The Music Guru

A recommendation software system

Raveendrababu Pasumarthi
4-12-2020

1.0 Introduction

With the overwhelming volume of online content and increasing ubiquity of Internet-enabled devices, pervasive use of the Web for content sharing and consumption has become our everyday routines. However, people seeking online access to content or items of interest are becoming more and more frustrated due to information overload. Deciding the content to consume from deluge of available alternatives becomes increasingly difficult. In this regard, an online content streaming company provides movies, TV shows, music and other services to millions of online users. The company makes profits from selling interesting content/ services of all genres to their customers. In the last couple of years, the fortune of the company has suffered due to dwindling sales. It has been decided to develop an intelligent service recommendation engine for company's online platform. The company has provided a dataset containing music tracks of various artists as well as the music features. This document briefly discusses about the problem understanding, requirements analysis, high level and low-level technical designs, coding & implementation and the limitations or future enhancements

The key implementations of the recommendations are

1. Processing the data set and successful loading into a Python module
2. Implementation of similarity measures as a separate module
3. A simple user interface to interact with the system by the user

The key considerations are

1. Data set must be loaded into two dictionaries with music properties and music features
2. Similarity measures must be functions so that they can called to the main module
3. The similarities functions to be implemented are
    a. euclidean_similarity
    b. cosine_similarity
    c. pearson_similarity
    d. jaccard_similarity
    e. manhattan_similarity

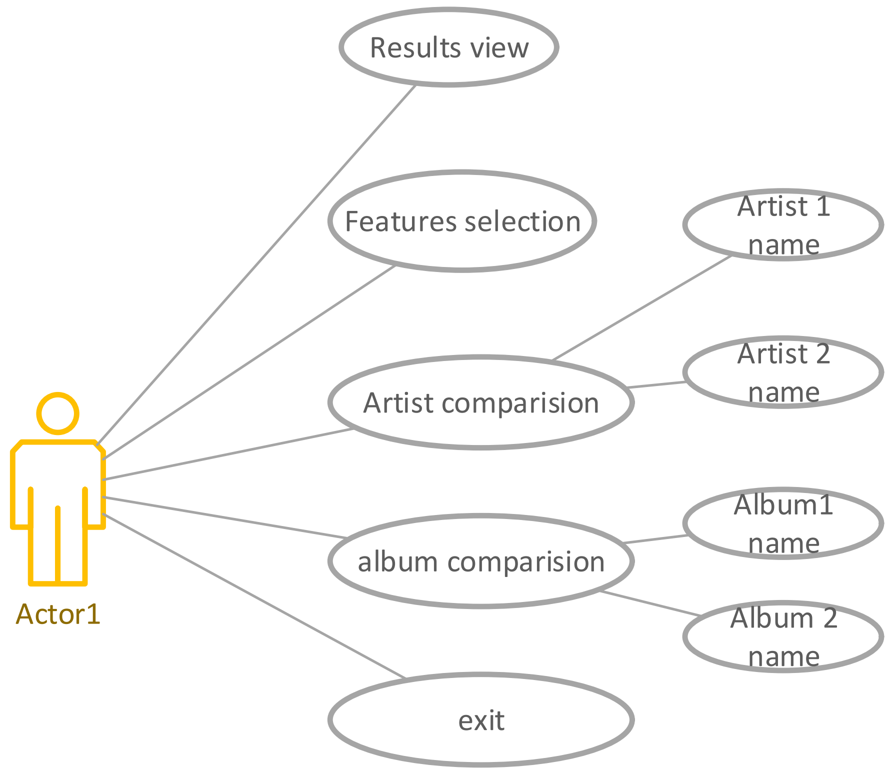
2.0 Requirements Analysis

The following requirements must be met by the recommendation system

1. Two artist comparison
2. Two albums comparison
3. Ability to select the music features for the comparison
4. Comparison results viewing
5. Reuse the system for new comparison
6. Smooth exit

The requirements implementations are analyzed with the appropriate use cases below

Use case

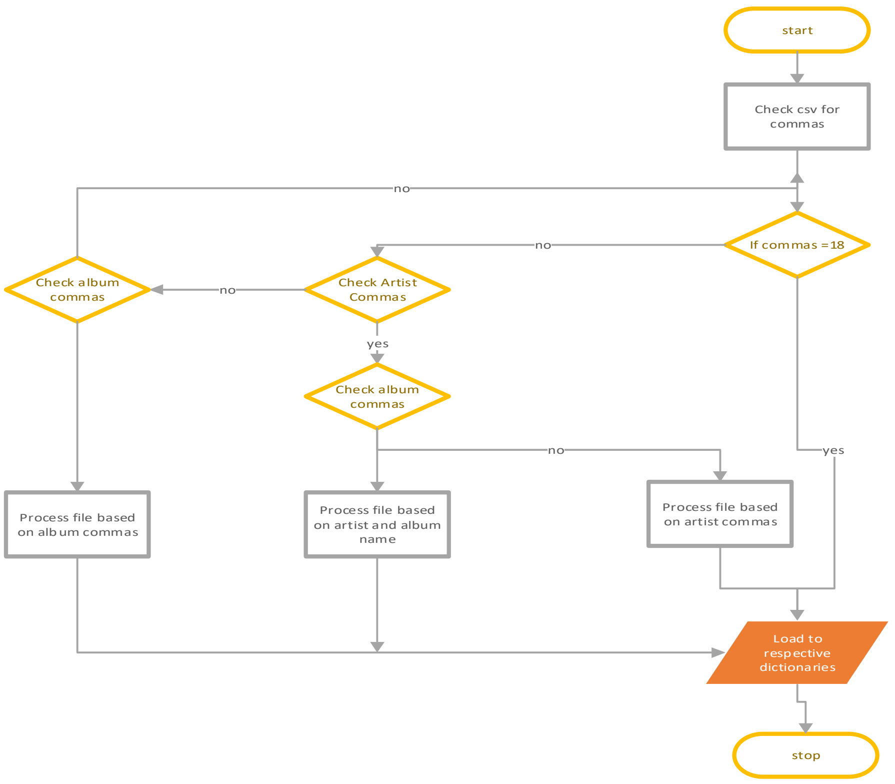


3.0 High Level Design

Key considerations of the design

1. Csv file parsing
2. Distinct artist names and Distinct album names

3.1 csv File Parsing:


