Python Programming and Practice

Development of a Personalized Perfume Recommendation System

Progress Report : 1

Date : 2023.11.26

Name : Sabin Wang

ID : 213407

1. Introduction

1) Background

The perfume market continues to grow steadily, unaffected by external factors such as economic uncertainty. Consequently, the industry anticipates the perfume market to remain stable for the foreseeable future. Additionally, with the diminishing impact of the COVID-19 pandemic, the reduced usage of masks has led to increased interest in perfumes. However, the expansion of the perfume market has resulted in the release of numerous products, making it challenging for customers to find a fragrance that suits them. Furthermore, exposure to too many scents at once can lead to olfactory fatigue. Therefore, to address these issues, a system recommending customized perfumes to customers is deemed necessary.

2) Project goal

The goal is to develop a system that, when provided with information about a customer's favorite or preferred fragrance, recommends similar perfumes to enhance customer satisfaction.

3) Differences from existing programs

The existing programs recommend new perfumes based on the customer's past purchase history. However, these recommendations may not align with the customer's current preferences as they rely on outdated buying patterns. In this system, the customer directly inputs their favorite fragrance, and the system recommends perfumes with similar characteristics. Additionally, the system takes into account not only the fragrance notes but also the description (feel and features) to provide recommendations, offering a differentiation from existing programs.

2. Functional Requirement

1) Function 1 (**Identifying Similar Fragrances**)

- A feature that locates perfumes with scents similar to the one entered by the customer.

(1) Detailed function 1

- Classifying Similar Fragrances Based on Notes and Finding the Most Similar Perfume to Customer's Input

(2) Detailed function 2

- Tokenizing Perfume Descriptions (Feel and Features) and Finding the Perfume with the Highest Similarity to User-Entered Description

2) Function 2 (**Adding Perfume to wishlist.txt**)

- Adding Recommended Perfume to Wishlist

(1) Detailed function 2

- Creating wishlist.txt File
- Adding Perfume Name and Brand to the Wishlist:
- Removing Previously Purchased Perfume from the List:

3. Progress

1) Implementing the features

(1) Main Module (Initial Menu Execution) – main.py

- Input: Menu number desired by the user
- Output: Execution of the respective module based on the entered menu number
- Upon program execution, the initial menu is displayed. When the user inputs the number corresponding to their desired menu, the program calls and executes the function associated with that menu. For efficient code modification and execution, each functionality has been modularized and imported into main.py.
- Applied Concepts: Loops, conditional statements, functions, file I/O, modules, etc.

- Code screenshot

```python
from similar_recommendation import *
from new_recommendation import *
from mood_recommendation import *
from wishlist import *

import pandas as pd

dataFrame = pd.read_csv("data/perfume_data_new.csv")
df1 = pd.read_csv("data/similar_perfumes_data.csv")
#df2 = pd.read_csv("data/mood_perfumes_data.csv")

#프로그램 첫 실행시 출력될 문구
print("******Perfume Recommendation System******")

def main_menu() :
    user_input = input("""
Enter the number of the menu
1. Recommend a product similar to what you used to do
2. Get a new perfume recommendation
3. Recommend a perfume according to your mood
4. Get a wish list file
5. Existing the program
""")

    if user_input == "1" :
        input_name = input("Enter a perfume name : ")
        if input_name in df1['Perfume'].values: #유저가 입력한 향수가 데이터셋에 존재하는 경우
            similar_recommendation(input_name, df1)
        else: #존재하지 않는 경우
            no_similar_recommendation()

```

```
    elif user_input == "2" :
        while True :
            input_option = input("""
You can search for similar perfumes by entering 1.Notes or 2.Description.
Which one would you like to use for the search?
Enter the number you'd like to use. [1 or 2]
""")
            if input_option == "1" :
                #Notes 클래스의 인스턴스 생성 및 메서드 호출
                notes_instance = Notes()
                similar_notes = notes_instance.notes_perfume()
                print(similar_notes)
                break
            elif input_option == "2" :
                #description_perfume()
                break
            else :
                print("잘못된 입력입니다. 다시 입력하세요.")


    elif user_input == "3" :
        #mood_recommendation()
        pass

    elif user_input == "4" :
        #wishlist_file()
        pass

    elif user_input == "5" :
        print("Exiting the program.")
        return
```

```
    else :
        print("잘못된 입력입니다. 다시 입력하세요.")


if __name__ == "__main__":
    main_menu()

