SentiXplorer

Software Code Documentation

Team Members

Shreya Vaidya (svaidya6)
Dhruv Kolhatkar (dukolhat)
Aniket Darp (adarp)
Chinmay Walinjkar (cpwalinj)

INTRODUCTION

Sentiment analysis is one of the fastest growing research areas in computer science, making it challenging to keep track of all the activities in the area. In our project we aim to achieve our goal in accurately predicting a users sentiment by analyzing the data provided in any of the five different methods. They are Document Analysis, Comment Analysis, Text Analysis, Product Analysis and Audio Analysis. This project, though currently in the initial stages of development, can be further applied to numerous domains which can be useful for society. This document provides a major perspective for the users to understand and take up the project as an Open source software and add on multiple features before releasing to the market. Also, the document aids the developers in understanding the code and acts as a reference point for starting the project.

The complete development was achieved using the following technologies and it is recommended that the next set of developers who take up this project have these technologies installed and keep them running before proceeding further:

- Python3
- Django
- HTML
- CSS
- Scrappy
- Vader Analysis Tool

Although we have used HTML and CSS for the FrontEnd, the users can merge the backend logic with any of the front end frameworks they wish to use such as React, angularJS, etc.

CODE FUNCTIONALITIES

This section covers major files used in Django and small descriptions of each file with their functionalities.

- Views.py - Backend logic is being written in this file in multiple functions
- urls.py - This is the start point of the project and contains the routes to which the web browser has to be routed once the user enters the URL.
- models.py - The database setup for the application is done here.
- templates/-This folder contains the html templates which have to be rendered with respect to each function's functionality.
- /media/-This contains the documents that are temporarily uploaded to process before being deleted.
- utilityFunctions.py- This provides functions to preprocess text, remove URLs, emojis, punctuation, and extra white spaces, and it allows for sentiment analysis using VADER.
- Signup.py- The backend validation of the signup page is done here.
- Functional_testing.py - This script performs sentiment analysis on various types of input data, including PDF files, text files, and audio files. It then prints the results of unit tests to verify the correctness of the sentiment analysis functions.
- Amazon_test.py - This Scrapy spider is designed to scrape Amazon product reviews, including star ratings and user comments, and save them into a JSON file named 'reviews.json.' The spider extracts data from multiple pages as specified by the URL range.
- Settings.py - This code sets up the foundational configurations for a Django project, including database settings, internationalization, static and media file handling.

In this software code Documentation,we majorly concentrate on the view.py file as it contains the major part of the development work. Now let us see each function listed in view.py along with the functionalities:

- def analysis(request): This renders the homepage of the web url that is localhost:8000
- def input(request): This function renders thehome.html page. This html page receives the input of the document to be analyzed. Here, please remember that the files to be inputted have to be of the following format only.
    - .pdf - For Document Analysis
    - txt - For Document Analysis
    - .wav - For Audio Analysis.

Once you upload the document, the documents initially get stored in the media/ folder and are converted into text. Here We Have A Variable “value” That stores the text in a list of strings as an array. “Value” serves as the input for the sentiment analysis analyser.

- defproductanalysis(request):This Function Receives the input as the link of the product reviews that has to be analyzed. Here the link to be given in the url input point is specific and is as follows :
    - Go to https://www.amazon.com
    - Select the product that needs to be reviewed.
    - Click on the ratings
    - Make sure that all star options are selected.
    - Please provide the following http link.
- def textanalysis(request): This function receives the input in textual format from the user and is used for raw text analysis.The text is converted into a list of strings and stored in the variable “value”.
- def get_clean_text(text): Cleans the text by removing hyperlinks, emojis, special characters, punctuations and Extra whitespaces. This is done using regular expression repackage. This method also removes top-words from the text which do not contribute to the meaning of the text.
- Def detailed_analysis(result): This function takes a list of text items, analyzes the sentiment of each item, and returns a dictionary that provides a detailed breakdown of the sentiment proportions within the input text collection.
- Def index(request): It is responsible for rendering the main page with appropriate user-specific content based on authentication.

