Intelligent Web System Assignment One

Student number: s3300154

Name: Ling Liu

1. Introduction to the project:

This assignment is implemented by Microsoft.NET Framework(C#). The development environment is Visual Studio 2010. It was tested on school's computer.

There are totally **nine** files in this project and the function for each file shows below:

StudentInfoInputForm.aspx: All the interface elements and layout is implemented in this file.

StudentInfoInputForm.aspx.cs: This is the main controller for this project. It receives and coordinates the whole operation of this system.

Student.cs: To use this class for storing the student information which read from student training data set, such as education background, sex, student type and GPA

Course.cs: To use this class for storing the course information which read from course training data, such as course ID, lecture ID, tutor ID etc.

StudentSimilarityCalculation_v3.cs: This class includes two functions for student similarity calculation. The one calculates attribute weight for the different student. The other calculates similarity score for each training student and also provides sorted results collection for further calculation.(Note: **StudentSimilarityCalculation_v6.cs** provides another implementation for these functions, but at the final version we choose the first algorithm. The reason will be explained later.)

CourseSimilarityCalculation_v2.cs: The course similarity calculation algorithm is implemented in this class.

SlopeOneCalculation.cs: This class implements the Slope One algorithm which is Item-based Rating-Based collaborative filtering algorithm.

CalculateCombinationSimilarity.cs: This class combines suggestion from different aspects to generate the final recommendation results.

2. Design and implementation

2.1 Introduction to basic idea of design

There are three major steps to implement this recommender system

Step 1. Using student's basic information(such as postgrad/undergrad,male/female etc) to find the most similar student(s), then we can recommend the courses which these similar students like to the new student.

Step 2. If new student provides course(s) information which he/she has done before, then we could use these information to predict the rating for other courses which he/she has not done. Then the courses which get the higher rating will be considered as candidate courses to new student.

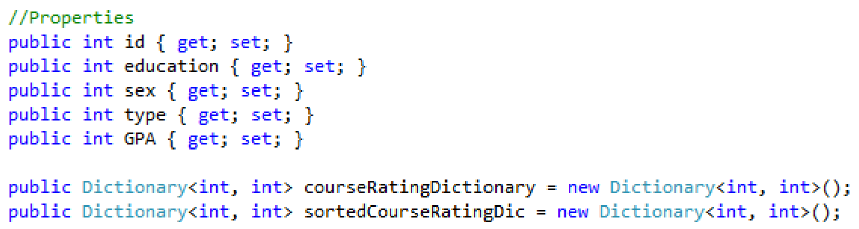
Step 3. At last step, we can combine previous recommendation results to generate more reasonable results.

2.2 Introduction to data structure

How to store student and course information:

In order to store such information, we firstly introduce two objects: student and course class.

Here is the main structure of **student class**:

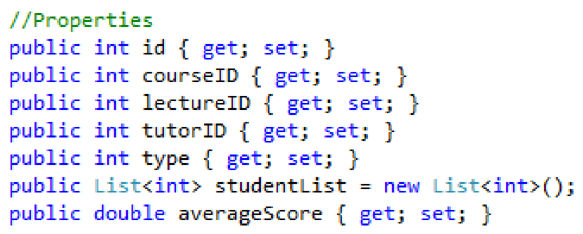


```
//Properties
public int id { get; set; }
public int education { get; set; }
public int sex { get; set; }
public int type { get; set; }
public int GPA { get; set; }

public Dictionary<int, int> courseRatingDictionary = new Dictionary<int, int>();
public Dictionary<int, int> sortedCourseRatingDic = new Dictionary<int, int>();
```

Each of properties corresponds to an attribute for each record which read from student.csv training data set. And "Dictionary<int, int> sortedCourseRatingDic" stores the course rating score by descending order given by current student which read from course.csv training data set.

Here is the main structure for **course class**:



```
//Properties
public int id { get; set; }
public int courseID { get; set; }
public int lectureID { get; set; }
public int tutorID { get; set; }
public int type { get; set; }
public List<int> studentList = new List<int>();
public double averageScore { get; set; }
```

Each of properties corresponds to an attribute for each record which read from course.csv file. **One important thing** has to be mentioned here is that we add one extra **id property** used for storing the unique id for each record in course.csv file. We will **always use this id attribute** to locate corresponding course object. It can be considered as the combination of course ID, lecture ID and tutor ID. The "List<int> studentList" attribute is used for storing all the rating score from different students. And the system uses "double averageScore" attribute to store average rating score for current course.

