Project Documentation - CS410 - NVK

Nikhil Khandekar (nikhilk5) (captain)
Vineet Chinthakindi (vineetc2)
Kyr Nastahunin (knasta2)

An overview of the function of the code:

There are several functions of the code. The intended use case of this code is to allow the user to enter a name of a business that's local to them (restaurant, gym, museum, etc.) and select it from the autocomplete list and get the most important keywords within the Google Maps reviews with the sentiment analyzed.

Implementation Documentation:

The code is implemented so that when the user selects their business of interest, the API makes a request to Google Maps Review APIs to retrieve the reviews to be analyzed. The UI app will then make a request to the flask API developed by our team. The API will fetch the top 5 reviews for the user-selected business and perform the RAKE keyword extraction algorithm on all of them. It will then return the list of keywords with their importance values to the UI application, which will display the result in a readable manner.

The RAKE algorithm is modeled after the Rapid automatic keyword Extraction as outlined in this paper: RAKE Research Paper Link. The code for this algorithm can be found in **rake_keyword_extract.py**. When given any text, this algorithm uses the idea that that keywords frequently contain multiple words but rarely contain standard punctuation or stop words. In the code within that file, we are delimiting the input text by stop words and punctuation and combining adjacent words into phrases. Then we are creating an 2-D adjacency graph and mapping how often and with what other words does a specific word in the text share the same phrase with. The score of degree(word)/count(word) is summed across all words in a phrase, and the phrases with the highest scores highlight the most important parts of the text.

The **google_helper.py** file contains the code to access the Google Reviews API for a specific business. You may notice that we have been using YELP reviews and Google API reviews interchangeably. This is because although our project was meant to be for Yelp Reviews, we could only gain access to the google review API.

App.py queries the reviews from a user inputted location and runs the keyword extraction and the sentiment analysis on the mined extracted keywords so that the user can see the most popular positive and negative attributes of the reviews for that place. Our demo shows an image of what is returned for the Green St. Mcdonalds

The JS-UI folder contains all the javascript and front end logic for the application.

The python/notebooks folder contains a lot of text notebooks that contain implementations of various things. For example, we have a notebook with the RAKE implementation and some review test data for our application.

Our project also includes using the VADER sentiment analysis tool. This is a module which takes in text and then outputs the sentiment of it. There is no pre-processing of the text that is required. The model works by evaluating the sentiment of individual words and then adjusts them if there are negations, capitalizations of all letters in a word, idioms, and uses of words such as 'but.' The model gives an overall score labelled as "compound." Upon trying multiple cutoff scores, we decided that if "compund" score is 0.10, the text will be labelled as "positive," if the score is less than -0.10 it will be labelled as 'negative,' and otherwise the text will be labelled as "neutral."

For the purposes of our project, upon extracting the keywords from the RAKE implementation, we would then pass each of them into the VADER sentiment analysis tool and then give the corresponding label based on the compound score.

Once all of the keywords are extracted with their corresponding sentiments, we send this as a json which will then get displayed to the user.

How to Run:

1. Clone the project repository
2. Install Conda v4.10.3 or better. Create conda environment: `conda create --name test python=3.9`
3. Activate conda environment: `conda activate test`
4. Open the `Python` folder and run the `pip install -r requirements.txt` command in the console to install the dependencies.
5. Start the app as a flask server from the `app.py` file. Here's an example run command: python -m flask run
6. Open the `js-ui` folder and run the `npm install` command to install dependencies.
7. After the dependencies are installed, run the `npm start` command to open the UI app.

8. After a while, the window with \`localhost:3000\` will open in the browser. You can start using the app.
9. Alternatively, the features of the app can be tested through the notebook. We have included an extra field at the end of the keyword extraction algorithm notebook: **CS_410_RAKE_Keyword_Extraction_Implementation.ipynb**, which allows the user to change the input value to the algorithm and see the results produced

Team Contribution:

a. Kyr Nastahunin (knasta2) - Responsible for finding and fetching the data, delivering the inputs to the algorithms, exposing the app as an API, and creating the UI.
b. Vineet Chinthakindi (vineetc2) - Responsible for the keyword extraction algorithm, cleaning up the data, and evaluating the importance of the extracted keywords.
c. Nikhil Khandekar (nikhilk5) - Responsible for the sentiment analysis model, evaluation of its result, cleanup of input and output data, and exposing the app as an API.