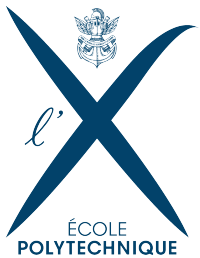
Data Stream project

Racism & hatred detection in tweets

K. Cottart, X. Loison, M. Noir

Institut Polytechnique de Paris

January 16, 2023



Outline

Table of Contents

Objective of the project

Background

During 2022 FIFA world cup, french football players have been targeted with racist insults. FFF expressed its desire to make a complaint. To do so, one should be able to identify and collect racist tweets so that it can file a complaint.

Objective

Identify racist tweets amid a tweet stream, display trends in real-time and keep a record of suspect users.

Table of Contents

Classification of tweets

General architecture of the classifier

Logical sequence

In order to identify tweets, we proceed in three successive steps:

- First, we build a classifier with two different data set
- Then, we identify tweets with negative polarity
- Finally, we identify tweets with racist terms

Principles

- Explainable
- Fast
- Adaptive

Classification of tweets

Model used

- Tokenize tweets from a labelled dataset with `Cleaner()`
- Convert tokens into features with TfidfVectorizer (statistical approach) to do keywords extraction (big corpus needed)[1]
- Fit the model with them (`LogisticRegression()`) to make predictions:
  - "*[...]this study uses five algorithms: [SVM, k-NNN, LR, NB & RF]. It is revealed from the results that out of the developed models,[...] LR model surpasses the other models in the IMDB dataset with an accuracy of 85.8% using the proposed system.*[2]"

[1] Gupta, Er. Tanya. "KEYWORD EXTRACTION : A REVIEW." (2017).

[2] Sayar Ul Hassan, Jameel Ahamed, Khaleel Ahmad. (2022). Analytics of machine learning-based algorithms for text classification. Sustainable Operations and Computers.

Classification of tweets

Datasets used

Desired features

- Possibility to switch between En & Fr
- Cleaned from anything but the text of the tweet and corresponding label

Data sets

- Pos/Neg tweets: sentiment140[3] (tweets with emoticons)
- Racist/non-racist tweets: MLMA[4] (made with DL)

[3] Go, Alec & Bhayani, Richa & Huang, Lei. (2009). Twitter sentiment classification using distant supervision. Processing. 150.

[4] Multilingual and Multi-Aspect Hate Speech Analysis (Ousidhoum et al., EMNLP-IJCNLP 2019)

Table of Contents

Architecture of the algorithm

Initialize the algorithm

To initialize the algorithm, we do:

- Initialize the application (multi-threading)
- Obtain API keys from `Secret()`
- Initialize the LR model with the `init` data set

Multi-threading is done with `multiprocessing` and `App()` :

```
def __init__(self, query, topic, lang, nb_tweets):
    """Class constructor [...]"""
    self.secrets = secret.Secret()
    self.ctx = mp.get_context('spawn')
    self.query = query
    self.topic = topic
    [...]
# Create the application
application = app.App(query, language[:2] + "_" + topic.lower() + "_tweets")
# Start the application
application.run()
```

Architecture of the algorithm

Multi-threading

`application.run()` :

```
def run(self):
    """ Run the program and create threads to analyse tweets"""
    try:
        # Start by training the classifier
        racism_hatred = analyser.Racist("./datasets/hatred_init_en.csv")
        racism_racist = analyser.Racist("./datasets/racist_init_en.csv")
        # Get tweets
        process_data_from_tweeter = self.ctx.Process(
            target=self.tweeter_to_kafka)
        process_data_from_tweeter.start()
        # Racist analysis
        process_analyse_racism = self.ctx.Process(
            target=self.analyse_racism_tweet, args=(
            racism_hatred, racism_racist))
        process_analyse_racism.start()
        # Clouds
        process_clouds = self.ctx.Process(target=self.generate_clouds)
        process_clouds.start()
```

