1. Challenge

The goal of this challenge is to detect tweets that have racist or sexist intent.

2. Framework:

Since this is clearly a natural language processing (NLP) task, the problem was divided in three sequential parts. Text processing, feature extraction and modelling.

For each of this steps and for the entire pipeline, different approaches were tested and combined. As an example, two of the solutions tested are listed below:

- Removal of punctuation, strange characters, words with possible no meaning like @user or links, lowercase all text and stemming **+** TFIDF **+** XGBoost
- Handling of emojis, removal of punctuation, strange characters, words with possible no meaning like @user or links, lowercase all text and Lemmatization **+** TFIDF, Doc2Vec **+** Logistic Regression

All different approaches tested for each step are listed below:

Text Processing:

- Remove @user, emails, numbers, links, # symbol, words with only one letter and double spaces
- Use python TextBolb library to correct words (havv => have), this didn't work out that well since it also changed relevant words like "retweet".
- Use python textacy library to fix Unicode and take care of contractions (fig. 1)
- Use python emojis library to handle emojis (fig. 1)
- Remove stop words
- Stemming
- Lemmatization

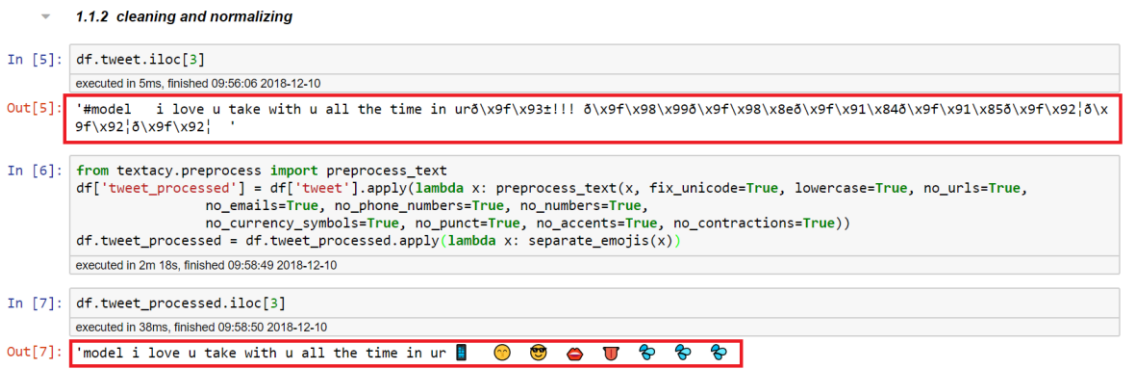

Figura 1 - example of the usage of textacy and emojis libraries to process the text and handle emojis

Feature Extraction:

- New features, such as: number of characters in each tweet, number of words, number of capitalized characters vs tweet length, number of exclamation marks
- Bag of words
- TFIDF
- Doc2Vec
- Glove pre-trained word vectors on twitter (200d) (fig. 2)
- Boolean variables for each of the 100 most common hashtags on racist/sexist tweets indicating if the specific hashtag is present or not in the tweet
- Boolean variables for each of the 1000 most common 1-gram/bi-gram on racist/sexist tweets, indicating if the specific 1-garm/bi-gram is present or not in the tweet
- Variable indicating for each tweet if it contained profanity or not (using python profanity library)
- Variables indicating the polarization and subjectivity of each tweet (using python TextBlob library)

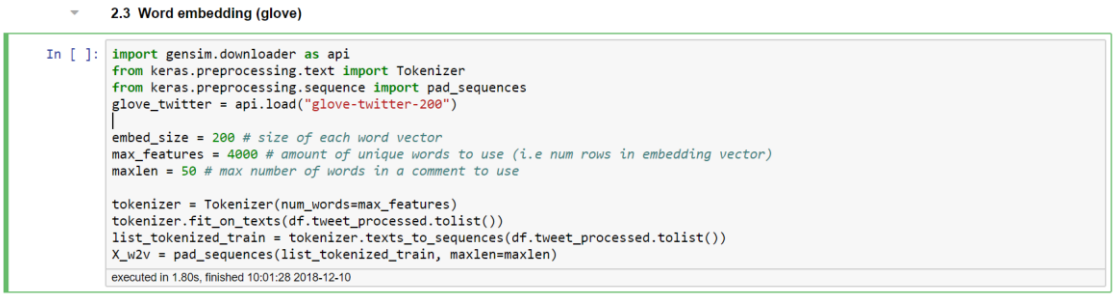

Figura 2 - code used to extract glove pre-trained word embeddings and to apply them to our data