In the main controller **StudentInfoInputForm.aspx.cs** file, we use "Dictionary<int, Student> **studentDictionary**" and "Dictionary<int, Course> **courseDictionary**" to store all the information read from training data set. For studentDictionary, if we give the student ID as key value then we can get the reference which points to a Student object. For courseDictionary, if we give the course id as key value then we can get the reference which points to a Course object. This is a very efficient method to store and access student and course information.

How to store student similarity information

Our system uses "Dictionary<int, double> **sortedSimilarityResult**" structure to store the student similarity calculation results. The key part of the dictionary represents the student ID and the value part records the student similarity score. The dictionary is also sorted by similarity score as descending order.

How to store course similarity information

The system uses "Dictionary<int, double> **sortedSimilarityResult**" structure to store the course similarity calculation result. The key part of the dictionary represents the course id and the value part records the course similarity score. The dictionary is also sorted by similarity score as descending order.

How to store final course recommendation score information

The system uses "Dictionary<int, double> **sortedFinalResultDictionary**" structure to store the final course recommendation results. The key part of the dictionary represents the course id and the value part records the final course recommendation score. This dictionary is also sorted by recommendation score as descending order.

Summary: From previous description, we can see this project mainly chooses Dictionary object as our major data structure for storing and searching purpose. In Microsoft.NET Framework library, the Hashtable class and the Dictionary class implement the IDictionary interface. The Dictionary generic class also implements the IDictionary generic interface. Therefore, each element in these collections is a key-and-value pair. The Hashtable structure has a very good performance for storing and searching task. The Dictionary class has the same functionality as the Hashtable. A Dictionary of a specific type (other than Object) has better performance than a Hashtable for value types because the elements of Hashtable are of type Object and, therefore, boxing and unboxing typically occur if storing or retrieving a value type. So we choose Dictionary as our major data structure can get the very high performance for this implementation.

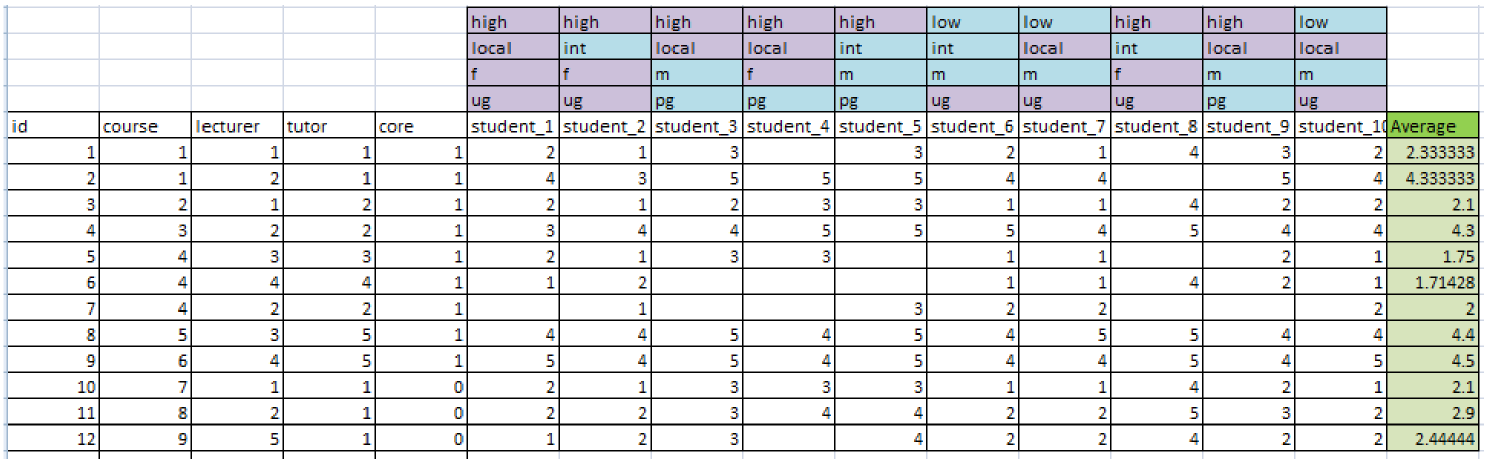
2.3 Introduction to important algorithms

How to calculate attribute weight

In this part, we need calculate two groups of weights for student and course table respectively.

