CS 410 Project Report, Fall 2022

Team: Tetra

Members: Dillon Harding (dth3), Kowshika Sarker (ksarker2), Nikhil Garg (nikhilg4)

Captain: Vaibhav Karanam (karanam5)

Overview

Holistic overview

Our project aims at recommending food items based on an individual's health or disease condition. Our system analyzes free text diet recommendations of different health conditions to extract which food items are recommended to intake or avoid for individuals with a particular disease. Currently we consider one disease at a time. Given a disease, our system outputs three food lists - recommended food list, detrimental food list, neutral list. These three lists provide ranked lists for which foods should be consumed for improvement of health condition, which foods should be avoided and which food items cannot be commented on based on the system's available knowledge.

We are presenting our system as a web application with the following components.

- Web scraper
- Sentiment analysis
- Website

Web scraper

Our web scraper module collects free-text dietary recommendations of different diseases. We used Wikipedia pages of different diseases for collecting diet recommendations.

Sentiment analysis

For a food item, we perform sentiment analysis on the textual diet recommendation data line by line to detect if the food is positively or negatively mentioned in that sentence in context of the disease. We detect three categories of sentiment - positive, negative, neutral.

Website

We present our system output on a website.

Implementation

We implemented our system in the Python programming language.

Web scraper

The team have built a simple but effective web scraper to gather data about the diet recommendations for different diseases. The web scraper python code uses libraries requests and beautiful soup to first get the html text data and then using beautiful soup to parse the html to find relevant information needed for the diet.

In this task we ran into some issues for which we had to use workarounds to get the result we wanted. The first challenge we ran into was that there was no standardization for the diet section on the website. We wanted to get the disease information from a reputable source such as Mayo Clinic or Webmd or CDC. But on each of those websites the information on diet was scattered throughout the webpage of the disease. This made it harder for us to capture the information. We could have come up with a strategy that could look for diet in every paragraph of the webpage but we concluded this will be an ineffective strategy as the number of food items to look at is high and going through each line will be computationally expensive as well. The best source for this project we found was Wikipedia. While the information on Wikipedia may not be certified by any expert like from sources such as Mayo Clinic, it provided the most easy format to web scrape text. We did not have to go any fancy way such as creating a headless browser and then web crawl. Our simple web crawler worked perfectly. Ideally we would have tried to challenge ourselves and go the more difficult route but with the time constraint and work to be done on other parts of the project, we decided to go the simple implementation route.

The second challenge we faced was that while we could navigate to the initial or seed webpage for a disease, the crawler was unable from there to find other distinct diseases. We found out that diseases do not have the best hierarchical structure of representation. While bigger diseases such as cancer and diabetes have many sub-divisions, their diet offered mostly the same text. This did not have much value for the project. Further it was also hard for the crawler to reach distinct diseases like for example going from cancer to diabetes. To solve this issue, we

decided that it would be best to fix the diseases we want to look into and make the crawler go through a static list of links. This made it feasible for us to validate the data and to make sure that the data collected provided value for the project.

Sentiment analysis

Pretrained Model: We perform *entailment analysis* for analyzing sentiments of food items in context of a disease diet recommendation. We utilize the pretrained RoBERTa model, which expects two textual inputs - *premise* and *hypothesis*. RoBERTa outputs whether the *hypothesis* is true given the *premise*. For a *premise-hypothesis* pair, RoBERTa yields probabilities for three sentiments - neutral (*hypothesis* is neither true nor false), entailment (*hypothesis* is true), contradiction (*hypothesis* is false). Some examples of the RoBERTa model are mentioned here.

Customization: Diet recommendations from Wikipedia are usually multi-line paragraphs where each nutrient or food item is usually mentioned in only one or two sentences. We checked that if individual sentences are used as *premise*, RoBERTa performs significantly well in extracting sentiments, but when the entire paragraph is used as a premise, the verdicts are usually neutral, even if the premise contains information regarding that ingredient. We suspect the reason is, for each ingredient most sentences in the paragraph mentions nothing and so the information content about that nutrient contained only in one or two sentences gets diluted. For this reason, we have decided to use each individual sentence as premise for each ingredient and aggregate the obtained sentiment probabilities through maximization to get the final output.