Modelling

- Logistic Regression (scikit-learn)
- Random Forest (scikit-learn)
- XGBoost (xgboost)
- Simple Multilayer Neural Network (Keras)
- Bidirectional LSTM (Keras) (fig. 3)

All models were tested using 5 fold cross validation and tuned using randomized search.

```
In [ ]: kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=20)
f_train = []
f = []
#start cross validation
for train_idx, test_idx in kf.split(X, y):
    x_train, x_test = X[train_idx], X[test_idx]
    y_train, y_test = y[train_idx], y[test_idx]

    #create model
    model = Sequential()
    model.add(Embedding(max_features, embed_size, input_length = maxlen, weights=[embedding_matrix],trainable=True))
    model.add(Bidirectional(LSTM(256, dropout_U = 0.05, dropout_W = 0.05, return_sequences=False)))
    model.add(Dense(128))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Dropout(0.15))
    model.add(Dense(2, activation = 'softmax'))
    model.compile(loss = 'sparse_categorical_crossentropy', optimizer = 'adam')

    for i in range(30):
        model.fit(x_train, y_train, nb_epoch = 1, batch_size = 512,
                  verbose = 2, class_weight={0:base_weights[0],1:base_weights[1]},
                  validation_data=(x_test, y_test))

        y_train_pred = model.predict(x_train).argmax(axis=1)
        y_test_pred = model.predict(x_test).argmax(axis=1)
        f1train = metrics.f1_score(y_train, y_train_pred)
        f1test = metrics.f1_score(y_test, y_test_pred)
        f_train.append(f1train)
        f.append(f1test)
execution queued 22:34:33 2018-12-02
```

Figura 3 - code used to create the bi-direction LSTM model

3. Best Solution

3.1 Text processing

To normalize the text, all tweets were lowercased. To correct irregularities, miswritten words commonly present in racist/sexist tweets were checked and corrected (fig. 4), and regex expressions were used to take care of other irregularities:

- Remove links
- Remove @user
- Remove all non-letters symbols
- Remove double spaces

To reduce the inflectional form of each word into a common root, lemmatization was used. And finally, to remove common words that usually don't add much meaning to a sentence, the usual process of removing stop words were conducted.

```
In [4]: def correct_words(text):
            for k,v in dict_changes.items():
                text = text.replace(k, v)
            return text

        dict_changes = {'michelleobama':'michelle obama','phillysuppophilly':'philly suppo philly',
                        'donaldtrump': 'donald trump','blacklivesmatter':'black lives matter',
                        'newyork':'ny','theresistance':'the resistance','new york':'ny','whiteprivilege':'white privilege',
                        'whitesupremacy':'white supremacy','putinschoice':'putins choice','jeff session':'jeffsession','nyc':'ny',
                        'stopracism' :'stop racism','putinspuppet':'putin puppet','happyholidays':'happy holidays',
                        'policebrutality':'police brutality','feminismiscancer':'feminism is cancer','miamiâ':'miami accent',
                        'feminismmuktbharat':'feminism mukt bharat','suppoers':'supporters','seashepherd':'sea shepherd',
                        'carlpaladino':'carl paladino','uselections2016' :'us elections sixteen','emiratisâ':'emiratis accent',
                        '2016in4words':'sixteen in four words','2016':'sixteen','trumpâ':'trump accent',
                        'sexualpredator':'sexual predator','new york':'ny','wâ':'w accent','p2':'p two','theâ':'the','whâ':'wh accent',
                        'htâ':'ht accent','sâ':'s accent','thatâ':'that accent','hâ':'h accent','retweetâ':'retweet','99c':'nine nine c',
                        '99p':'nine nine p','tâ':'t accent'}
executed in 22ms, finished 17:51:44 2018-12-06
```

Figura 4 - miswritten words that are frequent in racist/sexist tweets

3.2 Feature extraction

In this step, four types of features were combined:

i) General features computed from the tweet text.
ii) Features indicating the presence of words common in racist/sexist tweets
iii) TFIDF
iv) Feature indicating the presence of profanity

i) Since racist/sexist tweets may have a different style of writing when comparing to the average tweet, the following variables were added:

- Number of characters
- Number of exclamation marks
- Number of question marks
- Number capital letters vs total number of characters
- Number of punctuation symbols

ii) Word cloud (fig. 5) was used to identify the 1000 most common 1-gram/bi-gram in racist/sexist tweets, and features indicating the presence of each of those words were created.

