PROGRAMMING CONCEPT AND PRACTICE

(PCP)

Tenzin Rabyang-c0017118

Sheffield Hallam University

Assignment-I

REPORT

**Context:**

1. Implementation and Justification
2. How to execute the Program
3. Best Metrics/ Similarity functions
4. Algorithm
5. Flowchart
6. My experience
7. Problem analysis
8. Reference
9. **Implementation and justification.**

Here I will discuss the implementation and why?

* Firstly, I have loaded and read the data into two separate dictionaries and merged both of them with the same user id so that I can use only one dictionary rather than two.
* Then I have created a class called **other\_similarity**, which has a function or method called **list\_len(x,y).**

This function makes the list x and y of the same length if they are not the same.

* Because most of the, when I run the similarity function, Program throws an error of **IndexError.** It is coming because list "x" contains the rating value from the first input, and list "y" includes the rating value from the second input.
* Since it is implausible to have the same length, I append the "0" value to the lesser length of the list to make it the same length. It is not affecting the result. Hence **list\_len(x,y)** function is there.
* I have implemented so many different GLOBAL variables that these variables can be accessed by various modules and methods in the Program for more straightforward computation.
* At last, there is another class called **get\_simi().** This class contains many methods that take inputs from the user, which does all the verification if inputs are in the dictionaries and prints different kinds of options available for the user like exiting, try again, and what similarity function to print.
* It helps me put the Program in the loop unless the user chooses to exit.
* One of the best implementations is that the Program will automatically recognize whether the inputs are USERID to USERID, BOOK to BOOK, BOOK to USERID, or USERID to BOOK, etc. So that user doesn't have to don't have to give an explicit option. Instead, Program will do what to check.

1. **How to execute the Program.**

* First, all the code files and the data (CSV) must be in the same folder. Otherwise, Program will not work.
* Then open the **main\_function.ipynb** file, and you will notice a few lines of code.
* Execute and code and wait until asked to give a First input, followed by the second input. Give appropriate input. You can provide USER – USER, BOOK NAME – BOOK NAME, ISBN TO ISBN, or interchange this value.
* Then options will be displayed whether to check a particular similarity function, all functions, or exit the Program.
* The Program is in the loop, and it will ask you again and again unless you exit the Program, as explained below.
* To exit the Program, there are two different options available. First, when you give input, and Program will display all the similarity functions, and at the end, there are also options for exit.
* Secondly, once you get your first similarity, the Program will ask if you want to exit or not. Just type "y" for yes, and Program will stop, or type "n" and Program will continue for your input again.
* This loop will continue unless you stop the Program.

1. **Best Metrics/ Similarity functions.**

* I have implemented 7 different similarities matric or functions between book to book, user to user, ISBN to ISBN, ISBN to book, ISBN to the user, and vice versa.
* First, I can tell that all the metric produces a result with a reasonable accuracy value. I verified with an online calculator, like Cosine similarity, Square Euclidean distance, and Minkowski distance.
* The best-performed matric, I would say, **Squared Euclidean, Minkowski, Chebyshev, and Hamming distance similarity** because this function will work even with every fewer amount of data from both inputs given by the user. Say if the first user id doesn't have any data (rating in the dictionary). However, the second user id has only one data in the dictionary; these (above mentioned) functions can still run and produce similarities.
* However, the least good metric I would say is **Spearman correlation, Cosine similarity, and Pearson correlation**.Because to get an accurate result, first, we must have enough data to calculate.
* Both the input from the user must have at least more than 1 data to return the proper result.

1. **Algorithms**

**This algorithm is only for checking and returning similarity between two already given inputs as input1 and input2**

Step1: Start.

Step2: Define global variable total, b1, b2, b3, s1, s2, l1, l2, p, input1, input2, pref

Step3: Assign value as below.

Text

Description automatically generated

Step4: Then import module call other\_similarity as **othersi**

Import other\_similarity as othersi

Step5: Create find\_total(x,y) function with two-parameter of the list as x, y to calculate total and return total as below

Text

Description automatically generated

Step6: If both input1 and input2 are userID, check if both userID has the exact book name.

A screenshot of a computer

Description automatically generated with low confidence

Step7: If both input1 and input2 are book names, append all the ratings from input1 in b1 and append from input2 in b2.

Text

Description automatically generated

Text

Description automatically generatedStep8: Then make both lists b1, b2 with the same length by calling list\_len(x,y) from the other\_similarity module.



Step9: Then call find\_total() function to calculate total

Step10: If input1 and input2 are ISBN, append all the rating values from input1 to b1 and input2 to b2.

Text

Description automatically generated

Step11: Then make b1, b2 with the same length of the list and return total



Text

Description automatically generatedStep12: If input1 is ISBN and input2 is USERID, then append all the rating values of ISBN to b2 and append all the rating values of USERID to b1.

Step13: Make b1 and b2 with the same length and return total.



Text, letter

Description automatically generatedStep14: If input1 is USERID and input2 is ISBN, then append all the rating values of USERID into b1 and ISBN to b2.

Step15: Make b1 and b2 with the same length and return total.



Step16: Likewise, do the same procedure for every option.

Step17: At last, you return the actual similarity value using its formula. By simply returning the total value, we get the square Euclidean similarity. Use a different formula to return the corresponding similarity value for distinct similarities functions.

Step18: END

1. **Flowchart**

**Below Chart is a straightforward flow chart of the whole Program and module presented in the assessment.**

Diagram

Description automatically generated

1. **My Experience**

I would say that this assessment is like a refreshment to my python coding. Without consisting of practice, we tend to forget. However, whatever I have learned python during my bachelor's and in class, this assessment helps to regain most of the python basics, especially if and for loop, because the coding is done using these two statements. The most frequent problem that I had was with variables. Since I have a lot of variables in my code, most of the time, I get an error of local variables. So, I had to spend a long time understanding to solve the problem. So, in conclusion, I would say, since this coding is very raw, this assessment is perfect for practice.

1. **Problem Analysis**

A few problems that I have with my code are.

* Code will not accept book name without space; book name must have space
* Not all of the data are stored in the user\_preference dictionary.
* Only if both inputs are user id, only Program will check for the similar book by book name. Otherwise, Program will extract all the value available in the dictionaries.

1. **Reference**

<https://en.wikipedia.org/wiki/Euclidean_distance>

<https://people.revoledu.com/kardi/tutorial/Similarity/MinkowskiDistance.html>

<https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient>

https://www.sciencedirect.com/topics/engineering/spearman-rank

<https://en.wikipedia.org/wiki/Chebyshev_distance>

<https://www.tutorialspoint.com/what-is-hamming-distance>

<https://www.geeksforgeeks.org/cosine-similarity/>

<https://en.wikipedia.org/wiki/Pearson_correlation_coefficient#:~:text=Correlations%20equal%20to%20%2B1%20or,corr(Y%2CX)>.