For student table, we need to assign weight to pg/ug,m/f,int/local,GPA attribute. The algorithm describes as below:

First of all, we will combine both student and course table like this:



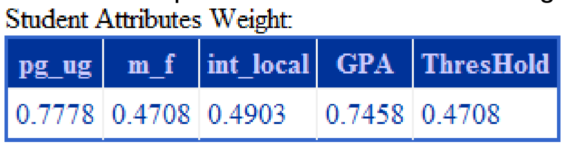
					high	high	high	high	high	low	low	high	high	low	
					local	int	local	local	int	int	local	int	local	local	
					f	f	m	f	m	m	m	f	m	m	
					ug	ug	pg	pg	pg	ug	ug	ug	pg	ug	
id	course	lecturer	tutor	core	student_1	student_2	student_3	student_4	student_5	student_6	student_7	student_8	student_9	student_10	Average
1	1	1	1	1	2	1	3		3	2	1	4	3	2	2.333333
2	1	2	1	1	4	3	5	5	5	4	4		5	4	4.333333
3	2	1	2	1	2	1	2	3	3	1	1	4	2	2	2.1
4	3	2	2	1	3	4	4	5	5	5	4	5	4	4	4.3
5	4	3	3	1	2	1	3	3		1	1		2	1	1.75
6	4	4	4	1	1	2				1	1	4	2	1	1.71428
7	4	2	2	1		1			3	2	2			2	2
8	5	3	5	1	4	4	5	4	5	4	5	5	4	4	4.4
9	6	4	5	1	5	4	5	4	5	4	4	5	4	5	4.5
10	7	1	1	0	2	1	3	3	3	1	1	4	2	1	2.1
11	8	2	1	0	2	2	3	4	4	2	2	5	3	2	2.9
12	9	5	1	0	1	2	3		4	2	2	4	2	2	2.44444

Figure -1 Explanation: Based on the course information, we add the student attribute information on the top and also calculate the average score for each course. Then the algorithm will try to calculate weight value for student's attribute. For example, If we try to work out the pg/ug weight value, the process shows below:

a. For each record, calculate the average score for both pg and ug attribute.
   Eg: for record 1: ug: (2+1+2+1+4+2) / 6 = 2 pg: (3+3+3)/2=3
b. Calculate the absolute value of difference between these two values:
   |2-3|=1

c. After repeating previous steps for each record, we can calculate average value for all the records using same attribute.

Then we can apply this process to all the attributes. The high value means that this attribute is more important than other attribute. Here we choose this average difference value as the weight for each attribute. The picture below shows the weight values which generated from training data set:



Student Attributes Weight:

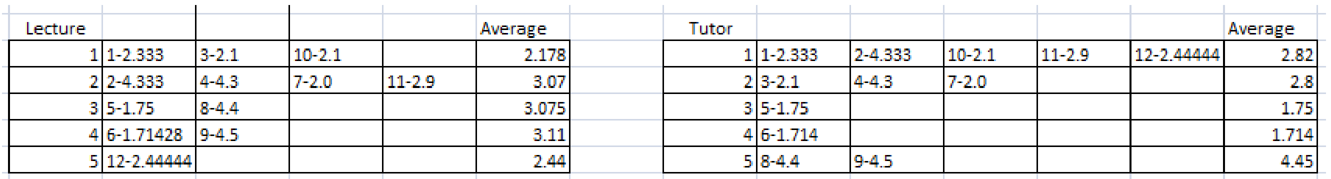
pg_ug	m_f	int_local	GPA	ThresHold
0.7778	0.4708	0.4903	0.7458	0.4708

Figure -2 Explanation: The first four columns show weight for each attribute. And we use last ThresHold attribute to calculate similarity score for each course. Only student whose similarity score is equal or larger than this score can be considered as a candidate. Then the program will try to find the most popular courses from these candidates. The value of ThresHold equals to the sum of two of the smallest weight values(If new student has not provided GPA, the ThresHold will be equal to the smallest attribute value).

Note: The **StudentSimilarityCalculation_v6.cs** file has implemented another algorithm for this part. The idea of this algorithm follows by the **Scope and Expectations for IWS Assignment 1** posted by Halil Ali. After doing some testing based on training data, we find the result is not as good as previous method(because we find some attribute weights are very low or equals to zero). So at the final version for this submission we use previous algorithm.

