Sprawozdanie do projektu Fake News Detector

1. Wstęp

Projekt Fake Tweet Detector realizuje funkcjonalność weryfikowania treści o fałszywym charakterze, pochodząca z Twitter'a. Fałszywy charakter postu określa się na podstawie :

- wiarygodności tweeta - oparta, głównie na analizie treści postu, ale również innych czynników tj. ilość retweetów, odpowiedzi, polubień itp.
- pochodzenia tweeta - sprawdzana jest wiarygodność autora tweeta, na podstawie wyniku weryfikacji przeprowadzonej przez serwis Twitter.com oraz ilości obserwujących, obserwowanych i ulubionych itp. konta autora danego tweetu. Dodatkowo sprawdzany jest profil autora pod względem tego, czy zachowuje się jak bot. (więcej poniżej)
- wiarygodności stron, zawartych w odnośnikach dołączonych do tweeta

2. Technologie

Python

Projekt jest zaimplementowany w języku Python 3.0. Dane do analizy pobierane są z Twitter'a za pośrednictwem udostępnionego przez service API. Korzystamy z tego API pośrednio korzystając z biblioteki tweepy.

MongoDB

Dane gromadzimy w dokumentowej bazie danych MongoDB.
W niektórych modułach programu zastosowano modele uczenia maszynowego.

3. Podział programu na moduły

Program podzielony jest na kilka modułów, z których można korzystać, nie wywołując pozostałych. Moduły te to :

TweetLoader

Implementacja tego modułu znajduje się w pliku src/mongoDB/TweetLoader.py . Odpowiada on za pobieranie danych z twitter.com i zapisywanie tych danych do bazy danych. W konstruktorze tej klasy znajdują się 2 argumenty :

- to_restart - determinuje czy mongoDB ma być zrestartowana (drop wszystkich kolekcji users i tweets)
- max_reply - limit odpowiedzi na tweeta, które mogą być wczytane

Najważniejsze metody, które pobierają, "czyszczą" z niepotrzebnych danych i zapisują tweety to :

saveUser(self, screen_name, to_print=**False**): - zapisuje do bazy danych json zawierający informacje o użytkowniku z loginem *screen_name.* Przykładowy JSON wygląda następująco:

```
{
        "contributors_enabled": false,
```

```
  "created_at": "Wed Mar 18 13:46:38 +0000 2009",
  "description": "45th President of the United States of
America\ud83c\uddfa\ud83c\uddf8",
  "favourites_count": 7,
  "followers_count": 60687236,
  "following": false,
  "friends_count": 47,
  "geo_enabled": true,
  "has_extended_profile": false,
  "is_translation_enabled": true,
  "is_translator": false,
  "lang": null,
  "listed_count": 103676,
  "location": "Washington, DC",
  "name": "Donald J. Trump",
  "profile_location": null,
  "protected": false,
  "screen_name": "realDonaldTrump",
  "statuses_count": 42122,
  "time_zone": null,
  "verified": true
}
```


saveTweet(self, id, to_print=**False**, with_author=**False**): - zapisuje do bazy danych json zawierający informacje o tweecie z podanym jako argument id. Przykładowy JSON wygląda następująco:


```
{
        'created_at': 'Tue May 28 08:11:10 +0000 2019',
        'id': 1133284566632787969,
        'full_text': 'Spoke with @GovStitt of Oklahoma last night from Japan because of the
        devastating tornadoes. Told him that @FEMA and the federal government are fully
        behind him and the great people of Oklahoma.',
         'display_text_range': [0, 194],
        'entities': {
                'hashtags': [], 'symbols': [], 'user_mentions': [{'screen_name': 'GovStitt',
                'name': 'Governor Kevin Stitt', 'id': 865210877678686208, 'id_str':
                '865210877678686208', 'indices': [11, 20]}, {'screen_name': 'fema', 'name':
                'FEMA', 'id': 16669075, 'id_str': '16669075', 'indices': [107, 112]}], 'urls': []
        },
        'in_reply_to_status_id': None,
        'in_reply_to_screen_name': None,
        'is_quote_status': False,
        'retweet_count': 13101,
        'favorite_count': 62629,
        'lang': 'en',
        'user_mentions': ['GovStitt', 'fema', 'GovStitt', 'fema'],
        'hashtags': [],
```

```
        'connected_with_tweet': None,
        'screen_name': 'realDonaldTrump',
}
```

saveAuthor(self, id) - zapisuje do bazy danych autora tweeta o id = *id*

saveReplies(self, tweet, to_print=False, with_author=False): - zapisuje do bazy odpowiedzi na tweeta, podanego jako argument metody (tweet w postaci JSON jak powyżej), parametr with_author określa, czy zapisać również autorów odpowiedzi.

