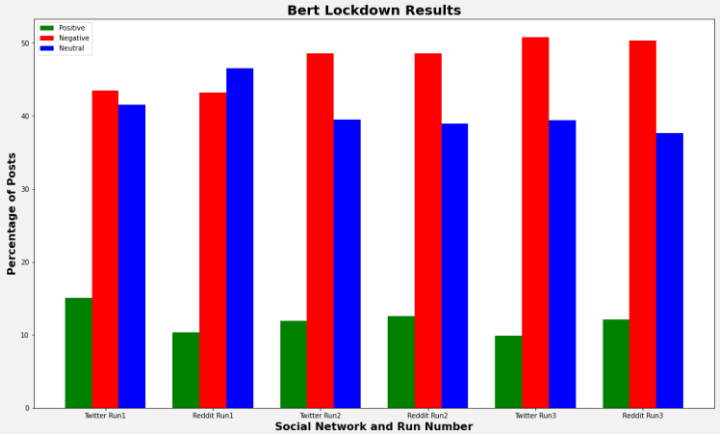


	Afinn	BERT	TextBlob	VADER
F1 score	0.423	0.549	0.400	0.417
Accuracy	0.470	0.617	0.478	0.463
Precision	0.436	0.554	0.407	0.437
Recall	0.512	0.638	0.433	0.505

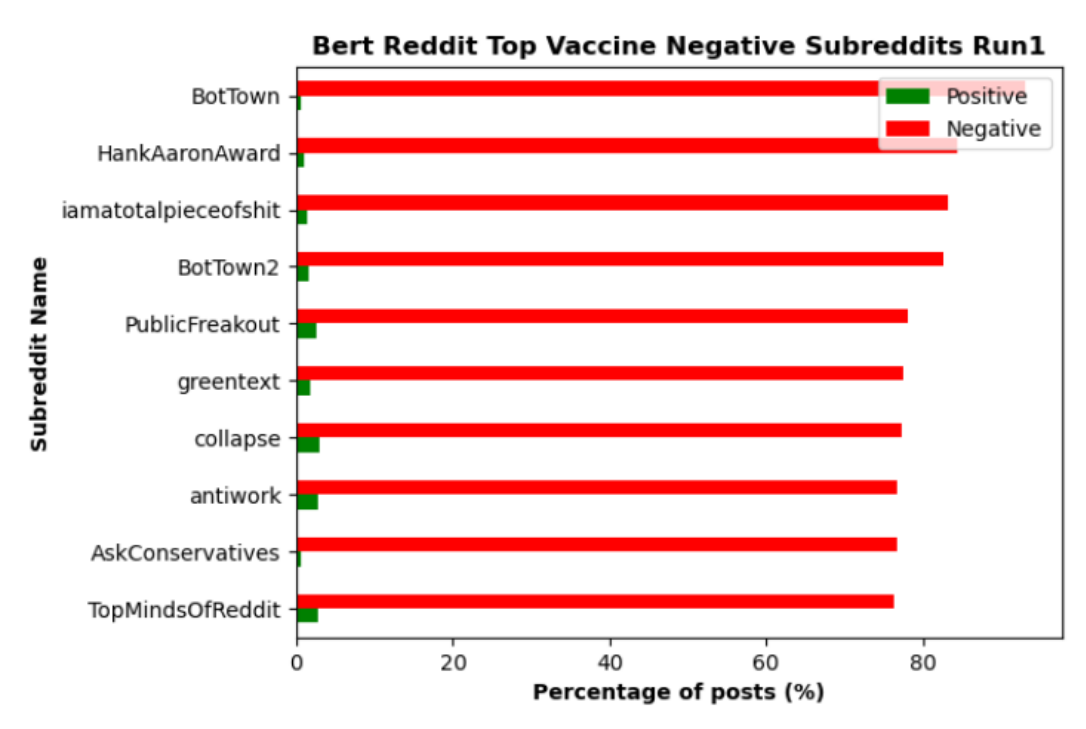
	Afinn	BERT	TextBlob	VADER
F1 score	0.431	0.549	0.408	0.421
Accuracy	0.481	0.617	0.495	0.471
Precision	0.440	0.554	0.411	0.438
Recall	0.513	0.638	0.435	0.502