For course table, we need to assign weight to course_id,lecturer_id,tutor_id attribute. In this part, I haven't use any dynamic algorithm to calculate the weight for these attributes. I only assign some fix value for these attributes. The weight for each attribute: course_id=0.5, lecturer_id=0.3, tutor_id=0.2.

Here is the explanation why we choose this value for each attribute: From the **Figure -1**, we summaries the follow tables:



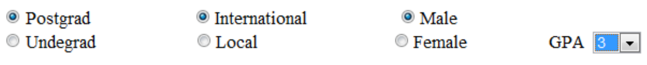
Lecture					Average
1	1-2.333	3-2.1	10-2.1		2.178
2	2-4.333	4-4.3	7-2.0	11-2.9	3.07
3	5-1.75	8-4.4			3.075
4	6-1.71428	9-4.5			3.11
5	12-2.44444				2.44

Tutor						Average
1	1-2.333	2-4.333	10-2.1	11-2.9	12-2.44444	2.82
2	3-2.1	4-4.3	7-2.0			2.8
3	5-1.75					1.75
4	6-1.714					1.714
5	8-4.4	9-4.5				4.45

From these two tables, it is very hard to say which attribute is more important than others. We can see that modifying only the lecture ID or Tutor ID has no big effect on overall ratings for some records. But from **Figure -1** we can see modifying the course ID has a large impact on the ratings. And we should also notice that: from the **first two records** which id equals 1 and 2, the lecture ID looks more important than tutor ID. Because modifying the lecture ID has a large impact on the rating. So the weighting on Course ID should be higher than the weighting on Lecture Id then tutor ID. In this part, if we have more training data, we may find better solution for calculating weight.

How to find out the most similar students for new student

From previous step, each attribute is assigned a weight value. Now we can compare input information with training data to generate similarity score for each student. Here is an example:

Input Student Information: 

[✓ Postgrad] Undegrad

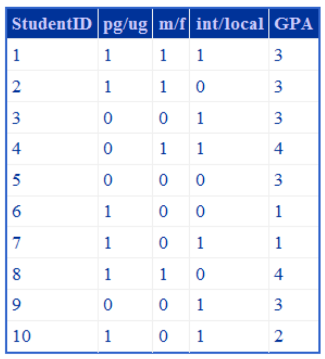
[✓ International] Local

[✓ Male] Female

GPA 3

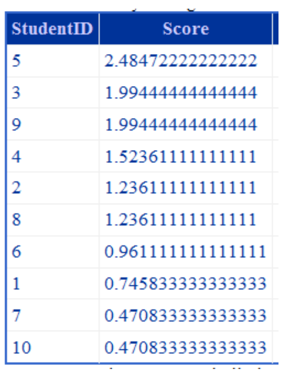
(0,0,0,3)

Training data set:

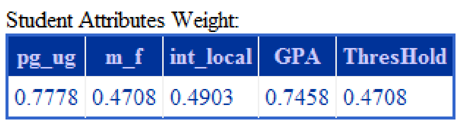


StudentID	pg/ug	m/f	int/local	GPA
1	1	1	1	3
2	1	1	0	3
3	0	0	1	3
4	0	1	1	4
5	0	0	0	3
6	1	0	0	1
7	1	0	1	1
8	1	1	0	4
9	0	0	1	3
10	1	0	1	2

The student similarity score:



StudentID	Score
5	2.48472222222222
3	1.99444444444444
9	1.99444444444444
4	1.52361111111111
2	1.23611111111111
8	1.23611111111111
6	0.961111111111111
1	0.745833333333333
7	0.470833333333333
10	0.470833333333333