```

(2) Change File Encoding – change_file_encoding.py

- Input: Original data file downloaded from Kaggle (perfume_data.csv)
- Output: New data file encoded in 'UTF-8' (perfume_data_new.csv)
- An issue arose while loading the original data file downloaded from Kaggle due to its encoding being 'Windows-1254.' Therefore, the file contents were encoded in 'UTF-8' and saved.
- Applied Concepts: File I/O, etc.

- Code screenshot

```python
import chardet

# 파일 경로
file_path = 'data/perfume_data.csv'

# 파일의 실제 인코딩 확인
with open(file_path, 'rb') as file:
    result = chardet.detect(file.read())

# 확인된 인코딩 출력
print(f"파일의 실제 인코딩: {result['encoding']}, 신뢰도: {result['confidence']}")

# 파일을 확인된 인코딩(Windows-1254)으로 읽어오기
with open(file_path, 'r', encoding=result['encoding'], errors='replace') as file:
    content = file.read()

# 새로운 파일이 저장될 경로 (UTF-8로 저장할 파일)
new_file_path = 'data/perfume_data_new.csv'


# 파일 내용을 'UTF-8'로 인코딩하여 저장
with open(new_file_path, 'w', encoding='utf-8') as file:
    file.write(content)

```

(3) Identifying similar fragrances Based on Notes & Name

– similar_perfumes_data_generate.py

- Input: 'perfume_data_new.csv'
- Output: 'similar_perfumes_data.csv'
- To implement the functionality of recommending perfumes similar to the one entered by the user, a dataset was created where perfumes with similar Notes are clustered. To enhance efficiency, it was deemed more effective to perform the calculation once during the initial phase and store it in a data file for subsequent use, rather than recalculating every time.
- Applied Concepts: Functions, file I/O, modules, etc.

- Code screenshot

```python
## Keras_text to word sequence ##

import pandas as pd
from keras.preprocessing.text import text_to_word_sequence

#데이터 불러오기
file_path = 'data/perfume_data_new.csv'
perfume_data = pd.read_csv(file_path)

#결측치 있는 행을 제거 (Notes 열에 대해 80개의 결측치 존재)
perfume_data = perfume_data.dropna(subset=['Notes'])

#Notes를 단어 리스트로 변환
perfume_data['WordSequence'] = perfume_data['Notes'].apply(lambda x: text_to_word_sequence(x))

#단어 중복 유사도를 계산하는 함수
def word_overlap_similarity(word_seq1, word_seq2):
    set1 = set(word_seq1)
    set2 = set(word_seq2)
    return len(set1.intersection(set2))

#가장 유사한 향수를 찾는 함수
def get_most_similar_perfume(name, df):
    # 대상 향수의 단어 시퀀스를 가져옵니다
    target_word_seq = df[df['Name'] == name]['WordSequence'].iloc[0]

    # 다른 향수들과의 유사도를 계산합니다
    similarities = []
    for _, row in df.iterrows():
        sim = word_overlap_similarity(target_word_seq, row['WordSequence'])
        similarities.append((row['Name'], sim))
```

```python

    # 유사도 점수를 기준으로 향수를 정렬합니다
    similarities = sorted(similarities, key=lambda x: x[1], reverse=True)

    # 자신을 제외하고 상위 5개를 가져옵니다
    top_similar = [perfume_name for perfume_name, _ in similarities if perfume_name != name][:5]

    return top_similar

# 각 향수에 대한 가장 유사한 향수들을 저장할 데이터프레임을 생성합니다
similar_perfumes = []

for perfume in perfume_data['Name']:
    # 각 향수에 대한 가장 유사한 향수들을 가져옵니다
    most_similar = get_most_similar_perfume(perfume, perfume_data)
    similar_perfumes.append({'Perfume': perfume, 'Similar Perfumes': ', '.join(most_similar)})

# 리스트를 데이터프레임으로 변환합니다
similar_perfumes_df = pd.DataFrame(similar_perfumes)

# 데이터프레임을 csv 파일로 저장합니다
similar_perfumes_df.to_csv('data/similar_perfumes_data.csv', index=False)
```

