Python programming and practice

Food Recipe Recommendations Based on User Preferences or Ingredients

Progress Report : 2

Date : 2023/12/08

Name : Seo Jun

ID :230750

1. Introduction (16 pt)

1) Background (14 pt)

Recipe recommendation systems make it easy for users to find information about cooking, save time compared to internet searches, and provide practical cooking ideas. Interest in cooking and food-related activities is on the rise, leading to an increased demand for recipe recommendation services.

2) Project goal

Goal is to create a program that recommends recipes based on customer preferences and available ingredients.

3) Differences from existing programs

There is an app called '만개의 레시피' that provides recipes for free and includes links to purchase cooking ingredients. Users can also upload their own recipes. However, my program differentiates itself by recommending recipes based on user preferences and available ingredients.

2. Functional Requirement

1) Collecting Information from Users

- Feature to Input Preferred Foods or Ingredients.

(1) Collecting Food and Ingredient Information by Creating an Arbitrary Text File

- Ability to add, delete, modify, and view ingredients in the text file.

2) Comparing Recipe Files with User Profiles

- Function to calculate similarity for each file

(1) Preprocessing each sentence, creating a token set to calculate similarity, and storing similarity scores in a dictionary.

- Sorting the similarity dictionary in descending order and ranking recipes based on similarity.

3) Extracting Recipes with High Similarity

- Implemented as a function to extract the top 3 recipes with high similarity and present them to the user.

(1) If the user is not satisfied, extract the next 3 in the ranked list.

- Access the next index in the list sorted by similarity after excluding the top 3.

3. Progress

1) Function Implementation

(1) Collecting Information from Users

-Input/Output

add_ingredient(ingredient_list, ingredient)

Input: ingredient_list - a list containing existing ingredients

ingredient - the name of the ingredient to be added

Output: None

display_ingredient(ingredient_list)

Input: ingredient_list - a list containing the current ingredients

Output: None

save_ingredients_to_file(ingredient_list, filename)

Input: ingredient_list - a list containing ingredients to be saved

filename - the name of the file to save the ingredients

Output: None

load_ingredients_from_file(filename)

Input: filename - the name of the file to be read

Output: a list containing the loaded ingredients

modify_ingredient(ingredient_list, index, new_ingredient)

Input: ingredient_list - List containing ingredients to be modified.

Index - Index of the ingredient to be modified.

new_ingredient - Name of the modified ingredient.

Output: None

def remove_ingredient(ingredient_list, index)

Input: ingredient_list - List containing ingredients to be removed.

Output: None

-Description

IngredientModule class:

A class containing functions for handling ingredient lists.

add_ingredient(ingredient_list, ingredient) function:

Adds a new ingredient to the list of ingredients.

display_ingredient(ingredient_list) function:

Displays the current list of ingredients.

save_ingredients_to_file(ingredient_list, filename) function:

Saves the list of ingredients to a file.

load_ingredients_from_file(filename) function:

Reads the list of ingredients from a file.

modify_ingredient(ingredient_list, index, new_ingredient):

Modify the ingredient by accessing it through the index.

def remove_ingredient(ingredient_list, index):

Remove the ingredient by accessing it through the index.

-Applied Concepts (e.g., loops, conditional statements, functions, classes, modules)

Modules, classes, functions, input/output, exception handling, loops, conditional statements, file input/output (File I/O), class, exception handling.

