Python Programming

Recipe for Users Development

Proposal

Date : 2023.11.26
Name : Park Min E
ID : mine-park

1. Introduction

1) Background

As the number of young people living alone, especially college students, has increased, many of them often struggle to properly prepare their meals due to busy schedules or lack of time and ingredients. Furthermore, with the advancement of food delivery apps, there is a growing preference for convenient delivery meals over cooking at home. Consuming only stimulating and limited food options can easily expose young people to nutritional imbalances and various diseases such as adult illnesses. To address this problem, a program is needed that recommends recipes that can be easily and conveniently prepared using ingredients at home, allowing young people living alone to have healthy and balanced meals.

2) Project goal

The goal is to create a program that analyzes the ingredients customers have on hand and recommends easy recipes that can be prepared with limited ingredients, allowing them to make healthy meals.

3) Differences from existing programs

Our approach is different from the traditional recipe recommendation methods seen on the internet or various TV programs, as they provide absolute recipes without considering the customer's ingredient availability. Many customers find it challenging to follow these recipes because they lack the necessary ingredients. We, on the other hand, aim to accurately assess the customer's ingredient inventory and recommend recipes that align with those ingredients. This way, we prevent customers from being unable to cook due to ingredient unavailability, setting us apart from existing recipe recommendation methods.

2. Functional Requirement

1) Function 1 Analyzing the customer's ingredient inventory.

- Functionality to determine the ingredients customers have on hand.

2) Function 2 Matching recipes that can be prepared

with the customer's available ingredients.

- A feature that matches the customer's gathered ingredient inventory with the ingredients in recipes and outputs only the recipes that have matching ingredients.

3) Function 3 In case there are no matching recipes.

- If there are no recipes that exactly match the ingredients entered by the user, the code will now calculate the similarity between the user's ingredients and the ingredients of existing recipes. It will rank these recipes based on their similarity and suggest them to the user. Additionally, it will provide alternative solutions.

3. Progress

1) functionality implementation

(1) Function 1

- Input / Output

user_input : The variable that receives the list of ingredients in the user's refrigerator.

user_ingredients : The variable that splits and stores user_input based on spaces is user_ingredients.

- Explanation

Functionality to determine the ingredients customers have on hand.

- The applied learned concepts include

Input / Output, Word Splitting(split, strip)

- Code Screenshot

```
# 사용자 입력을 받아 재료 리스트로 변환
user_input = input("냉장고에 있는 재료 목록을 입력하세요 (띄어쓰기로 구분): ")
user_ingredients = [ingredient.strip() for ingredient in user_input.split(' ')]
```

(2) Function 2

- Input / Output

user_ingredients : Receiving information about the ingredients the user has.

matched_recipes : Comparing user_ingredients with recipe.txt and outputting matching recipes.

- Explanation

A feature that matches the customer's gathered ingredient inventory with the ingredients in recipes and outputs only the recipes that have matching ingredients.

- The applied learned concepts include

Looping, Conditional Statements, List Append(append)

- Code Screenshot

```
if matched_recipes:
    print("\n냉장고 재료로 만들 수 있는 레시피 목록")
    print("----------------------------------------")
    for i, recipe in enumerate(matched_recipes, 1):
        print(f"{i}. {recipe['recipe_name']}")
        print("< 재료 > ", ', '.join(recipe['ingredients']))
        for step in recipe['directions']:
            print(step)
        print()
else:
    print("냉장고 재료로 만들 수 있는 레시피를 찾을 수 없습니다.")
```

(3) Function 3

- Input / Output

user_ingredient / recipes : Receiving information about both the user's ingredient and existing recipe information.

similar_recipes : When there are no matched_recipes, outputting the calculated similarity values between the user's ingredients and the recipes.

- Explanation

If there are no recipes that exactly match the ingredients entered by the user, the code will now calculate the similarity between the user's ingredients and the ingredients of existing recipes. It will rank these recipes based on their similarity and suggest them to the user. Additionally, it will provide alternative solutions.

- The applied learned concepts include

Looping, Conditional Statements, List Append(append)

- Code Screenshot

```
# 유사한 레시피 탐색
similar_recipes = []
for recipe in recipes:
    ingredients_set = set(recipe['ingredients'])
    similarity = len(set(user_ingredients) & ingredients_set) / len(set(user_ingredients) | ingredients_set)
    similar_recipes.append((recipe, similarity))

# 유사도 리스트 정렬
similar_recipes.sort(key=lambda x: x[1], reverse=True)

# 상위 3가지 레시피 출력
print("\n유사한 레시피를 고려해보세요:")
for i, (recipe, similarity) in enumerate(similar_recipes[:3], 1):
    print(f"{i}. {recipe['recipe_name']} (유사도: {similarity:.2%})")
    print("< 재료 > ", ', '.join(recipe['ingredients']))
    print()

# 사용자에게 다른 방안 제시
print("\n다른 방안을 고려해보세요:")
print("1. 다른 재료로 검색해보세요.")
print("2. 심플한 요리를 시도해보세요. (예: 계란후라이, 김치볶음밥 등)")
```

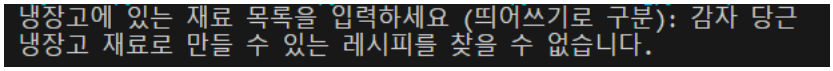
2) Result of test

(1) Function 1

- Explanation

This code segment receives a list of ingredients from the user for items in their refrigerator. It then splits the input based on spaces, creating a list. The final list, named user_ingredients, is obtained by removing leading and trailing spaces from each ingredient in the previously created list comprehension.