- Def register(request): It manages the user registration process, including form validation, saving user data, and authentication. It then redirects the user to the main page upon successful registration.
- Def pdfparser(data): This function reads a PDF file, extracts the text content, and preprocesses it to obtain a list of sentences. It can be used to convert PDF text into a format that is easier to analyze or work with.
- Def audioanalysis(request) & def ytanalysis(request) respectively analyze audio and Youtube comments to assess their sentiments.
- Def get_video_captions(video_id): Retrieves captions for a YouTube video identified by video_id using the YouTube Data API.
- Def ytcaptions(request) : to analyse Youtube Captions/transcripts to assess their sentiments.
- Def speech_to_text(filename): This function transcribes audio content into text
- Def face_analyser(filename): This function accepts an image with a face as input, analyses the emotions and classifies its percentages (sad, happy, disgusted, angry, afraid etc). It employs the deepface library to do the same.
- Def sentiment_analyzer_scores(sentence): This function analyzes the sentiment of text content.
- The **reddit_analysis** function is a Django view that facilitates sentiment analysis on Reddit posts based on a user-provided keyword. When the user submits a keyword through a form, the function retrieves titles of Reddit posts containing that keyword using PRAW, the Python Reddit API Wrapper. It then conducts sentiment analysis on the collected titles and displays the results on a webpage. This function is designed for authenticated users, and if no keyword is provided, it prompts the user to enter one. The integration of PRAW allows seamless interaction with the Reddit API, enabling the extraction of relevant data for sentiment analysis.
- The **get_face_analysis function** is a utility function that performs facial emotion analysis using the DeepFace library. Given the file path to a facial image, it reads the image, analyzes the emotions present in the face, and returns the dominant emotion. The faceAnalysis function is a Django view that utilizes the get_face_analysis function. When a user uploads a facial image through a form, the function saves the image, retrieves its file path, and calls get_face_analysis to obtain the emotion analysis results. The

function then displays the sentiment analysis results on a webpage dedicated to facial analysis. Both functions are protected by the @login_required decorator, ensuring that only authenticated users can access the facial analysis functionality. After analysis, the uploaded files are removed to maintain user privacy and clean up the storage space. This integration provides users with a convenient way to analyze emotions depicted in facial images, offering insights into the emotional content of the provided photos.

FUNCTIONAL TESTING

For functional testing of the code, a separate testing script is written to validate the output generated by the CELT. The scripts are written in:

SE_Project1/sentimental_analysis/realworld/functional_testing.py

The original code has been copied into the functional_testing.py file, and a Scrapy environment has been recreated with another file, amazon_test.py. All the HTML and CSS dependencies have been removed to have an independent execution of the functional test. Initially, there are three test cases that have been used for the validation of the software.

Test Case 1:
Here, a pre-configured string has been passed to the text analysis function, which outputs the sentimental analysis values in the dictionary format. These values are further verified with our pre-configured values.

Test Case 2:
The Amazon product feedback link that has to be analyzed is initially stored in the `productanalysis.txt` file. The path of the `productanalysis.txt` file is parsed into the `productanalysis` function, which invokes the Scrapy execution and provides us with a `reviews.json` file. From the `reviews.json` file, we collect the comments of all the feedback and pass them to the CELT. The output from the CELT is compared with the precomputed values and verified.

Test Case 3:
We initially store the file that has to be analyzed in a particular folder and copy that path and parse it into the `input` function. This then checks for the extension name of the file and processes it with the CELT, irrespective of it being a PDF, txt, or wav file.

If needed we can provide more test cases in this file without any dependencies with the main software.

FUTURE SCOPE

- Store history for each user and the corresponding analysis results.
- Recommendation system based on analysis results.
- Live speech to text sentiment analysis.
- Enhance the analysis by taking into consideration the number of users rated for each product.
- Audio diarization to split input audio in sections according to different speakers, to analyze individual speaker sentiment.