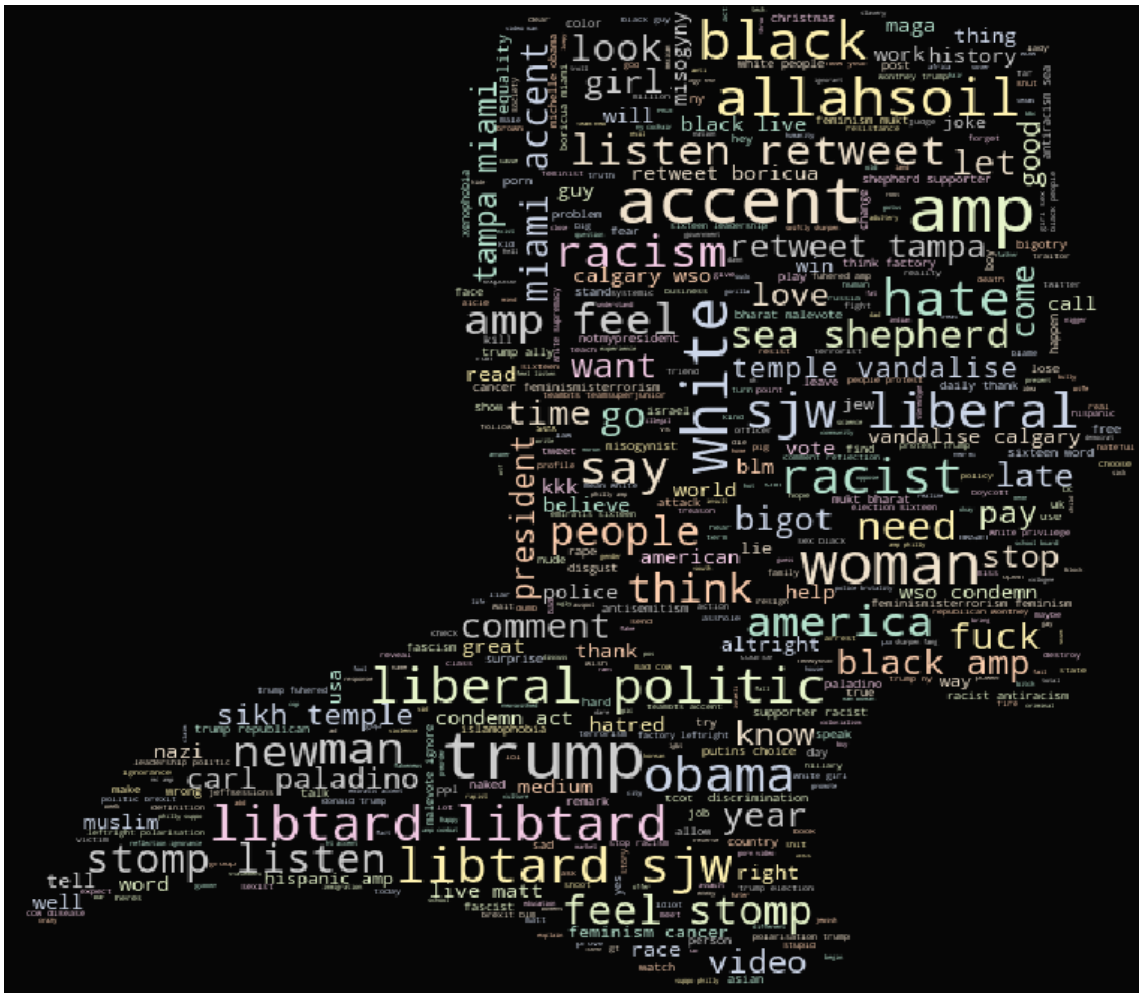

Figura 5 - word cloud gernerated based on the racists/sexist tweets

iii) To reflect the importance of a word for each tweet, a tfidf matrix using the 4000 more common words was created.

iv) To detect profanity and generate a variable indicating its presence in each tweet, a python library called “profanity” was used (fig. 6)

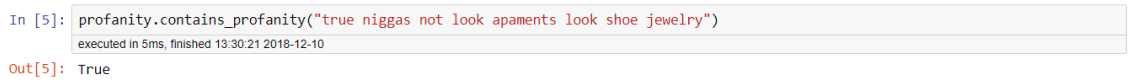

Figura 6 - example of the usage of profanity library to detect profanity in text

3.3 Modelling

3.3.1 Approach

Before choosing the model, the evaluation metric must be defined. In this case, and since we are in the presence of an unbalanced data set, f1-score and precision-recall curves were used to evaluate the outputs of each model.

