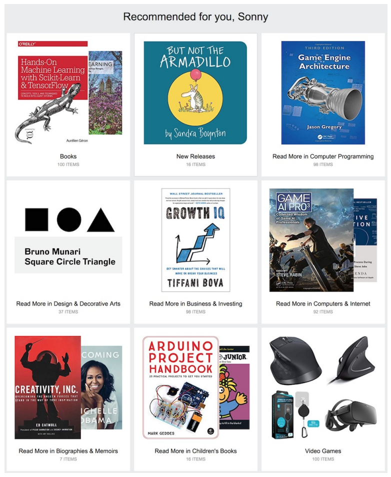
CPSC 231 ASSIGNMENT #3

You Might Also Like...


Recommended for you, Sonny
Books
New Releases
Read More in Computer Programming
Read More in Design & Decorative Arts
Read More in Business & Investing
Read More in Computers & Internet
Read More in Biographies & Memoirs
Read More in Children's Books
Video Games

This is what Amazon.ca believes your professor will likely purchase next.

Have you noticed that our computer systems are getting to know us almost better than we know ourselves? For example, if you watch television shows or movies on Netflix, they'll happily show you all the titles you haven't seen, but will probably like. Spotify's Discover will find music tracks for you that will end up being your new favourites. And if you shop on Amazon, they have a whole personalized virtual store for you filled with stuff you never knew you needed. Pretty scary, isn't it?

If you're under the impression that these companies are running very sophisticated machine learning and artificial intelligence algorithms to find recommendations for you, you are probably correct. However, you might be surprised at how effective a few simple pattern recognition and classification techniques can be in some situations. The goal of this assignment is to apply what we've learned about computing so far to explore the foundations behind such recommendation systems, using our very own class as a test sample.

The individual component will have you write a small program that will read a list of movies from a text file, and allow you to interactively give each one a rating based on how much you liked it. We will assemble all your individual ratings to create a large data file representing the movie tastes of our class. In the paired component, you will write a program that uses this data to provide individualized movie recommendations.

Due Dates

Individual component	Friday, October 12, 11:59 PM
Paired component	Friday, October 19, 11:59 PM

Individual Component

First, find the text file `cpsc231-movies.txt` which contains a curated list of movie titles that we'll use for this assignment. You can download it from the Assignments section of the course web page, or grab it from D2L. The main goal of this first part of the assignment is for you to write a program that will allow you to interactively give a rating to each title in the movie list.

You can use the exact same programming environment you set up for previous assignments to complete this assignment. Create and submit a separate Python program for each problem.

The file `cpsc231-movies.txt` contains movie titles formatted like this:

```
Wall-E (2008)
The Ring (2002)
Minions (2015)
The Mummy (1999)
...
```

Problem 1: Your Movie Ratings

Write a Python program that will read a list of movie titles from a file, `cpsc231-movies.txt`, and present the titles one at a time on the screen, asking for a rating for each one. First, ask the user for his or her name, then ask the user to rate each of the titles according to the scale shown on the right.

Use the following scale to rate the 100 movies in the list:

0	Never seen it.
1	It was terrible!
2	Didn't like it.
3	It was OK.
4	Liked it.
5	It was awesome!

Your program should write the final ratings for all the titles to a text output file once the interactive rating process is complete. You may name your output file however you'd like, but use a `.txt` file extension. The file should contain exactly two lines of text: line 1 should contain the name provided at the beginning of the run and line 2 should contain the ratings for all 100 movies, in the order presented and separated by commas. Please don't write a comma after the last rating, and ensure that the second line ends with a newline (`'\n'`) character!

When you are satisfied with your program, run it to give your own rating to each title in our movies list. Feel free to use a fake name, partial name, nickname, or simply write "anonymous" if you'd like to keep your movie preferences private. We thought it might be neat to potentially find movie buddies though this exercise if you're not embarrassed to include your name with your ratings. We will not divulge your real identity to your classmates nor anyone else through this assignment unless you volunteer it explicitly here.

Ensure that the output file created by your program, which contains your ratings, is formatted exactly as specified above when you are finished. Submit your ratings file along with your program for this problem.

Example Problem 1 output:

```
What's your name?  Anonymous

Hi, Anonymous!
Please tell us your favourite
movies using the following scale:
  0 = Never seen it.
  1 = It was terrible!
  2 = Didn't like it.
  3 = It was OK.
  4 = Liked it.
  5 = It was awesome!

Wall-E (2008)?  5
The Ring (2002)?  1
Minions (2015)?  3
The Mummy (1999)?  0
...

Thank you!
Your ratings were saved to
my-ratings.txt
```

Inputs: A file containing our list of movie titles, specified as a command line argument. For example, you might run your program with the following invocation:

```
$ python3 my-p1.py cpsc231-movies.txt
```

Outputs: An interactive prompt for each of the titles in the file specified, asking for a rating for each title, similar to that shown on the right. At the end of your program, the ratings should be saved to a text file in the exact format specified. The output file from this sample run (`my-ratings.txt`) would look like:

```
Anonymous
5,1,3,0,⟨ratings for the other 96 movies...⟩
```

Problem 2: Your Personal Summary

It's always a good idea to double-check your results before you submit them. The format of the output file makes it tricky to do by inspection, but we can create a program that both confirms some of your ratings while showing some interesting information.

