segalgouldn Directory cleaning and file exporting. adb4c40 2 minutes ago

1 contributor

360 lines (273 sloc) | 12.6 KB

Script for running a Twitter bot that interacts with subtweets

Import some libraries

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction import text
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
from sklearn.model_selection import KFold
from sklearn.externals import joblib
from nltk.corpus import stopwords
from string import punctuation
from pprint import pprint
from random import choice
from time import sleep

import pandas as pd
import numpy as np

import itertools
import enchant
import tweepy
import nltk
import json
import re
```

Prepare the probability threshold for interacting with a potential subtweet and the duration for which the bot should run

```
THRESHOLD = 0.75 # 75% positives and higher, only
DURATION = 60*15 # 15 minutes
```

Set up regular expressions for genericizing extra features

```
hashtags_pattern = re.compile(r'(\#[a-zA-Z0-9]+)')
```

```
urls_pattern = re.compile(r'(?i)\b((?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-z]{2,4}/)(?:[^\s()<>]|\(([^\s()<>]+|(
```

```
at_mentions_pattern = re.compile(r'(?<=^|(?<=[^a-zA-Z0-9-\.]))@([A-Za-z0-9_]+)')
```

Load the classifier pipeline which was previously saved

```
sentiment_pipeline = joblib.load("../data/other_data/subtweets_classifier.pkl")
```

Load the Twitter API credentials

```
consumer_key, consumer_secret, access_token, access_token_secret = open("../../credentials.txt").read().split("\n")
```

Connect to the API

```
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth, retry_delay=1, timeout=120, # 2 minutes
                 compression=True,
                 wait_on_rate_limit=True, wait_on_rate_limit_notify=True)
```

Create lists to fill with tweets while the bot is streaming

```
subtweets_live_list = []
non_subtweets_live_list = []
```

Use pyenchant to check if words are English

```
english_dict = enchant.Dict("en_US")
```

Use NLTK for tokenizing

```
tokenizer = nltk.casual.TweetTokenizer(preserve_case=False, reduce_len=True)
```

Create a custom StreamListener class for use with Tweepy

```
class StreamListener(tweepy.StreamListener):
    def on_status(self, status):
        choices = ["retweet", "like", "retweet and like", "reply"]

        id_str = status.id_str
        screen_name = status.user.screen_name
        created_at = status.created_at
        retweeted = status.retweeted
        in_reply_to = status.in_reply_to_status_id_str

        # The text and media of the tweet vary based on if it's extended or not
        if "extended_tweet" in status._json:
            if "full_text" in status._json["extended_tweet"]:
                text = status._json["extended_tweet"]["full_text"]
                has_media = "media" in status._json["extended_tweet"]["entities"]
            else:
                pass # Something else?
        elif "text" in status._json:
            text = status._json["text"]
            has_media = "media" in status._json["entities"]

        # Genericize extra features and clean up the text
        text = (hashtags_pattern.sub("❶",
                urls_pattern.sub("❷",
                at_mentions_pattern.sub("❸",
                text)))
                .replace("\u2018", "'")
                .replace("\u2019", "'")
                .replace("\u201c", "\"")
                .replace("\u201d", "\"")
                .replace("&quot;", "\"")
                .replace("&amp;", "&")
                .replace("&gt;", ">")
```

```
            .replace("&lt;", "<"))

        tokens = tokenizer.tokenize(text)

        english_tokens = [english_dict.check(token) for token in tokens]
        percent_english_words = sum(english_tokens)/len(english_tokens)

        # Make sure the tweet is mostly english
        is_mostly_english = False
        if percent_english_words >= 0.5:
            is_mostly_english = True

        # Calculate the probability using the pipeline
        positive_probability = sentiment_pipeline.predict_proba([text]).tolist()[0][1]

        row = {"tweet": text,
               "screen_name": screen_name,
               "time": created_at,
               "subtweet_probability": positive_probability}

        print_list = pd.DataFrame([row]).values.tolist()[0]

        # Only treat it as a subtweet if all conditions are met
        if all([positive_probability >= THRESHOLD,
                "RT " != text[:3], is_mostly_english,
                not retweeted, not in_reply_to, not has_media]):

            decision = choice(choices)
            if decision == "retweet":
                api.update_status(("Is this a subtweet? {:.3%} \n" +
                                   "https://twitter.com/{}/status/{}").format(positive_probability,
                                                                              screen_name,
                                                                              id_str))
                print("Retweet!")

            elif decision == "like":
                api.create_favorite(id_str)
                print("Like!")

            elif decision == "retweet and like":
                api.update_status(("Is this a subtweet? {:.3%} \n" +
                                   "https://twitter.com/{}/status/{}").format(positive_probability,
                                                                              screen_name,
                                                                              id_str))
                api.create_favorite(id_str)
                print("Retweet and like!")

            elif decision == "reply":
                api.update_status("@{} Is this a subtweet? {:.3%}".format(screen_name,
                                                                          positive_probability),
                                  id_str)
                print("Reply!")