Diet Sentiment Analysis: We use lines from free-text diet recommendations as *premise* input of the RoBERTa model. We combine dummy positive and negative phrases with food items to form positive/negative sentences. For example, some positive sentences for the food item ‘sugar’ are ‘sugar is beneficial’, ‘sugar is recommended’, ‘high sugar is recommended’ and some negative sentences are ‘sugar is harmful’, ‘sugar is risky’, ‘high sugar is harmful’ etc. For sake of robustness, we use multiple positive phrases and multiple negative phrases. We maximize overall positive phrases and negative phrases separately to get the final result for a single line of diet recommendation. If we get a positive/negative association of a food item for any line in the

entire diet recommendation paragraph, we return the final verdict as positive/negative, otherwise the output of that food is neutral for that disease. In our code, prob_for_single_food and sort_food functions perform the sentiment analysis task.

Website

We are creating a website to host our results. For simplicity, we first computed the results for a number of diseases and food items and exported that into a JSON file. Then we read the JSON file and pull the results to display on the webpage.

The website will feature a simple design which will feature a drop down menu where users can select the disease they want diet information on. To display the result we have decided to make a view of three lists - positive, negative, and neutral. The food items in the positive list were the items that we found to have positive entailment with the text collected from wikipedia. Similarly, negative list items showed negative entailment with the text. Neutral list items did not produce a label of either positive or negative. This was mostly because these food items were not part of the diet and did not show up in text. Thus, the entailment model came up with the label of neutral.

The website will be created using HTML, CSS, and Javascript. For Javascript we will be using the JQuery library. This library is easy to use and provides a simple way to display results on the webpage. The results for each disease itself are stored in a JSON file from which JQuery will read the data. At first we thought to integrate our python library with the webpage and run the script every time a user selects the disease. But this turned out to be a difficult task in itself. Primarily, the python script takes a long time to run. When we were running the script on Google Colab with purchased credit hours and premium GPU runtime, it took almost one hour. Therefore, integrating a python script with a website will make the user wait and ruin the user experience. Also, integrating the python script with the machine learning model means we will have to use a strong environment to host such a script such as a paid version of AWS Lambda. This meant additional hours to configure the lambda function and perfectly integrate it to the website's javascript. But this still will not fix the delay it takes for the script to run. Since the

script was created to produce results for all the diseases and the results do not change when run at different time periods, we felt it was a safe option to do a database model for our website where the JSON acts as our database.

Usage

Our website is hosted at https://vkara5.github.io/. The website has a dropdown menu of disease conditions for the user to choose one from. The website displays the food item and food group/subgroup names that we are using for ranking with respect to the user selected disease condition. Once the user selects a disease condition, the website displays three resultant food lists from our system - a list of beneficial foods, a list of detrimental foods, a list of neutral foods.

Link to the presentation video:
https://drive.google.com/file/d/1WUj6jc0U358KhNme9CoHL4s54f317I6v/view?usp=sharing

Contribution

Nikhil: Wrote the code for categorizing the results of the sentiment analysis. Discussed and decided what criteria should be used to determine and interpret the results and how to ensure accuracy of the sentiment analysis. Worked to create the slideshow for the project presentation.

Dillon: Creation and implementation of a Python web scraper, which would retrieve parsed HTML data from Wikipedia.org. Development of interactive Web application to display data for diets associated with select diseases.

Kowshika: Performed the sentiment analysis of the web crawled diet recommendation paragraphs and food items with pre-trained RoBERTa model.

Vaibhav: Contributed on JSON parser for web scraper and auxiliary functions to run the two functions for sentiment analysis and maximizing score.
Helped with the scoring function to come up the threshold idea and utilized it with the overall sentiment analysis.
Created function to export the result data to JSON to be used by webpage.
Maintained the Github repository and administrative tasks as team leader.