saveTweetsWithWords(self, words, connected_with_tweet=None, verified_authors_only=False, with_authors=False, to_print=False): zapisuje do bazy tweety, które zawiera słowa podane jako argument "words". Parametr with_author określa, czy zapisać również autorów tych tweetów. Limit ilości tzw. tweetów powiązanych ze słowami określany jest argumentem limit. Jeżeli verified_authors_only ustawione jest na True, zapisywane są tylko te tweety, których autorzy są zweryfikowani przez portal Twitter. .

saveLastTweetsOfAuthor(self, screen_name,to_print=**False**,size_for_bot=10) - zapisuje *size_for_bot* ostatnich tweetów autora. Ustawia *connected_with_tweet* w Jsonie każdego z tych tweetów jako *screen_name*

saveTweetWithAllData(self, id = -1, to_print = **False**, with_author=**True**, with_authors_of_replies=**False**, connected_tweets=**False**, verified_authors_only=**True**): - jest to najważniejsza metoda w klasie TweetLoader (i prawdopodobnie jedyna wywoływana bezpośrednio przez użytkownika/ w mainie). Argumenty wywołania decydują jak dużo danych zostanie zassanych do bazy danych.

- with_author - decyduje o tym czy autor tweeta z id = *id* będzie zapisany do bazy
- with_authors_of_replies - decyduje o tym, czy autorzy odpowiedzi na tweeta oraz ew. tweetów powiązanych, będą zapisani do bazy
- connected_tweets - decyduje o tym, czy zapisane zostaną tweety powiązane. Metoda wybiera z treści tweeta "istotne" dla tweet'a słowa i wywołuje metodę **saveTweetsWithWords** z argumentem words z tymi słowami.
- verified_authors_only - określa czy w tweetach powiązanych brać pod uwagę tweety tylko zweryfikowanych użytkowników czy wszystkich.

Fetcher

Klasa odpowiadająca za pobieranie danych z bazy danych. Z metod tej klasy korzystają wszystkie pozostałe moduły (oprócz **TweetLoader)**
Metody typu 'Get':

get_tweet(self,id) - zwraca JSON z tweetem o konkretnym *ID*

get_user(self,screen_name) - zwraca JSON z userem o konkretnym *screen_name*

get_author_of_tweet(self,id) - zwraca JSON z użytkownikiem, który jest autorem, tweeta o konkretnym *ID*

get_users_last_tweets(self,screename): zwraca tablice wszystkich ostatnich tweetów użytkownika screen_name z BD

get_connected(self,id) - zwraca tablice JSONów z tweetami, które są powiązana z tweetem o konkretnym *ID*

get_replies(self, id,verified_authors_only=**False**):zwraca tablice JSONów z tweetami, które są odpowiedziami na tweeta o konkretnym *ID*

Poniższe Moduły zwracają wartości od 0 do 1, które oznaczają prawdopodobieństwo tego, że analizowany news jest prawdziwy (nie jest fake), lub -1 w przypadku, gdy brak jest danych, by cokolwiek orzec.

PostCredibility

Klasa odpowiadająca za ocenianie wiarygodności, głównie na podstawie popularności tweeta, a także ładunku emocjonalnego jego treści.
Klasa zawiera metody:
evaluate(self, id): metoda analizująca:

-sentyment tweeta
-średni sentyment odpowiedzi
-subiektywność tweeta
-średni sentyment odpowiedzi
-procent zweryfikowanych kont, które napisały podobne tweety
-ilość obserwujących, retweetów, ulubionych

Prawdopodobieństwo fałszywości obliczane jest na podstawie zliczania punktów, które dostaje tweet za “kamienie milowe”, takie jak np. co najmniej 100 retweetów, czy 1000 obserwowanych. Punkty są odejmowane od tej sumy, w zależności od natężenia emocjonalnego w tweecie.

Jeśli analizowany tweet jest wysłany ze zweryfikowanego konta, nadawane jest mu prawdopodobieństwo fałszywości ‘0’.