Write a program that that takes as input our list of movies and a saved ratings file (the output of Problem 1), both specified as command line arguments, and prints the following to the screen:

1. The number of movies seen out of the total number of movies.
2. A list of favourite titles (with the highest possible rating).
3. A list of least favourite titles (with the lowest possible rating).

Example Problem 2 output:

```
Hello Anonymous!

It looks like you've seen 23 of
the 100 movies.

Your favourite movies were:
  Wall-E (2008)

Your least favourite movies were:
  The Ring (2002)
```

Inputs: The movie list file (`cpsc231-movies.txt`) and a saved ratings file, both specified by command line arguments.

```
$ python3 my-p2.py cpsc231-movies.txt my-ratings.txt
```

Outputs: Summary information from the ratings, presented in a form similar to that shown on the right.

Paired Component

We encourage you to work with a partner to complete the remainder of this assignment. You may choose your own partner, but remember that you must work with a *different partner* for each assignment in this class! If you are solving these problems as a pair, one submission is sufficient for both students.

The main goal of this part of the assignment is to create a program that will use all the data collected from the individual component to make individualized movie recommendations from a person's own movie tastes. We will compile all the individual ratings files into a class data set, `cpsc231-ratings.txt`,[1] that your programs can use to solve the two problems in this component. As before, create and submit a separate Python program for each problem that follows.

[1] This data file will obviously not be available until everyone has submitted their ratings from the individual component. However, you shouldn't let that stop you from proceeding with solving the problems in this part! If you're working as a pair, you should have two files that you can immediately combine. If you get a few friends to rate the movies list as well, you should have plenty to work with until the class data file is ready.

Problem 3: Class Statistics

Now that we have all the movie ratings from everyone in the class, it might be interesting to compute a few movie-watching statistics on our class as a whole. Write a program that will take an aggregate ratings file name as a command-line argument, open the file, read the data, and compute the following statistics:

Average number of movies seen. How many of our 100 movies, on average, has a student in our CPSC 231 class seen? Calculate the average number of non-zero ratings per individual in our data set.

Most popular movies. Which of our movies have been seen by the most students? Find the 5 movies that have the most number of non-zero ratings from our students.

Least popular movies. Which movies have the fewest of us seen before? Find the 5 movies that have the most number of zero ratings.

Highest rated movies. Which movies from the list are the "best" movies according to our class? Find the 5 movies that have the highest average rating from our class. Do not include 0 ratings (never seen it) when calculating the average and, to minimize outliers, only include movies that have at least 10 non-zero ratings.

Lowest rated movies. Which are the worst movies? Find the 5 movies that have the lowest average rating. Again, do not include 0 ratings and only include movies with at least 10 non-zero ratings.

Most contentious movies. Which movies are most likely to cause a debate within our class? Find the 5 movies that have the highest variance[2] in their ratings. Do not include 0 ratings when calculating the variance and, once again, only include movies with at least 10 non-zero ratings.

[2] Recall that *variance* is the square of the standard deviation, σ, and can be calculated on a discrete set of samples by the following formula:

$$\sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu)^2$$

N is the number of samples (ratings), x_i is the value of each individual sample, and μ is the mean sample value.

The format of the class ratings input file, `cpsc231-ratings.txt`, is the same as the output format of Problem 1, and its contents are simply all of your Problem 1 outputs concatenated together. The first line, and every odd line thereafter, will contain a movie rater's name. The second line, and every even line thereafter, will contain a comma-separated list of 100 ratings corresponding to our list of 100 movies. For example, the file may look like this:

```
Alice
5,4,1,0,⟨ratings for the other 96 movies...⟩
Bob
3,5,2,1,⟨ratings for the other 96 movies...⟩
Carol
5,2,5,3,⟨ratings for the other 96 movies...⟩
Dave
3,4,0,4,⟨ratings for the other 96 movies...⟩
```

Inputs: The movie list file and our class ratings file, both specified by command line arguments, that your program should read and compute statistics on. For example, you might run your program on the command prompt as follows:

```
$ python3 my-p3.py cpsc231-movies.txt cpsc231-ratings.txt
```

Outputs: Summary information from the class ratings, presented in a form similar to that shown on the right.