(4) Identifying similar fragrances Based on Notes & Name

– similar_recommendation.py

- Input: Name of the perfume entered by the user
- Output: Display the most similar perfume to the entered one
- If the perfume entered by the user exists in 'similar_perfumes_data.csv,' the system outputs the most similar perfume. To provide additional recommendations, the system can also display four more similar perfumes if the user desires more suggestions. If the entered perfume is not found in 'similar_perfumes_data.csv,' the system calls a function to inquire whether the user wants to search for the perfume using a different method or return to the initial menu.
- Applied Concepts: Loops, conditional statements, functions, modules, etc.

- Code screenshot

```python
#user_input == 1
from new_recommendation import *
from wishlist import *

from middle import call_main_menu

#main.py의 메인 메뉴 호출을 위한 함수
def return_to_menu():
    call_main_menu()

#사용자가 입력한 향수가 데이터셋에 존재하는 경우
def similar_recommendation(input_name, df1) :
    similar_perfumes = df1[df1['Perfume'] == input_name]['Similar Perfumes'].values[0]
    similar_perfumes_list = similar_perfumes.split(', ')

    print("<Similar Perfume>")
    print('"',similar_perfumes_list[0],'"') #유사한 향수 1개 출력
    print()
    more_help = input("""Enter the number you want
        1. Save to Wishlist
        2. More recommendation
        3. Return to menu
        """)
    if more_help == "1" : #위시리스트에 저장
        pass #wishlist.py의 위시리스트 관련 모듈 불러오기
    elif more_help == "2" :
        print("<More Recommendation>")
        for i in range(1, len(similar_perfumes_list)) : #유사한 향수 4개 더 출력
            print('"',similar_perfumes_list[i],'"')
    elif more_help == "3" : #초기 메뉴로 돌아가기
        return_to_menu()

```

```
33
34  #사용자가 입력한 향수가 데이터셋에 존재하지 않는 경우
35  def no_similar_recommendation() :
36      print("Sorry, Not Found.. :<")
37      more_help = input("""1. Enter Notes to search Or 2. Return to menu
38  Enter the number you want
39  """)
40      if more_help == "1" :
41          # Notes 클래스의 인스턴스 생성 및 메서드 호출
42          notes_instance = Notes()
43          similar_notes = notes_instance.notes_perfume()
44          print(similar_notes)
45      elif more_help == "2" :
46          return_to_menu()
47
48
```

(5) Identifying similar fragrances Based on Notes – new_recommendation.py

- Input: Notes
- Output: Display the perfume most similar to the Notes entered by the user
- The system takes user input for Notes and calculates similarity by counting the overlapping words with the dataset's Notes. Subsequently, it outputs and displays the most similar perfume.
- Applied Concepts: Loops, conditional statements, functions, classes, file I/O, modules, etc.

