Title: Cooking Recipe Search

Author: Tianjun Cai, Guangyao Dou, Alicia Sheng

Date: 2022/05/09

Description: The program is a personalized cooking recipe search engine. The users can search for cooking recipes related to specific titles, cuisine types, or health information.

Dependencies:

- First, this system requires a Conda version of 4.11.0, with a python version of 3.8.12. You can download the Conda from here: https://docs.conda.io/projects/conda/en/latest/user-guide/install/macos.html
- Create a conda environment by running `conda create <environment_name>`, and activate it by running `conda activate <environment_name>`.
- Also, this system requires the installation of Flask, and the version is 2.0.2. We can install it by running "pip install flask" in our terminal.
- This program also uses Elasticsearch 7.10.2, and you can download it from here: https://www.elastic.co/downloads/past-releases#elasticsearch
- You should also run `conda install pandas` to install the pandas package. This program requires a panda version of at least 1.4.2.
- Lastly, you should run `conda install -c anaconda scikit-learn`, and this program requires the sklearn version to be of at least 1.0.2.
- Run `conda install pip` to install pip into your vene directory.
- Run `pip install -r requirements.txt` to install all of the required packages.
- Download all of the data in this google drive:https://drive.google.com/drive/folders/1Nz0fuX5p4t6hhrDNrsN53ySlQKquk1Cs?usp=sharing and put these files into a folder called "data."

Build Instructions:

1. Activate the conda environment
    a. $ conda activate <environment_name>
2. Start the ES engine:
    a. $ cd elasticsearch-7.10.2/
    b. $ ./bin/elasticsearch
3. Test your ES is running correctly:
    a. open http://localhost:9200/ on your browser
4. Activate the embedding server:
    a. $ python -m embedding_service.server --embedding sbert --model msmarco-distilbert-base-v3
5. Run sbert embedding on both the training set and the recipe dataset:

    a. uncomment line 79 and 80 of embedding.py
        i. load_train("data/whats-cooking/train.json")
        ii. load_doc("data/recipes_with_nutritional_info.json")
    b. run embedding.py
        i. $ python embedding.py
6. (optional) Run machine learning algorithms to see which one has the highest accuracy:
    a. run cuisine_prediction.py with the output of step 4
    b. check the accuracy in output/classifiers_accuracy_runtime.csv
7. Run SVM (highest accuracy model) on the embedded dataset to get the labels:
    a. open jupyter lab
        i. $ jupyter lab
    b. run cuisine_labels.ipynb from the beginning to the end
8. Load recipe and label datasets to Elasticsearch:
    a. $ python load_es_index.py --index_name cooking_recipe --doc_path data/recipes_with_nutritional_info.json --label_path output/SVM_predicted_label.csv
9. Start the program:
    a. $ python app.py
    b. open http://127.0.0.1:5000/ on your browser

Run Instructions:

1. Follow step 2, 3, and 9 in Build Instructions section

User Interface:

Home page (General Search):
This home page mainly consists of two segments. At the top, a “Recipe Search” hyperlink could bring users back to the home page. “Switch to HEALTHY search” is another hyperlink that allows users to head to “Healthy Recipe Search,” introduced later in this section.

In the middle of the home page, there is one search box and three selections, followed by a “Search” button. Users could type in their needs for recipes into this query box for searching. The first selection box asks for different ways to search. For example, “Relevance” is the default search by ElasticSearch. “Complexity” measures how many instructions are needed for each recipe. “Healthiness” evaluates the general health status of recipes, with fat, salt, saturates, and sugars taken into consideration. “Ingredients,” as its name suggests, support users to search by ingredients. Continuing in the selection, users could select whether to search through all recipes or narrow results to 20 different cuisines. Last but not least, the user could choose in which order the results would be displayed. Please note that this only affects when users select “Complexity” or “Healthiness” in the previous selection box.

Result page (General Search):
A max number of 8 snippets of content would be displayed on this result page. Within each snippet, one could see the title, cuisine, ingredients, health status, and complexity of this recipe. Clicking the title would lead one to a detailed document page. To elaborate on health status, the

color green represents healthy (i.e., low amount contained in the dish); the color yellow represents normal; the color red represents not healthy. Users could get a broad view of their health status through this simple label. As for complexity, we evaluate the number of instructions in each recipe. We set 10 to be full complexity; all recipes requiring more than ten instructions are also counted as full complexity. Other recipes with less than ten instructions would show their complexity accordingly based on the ratio of the number of instructions to 10.

At the end of the results page, one would see “Previous,” “Next,” and several buttons for pagination. Users could click one button and go directly to desired pages. On the first page, the “Previous” button is disabled. Similarly, the “Next” button is disabled on the last page.

Document page (General Search):
Clicking on the document page, users could read more detailed information about this recipe. The page starts with a title and has four sections: ingredients, instructions, health level per nutrition, and nutrition values per 100 grams. All ingredients are displayed in flex boxes, with descriptions and amounts specified. Following that are instructions for users to follow. Health level per nutrition has the same information displayed on the results page, yet it offers users a broader sense of their health status. Lastly, users could find more detailed health-related data under the Nutrition values per 100-gram section.

Health Search page (Health Search):
After clicking “switch to HEALTHY search,” users are led to this “Healthy Recipe Search.” Though the UI is rather similar, some new functionalities are worthy of introduction. Firstly, users could still type in the query box for keywords. However, the selection following now enables users to search for detailed nutrition categories they are most concerned about, including energy, fat, protein, salt, saturates, and sugars. The remaining two selections are the same as that on the home page.