EVALUATION – SENTIMENT ANALYSIS

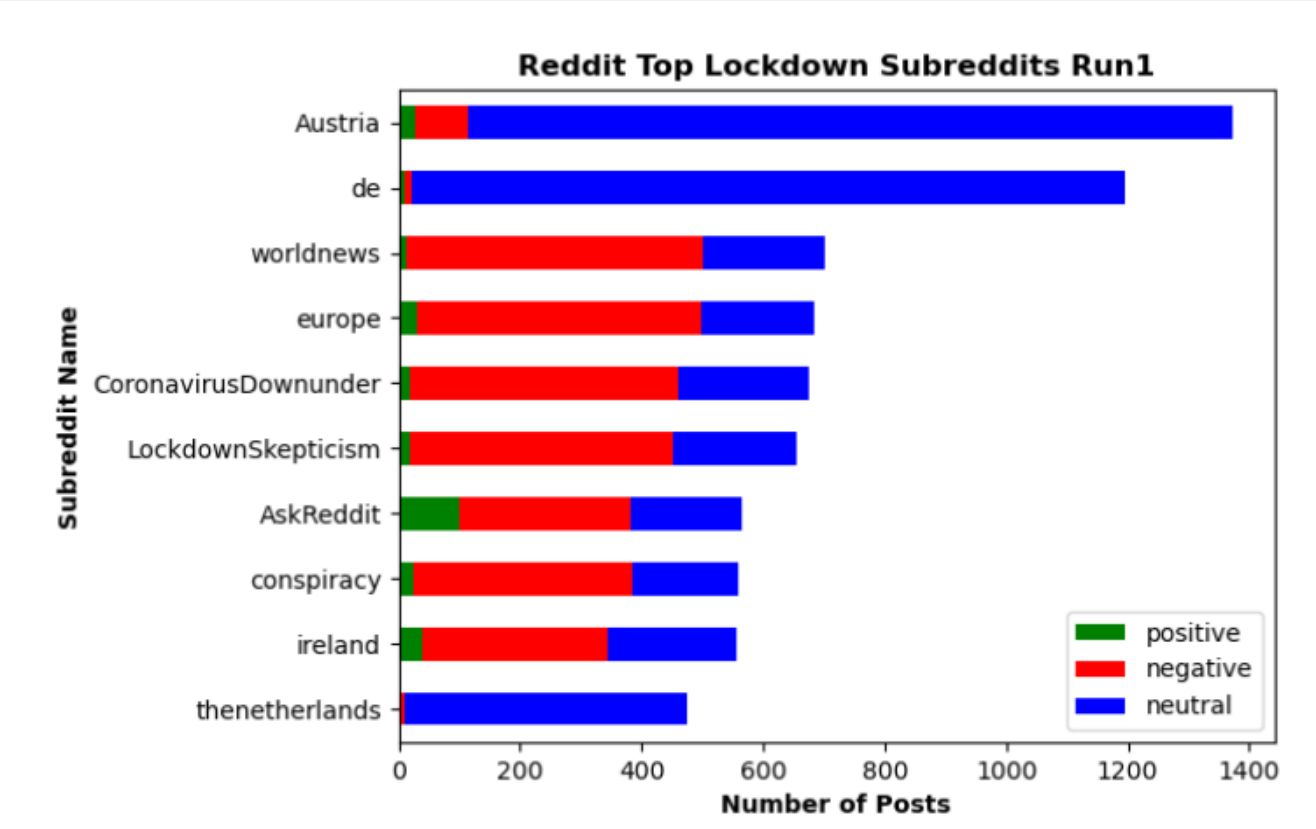




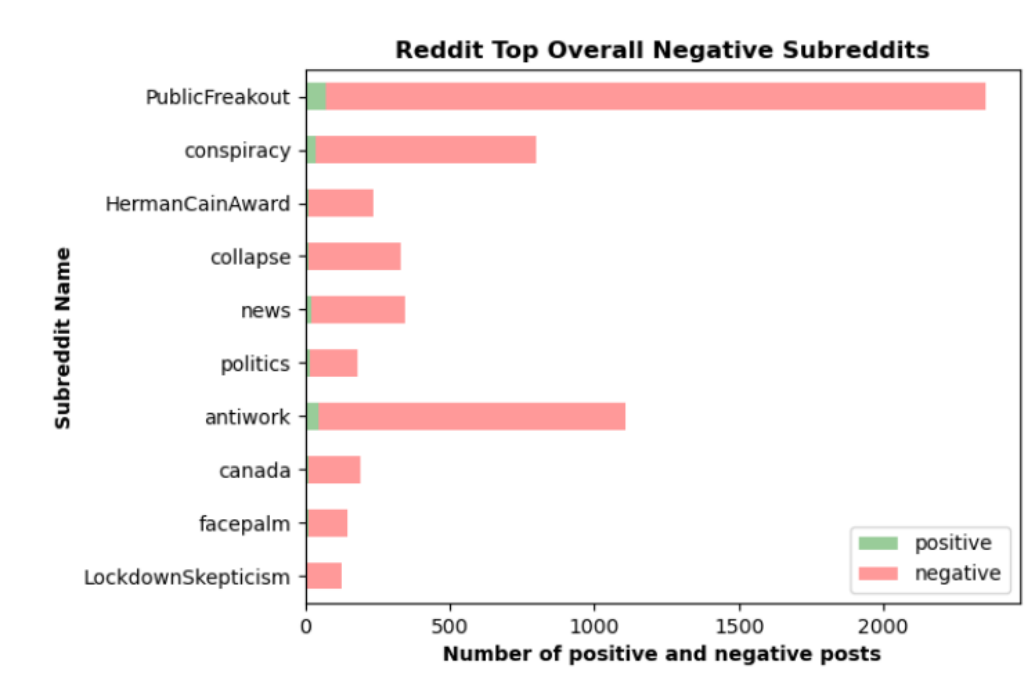
EVALUATION – SUBREDDIT SENTIMENT ANALYSIS