- Test result screenshot

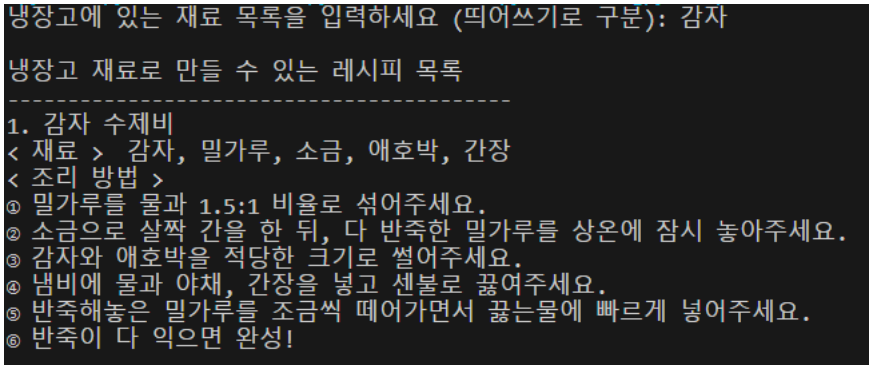


(2) Function 2

- Explanation

Initialize the matched_recipes list. Check each recipe for a match with the user's ingredients. Add matching recipes to the matched_recipes list. If matching recipes exist, print their details. Display the recipe number, name, ingredients used, and cooking steps. If no matching recipes are found, print a message: "Unable to find recipes with the provided ingredients."

- Test result screenshot

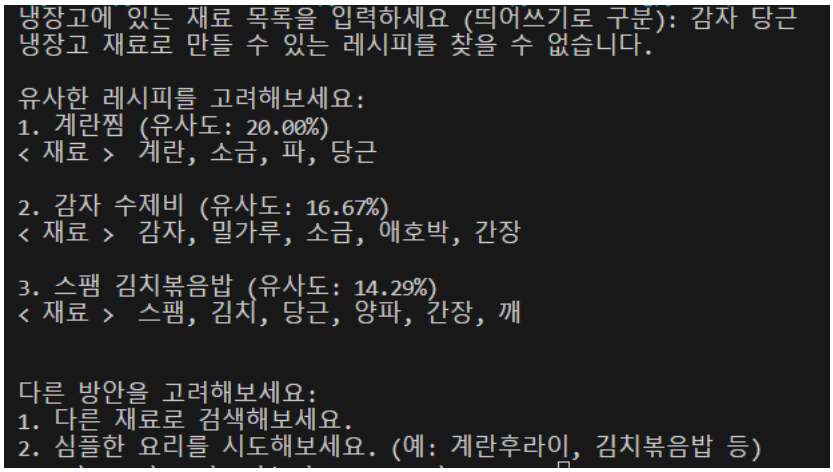


(3) Function 3

- Explanation

The code calculates the similarity between the user's input ingredients and each recipe in the dataset using the Jaccard similarity coefficient. It then sorts the recipes in descending order based on their similarity. The top 3 similar recipes are displayed, showing the recipe name, similarity percentage, and the list of ingredients used. Additionally, the code provides various suggestions to the user, such as trying a different set of ingredients or attempting simpler recipes.

- Test result screenshot



4. Changes in Comparison to the Plan

1) In the case where there is no matching recipe, alternative suggestions

- Before

Print 'Unable to find recipes with the provided ingredients.'

- After

Print 'Unable to find recipes with the provided ingredients.' Afterward, rank and present recipes that somewhat match the user-entered ingredients based on similarity. Additionally, provide alternative solutions.

- Reason

In cases where there are no ingredients for a recipe, it contradicts the purpose of the program, which is to recommend recipes for the user. Therefore, it is to complete the program as one that does its best to recommend recipes even when an exact match is not found.

5. Schedule

Work	11/3	11/10	11/12	11/15	11/20	11/22	11/24	11/26	12/1	12/15	12/22
Proposal	Clear										
Text File	Clear										
Source Code		Clear									
Function1		Clear									
Feedback1			Clear								
Function2				Clear							
Feedback2				Clear							
Function3						Clear					
Feedback3								In Progress			
All Feedback								Not Started			
Final Report										Not Started	