-Code Snapshot

```python
class ingredient_module:
    #재료 추가하는 함수 생성
    def add_ingredient(ingredient_list, ingredient):
        ingredient_list.append(ingredient)
    #재료 보여주는 함수 생성
    def display_ingredient(ingredient_list):
        if not ingredient_list:
            print("재료 목록이 비어 있습니다.")
        else:
            print("재료 목록:")
            for i in range(len(ingredient_list)):
                ingredient = ingredient_list[i]
                print(f"{i+1}. {ingredient}")
    #재료를 파일에 저장하는 함수 생성
    def save_ingredients_to_file(ingredient_list, filename):
        write_fp = open(filename, 'w', encoding="utf8")
        for ingredient in ingredient_list:
            write_fp.write(ingredient+"\n")
        write_fp.close()
        print("재료가 파일에 저장되었습니다.")
```

```python
#재료를 읽어오는 함수
def load_ingredients_from_file(filename):
    try:
        ingredient_list = []
        read_fp = open(filename,'r', encoding="utf8")
        lines = read_fp.readlines()
        for line in lines:
            ingredient_list.append(line.strip())
        read_fp.close()
        return ingredient_list
    except FileNotFoundError:
        print("파일을 찾을 수 없습니다.")
# 재료를 수정하는 함수
def modify_ingredient(ingredient_list, index, new_ingredient):
    if 0 <= index < len(ingredient_list):
        ingredient_list[index] = new_ingredient
        print("재료가 수정되었습니다.")
    else:
        print("유효하지 않은 인덱스입니다.")
#재료를 제거하는 함수
def remove_ingredient(ingredient_list, index):
    if 0 <= index < len(ingredient_list):
        removed_ingredient = ingredient_list.pop(index)
        print(f"{removed_ingredient}이(가) 제거되었습니다.")
    else:
        print("유효하지 않은 인덱스입니다.")
```

```python
from ingredient_module import IngredientModule as im
from recipe_module import calculate_similarity, display_top_n_recipes
import numpy as np  # NumPy 모듈 임포트 추가

def read_text_file(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        content = file.read()
    return content

def main():
    # 사용자 프로필 및 레시피 파일 경로
    user_profile_path = 'ingredient.txt'
    recipe_file_path = 'food_info.txt'

    # 사용자 프로필 및 레시피 파일 내용 가져오기
    user_profile = ''
    ingredient_list = []
```

```python
try:
    file_name = "ingredient.txt"
    ingredient_module = im()
    ingredient_list = im.load_ingredients_from_file(file_name)
    while True:
        print("\n레시피 추천 프로그램")
        print("1. 재료 추가")
        print("2. 재료 수정")
        print("3. 재료 제거")
        print("4. 재료 목록 표시")
        print("5. 종료")
        choice = input("선택: ")
        if choice == '1':
            ingredient = input("재료를 입력하세요: ")
            im.add_ingredient(ingredient_list, ingredient)
            print("재료가 추가되었습니다.")
        elif choice == '2':
            index = int(input("수정할 재료의 인덱스를 입력하세요(1번부터 입력): ")) - 1
            new_ingredient = input("새로운 재료를 입력하세요: ")
            im.modify_ingredient(ingredient_list, index, new_ingredient)
        elif choice == '3':
            index = int(input("제거할 재료의 인덱스를 입력하세요(1번부터 입력): ")) - 1
            im.remove_ingredient(ingredient_list, index)
        elif choice == '4':
            im.display_ingredient(ingredient_list)
```

```python
        elif choice == '5':
            im.save_ingredients_to_file(ingredient_list, file_name)
            print("프로그램을 종료합니다.")
            user_profile = ' '.join(ingredient_list)
            break
        else:
            print("잘못된 선택입니다. 다시 시도하세요.")
except FileNotFoundError:
    print("파일을 찾을 수 없습니다.")
```

(2) Comparing Recipe Files with User Profiles

(1) Preprocessing each sentence, creating a token set to calculate similarity, and storing similarity scores in a dictionary.

-Input/Output

Input:

- user_profile: String representing user profile.

- recipes: List containing recipes.

- text: Original text for preprocessing and tokenization.

Output:

- sorted_recipe_indices: NumPy array containing the indices of recipes sorted by similarity.

- tokens: List of tokens obtained by tokenizing the text.

- Description

def preprocess_text(text):

Role in preprocessing and tokenization of text.

def calculate_similarity(user_profile, recipes):

The function calculates the similarity between the given user profile and various recipes, and it sorts the recipes based on the calculated similarity.

-Applied Concepts (e.g., loops, conditional statements, functions, classes, modules)

Array and indexing, conditional statements, loops, functions, modules, frequency,

tokenization, preprocessing.

-Code Snapshot

```python
from soynlp.tokenizer import MaxScoreTokenizer
from collections import Counter
import numpy as np

def preprocess_text(text):
    tokenizer = MaxScoreTokenizer()
    tokens = tokenizer.tokenize(text)
    return tokens

def calculate_similarity(user_profile, recipes):
    # 텍스트 전처리 및 토큰화
    user_tokens = preprocess_text(user_profile)
    recipe_tokens_list = [preprocess_text(recipe) for recipe in recipes]

    # 사용자 프로필과 레시피의 단어 빈도 계산
    user_counter = Counter(user_tokens)

    similarities = []
    for recipe_tokens in recipe_tokens_list:
        # 레시피 블록을 토큰화하고 사용자 프로필과의 유사도 계산
        intersection = sum((user_counter & Counter(recipe_tokens)).values())
        union = sum((user_counter | Counter(recipe_tokens)).values())
        similarity = intersection / union if union != 0 else 0
        similarities.append(similarity)

    # 유사도에 따라 레시피를 정렬
    sorted_recipe_indices = np.argsort(similarities)[::-1]

    return sorted_recipe_indices
```

```python
recipes = read_text_file(recipe_file_path).split('\n\n')
sorted_recipe_indices = calculate_similarity(user_profile, recipes)
```

3) Extracting Recipes with High Similarity

(1) If the user is not satisfied, extract the next 3 in the ranked list.

