APPENDIX

SCRAPPING CODING:

1. **IMPORT LIBRARY**

```
1.import json
2.import csv
3.import tweepy
4.import re
```

1. **CALL FUNCTION**

```
1.def   search_for_hashtags(consumer_key,   consumer_secret,
access_token, access_token_secret, hashtag_phrase):
2.#create authentication for accessing Twitter
3.auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
4.auth.set_access_token(access_token, access_token_secret)
5.#initialize Tweepy API
6.api = tweepy.API(auth)
7.#get the name of the spreadsheet we will write to
8.fname = '_'.join(re.findall(r"#(\w+)", hashtag_phrase))
9.#open the spreadsheet we will write to
10.with open('%s.csv' % (fname), 'w') as file:
11.w = csv.writer(file)
12.head=['timestamp',        'tweet_text',        'username',
'all_hashtags', 'followers_count']
13.#write header row to spreadsheet
14.w.writerow(head)
```

```
15.with open('%s.csv' % (fname), 'a', encoding="utf-8") as file:
16.w = csv.writer(file)
17.#for each tweet matching our hashtags, write relevant info to the spreadsheet
18.for tweet in tweepy.Cursor(api.search, q=hashtag_phrase+' -filter:retweets', lang="en", tweet_mode='extended').items(100):
19.w.writerow([tweet.created_at, tweet.full_text.replace('\n',' ').encode('utf-8').decode('utf-8'), tweet.user.screen_name.encode('utf-8').decode('utf-8'), [e['text'].encode('utf-8').decode('utf-8') for e in tweet._json['entities']['hashtags']], tweet.user.followers_count])
```

3. REQUEST API KEY, CONSUMER KEY, AND ACCESS TOKEN

```
1.consumer_key = input('Consumer Key ')
2.consumer_secret = input('Consumer Secret ')
3.access_token = input('Access Token ')
4.access_token_secret = input('Access Token Secret ')
5.hashtag_phrase = input('Hashtag Phrase ')
6.if __name__ == '__main__':
7.    search_for_hashtags(consumer_key, consumer_secret, access_token, access_token_secret, hashtag_phrase)
```

PRE-PROCESSING

```
1.import nltk
2.nltk.download('stopwords')
3.nltk.download('punkt')
```

```
4.from nltk.corpus import stopwords
5.from nltk.tokenize import RegexpTokenizer
6.Stopwords = set(stopwords.words('english'))
7.import numpy as np
8.import matplotlib.pyplot as plt
9.import pandas as pd
10.from nltk.tokenize import word_tokenize
11.data=pd.read_csv('file/SumpahPemuda.csv')
12.tweets = data.iloc[:,1]
13.label = data.iloc[:,5]
```

1. TOKENIZE AND STOPWORDS

```
1.import re
2.def tokens(text):
3.    return re.findall('[\w]+', text.lower())
4.listalltoken = []
5.filtered = []
6.for t in tweets:
7.    words_token = tokens(t)
8.    listalltoken.append(words_token)
9.    filtered_word = []
10.    for w in words_token:
11.        if w.lower() not in Stopwords:
12.            filtered_word.append(w.lower())
13.    filtered.append(filtered_word)
14.print(filtered)
```

1. STEMMING AND LISTING

```
1.from nltk.stem import PorterStemmer
2.porter = PorterStemmer()
3.liststem = []
4.for sf in filtered:
5.    listwordstem = []
6.    for wf in sf:
7.        listwordstem.append(porter.stem(wf))
8.    liststem.append(listwordstem)
9.print (liststem)
10.import numpy as np
11.vectorword =[]
12.for d in liststem:
13.    for t in d:
14.        vectorword.append(t)
15.list_term = np.unique(vectorword)
16.for k in np.unique(vectorword):
17.    print(k)
```

CALCULATE AND ALGORITHM

1. TF-IDF DATA TESTING

```
1.term_info_freq =[]
2.for term in list_term:
3.    i = 0;
4.    temp = []
```

```
    jml = 0
    for lists in liststem:
        count = lists.count(term)
        jml += count
        temp1 = {"doc": i, "count" : count}
         temp.append(temp1)
         i +=1
    term_info = {"term": term, "info":temp, "df" : jml}
    term_info_freq.append(term_info)
print(term_info_freq)
import math
idf = []
for tf in term_info_freq:
    idf.append(math.log(len(liststem)/tf["df"]))
print(idf)
list_tfidf=[]
i =0
for tf in term_info_freq:
    x =0
    tf_idf = []
    for doc in tf["info"]:
        temp = {"doc":x , "w": doc["count"] * idf[i]}
        tf_idf.append(temp)
        x+=1
    list_tfidf.append({"term": tf["term"], "info": tf_idf})
    i+=1
```

```
31.print(list_tfidf)
```

1. TF-IDF DATA TRAINING

```
1.words = [
```

"bullshit. This is why we stand against you. For everyone who suffered for your bullshit. #NightOfBrokenGlass #Racism #NeverAgain #NeverForget",