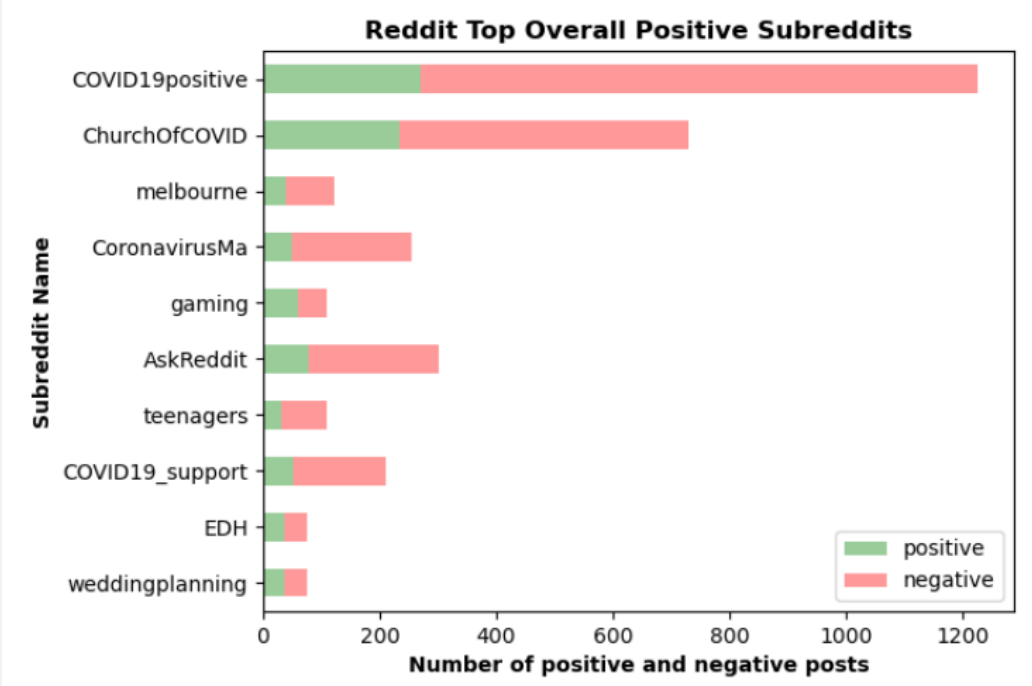


EVALUATION – SUBREDDIT SENTIMENT ANALYSIS



EVALUATION – SUBREDDIT SENTIMENT ANALYSIS





CONCLUSION

- Summary
- Reflection and Limitations
- Future work