C.E.L.T
The Sentimental Analyser

Software Code Documentation

Team Members

Mangalnathan Vijayagopal (mvijaya2)
Nischal Badarinath Kashyap (nkashya)
Amogh Rameshappa Devapura (arames22)
Niranjan Pandeshwar (nrpandes)
Sharath Bangalore Ramesh Kumar (sbangal2)

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 analysing the data provided in any of the four different methods. They are Document 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 the 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

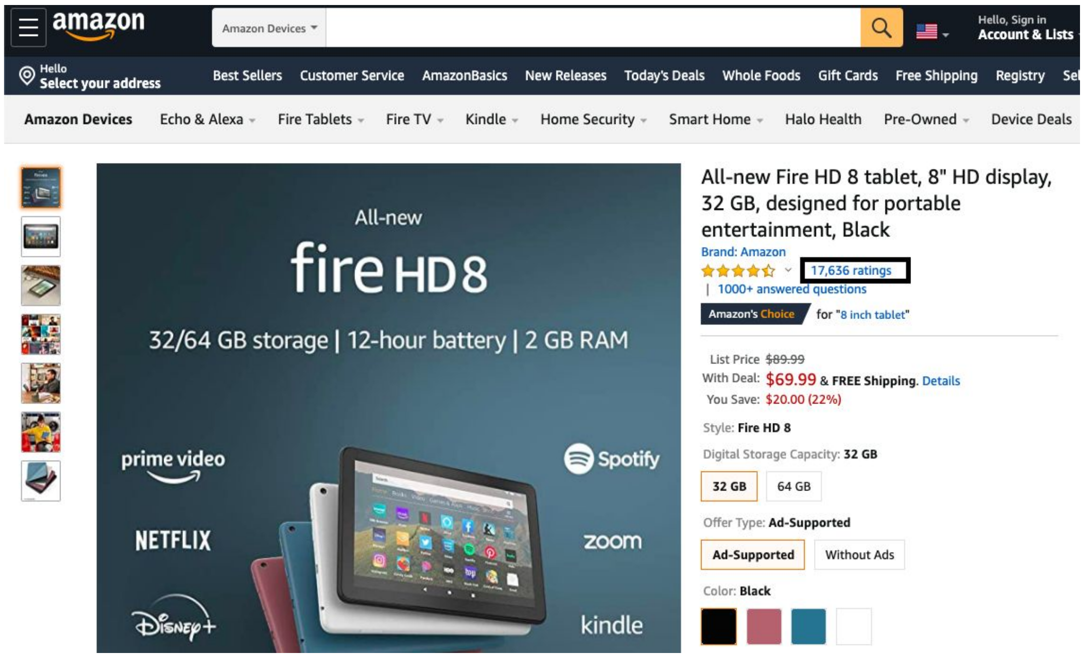
This section covers major files used in Django and small descriptions of each file with their functionalities.

- 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 - we currently do not have any database usage in the project and this file contains the section which can be used for the database setup
- 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.
- Scrapy Tool functionality:
    - Initially the scrapy project needs to be created. Once we have the project in place, a spider needs to be created. A spider is a chunk of python code which determines the content of the web page that needs to be scrapped. The generation of the scrapy project and related files are taken care of by our software.
    - Some of the important files which will be created by running scrapy.
    - Spider Folder: /Amazon_Comments_Scrapper/amazon_reviews_scraping/amazon_reviews_scraping/spiders
    - The above directory will have all the python classes spider/crawlers. The scrapy.py will have the xpath for the reviews and comments of the amazon web page. These data are saved into the json format. These data are passed for sentiment analysis.

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 the home.html page. This html page receives the input of the document to be analysed. Here please remember that the files to be inputted has 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.
- def productanalysis(request): This function receives the input as the link of the product reviews that has to be analysed. 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



All-new Fire HD 8 tablet, 8" HD display, 32 GB, designed for portable entertainment, Black

Brand: Amazon

17,636 ratings | 1000+ answered questions

Amazon's Choice for "8 inch tablet"

List Price: ~~$89.99~~

With Deal: $69.99 & FREE Shipping. Details

You Save: $20.00 (22%)

Style: Fire HD 8

Digital Storage Capacity: 32 GB

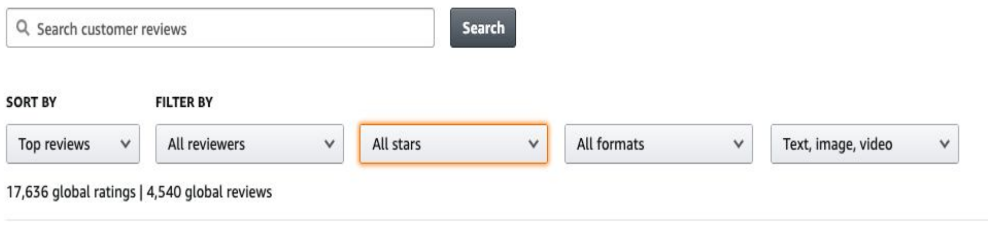
32 GB | 64 GB

Offer Type: Ad-Supported

Ad-Supported | Without Ads

Color: Black

    - Make sure that all star options is 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 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 white spaces. This is done using regular expression re package. This method also removes stop-words from the text which do not contribute to the meaning of the text.

- def detailed_analysis(result): This function takes a list of strings as input and performs cleaning and sentiment analysis using Vader to generate a sentiment dictionary. Then the scores for the strings within the list are added up and divided by the total score.

- def sentiment_scores(text): This function is responsible for leveraging SentimentIntensityAnalyzer() which is a component of vaderSentiment, the core tool used for text sentiment analysis in our application.

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_tesing.py

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

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

Test case 2: The amazon product feedback link that has to be analysed is initially stored in the productanalysis.txt file. The path of the productanalysis.txt file is parsed into the product analysis function which invokes the scrapy execution and provides us with review.json file. From review.json file we collect the comments of all the feedback and pass it 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 analysed in a particular folder and copy that path and parse it into input function. This then checks for extension name of the file and processes it to CELT irrespective of it being a PDF, txt or wav file. The output from the CELT is again verified with the pre-computed values.

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

FUTURE SCOPE

- Implement user authentication to store history for each user.
- 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!
- Extend the analysis to the Facebook, Twitter and LinkedIn Posts