"Oversights, bad planning, inattention to important details affects lives. This event hosting is important to the Philippines but mistakes need to be pointed out and learned from. #SeaGames2019 #SEAGames2019fail #SEAG2019NOTAFAILURE #WeWinAsOne #SeaGames",

"#Indonesia, a monolithic, unproductive, resigned, awfully religious clusterof islands stripped off almost all of its natural resources as well as original fauna and flora; islands polluted, inhabited by uneducated and increasingly aggressive and intolerant people #WestPapua"

"And they called it #WhiteSupremacy? No, it's #WhiteStupidity! #Bullshit"

"Oh my god, what a horrific and disgusting story - trigger warning - two Indian teenagers urinated on a young black girl and called her the N word in NJ - #AntiBlackViolence #AntiBlackRacism #WhiteSupremacy"

"Last day to donate to Wreaths Across America, which honors fallen veterans during the holidays. Any amount helps! #veterans #america #freedomisntfree #khs"

"What I'm sayin is the average American is a lazy, stupid, myopic, losers, who would rather get the next IPhone than protest to preserve what the greatest generation fought and died for. #freedomisntfree"

"Most Americans will enjoy their Holiday at home. Others will be on Mess Duty, Fire watch, Gaurd Duty or Patrol. Thank you for you service and sacrifice brothers and sisters. @USMC @USNavy @USArmy @usairforce @USCG #Freedomisntfree #SemperFi"

"With protests continuing the Hong Kong government are still considering a full or partial #internetshutdown in an attempt to quell unrest. This tactic will only exacerbate the problem. We urge authorities to #KeepItOn."

"The lawsuit against the Minister of Telecommunication in Iraq states also that the minister should pay all fees and expenses related to the network disruptions. #KeepItOn"

```
      ]
2.import re
3.def tokens(text):
4.    return re.findall('[\w]+', text.lower())
5.ujitoken = []
6.try_filter = []
7.for t in words:
8.    uji_token = tokens(t)
9.    ujitoken.append(uji_token)
10.    filtered_word = []
11.    for w in uji_token:
12.        if w.lower() not in Stopwords:
13.            filtered_word.append(w.lower())
14.    try_filter.append(filtered_word)
15.print(try_filter)
16.from nltk.stem import PorterStemmer
17.porter = PorterStemmer()
```

```
ujistem = []
for sf in try_filter:
    uji_stem = []
    for wf in sf:
        uji_stem.append(porter.stem(wf))
    ujistem.append(uji_stem)
print (ujistem)
term_uji_freq =[]
for term in list_term:
    i = 0;
    temp = []
    jml = 0
    for lists in ujistem:
        count = lists.count(term)
        jml += count
        temp1 = {"doc": i, "count" : count}
        temp.append(temp1)
        i +=1
    term_info = {"term": term, "info":temp, "df" : jml}
    term_uji_freq.append(term_info)
print(term_uji_freq)
uji_tfidf=[]
i =0
for d in ujistem:
    results = dict.fromkeys(list_term, 0.0)
    for tf in term_uji_freq:
```

```
44.         x =0
45.         tf_idf = []
46.         doc = tf["info"]
47.         results[tf['term']] = doc[i]["count"] * idf[i]
48.     i+=1
49.     uji_tfidf.append(results)
50.print(uji_tfidf)
```

K-NEAREST NEIGHBOUR ALGORITHM

```
1.import math
2.import numpy as np
3.temp_euclid = []
4.for b in uji_tfidf:
5.     i = 0
6.     e_doc = np.zeros(len(liststem))
7.     for doc in liststem:
8.         for a in list_tfidf:
9.             square = a['info'][i]['w']-b[a['term']]
10.             temp = math.pow(square,2)
11.             e_doc[a['info'][i]['doc']] += temp
12.         e_doc[i] = math.sqrt(e_doc[i])
13.         i+=1
14.     temp_euclid.append(e_doc)
15.for d in temp_euclid:
16.     minv = min(d)
17.     result = (np.where(d == np.amin(d)))
```

```
18.     print(label[result[0]])
19.     print(minv)
20.     print(d)
```

UNICHECK

Submission author:
15k10065 REVIVAL DANISAPUTRA

Check date:
17.01.2020 13:28:48 GMT+0

Report date:
17.01.2020 23:29:32 GMT+0

Check ID:
15996269

Check type:
Doc vs Internet + Library

User ID:
29129

Unika SOEGIJAPRANATA

File name: 15.K1.0065_RevivalDanisaputra.docx

File ID: 20298497 Page count: 8 Word count: 3584 Character count: 21423 File size: 42.99 KB

1.14% Matches

Highest match: 0.25% with source https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/3/html-single/archite...

1.14% Internet Matches 5 ..... Page 10

No Library Sources Found

0.31% Quotes

Quotes 1 ..... Page 11

No references found

0% Exclusions

No exclusions found

Replacement

No replaced characters found