Example Problem 3 output:

```
The average student in CPSC 231
has seen 25.9 of the 100 movies.

The most popular movies were:
  <movie title>
  ...

The least popular movies were:
  <movie title>
  ...

The highest rated movies were:
  <movie title>
  ...

The lowest rated movies were:
  <movie title>
  ...

The most contentious movies were:
  <movie title>
  ...
```

Problem 4: Movie Recommendations

Now it's finally time to make some recommendations! If we have a list of ratings of our movies from a specific "customer", which movies might we recommend that he or she see next? The most basic and naïve approach would be to simply that the highest rated movies (as found in the previous problem) and recommend to the customer that ones that he or she has not seen (rated 0). While this approach does work reasonably well, it's very generic and essentially recommends the same movies to every customer.

We can do better by taking into account a customer's unique preferences as indicated by his or her ratings of movies that he or she has seen, and comparing these to the ratings of other customers. To understand how this might work, consider how you might decide on movie recommendations from your own friends. If a friend talks to you about movies that you both enjoyed watching, and he or she then recommends a movie you've never seen, you'd probably be inclined to go see it. If, on the other hand, you and another friend tend to disagree about movies, you're not likely going to run out and see the next movie he or she is raving about.

For this problem, we can use the following two steps to make individualized movie recommendations for a specific person, given his or her ratings of other titles:

1. Find the student from our class whose ratings are most similar to those of the person for whom we wish to make recommendations.

2. Find the highest-rated movies in the most similar student's set that the person has not seen, and recommend those titles.

The problem that remains is, what does it mean to be "most similar"? It turns out that if we were to try to compare ratings on our 0-5 scale directly, the results won't be very good. But if we re-assigned specific values to the ratings according to the table on the right, we can calculate a numeric similarity score between two sets of ratings as a sum of products.

Assign the following values to the ratings when calculating similarity:

Rating	Meaning	Value
0	Never seen it.	0
1	It was terrible!	-5
2	Didn't like it.	-3
3	It was OK.	1
4	Liked it.	3
5	It was awesome!	5

For example, let us consider just the first three titles in the example ratings from Problem 3. The values for Alice's ratings would be [5, 3, -5], Bob's [1, 5, -3], Carol's [5, -3, 5], and Dave's [1, 3, 0]. These specific values make it so that if we multiply two people's rating value for a specific movie, we get a positive score if their ratings agree, and a negative score if they don't. We can take the sum of these individual products as the total similarity score. Suppose we wanted to calculate how similar Alice's preferences are to those of Bob, Carol, and Dave. The scores would look like this:

$$\begin{array}{lll} \text{Alice to Bob} & (5 \times 1) + (3 \times 5) + (-5 \times -3) & = 35 \\ \text{Alice to Carol} & (5 \times 5) + (3 \times -3) + (-5 \times 5) & = -9 \\ \text{Alice to Dave} & (5 \times 1) + (3 \times 3) + (-5 \times 0) & = 14 \end{array}$$

Thus, Bob's preferences are most similar to Alice's, and we can use Bob's ratings to make recommendations to Alice.

Write a program that uses this strategy to make individualized movie recommendations for a someone who has specified his or her movie preferences as a set of ratings. This "someone" can of course be yourself, and you can certainly test how well your program works by using your own ratings file from Problem 1 as input. Note that you might very well find yourself as your own most similar rating, because your own ratings are included in the class data set! Then you would have no movies to recommend because there are no good movies you've seen that you haven't seen yourself. You may also happen to find a classmate with a "most similar" set of ratings who doesn't like any movies that you haven't seen already. In such cases, you should take the next-most similar rating set, and proceed until you manage to find at least one good recommendation.

You might already have noticed that the strategy prescribed is still far from optimal. For instance, it only uses one other person's preferences to make movie recommendations for a given set of ratings. You can imagine that if you were to use some combination of the many other ratings you have, you ought to get a better result.

Feel free to come up with a better strategy for making recommendations if you are not satisfied with the results. If you do use a different strategy, you must also submit a description of your strategy and why you believe it would generate superior results.

Note that this is not an "official" bonus of the assignment. However, if your improvements are substantial, your TA may use his or her discretion to award you a + grade.

Inputs: Your program for this problem will need to read the movies list file, the class ratings file, and an individual ratings file formatted as described in Problem 1. Specify the names of all three files, in this order, as command-line arguments so that your program may be invoked as follows:

```
$ python3 my-p4.py cpsc231-movies.txt cpsc231-ratings.txt my-ratings.txt
```

Outputs: A list of personalized movie recommendations for the person whose preferences are indicated in the ratings file supplied, similar to what is shown on the right. Ensure that your program makes at least one recommendation, and ideally no more than a dozen or so.

Example Problem 4 output:

```
Hello Anonymous!

From your ratings of our 100 movies,
our CPSC 231 class believes that you
might also like:
  Arrival (2016)
  The Shawshank Redemption (1994)
  King Arthur (2004)
```

Bonus Problem: Guess Your TAs' Ratings!