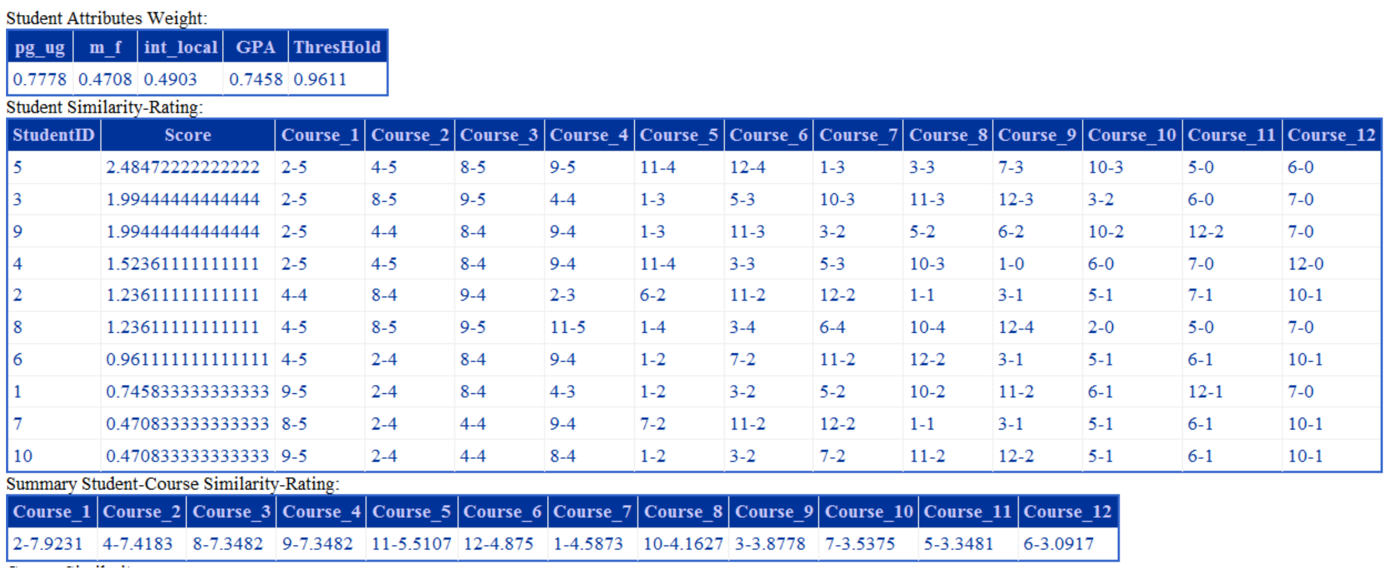
Student Attributes Weight:

pg_ug	m_f	int_local	GPA	ThresHold
0.7778	0.4708	0.4903	0.7458	0.4708

Figure -3 Explanation: the user provides vector: 0,0,0,3. The program will compare this vector with each record from training data. One thing has to be mentioned here is that we normalize the value of GPA as this rule: value 1, 2 convert to 0 and value 3, 4 convert to 1. For example, the normalized vector of student 5 is: 0,0,0,1. The normalized input vector is also: 0,0,0,1. The final similarity score for student 5 is: 0.7778+0.4708+0.4903+0.7458=2.4847. So according to the attribute weight value, we can calculate the similarity score for each student and sort them by descending order.

How to combine the recommendation results from most similar students

We will use an example to explain this process. The testing data shows below:



Student Attributes Weight:

pg_ug	m_f	int_local	GPA	ThresHold
0.7778	0.4708	0.4903	0.7458	0.9611

Student Similarity-Rating:

StudentID	Score	Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10	Course_11	Course_12
5	2.48472222222222	2-5	4-5	8-5	9-5	11-4	12-4	1-3	3-3	7-3	10-3	5-0	6-0
3	1.99444444444444	2-5	8-5	9-5	4-4	1-3	5-3	10-3	11-3	12-3	3-2	6-0	7-0
9	1.99444444444444	2-5	4-4	8-4	9-4	1-3	11-3	3-2	5-2	6-2	10-2	12-2	7-0
4	1.52361111111111	2-5	4-5	8-4	9-4	11-4	3-3	5-3	10-3	1-0	6-0	7-0	12-0
2	1.23611111111111	4-4	8-4	9-4	2-3	6-2	11-2	12-2	1-1	3-1	5-1	7-1	10-1
8	1.23611111111111	4-5	8-5	9-5	11-5	1-4	3-4	6-4	10-4	12-4	2-0	5-0	7-0
6	0.961111111111111	4-5	2-4	8-4	9-4	1-2	7-2	11-2	12-2	3-1	5-1	6-1	10-1
1	0.745833333333333	9-5	2-4	8-4	4-3	1-2	3-2	5-2	10-2	11-2	6-1	12-1	7-0
7	0.470833333333333	8-5	2-4	4-4	9-4	7-2	11-2	12-2	1-1	3-1	5-1	6-1	10-1
10	0.470833333333333	9-5	2-4	4-4	8-4	1-2	3-2	7-2	11-2	12-2	5-1	6-1	10-1

Summary Student-Course Similarity-Rating:

Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10	Course_11	Course_12
2-7.9231	4-7.4183	8-7.3482	9-7.3482	11-5.5107	12-4.875	1-4.5873	10-4.1627	3-3.8778	7-3.5375	5-3.3481	6-3.0917

Figure-5 Explanation: the first table shows student attributes weight and ThresHold value. The second table shows sorted similarity score for each student compared with new student by descending order. It also provides the rating value for each course for specific student. The last table shows the combined results from previous tables.

