flairDetector Documentation

Release latest

Feb 02, 2020

Contents

# flairDetector

A reddit flair detector project for Precog :heart:

This is a flask based project hosted on [Heroku] (https://evening-basin-22130.herokuapp.com/). It is developed entirely using Python.

PRAW was used to extract **170 posts** belonging to each of the flairs mentioned at */r/india*. 170 data per flair were chosen because a *memory limit error* showed up while trying to store more data than that on mongoDB. Running this project on Heroku often gave a *timeout error* when the data was extracted from mongoDB Atlas as it returns data in a batch of few hundreds and stops in between batches. To prevent that from happening, the code was tweaked a little to read from a **_pickle_** file rather than from the database. Everything that is on mongo database was also store in a file called *data.pickle* for Heroku. Since the file was too big and the code runs fine on local machine, the file was disregarded while uploading the project to Github.

The document was processed using *nltk* and stored on mongoDB Atlas.

Whenever required, all the data is extracted from the database (or the pickle file) and stored in the following format to make it compatible with the pandas library: ‘‘‘ {“title” : [list of titles of all the posts], “comments” : [list of comments on all extracted post], . . . }

‘‘‘

Once it is processed to the format shown above, the words and tokenized and the data set are split into training and testing data set. Following features were tested:

- Comments
- Title
- Body
- Title + comments
- Title +body
- Title + comments + body

It was seen that *title* + *comments* gave the highest accuracy with any model and *body* had the lowest accuracy. While going through the posts, it was seen that a lot of the posts did not have anything on their body, which might have been the reason for the low accuracy. Following models were used:

- Naïve bayes
- Logistic regression
- Random forest
- MLP classifier

### Accuracy

One of the highest accuracies was observed when *logistic regression* was used with *title* + *comments* as the feature. So, this model is used for the prediction.

The list of all the features and the models used, along with the accuracy of their performances on said features is given below:

#### Naïve Bayes

- Title: 0.5
- Body: 0.08045977011494253
- Comments: 0.3103448275862069
- Title + Comments: 0.3735632183908046

#### Regression

- Title: 0.5287356321839081
- Body: 0.16091954022988506
- Comments: 0.5057471264367817
- Title + Comments: 0.5747126436781609

#### Random Forest

- Title: 0.43103448275862066
- Body: 0.16666666666666666
- Comments: 0.27586206896551724
- Title + Comments: 0.367816091954023

#### MLP

- Title: 0.5402298850574713
- Body: 0.1724137931034483
- Comments: 0.5172413793103449
- Title + Comments: 0.5862068965517241

### Comment on the accuracy:

While searching for a particular flair as a keyword instead of using a query, the accuracy went up to 70+. But the predictions were off by a lot. Later it was found that searching for data using a keyword resulted in a lot of data belonging to popular flairs like Politics and very few data on flairs like Food and AMA. This increased the accuracy as a lot of posts were predicted as Politics. Using queries to find posts, that strictly belongs to a certain flair, resulted in decrease in the accuracy percent but led to more matches in terms of prediction. The results obtained using MLP were the best out of all the models but the model was discarded because of latency.

### Dependencies All the dependencies have been listed on the file [requirements.txt] (https://github.com/dibyaaaaax/flairDetector/blob/master/requirements.txt).

## How to run

### Clone the project to the local machine.

### Create a virtual environment and activate it. *virtualenv env envScriptsactivate*

### Install the dependencies using: *pip install -r requirements.txt*

### Run ‘python app.py ‘