- Code screenshot

```
1   #user_input == 2
2   from keras.preprocessing.text import text_to_word_sequence
3   from middle import call_main_menu
4
5   #main.py의 메인 메뉴 호출을 위한 함수
6   def return_to_menu():
7       call_main_menu()
8
9   class Notes:
10      import pandas as pd
11      from keras.preprocessing.text import text_to_word_sequence
12      from middle import call_main_menu
13
14      #main.py의 메인 메뉴 호출을 위한 함수
15      def return_to_menu():
16          call_main_menu()
17
```

```
    # 데이터 불러오기
    file_path = 'data/perfume_data_new.csv'
    perfume_data = pd.read_csv(file_path)

    # 결측치 있는 행을 제거 (Notes 열에 대해 80개의 결측치 존재)
    perfume_data = perfume_data.dropna(subset=['Notes'])

    # 각 향수의 'Notes'를 단어 시퀀스로 변환하여 새로운 열에 저장
    perfume_data['NotesSequence'] = perfume_data['Notes'].apply(lambda x: text_to_word_sequence(x))

    # 단어 중복 유사도를 계산하는 함수
    def notes_overlap_similarity(notes_seq1, notes_seq2):
        set_1 = set(notes_seq1)
        set_2 = set(notes_seq2)
        return len(set_1.intersection(set_2))

    # 가장 유사한 향수를 찾는 함수
    def most_similar_notes(self, user_notes_seq, df, num_results=5):
        similarity = []
        for _, row in df.iterrows():
            similar = Notes.notes_overlap_similarity(user_notes_seq, row['NotesSequence'])
            similarity.append((row['Name'], similar))

        similarity = sorted(similarity, key=lambda x: x[1], reverse=True) #유사도를 기준으로 정렬
        return [perfume_name for perfume_name, _ in similarity][:num_results]
```

```
    def notes_perfume(self):
        notes = []
        notes_input = input("단어 사이에 ,을 넣어 Notes를 입력하세요. : ")
        notes.append(notes_input)
        while True:
            more_notes = input("더 입력하시겠어요? [y/n]")
            if more_notes == 'y':
                notes_input = input("단어 사이에 ,을 넣어 Notes를 입력하세요. : ")
                notes.append(notes_input)
                continue
            elif more_notes == "n":
                #입력된 모든 노트를 하나의 문자열로 합치기
                all_notes = ', '.join(notes)
                #텍스트를 단어 시퀀스로 변환
                user_notes_sequence = Notes.text_to_word_sequence(all_notes)
                #유사한 향수 상위 1개 찾아서 출력
                most_similar = self.most_similar_notes(user_notes_sequence, self.perfume_data, 1)
                print("가장 유사한 향수:", most_similar[0])

                more_notes = input("""
    Enter the number you want
    1. Save to Wishlist
    2. More recommendation
    3. Return to menu
    """)
```

```
                if more_notes == "1" :
                    pass #wishlist.py의 위시리스트 관련 모듈 불러오기
                elif more_notes == "2" :
                    additional_similar = self.most_similar_notes(user_notes_sequence, self.perfume_data, 5)
                    print("추가 추천 향수:", additional_similar[1:])  # 첫 번째 추천을 제외하고 출력
                    notes_more_input = input("""
    Enter the number you want
    1. Save to Wishlist
    2. Return to menu
    """)
                    if notes_more_input == "1" :
                        pass #wishlist.py의 위시리스트 관련 모듈 불러오기
                    elif notes_more_input == "2" :
                        return_to_menu()
                        break
                elif more_notes == "3" : #초기 메뉴로 돌아가기
                    return_to_menu()
                    break
            else:
                print("잘못된 입력입니다. 다시 입력하세요.")
```

2) Test Results

(1) Identifying Similar Fragrances - Based on Name

- Explanation : If the user selects the feature to recommend perfumes similar to the ones they have previously used from the initial menu:

1. The system prompts the user to enter the name of the perfume.
2. If the entered perfume exists in the existing dataset (similar_perfumes_data.csv), the system outputs one perfume that is similar.
3. If the user wishes to save the recommended perfume to their wishlist, the system invokes the wishlist module (to be added later).
4. If the user desires more recommendations, the system displays four additional perfumes that are similar.
5. If the user wants to return to the initial menu, they can do so.

If the entered perfume is not found in the existing dataset:

1. The system prompts the user to choose between two options: a. Receive recommendations for perfumes in a different way. b. Return to the initial menu.