BotChecker

Moduł sprawdzający fake’owatość tweeta na podstawie wykrywania użytkowników jako botów i badaniu załączonych odnośników URL w danym tweecie

przykładowy wynik tego modułu:

```
'probability': 1,
'description': 'Tweet był retweetowany - to nie jest bot'
```

Algorytm i schemat postępowania:

Moduł oparty jest na dwóch metodach - is_fake_based_on_user i is_fake_based_on_external_urls:

- is_fake_based_on_user(self, tweetId) - metoda sprawdza czestotliwosc postowania tweetow.
  Algorytm:

1. jesli sa jakies retweety tego tweeta to znaczy ze to nie jest bot czyli nie jest to fake news prawd.=1
2. jesli user nie opublikowal min 10 tweetow - moze to byc bot - tweet ustawiany na fake z prawd. = 0.3
3. jesli uzytkownik publikuje tweety czesciej niz co 2 dni - nie jest to bot, news nie jest fake z prawd.=1
4. jesli uzytkownik publikuje tweety czesciej niz 2 dni ale rzadziej niz 5 dni jest ustawiany tweet jako fake z prawd.=0.4
5. jesli uzytkownik publikuje rzadziej niz 5 dni tweet jest ustawiany na fake z prawd.=0.1

- is_fake_based_on_external_urls(self, tweetId, isMachineLearning) - metoda sprawdza zalaczone w tweecie odnosniki jako parametry wejsciowe oprocz tweetId przyjmuje wartosc True lub False
  Algorytm z isMachineLearning ustawionym na True:

1. jesli nie ma zadnego odnosnika w tweecie zwracamy wartosci -1
2. jesli ma odnosniki uzywamy uczenia maszynowego z klasy UrlMachineLearner ktora zwraca nam wartosc fake true albo false wrac z prawd.

Algorytm z isMachineLearning ustawionym na false:

1. Probujemy otworzyc strone z odnosnika
2. jesli kod responsu to error (kody z wartoscia wieksza od 400) to znaczy ze jest to fake i ustawiamy prawd.=0
3. w innym wypadku nie jest to fake z prawd.=1
4. Gdy nie jestesmy w stanie wejsc na odnosnik to znaczy ze jest to fake z prawd.=0

DataLoader

Moduł odpowiada za obróbkę danych przed wektoryzacją, oraz wczytanie danych uczących. Implementuje funkcje:
stem (*sentence*) - przyjmuje za argument listę tweetów w zmiennych tekstowych, każde ze słów poddaje funkcji stem obiektu *SnowballStemmer* z pakietu *NLTK*. Zwraca listę zmienionych tweetów.
lemmatize (*sentence*) - działa analogicznie analogicznie do funkcji stem, używając obiektu *WordNetLemmatizer* z pakietu *NLTK*.

Classifier

Moduł odpowiadający za znalezienie optymalnych parametrów wektoryzacji i funkcji klasyfikującej. Za pomocą biblioteki *joblib* zapisuje najlepszy model w pliku *prediction_model.sav*.

Predictor

Moduł odpowiedzialny za zwrócenie odpowiedzi z klasyfikatora.
predict(*self*, *id*): jako argument przyjmuje numer id tweeta z bazy danych. Pobiera go za pomocą modułu *Fetcher*, wczytuje model *prediction_model.sav*, a następnie zwraca jego klasyfikację tweeta w formie słownika zawierającego opis przypisania i prawdopodobieństwo, że news jest prawdziwy.

4. Działanie programu

W podstawowej wersji, program realizuje swoją funkcjonalność w następujących krokach.

1. wywołujemy program skryptem main.py z argumentem id tweeta, który chcemy zweryfikować oraz
2. Tweet Loader zapisuje wszystkie istotne dla działania pozostałych modułów dane - post, posty powiązane, odpowiedzi na post oraz autorów odpowiednich postów
3. Następnie wykonywana jest analiza danych pozostałymi modułami w takiej kolejności : PostCredibility, BotChecker, HyperLinkChecker, Predictor

4. Kolejnym krokiem jest podsumowanie wyników zebranych z powyższych modułów i zwrócenie werdyktu - Procentową szansę, że tweet jest fake newsem

Opcjonalnie można najpierw uruchomić skrypt TweetLoader.py z odpowiednim id jako argument, by pobrać dane i zachować je w bazie MongoDB. Następnie można uruchomić main.py z odpowiednimi parametrami, by pominąć etap ładowania danych do BD.

Schemat działania przedstawia grafika poniżej: