Fetch Rewards App Overview

Dependencies

- Developed using Python 3

How to Run This App

You have two choices when running this application.

Option 1: Use Docker to run it (Example on Windows using Docker Desktop)

1. Open Docker Desktop
2. Clone the application from DockerHub
3. Find the image *fetchrewards*
    - Click RUN on the image
    - Specify a port under *Local Host* to run the container on (for example, `5000`)
    - The container should now be running on `127.0.0.1:5000`
4. Interact with the endpoints using your favorite tool(s)

Option 2: Fire it up using a command line and Python (Example on Windows)

1. Clone the application from GitHub
2. Create and activate a virtual environment in the app directory
    - `cd .\FetchRewards`
    - `python -m venv env`
    - `env\Scripts\activate.bat`
    - `pip install -r requirements.txt`
3. Activate the application
    - `python main.py`
    - By default, Flask will start on port `5000`; you can verify this on the terminal output
    - The application should now be running on `127.0.0.1:5000`
4. Interact with the endpoints using your favorite tool(s)

Endpoints

Base Endpoint

`/`

This endpoint returns a simple HTML landing page related to the application.

Info Endpoint

`/info`

This endpoint returns my decisions to the points made in the assignment. It scrapes data from the HTML page on your S3 bucket and provides my answers as key:value pairs.

This endpoint accepts GET requests.

Example response JSON

```
{
    "Question 1: Do you count punctuation or only words?": "Answer 1: My
application does not count punctuation",
    "Question 2: Which words should matter in the similarity comparison?": "Answer
2: My application removes common stop words"
}
```

Text Similarity Endpoint

`/textsimilarity`

This endpoint accepts JSON from a POST request. It will accept more than two text samples and returns a similarity comparison between all samples. The similarity metric ranges between 0 and 1; a score of 0 means no words match while a score of 1 means they are a perfect match.

For example, if you provide three samples (s1,s2,s3), the endpoint returns comparisons between s1 and s2, s1 and s3, and s2 and s3.

Example Request

```
{
   "Sample 1": "The easiest way to earn points with Fetch Rewards is to just shop
for the products you already love. If you have any participating brands on your
receipt, you'll get points based on the cost of the products. You don't need to
clip any coupons or scan individual barcodes. Just scan each grocery receipt after
you shop and we'll find the savings for you.",
   "Sample 2": "The easiest way to earn points with Fetch Rewards is to just shop
for the items you already buy. If you have any eligible brands on your receipt,
you will get points based on the total cost of the products. You do not need to
cut out any coupons or scan individual UPCs. Just scan your receipt after you
check out and we will find the savings for you.",
   "Sample 3": "We are always looking for opportunities for you to earn more
points, which is why we also give you a selection of Special Offers. These Special
Offers are opportunities to earn bonus points on top of the regular points you
earn every time you purchase a participating brand. No need to pre-select these
offers, we'll give you the points whether or not you knew about the offer. We just
think it is easier that way."
}
```

Example Response

```
{
    "Sample 1 vs Sample 2": "Similarity between two provided samples is:
```

```
0.7692307692307693",
    "Sample 1 vs Sample 3": "Similarity between two provided samples is:
0.38028169014084506",
    "Sample 2 vs Sample 3": "Similarity between two provided samples is:
0.3088235294117647"
}
```

Application Decisions

- Question 1: Do you count punctuation or only words?
    - My application does not count punctuation
- Question 2: Which words should matter in the similarity comparison?
    - My application removes common stop words
- Question 3: Do you care about the ordering of words?
    - My application does not care about the ordering of words
- Question 4: What metric do you use to assign a numerical value to the similarity
    - Number of unique matched words divided by total unique words in both samples
- Question 5: What type of data structures should be used? (Hint: Dictionaries and lists are particularly helpful data structures that can be leveraged to calculate the similarity of two pieces of text.)
    - Dictionaries, lists, sets, and tuples are employed. Sets are used for comparing unique words between two samples and are helpful because they are more performant for lookups than lists.