Here is the implementation for this algorithm:

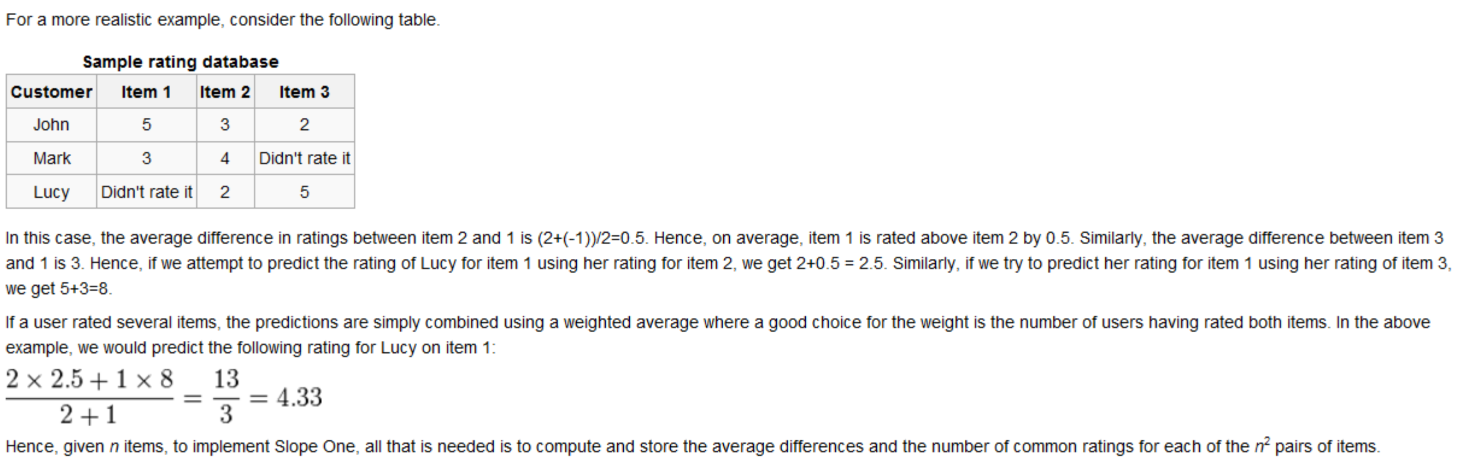
a. Choose all the students whose score are equal or larger than ThresHold.
b. Calculate the average rating value for each course from all the candidates(consider both score and rating value to work out the average rating value).
c. Sorted the final average rating score by descending order.

For example: how to calculate the score for course id=8

a. Given ThresHold: 0.9611, select student 5,3,9,4,2,8,6 as candidates.
b. the average rating value is calculated as below: (5*2.48+5*1.99+4*1.99+4*1.52+4*1.24+5*1.24+4*0.96)/7=7.348

How to predict course(s) rating according to provided course(s) information

Collaborative filtering is a technique used by many recommender systems. It aims to predict the ratings of one individual based on his past ratings and on a (large) database of ratings contributed by other users. We have learned two Collaborative filtering methods for rating prediction. The one is Amazon's item-to-item algorithm. The other is Slope One algorithm. Amazon's item-to-item algorithm computes the cosine between binary vectors representing the purchases in a user-item matrix. It can only use for binary data. Slope One algorithm implements Item-based Rating-Based collaborative filtering algorithms. It can use for rating value prediction. So Slope One algorithm is more suitable for this assignment. Here is an example:



For a more realistic example, consider the following table.

Sample rating database

Customer	Item 1	Item 2	Item 3
John	5	3	2
Mark	3	4	Didn't rate it
Lucy	Didn't rate it	2	5

In this case, the average difference in ratings between item 2 and 1 is (2+(-1))/2=0.5. Hence, on average, item 1 is rated above item 2 by 0.5. Similarly, the average difference between item 3 and 1 is 3. Hence, if we attempt to predict the rating of Lucy for item 1 using her rating for item 2, we get 2+0.5 = 2.5. Similarly, if we try to predict her rating for item 1 using her rating of item 3, we get 5+3=8.