            subtweets_live_list.append(row)
            subtweets_df = pd.DataFrame(subtweets_live_list).sort_values(by="subtweet_probability",
                                                                         ascending=False)

            subtweets_df.to_csv("../data/data_from_testing/live_downloaded_data/subtweets_live_data.csv")

            print(("Subtweet from @{0} (Probability of {1:.3%}):\n" +
                   "Time: {2}\n" +
                   "Tweet: {3}\n" +
                   "Total tweets acquired: {4}\n").format(print_list[0],
                                                          print_list[1],
                                                          print_list[2],
                                                          print_list[3],
                                                          (len(subtweets_live_list)
                                                           + len(non_subtweets_live_list))))

            return row
        else:
```

```
            non_subtweets_live_list.append(row)
            non_subtweets_df = pd.DataFrame(non_subtweets_live_list).sort_values(by="subtweet_probability",
                                                                                 ascending=False)
            non_subtweets_df.to_csv("../data/data_from_testing/live_downloaded_data/non_subtweets_live_data.csv")

            return row
```

Create a function for downloading IDs if users I follow who also follow me

```
def get_mutuals():
    my_followers = [str(user_id) for ids_list in
                    tweepy.Cursor(api.followers_ids,
                                  screen_name="NoahSegalGould").pages()
                    for user_id in ids_list]
    my_followeds = [str(user_id) for ids_list in
                    tweepy.Cursor(api.friends_ids,
                                  screen_name="NoahSegalGould").pages()
                    for user_id in ids_list]

    my_mutuals = list(set(my_followers) & set(my_followeds))

    bots = ["890031065057853440", "895685688582180864",
            "894658603977777152", "970553455709446144",
            "786489395519983617", "975981192817373184"]

    # Remove known twitter bots
    my_mutuals = [m for m in my_mutuals if m not in bots]

    with open("../data/other_data/NoahSegalGould_Mutuals_ids.json", "w") as outfile:
        json.dump(my_mutuals, outfile, sort_keys=True, indent=4)

    return my_mutuals
```

Create a function for downloading IDs of users who follow my mutuals who are also followed by my mutuals

```
def get_mutuals_and_mutuals_mutuals_ids(mutuals_threshold=250):
    my_mutuals = get_mutuals()
    my_mutuals_mutuals = my_mutuals[:]

    for i, mutual in enumerate(my_mutuals):
        start_time = time()
        user = api.get_user(user_id=mutual)
        name = user.screen_name
        is_protected = user.protected
        if not is_protected:
            mutuals_followers = []
            followers_cursor = tweepy.Cursor(api.followers_ids, user_id=mutual).items()
            while True:
                try:
                    mutuals_follower = followers_cursor.next()
                    mutuals_followers.append(str(mutuals_follower))
                except tweepy.TweepError:
                    sleep(30) # 30 seconds
                    continue
                except StopIteration:
                    break
            mutuals_followeds = []
            followeds_cursor = tweepy.Cursor(api.friends_ids, user_id=mutual).items()
            while True:
                try:
                    mutuals_followed = followeds_cursor.next()
                    mutuals_followeds.append(str(mutuals_followed))
                except tweepy.TweepError:
                    sleep(30) # 30 seconds
                    continue
                except StopIteration:
                    break
```

```
            mutuals_mutuals = list(set(mutuals_followers) & set(mutuals_followeds))
            print("{} mutuals for mutual {}: {}".format(len(mutuals_mutuals), i+1, name))
            if len(mutuals_mutuals) <= mutuals_threshold: # Ignore my mutuals if they have a lot of mutuals
                my_mutuals_mutuals.extend(mutuals_mutuals)
            else:
                print("\tSkipping: {}".format(name))
        else:
            continue
        end_time = time()
        with open("../data/other_data/NoahSegalGould_Mutuals_and_Mutuals_Mutuals_ids.json", "w") as outfile:
            json.dump(my_mutuals_mutuals, outfile, sort_keys=True, indent=4)
        print(("{0:.2f} seconds for getting the mutuals' IDs of mutual {1}: {2}\n")
              .format((end_time - start_time), i+1, name))
    my_mutuals_mutuals = [str(mu) for mu in sorted([int(m) for m in list(set(my_mutuals_mutuals))])]
    with open("../data/other_data/NoahSegalGould_Mutuals_and_Mutuals_Mutuals_ids.json", "w") as outfile:
        json.dump(my_mutuals_mutuals, outfile, indent=4)
    return my_mutuals_mutuals
```

```
# %%time
# my_mutuals_mutuals = get_mutuals_and_mutuals_mutuals_ids()
```

Load the IDs JSON

```
my_mutuals_mutuals = json.load(open("../data/other_data/NoahSegalGould_Mutuals_and_Mutuals_Mutuals_ids.json"))
```

```
print("Total number of my mutuals and my mutuals' mutuals: {}".format(len(my_mutuals_mutuals)))
```

```
Total number of my mutuals and my mutuals' mutuals: 4218
```

Begin streaming

```
stream_listener = StreamListener()
stream = tweepy.Stream(auth=api.auth, listener=stream_listener, tweet_mode="extended")
```

```
%%time
# stream.filter(locations=[-73.920176, 42.009637, -73.899739, 42.033421],
# stall_warnings=True, languages=["en"], async=True)
stream.filter(follow=my_mutuals_mutuals, stall_warnings=True, languages=["en"], async=True)
print("Streaming has started.")
sleep(DURATION)
stream.disconnect()
```

```
Streaming has started.
Retweet!
Subtweet from @fka_zigs (Probability of 75.831%):
Time: 2018-04-23 02:28:26
Tweet: my mom thinks everyone on the internet is a catfish
Total tweets acquired: 872

Retweet!
Subtweet from @JillMurphy0421 (Probability of 77.362%):
Time: 2018-04-23 02:29:35
Tweet: get you a roommate who wallows in mutual lonesome with you like mine does
Total tweets acquired: 951

CPU times: user 22.1 s, sys: 3.57 s, total: 25.7 s
Wall time: 15min
```