-Input/Output

Input

sorted_recipe_indices: List of recipe indices sorted by similarity.

sorted_indices: List of recipe indices sorted by similarity.

Output

None

- Description

def display_top_n_recipes: Sorting the recipe indices list based on similarity, taking the entire recipe list, and outputting the top n recipes.

-Applied Concepts (e.g., loops, conditional statements, functions, classes, modules)

Lists, arrays and indexing, conditional statements, loops, functions, modules.

-Code Snapshot

```python
def display_top_n_recipes(sorted_indices, recipes, n=3):
    for i in range(n):
        idx = sorted_indices[i]
        recipe = recipes[idx]
        print(f"\n레시피 {i + 1}:\n{recipe}")
```

```
display_top_n_recipes(sorted_recipe_indices, recipes, n=3)

satisfied = input("만족하시나요? (y/n): ")

while satisfied.lower() != 'y':
    current_top_indices = sorted_recipe_indices[:3]

# 다음으로 추천될 레시피 중에서 현재 상위 3개의 인덱스를 제외한 인덱스들
    next_top_indices = []
    for idx in sorted_recipe_indices:
        if idx not in current_top_indices:
            next_top_indices.append(idx)

    # 상위 3개를 모두 찾았으면 반복 종료
        if len(next_top_indices) == 3:
            break
    display_top_n_recipes(next_top_indices, recipes, n=3)
    satisfied = input("만족하시나요? (y/n): ")
```

Test Result

(1) Collecting Information from Users

- Description

I tested inputting, saving to a file, and program termination in the file.

The test revealed an issue where the module was not being called due to a conflict between the class and the name. After renaming the class, I successfully called the module.

-Test Results Screenshots

```
레시피 추천 프로그램
1. 재료 추가
2. 재료 목록 표시
3. 종료
선택: 1
재료를 입력하세요: 당근
재료가 추가되었습니다.

레시피 추천 프로그램
1. 재료 추가
2. 재료 목록 표시
3. 종료
선택: 1
재료를 입력하세요: 파
재료가 추가되었습니다.

레시피 추천 프로그램
1. 재료 추가
2. 재료 목록 표시
3. 종료
선택: 2
재료 목록:
1. 당근
2. 당근
3. 양파
4. 코코아
5. 당근
6. 파

레시피 추천 프로그램
1. 재료 추가
2. 재료 목록 표시
3. 종료
선택: 3
재료가 파일에 저장되었습니다.
프로그램을 종료합니다.
```

- Description

During the process of removing and modifying ingredients in a file, there was an inconvenience for the user because the input index and the visible index did not match. To address this, I added -1 to the index, aligning the user's input index with the actual file index.

-Test Results Screenshots

4. 재료 목록 표시
5. 종료
선택: 4
재료 목록:
1. 당근
2. 소금
3. 코코아
4. 당근
5. 파

레시피 추천 프로그램
1. 재료 추가
2. 재료 수정
3. 재료 제거
4. 재료 목록 표시
5. 종료
선택: 2
수정할 재료의 인덱스를 입력하세요(1번부터 입력): 3
새로운 재료를 입력하세요: 후추
재료가 수정되었습니다.

레시피 추천 프로그램
1. 재료 추가
2. 재료 수정
3. 재료 제거
4. 재료 목록 표시
5. 종료
선택: 4
재료 목록:
1. 당근
2. 소금
3. 후추
4. 당근
5. 파

레시피 추천 프로그램
1. 재료 추가
2. 재료 수정
3. 재료 제거
4. 재료 목록 표시
5. 종료
선택: 5
재료가 파일에 저장되었습니다.
프로그램을 종료합니다.

4. Changes in Comparison to the Plan

Title of the Change

-Before

Output remains the same even if the user knows the recipe.

-After

If the user knows the recipe, prompt for input and output a different recipe.

-Reason

Initially, individual cooking knowledge was not considered. However, based on feedback that chef's recipes are too common, this modification is made to address the issue.

5. Schedule

업무		11/3	11/10	11/30	12/5	12/15
Create a proposal		Completion				
Collecting Information from Users	Collecting Food and Ingredient Information by Creating an Arbitrary Text File		Completion			
Comparing Recipe Files with User	Preprocessing each sentence,			Completion		

Profiles	creating a token set to calculate similarity, and storing similarity scores in a dictionary.				
Extracting Recipes with High Similarity	If the user is not satisfied, extract the next 3 in the ranked list.			Completion	
TEST					------->