Ann Arbor Restaurant Recommender

Description

Ann Arbor restaurant recommender is a query application which prompts the user for a query and searches Ann Arbor restaurant menu data for a best match. The application then returns a restaurant and a menu item suggestion

- What was your motivation? - When you have a craving for a certain food or beverage it can take time to find the place to go to best fill the craving. When searching on Google, the search items returned are often not what the user is looking for, as often review information is used in the query. Even when these reviews say, 'I wish this restaurant had more **vegetarian** items,' when you search for 'vegetarian restaurants,' it return the restaurant as a match. In order to find food a user is looking for, the relevant menus should be searched. That is the aim of our project.
- What did you learn? - Many tools, algorithms, and processes were implemented that were new to us were used to implement this project including sentence transformers, similarity metrics, pytorch tensors and evaluation data annotation.

Table of Contents

Installation

There is a requirements.txt file which includes all file dependencies. These dependencies can be installed using: `$ pip install -r requirements.txt`

Usage

To utilize the query application, skip to Runtime folder and run runtime_query.py. All previous file dependencies are included in the folder

Folders and Files

Retrieve_Data folder

1. get_urls.py

```
Run: In the Retrieve_Data folder $ python3 get_urls.py
Input: None
Output: restaurant_menu_urls.csv
```

Restaurant Name	URL
Aventura	aventura-ann-arbor?osq=Restaurants

```
Functionality: This file utilizes beautifulSoup and the python request library to scrape the Ann Arbor Restaurant search page on Yelp to retrieve all Ann Arbor restaurants menu urls that are available on Yelp. These urls are then stored in restaurant_menu_urls.csv
```

2. menu_scraper.py

```
Run: In the Retrieve_Data folder $ python3 menu_scraper.py
Input: restaurant_menu_urls.csv
Output: menu_data.csv
```

Restaurant Name	Menu Item	Item Description
AmA Bistro	Perfect Burger	Cheese, lettuce, tomatoes, pickles, and onions.
Functionality: This python file utilizes beautifulSoup and the python request library to crawl the links in restaurant_menu_urls.csv and retrieve menu items and menu item descriptions at those urls. This data is then stored in menu_data.csv		

Compute_Embeddings folder

1. embedding_similarity.py

```
Functionality : This file contains the functions encode(), mean_pooling(), max_pooling(), normalize(), cosine_similar(), and euclidean_distance(). The sentence BERT is also loaded into this file, the sentence BERT used can be changed by loading in a different model. The functions in this file are used in all of the following files.
```

2. embeddings_to_csv.py

```
Run: In the Compute_Embeddings folder $ python3 embedding_similarity.py
Input: menu_data.csv
Imported files: embedding.py
Output: dish_embeddings.csv and restaurant_embeddings.csv
```

Restaurant embeddings

Restaurant Name	Food Item	BERT embedding
AmA Bistro	Perfect Burger Burger with lettuce,	[0.041763704270124435, -0.05297095328569412,..

	tomato, and onion	
Functionality : This file takes in the restaurant menu data in menu_data.csv		
and stores embeddings in 2 parts		
1. The file concatenates all menu information for each restaurant into one line, pass this information into the encode function in embedding.py, and stores the embedding into the csv file restaurant_embeddings.csv, shown in the table above		
2. The file concatenates food item and item description for each line and similarly gets an embedding for each line. This information is output in the dish_embeddings.csv file.		

Runtime folder

1. runtime_query.py

```
Run: In the Runtime folder $ python3 embedding_similarity.py
Input: dish_embeddings.csv and restaurant_embeddings.csv
Imported files: embedding.py
Output: Restaurant and Menu item suggestion on the command line
Functionality: This file loads the files with the stored embeddings, prompts the user for a query on the command line. Retrieves an embedding for the query, and compute cosine similarity with restaurant_embeddings and dish_embeddings in order to retrieve a best match. The application then prints to the command line a restaurant and food item suggestion.
```

Evaluate folder

1. evaluate_queries.py

```
Run: In the Evaluate folder $ python3 evaluate_queries.py
Input: test_queries.csv, dish_embeddings.csv, restaurant_embeddings.csv
Output: query_results.csv
Functionality: This file gets embeddings for each query, computes cosine similarities with menu items in the dish_embeddings.csv and writes the top 5 dish matches to a csv file. These outputs are then being evaluated by hand to rate the functionality of the application. These will be tested with different metrics. When changing metrics in this fill. The menu dataset embeddings will be recomputed with the corresponding metrics. This will involve changing the sentence BERT model, the similarity metric and pooling method. The metrics for the corresponding output column in query_results are tracked in the table below.
```

Test Number	model	pooling	similarity metric

1	msmarco-distilbert-cos-v5	mean	cosine similarity

Credits

Maira Gajda, Saanika Kulkarni, Hilary Lam, Kate Shenton, Sylvia Wei

Resources Used