The bonus problem in this assignment is our own miniature version of the 2006 Netflix Prize.[3] Each of your Teaching Assistants has also assigned ratings to the same list of movies, but we did not include their ratings with the class data set. Instead, we randomly selected 30% of the movies each TA has seen and hid his or her ratings for those titles to provide you with a partial data set for this problem.

[3] In October 2006, Netflix started an open competition with a prize of one million dollars to be awarded to anyone who could come up with a movie rating prediction algorithm that can beat their own system, Cinematch, by at least 10%. The prize was won in 2009. Details of the contest can still be found at https://www.netflixprize.com

Your goal for the bonus problem is to use the partial ratings from each TA, combined with the information you have from the class ratings, to predict their missing ratings. You may use any strategy you'd like to make these predictions. One bonus will be awarded to the program that makes the most accurate predictions for each TA, for a total of 7 guaranteed bonuses. You can't double-dip, so if you've got the best prediction for more than one TA, we'll choose the 2nd or 3rd best, and so forth, as needed. We may also award bonuses to runners-up if the prediction accuracy is close.

The accuracy each of your predictions will be scored on our original rating scale as follows:

exact prediction	= 2 points
close prediction (1 rung difference)	= 1 point
bad prediction (3 rung difference)	= -1 point
terrible prediction (*e.g.* 1 vs 5)	= -2 points

Your total prediction score for each TA is the sum total of your prediction scores for each title that was hidden.

A partial ratings file for each TA will be provided for you in the same format as a regular ratings file (from Problem 1), except that hidden ratings will be indicated by a -1. For example, a partial ratings file may look like this:

```
Teaching Assistant #1
5,-1,3,0,⟨ratings for the other 96 movies...⟩
```

Your program should write your predictions to an output file with the exact same format, but replace all the -1 entries with your predictions of the TA's ratings.

Inputs: As before, your program should take the names of the movies list file and the class ratings file as its first two command-line arguments. The third and fourth arguments should be the name of the partial ratings file and the file to save your predictions to, respectively. An invocation may look like this:

```
$ python3 my-pb.py cpsc231-movies.txt cpsc231-ratings.txt ta-partial.txt ta-predicted.txt
```

Outputs: Predictions for the ratings that were hidden in the partial ratings file, written to the file indicated by the fourth argument in the same format as the input ratings. You do not need to print anything to the screen for this problem.

Java Bonus

We want to continue encouraging you to become a multi-lingual computer scientist. As with the previous assignment, we will award you another bonus if you submit correct Java programs for Problems 3 and 4, in addition to your Python programs.[4]

[4] Note that you should consider your Python programs to be your de facto solutions for this assignment. You will not get credit for solving these problems unless your Python solutions are correct.

You may set up your Java programs however you'd like, as long as it is convenient enough for your TA to run them on the CPSC lab machines. There exists a version of our course text that uses the Java programming language, titled *Computer Science: An Interdisciplinary Approach*.[5] It has corresponding support libraries that can be found at `https://introcs.cs.princeton.edu/java/home/`. If you're not sure where to start with Java, you may find it most convenient to follow the instructions from that booksite. You may also use functionality from their Java Booksite Library if that helps you to create Java-language equivalents of your Python programs.

[5] Robert Sedgewick and Kevin Wayne. *Computer Science: An Interdisciplinary Approach*. Addison-Wesley, 2016

Submission

When you've completed the individual component, submit your two Python programs (`.py` files) as well as the output text file (`.txt`) containing your own ratings of the movies in our list from Problem 1. After completing the paired component, submit your solutions to those problems as `.py` files as well. If you completed the Java Bonus, submit the source of your Java programs (`.java` files). Please ensure that your files are named descriptively, with the problem number included, so that your TA can easily see which program is associated with each problem.

Use the University of Calgary Desire2Learn system[6] to submit your assignment work online. Log in using your UofC eID and password, then find our course, CPSC 231 L01 and L02, in the list. Then navigate to Assessments → Dropbox Folders, and find the folder for Assignment #3 here. Upload your programs for problems 1 and 2 in the individual folder, and your other programs in the paired folder (if your partner hasn't already done so). One submission will suffice for each pair of students.

[6] `http://d2l.ucalgary.ca`

If you are using one or more of your grace "late days" for this assignment, indicate how many you used in the note accompanying your submission. Remember that late days will be counted against *both* partners in the paired component. If you completed any of the bonus problems, please indicate that here as well.

Credits

The mechanics of the personalized recommendation algorithm used in this assignment are adapted from an assignment developed by Michelle Craig at the University of Toronto. Their original specification had students rate books, but we thought that in this day and age, you're more likely to have seen a bunch of the movies in our list than read many of their books.