CS110 Project 4: A Book Recommendation System

Prof. Karpenko

Due: Wed, Dec 3d, 2014, 11:59pm

For project 4 you will develop a book recommendation system[1]. Websites like goodreads.com ask a person to create a user profile where they specify the book titles they enjoyed reading, and then the system provides recommendations of other books to read. The recommendations are based on the user profile and the ratings data collected from other users. Similar ideas are used by various businesses: for instance, Netflix website recommends which movies a person might like, and amazon.com recommends a list of books you might want to purchase based on your previous order history and based on what other people bought in the past.

How can we predict what books a person (let us call her Alice) might like? One simple approach would be to ask Alice to rate some of the books she read previously, and compare these ratings to the ratings data of other users. The system can then find several users that are most **similar** to Alice, look at their ratings, and recommend the books they liked to Alice if she has not read them. The idea is that if you and some other person both liked the same books in the past, then the chances are, you might like some other books they rated well.

Similarity Metric

To measure how similar two users are, we will treat their ratings as two vectors v1 and v2, and compute the dot product of v1, v2. For example, suppose we have only 3 books in books.txt, and we want to compute the similarity between the following two users: Alice, who rated the books 5 0 -3 and Jeff, whose ratings for the three books are 3 -5 1. Here v1 = [5, 0, -3] and v2 = [3, -5, 1]. The similarity would be the dot product of v1 and v2: 5*3 + 0*-5 + (-3)*1 = 12.

Suppose we have another user, Kaiming, who rated the books as follows: 5 3 -2. The similarity between Alice and Kaiming is 5*5+0*3+(-3)*(-2) = 31. We can say that Kaiming's taste is more similar to Alice's taste than Jeff's, so we are more likely to recommend the books Kaiming liked to Alice (the second book in this example).

Your program will compute the pair-wise similarity between a given user, for example Alice, and all the other users. Then your program will identify the **top five** users who are "most similar" to Alice, and will write a list of recommended books for Alice to a file called "recommendedBooks.txt". The recommended books are the ones that Alice has not read and that were rated well (with a rating of 5) by any of the five most similar users.

[1] This assignment is based on the assignment originally designed by Professor Michelle

Data[2]

You will be using the previously collected data that contains ratings for a list of 55 books. You are provided with two files**: books.txt** and **ratings.txt.**

books.txt contains a list of books (one book per line):

```
Douglas Adams,The Hitchhiker's Guide To The Galaxy
Richard Adams,Watership Down
...
```

ratings.txt contains the book ratings provided by some users in the following format: the name of the user, followed by the row of numbers on the next line where the first number is the rating for the first book in books.txt, the second number is the rating for the second book in books.txt etc.:

```
Ben
5 0 0 0 0 0 0 1 0 1 -3 5 0 0 0 5 5 0 0 0 0 5 0 0 0 0 0 0 0 0 1 3 0 1 0 -5 0 0 5 5 0 5 5 5 0 5 5 0 0 0 5 5 5 5
Moose
5 5 0 0 0 0 3 0 0 1 0 5 3 0 5 0 3 3 5 0 0 0 0 0 5 0 0 0 0 0 3 5 0 0 0 0 0 5 -3 0 0 0 5 0 0 0 0 0 0 5 5 0 3 0
...
```

The ratings are separated by a white space. The ratings are in the range from -5 to 5, and should be interpreted as follows:

0 means the user did not read the book.
-5 means they strongly disliked it
-3 means they disliked it
3 means the liked the book
5 means they strongly liked it.

In homework 11, you will write a piece of code that asks the user to provide the ratings for all the books from books.txt, and saves this info into a file called **"profile.txt":**

```
Alice
2 0 0 0 0 0 0 1 0 1 -3 5 0 0 0 5 4 0 2 0 0 5 0 0 0 0 0 0 0 0 1 3 0 1 0 -5 0 0 5 5 0 5 5 5 0 5 5 0 4 0 5 5 5 5 0
```

The format is the same as ratings.txt except **profile.txt** contains book ratings for a single user.

Basic Requirements

Write a program that takes three data files: **books.txt, ratings.txt** and **profile.txt** (profile.txt will be created in homework 11), and first creates the following data structures:

- a list of books (from books.txt)

[2] Courtesy of Professor Michelle Craig.

- a list of user ratings (created from profile.txt)
- a dictionary where each key is a username from ratings.txt , and the value is the list of ratings for this user. For example, here is a (key, value) entry from the dictionary:

'Claire': [5, 3, 0, 0, 0, 0, 0, 5, 0, 3, 1, 0, 0, 0, 1, 0, 1, 0, 3, 0, 0, 0, 0, -3, 0, 5, 0, 0, 0, 0, 5, 5, 0, 1, 0, -5, 5, 0, 3, 3, 0, 5, 5, 5, 0, 0, 0, 0, 0, 5, 5, 0, 1, 0, 1]

Then, your program should compute the pair-wise similarity between a given user (whose data is in the list of user ratings) and all the other users from the dictionary. This should produce a list of lists, where each sublist contains the username and the corresponding similarity score, for instance:
[['Cust8', 170], ['Claire', 168], ['Tiffany', 165], ['Moose', 159], ['Megan', 142]]

Then your program should sort this list based on similarity, take the **top five** users who are "most similar" to a given user, and write a list of recommended books to a file called "recommendedBooks.txt" in the same format as books.txt (one recommended book per line). The recommended books are the ones that a given user has not read and that were rated well (with a rating of 5) by any of the five most similar users.

Extensions (each extension is worth 5% of the grade):

You need to implement at least one of these extensions to get 100% on the project.

1. Make your program object-oriented: write classes Book, Reader, Ratings etc.
2. Turn the program into a flask web application that lets the user enter their ratings for some of the books, and then displays a webpage with the list of recommended books.
3. Alternatively, you can use Tkinter to create a GUI for the system.
4. Come up with a more interesting recommendation algorithm (details will be posted soon).
5. Any other ideas you might have (discuss them with the instructor prior to implementation).

Tests:

When we grade your code, we will be running several tests on your code comparing the output of your program with the expected output. To pass these tests, you need to make sure the names of your files and the format of your files is exactly as specified in this handout.

Extra Credit:

Advanced students may choose to work on a harder project, a movie recommendation system, using a 100K data set from the MovieLens website:
http://grouplens.org/datasets/movielens/
Please talk to me if you want to work on this project.

Grading:

Completing the basic set of requirements will get you 95% of the grade for this project. To get the remaining 5%, you would need to add one or more of the suggested enhancements (or propose your own; you would need to discuss them with the instructor).

This project is worth 10% of your total grade. A student can not earn more than 5% of extra credit on the project.

You are not allowed to use any code from the web or collaborate on the project with anybody. You may receive help only from the instructor, the TAs or the CS tutors. I will randomly select several people from each section of cs110, and ask them to come for an interactive code-walkthrough. If you submit the code that you cannot explain to me during the code review, you will get a 0 for the project.

Submission: Save your file in **project4.py** . Thoroughly test your code before submitting it and **add comments to your code**. Upload all your files to Canvas.

Prize:

The names of 6 students with the highest score for the project will be entered in a lottery to win a small prize. The prize will be announced shortly.