- Test results screenshot

```
******Perfume Recommendation System******

Enter the number of the menu
1. Recommend a product similar to what you used to do
2. Get a new perfume recommendation
3. Recommend a perfume according to your mood
4. Get a wish list file
5. Existing the program
1
Enter a perfume name : Velvet Fantasy Eau de Parfum
<Similar Perfume>
" 20 Mars 2022 Eau de Parfum "

Enter the number you want
        1. Save to Wishlist
        2. More recommendation
        3. Return to menu
        2
<More Recommendation>
" Artemisia Eau de Parfum "
" L'Aimee Eau de Parfum "
" Coup de Foudre Eau de Parfum "
" Elixir Pour Femme Extrait de Parfum "
```

(2) Identifying Similar Fragrances – Based on Notes

- Explanation :

If the user selects the feature to receive recommendations for a new perfume based on Notes from the initial menu:

1. The system prompts the user to enter Notes.
2. After the Notes input, the system combines all entered Notes into a single string and applies `text_to_word_sequence` to convert it into a word sequence.
3. It applies `text_to_word_sequence` to the Notes in the dataset (perfume_data_new.csv) as well.
4. The system calculates the similarity between the user's entered Notes and the Notes in the existing dataset, then outputs the most similar perfume to the user.
5. If the user wishes to save the recommended perfume to their wishlist, the system invokes the wishlist module (to be added later).
6. If the user desires more recommendations, the system displays four additional perfumes that are similar.
7. If the user wants to return to the initial menu, they can do so.

- Test results screenshot

```
******Perfume Recommendation System******

Enter the number of the menu
1. Recommend a product similar to what you used to do
2. Get a new perfume recommendation
3. Recommend a perfume according to your mood
4. Get a wish list file
5. Existing the program
2

You can search for similar perfumes by entering 1.Notes or 2.Description.
Which one would you like to use for the search?
Enter the number you'd like to use. [1 or 2]
1
단어 사이에 ,을 넣어 Notes를 입력하세요. : coffee, musk
더 입력하시겠어요? [y/n]y
단어 사이에 ,을 넣어 Notes를 입력하세요. : vanilla
더 입력하시겠어요? [y/n]n
가장 유사한 향수: Velvet Fantasy Eau de Parfum

    Enter the number you want
    1. Save to Wishlist
    2. More recommendation
    3. Return to menu
    2
추가 추천 향수: ['Fusion Sacree - Obscur Eau de Parfum', 'Ristretto Intense Cafe Extrait de Parfum', 'Purple Sakan', 'Sensual Instinct Eau de Parfum']

    Enter the number you want
    1. Save to Wishlist
    2. Return to menu
2

Enter the number of the menu
1. Recommend a product similar to what you used to do
2. Get a new perfume recommendation
3. Recommend a perfume according to your mood
4. Get a wish list file
5. Existing the program
5
Exiting the program.
```

4. Changes in Comparison to the Plan

1) Change History

- There is no change except for adding feature.

2) Add Feature

[Perfume recommendation based on emotions]

The existing planned perfume recommendation system had three main features : A function recommending perfumes similar to those the user has used before, A recommendation based on "Notes," and A recommendation based on "Description." In addition to these, I am planning to add a new feature that recommends perfumes based on the user's emotions. This new feature is designed to be more user-friendly for customers who may find it difficult to input their desired fragrance due to a lack of familiarity with perfumes. Unlike the existing recommendation features, this new addition aims to enhance perfume sales by reaching out to customers who are not yet well-acquainted with fragrances. It is also anticipated to provide a novel and enjoyable experience for existing customers.

5. Schedule

WORK		11/3	11/10	11/17	11/26	12/3	12/10	12/15	12/22
Create a proposal		Done							
Structure design, Progress report creation		Done							
Function 1 (Identifying Similar Fragrances)	Detailed function 1 (Based on Notes)		Done						
Function 1 (Identifying Similar Fragrances)	Detailed function 2 (Based on Description)			In progress					
Function 3 (Perfume recommendation based on emotions)									
Function 2 (Adding Perfume to wishlist.txt)									
create a progress report #2									
Finalizing a project, Making a final report									