Then, to assess the model and understand if its results are expected to generalize to the test set, the models were cross validated. And to tune the model’s hyper parameters randomized search was used.

Random forest was the model that yielded the best result. To cross-validate its results, a 5 fold cross-validation schema was used. And to tune the hyper parameters different parameters values, from the ones listed below, were combined:

- Whether bootstrap samples are used to build the trees: [True, False],
- Maximum depth of each tree: [5, 10, 20, 40, 60, 80, 100, None],
- Number of features to consider [sqrt(number of features), log2(number of features), number of features],
- Minimum number required to be at a leaf node: [1, 2, 4],
- Minimum number of samples required to split an internal node: [2, 5, 10],
- Number of trees: [100, 300, 500, 800]
- Class weights: [None, 0.5x(occurrences max class/occurrences min class), balanced]

Finally, to move the prediction in the precision and recall curve, with the intent of maximizing f1-score, different thresholds for the level of confidence score needed to predict that a tweet is racist/sexist, were tested.

3.3.2 Best Model

After iterating different models, and parameters, the best model at classifying racist/sexist tweets seems to be a Random forest with the following parameters:

- Whether bootstrap samples are used to build the trees: True,
- Maximum depth of each tree: None
- Number of features to consider: sqrt(number of features),
- Minimum number required to be at a leaf node: 1,

- Minimum number of samples required to split an internal node: 2,
- Number of trees: 300
- Class weights: None

The correspondent precision and recall curve is shown below:

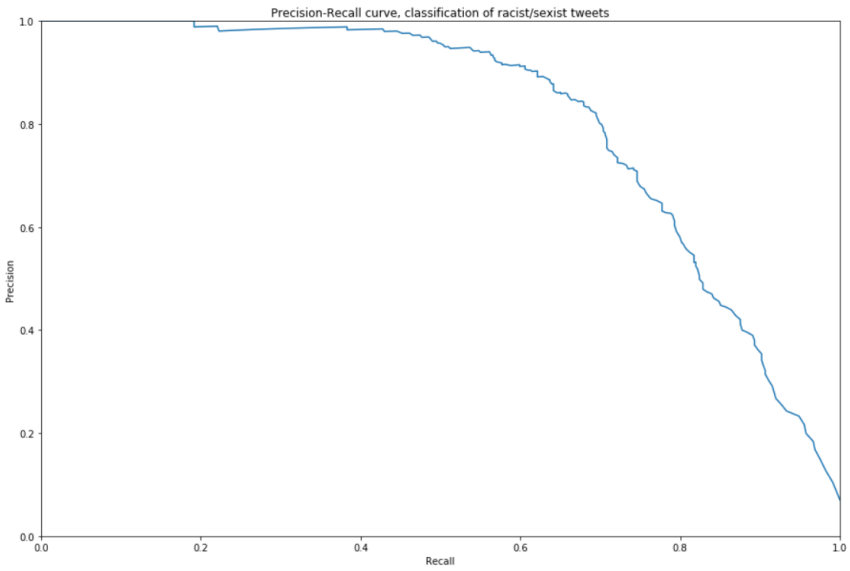

Figura 7 -precision-recall curve for the best model found

The classification threshold that seems to maximize f1 score the most, is 0.37, and yielded the following results:

	precision	recall	f1-score	support
0	0.98	0.99	0.98	5944
1	0.85	0.67	0.75	449
micro avg	0.97	0.97	0.97	6393
macro avg	0.91	0.83	0.87	6393
weighted avg	0.97	0.97	0.97	6393

4. Future works

There are some things that were not tested and can be done to improve the final solution:

- Generate new racist/sexist tweets to diminish class imbalance. This can be done by translating the existing racist/sexist tweets to other languages and then back to English.

- Analyze the tweets that are not being correctly labeled as racist/sexist. A simple analysis of word frequencies or other aspects of these tweets can be reviling, and may show us different solutions to tackle them.

- Use tools, like shapely additive explanations, to explain the current Machine Learning Model. That will tell us what features are relevant to our model when preforming classification, and thus it may help us understand the data and give us ideas to create new features and approaches.

- Use features others than word embedding in the LSTM. Concatenate tweet level features (tfidf, profanity, and other features) with the output of the LSTM before the fully connected layer, may be a good idea to improve the classification performance showed by these models in the tests I run.