One main new functionality added to healthy search is the “Advanced & Customized Search” section. After clicking, users can now type in exact numeric values as requirements for their recipes. For example, one could type 100 and 300 for “Energy” so that recipes in the results would all meet this requirement.

Health Results page (Health Search):
The entire demonstration of health search results is similar to that of general search despite few differences. The first is that if users select certain nutrition to be sorted upon, the results page would show that nutrition value per 100 gram within each snippet. This helps users to better choose recipes. The second is that if users type in numeric values and search for results, the entire table of nutrition values per 100 gram would be included in each snippet of content.

Health Document page (Health Search):
This document is similar to the general document page.

Testing:

For sample data:

1. $ conda create <environment_name>
2. $ conda activate <environment_name>
3. $ pip install -r requirements.txt
4. $ cd elasticsearch-7.10.2/
5. $ ./bin/elasticsearch
6. $ python -m embedding_service.server --embedding sbert --model msmarco-distilbert-base-v3
7. $ python load_es_index.py --index_name cooking_recipe --doc_path sample_data/recipes_sample_data.json --label_path sample_data/SVM_predicted_label.csv
8. $ python app.py
9. go to http://127.0.0.1:5000/

We created a small corpus of 20 data samples from different cuisine types (based on our common sense knowledge) in the folder “sample_data.” Then, we used Support Vector Machine (SVM) to create a CSV file that contains their cuisine predictions. We ran the command `python load_es_index.py --index_name cooking_recipe --doc_path sample_data/recipes_sample_data.json --label_path sample_data/SVM_predicted_label.csv` to load the sample data set.

Once we opened the app with a new index, we searched for some key terms such as “chicken,” and it displayed search results such as “Orange Chicken” and “Shanghai Chicken and Noodles” as Chinese Cuisine and ”Chicken Spaghetti” as Italian Cuisine. When we searched Sushi among all regions, the result page displayed “Hand-Rolled Sushi” as Japanese Cuisine. If we searched “Sushi” again in the other areas, such as British or Brazil, nothing was displayed. We also tested the search by ingredients. For example, if we search “spice” for ingredients, then results in new recipes such as “Taco sauce” and “Toasty Cheese Ravioli” that use spice as one of their key ingredients.

We also tested the “Healthy” search, in which we allowed users to search based on the specific amount of energy, fats, protein, salt, saturates, and sugars. The result page displayed results based on the user's needs, and all of the results matched the inputs of our sample corpus.

Lastly, we also tested our customized healthiness function and tested the sorting based on the level of complexity (number of steps to complete a recipe). The results page matched our sample corpus perfectly.

Examples:

Recipe Search

- Search “Chicken” with Relevance search, and specify all regions
- Search “Chicken” with Relevance search, and select Chinese cuisine

- Search “Chicken” with Relevance search, and select Italian cuisine
- Search “Sushi” with Relevance search, and select all regions
- Search “Sushi” with Relevance search, and specify Japanese cuisine
- Search “Spice” with ingredient search, and specify all regions
- Perform any of the above searches, and you can select sort by healthiness or complexity as you wish.

Healthy search:

- Search “Chicken/Sushi/Spaghetti/Taco” and specify the type of nutritional information you want it to be sorted on.
- Search “Chicken/Sushi/Spaghetti/Taco,” click advanced search, and specify a range of values you want on the result page.

Code submitted by: Alicia Sheng

Team Member Contributions:

Gordon Dou:

I was primarily responsible for applying machine learning techniques to our dataset. I was responsible for writing codes (cuisine_prediction.py) that run different machine learning classifiers such as SVM, LDA, and Random Forest from scikit-learn to perform stratified k-fold cross-validation deals with imbalanced datasets. Our group identified that the SVM would be the best classifier for accuracy and code runtime. Then I wrote cuisine_labels.ipynb to train and predict our dataset using the SVM and save the result as a CSV file. I also created a sample corpus for our team to test the performance of our web app.

I also helped my team members to define the elasticsearch schema definition, build the index of our dataset, and process the original JSON files into data that is more convenient for us to preprocess. Also, I helped to write and test search methods based on ingredients and make sure that this search method functions well on our web app.

Alicia Sheng:

- Data processing
    - utils.py
        - load the original JSON file and the cuisine labels and process the data to have the correct structure and information we want
- Embedding
    - embedding.py
        - implement sbert embedding on the training set and recipe dataset to turn the ingredients into n * 768 CSV files
- Machine Learning
    - help run machine learning algorithms, including Gradient Boosting Regressor, Nearest Neighbors, Decision Tree, and MLP
- Elasticsearch:

- index.py
  - load the data from the preprocessed data to elasticsearch data structures
- load_es_index.py
  - call the functions in index.py with the correct data paths to load data into ES
- app.py
  - search methods
    - implement filter, query, and sort methods with different requirements such as cuisine type and healthiness
  - routing methods
    - help build front-end and back-end interaction between HTML forms and different search methods

Tianjun Cai:

I was mainly responsible for building user interfaces and implementing search by ingredients as well as search with detailed numeric values in health search. In this sense, I finished all HTML files in the templates folder with multiple tryouts on format display. Apart from that, I was responsible for building front-end and back-end interaction codes in app.py. Lastly, I have also implemented health_nutr_num_search, which supports numeric value search for nutritions. Finally, I helped with testing the search functionalities of this website.