Architecture of the algorithm

Retrieve tweets

To retrieve tweets, we:

- Instantiate `Ingestor` class & collect tweets from Twitter API
- With `Ingestor`, send tweets to Kafka through a producer
- Instantiate `Retriever` class & retrieve tweets

Retrieving is done with `Retriever()` :

```
class Retriever():
    def __init__(self, topics):
        """Class constructor"""
        self.consumer = KafkaConsumer(
            bootstrap_servers=['localhost:9092'],
            auto_offset_reset='earliest',
            value_deserializer=lambda x: loads(x.decode('utf-8')))
        self.topics = topics
        self.consumer.subscribe(self.topics)
```

Architecture of the algorithm

Ingesting tweets continuously, by batch (not using `Tweepy.StreamingClient()`)

`get_data_continuously()` :

```
def get_data_continuously(self, query, limit, topic, lang, timeLimit=0):
        """Get tweets continuously [...]"""
        is_time_true = True if timeLimit == 0 else False
        true_end_time = (datetime.datetime.utcnow()
                         + datetime.timedelta(seconds=timeLimit))
        start_time = (datetime.datetime.utcnow()
                      - datetime.timedelta(seconds=40))
        end_time = (datetime.datetime.utcnow()
                    - datetime.timedelta(seconds=30))
        while is_time_true or not true_end_time < datetime.datetime.utcnow():
            tweets = self.get_recent_tweets(query, limit, start_time, end_time)
            self.send_to_kafka(tweets, topic, lang, verbose)
            start_time = end_time
            end_time = start_time + datetime.timedelta(seconds=10)
            time.sleep(10)
```

Architecture of the algorithm

Classify & stream tweets

To stream racist tweets, we do:

- Apply the classifier to the tweet that has been "cleaned"
- Select hatred/racist tweets depending on a threshold
- Stream and display tweets, collect users' pseudos

Classifying tweets is done with `tweet_to_racism()` :

```
def tweet_to_racism(self, tweet, verbose=False):
    """Racist tone analysis of a tweet [...]"""
    cleaner = Cleaner(tweet, self.lang, self.stoplist)
    tweet = cleaner.to_tokens()
    new_features = self.vectorizer.transform([tweet])
    probability = self.model.predict_proba(new_features)[0][1]
    negative, score = self.negative_tweets(tweet)
    racist = negative and probability > self.threshold
```

Table of Contents

Outlooks

Difficulties, results and possible outlooks

Difficulties

- Find the good data sets
- Make the algorithm real-time

Results

- Effective algorithm, correct distribution but high FPR
- Real-time display

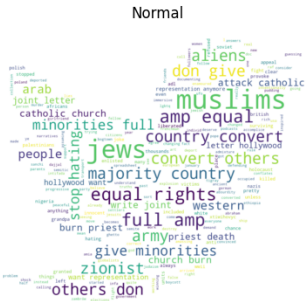
Outlooks

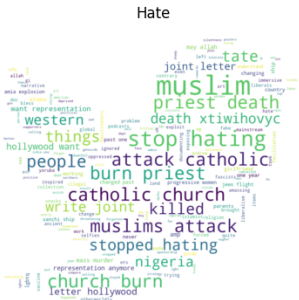
- Reduce FPR
  - Change the vectorizer (`GloveEmbedding`?)
- Use MLMA for homophobic tweets, switch to french...

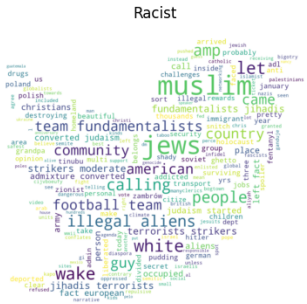
Outlooks

Results : Wordcloud words distribution

- Correct distribution, less obvious in real-time
- Trade off between window_size (memory) & distribution







Outlooks

- Program demonstration
- Questions & answers