start
Check csv for commas
no
no
If commas =18
Check album commas
no
Check Artist Commas
yes
Check album commas
no
yes
Process file based on album commas
Process file based on artist and album name
Process file based on artist commas
Load to respective dictionaries
stop

3.2 Artist distinct names:

- One artist example ['Taylor Swift']
- Two artist example "['Bethel Music', 'Amanda Lindsey Cook']"
- Handle the double quotes square brackets with reg ex.,
- Two artists split them by commas and separate
- Take them into a list and then to dictionary to deduplicate
- Special cases of artists were taken care with UTF-8 and other patterns with regex

  E.g,1**'Orchestra E Coro Del Teatro Alla Scala / Giuseppe Antonicelli / Vittore Veneziani'**

4.0 Low Level Design/Implementation

- load module functions:

```
artist_music_dict()
music_features_dict()
=flattenartists(lis)
no_of_features() #select no.of features option
multi_feat_sel_func() # any three features selection
allfeatfunc() # All features seletion
```

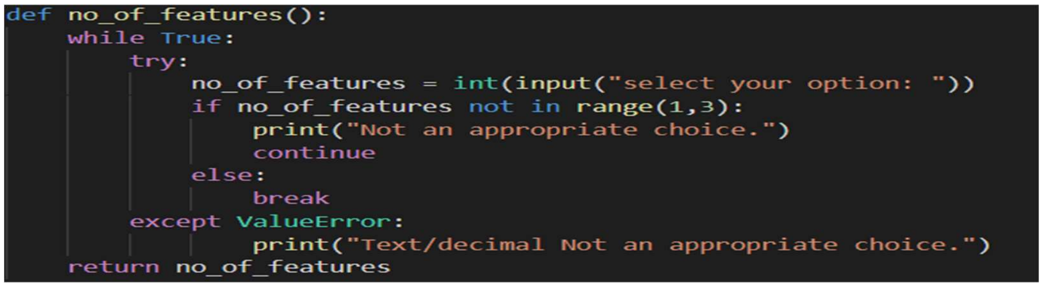
- similarity module functions:

```
vector_xy(d,id1,id2) # a vector function
euclidean_similarity_func(d,id1,id2)
cosine_similarity_func(d,id1,id2)
manhattan_similarity_func(d,id1,id2)
jaccard_similarity_func(d,id1,id2)
# function for any three features

euclidean_similarity_one_func(x,y)
cosine_similarity_one_func(x,y)
manhattan_similarity_one_func(x,y)
jaccard_similarity_one_func(x,y)
```

Exception Handling:

- At every user interaction, exceptions were implemented with appropriate messages



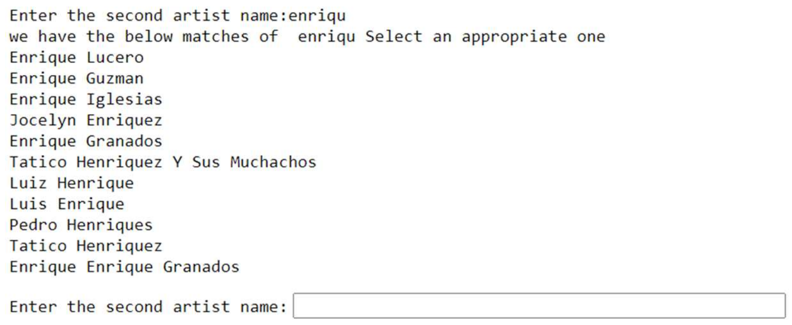
```
def no_of_features():
    while True:
        try:
            no_of_features = int(input("select your option: "))
            if no_of_features not in range(1,3):
                print("Not an appropriate choice.")
                continue
            else:
                break
        except ValueError:
                print("Text/decimal Not an appropriate choice.")
    return no_of_features
```

Modules imported: `import re` and `import math`

4.0 Testing & Limitations/Enhancements

Testing:

- Thorough testing was conducted on the artists and albums
- If user enters a substring, if there is an album/artist with the substring is considered, otherwise 20 (max) results will be displayed



```
Enter the second artist name:enriqu
we have the below matches of  enriqu Select an appropriate one
Enrique Lucero
Enrique Guzman
Enrique Iglesias
Jocelyn Enriquez
Enrique Granados
Tatico Henriquez Y Sus Muchachos
Luiz Henrique
Luis Enrique
Pedro Henriques
Tatico Henriquez
Enrique Enrique Granados

Enter the second artist name:
```

- Few rows have only one letter artists and as n/a , that's not handled
- E.g., n,a,X,-,A,Z,D,O,M,V,K
- Implemented similarities working fine

Limitations:

- If an artist name is a substring in another artist, then there is chance of another artist result will be displayed (e.g., "Ramchandra Pal" will be displayed for "Ramchandra")
- If user enters a single character in any of the character in the list "n,a,X,-,A,Z,D,O,M,V,K" as input, then results will be wrong
- "Sky fall" album name was repeatedly asked and not handled by the system
- Main notebook was not converted into a main function (indentation issues were occurred due to late conversion of code into a function , it's a learning curve and a good lesson)