If a user rated several items, the predictions are simply combined using a weighted average where a good choice for the weight is the number of users having rated both items. In the above example, we would predict the following rating for Lucy on item 1:

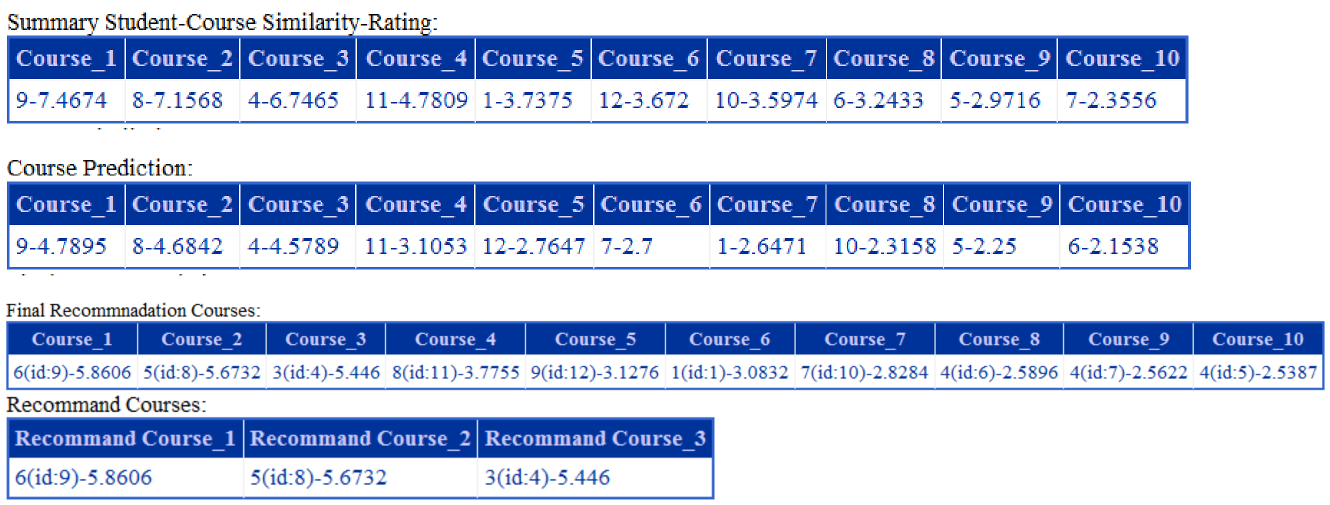
$$\frac{2 \times 2.5 + 1 \times 8}{2+1} = \frac{13}{3} = 4.33$$

Hence, given n items, to implement Slope One, all that is needed is to compute and store the average differences and the number of common ratings for each of the n^2 pairs of items.

Figure -4 Explanation: This example is from Wikipedia and the implementation for our algorithm follows this algorithm.

How to combine student and course recommendation to generate final results

In order to generate final results, our solution uses weighting mechanism to combine both student and course recommendation results. The system assigns course weight to 0.6 and student weight to 0.4. That's because we assume course recommendation is more important than student recommendation. Then based on weight value and similarity score, we can calculate the final result. Here is an example:



Summary Student-Course Similarity-Rating:

Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10
9-7.4674	8-7.1568	4-6.7465	11-4.7809	1-3.7375	12-3.672	10-3.5974	6-3.2433	5-2.9716	7-2.3556

Course Prediction:

Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10
9-4.7895	8-4.6842	4-4.5789	11-3.1053	12-2.7647	7-2.7	1-2.6471	10-2.3158	5-2.25	6-2.1538

Final Recommnadation Courses:

Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10
6(id:9)-5.8606	5(id:8)-5.6732	3(id:4)-5.446	8(id:11)-3.7755	9(id:12)-3.1276	1(id:1)-3.0832	7(id:10)-2.8284	4(id:6)-2.5896	4(id:7)-2.5622	4(id:5)-2.5387

Recommand Courses:

Recommand Course_1	Recommand Course_2	Recommand Course_3
6(id:9)-5.8606	5(id:8)-5.6732	3(id:4)-5.446

Figure-6 Explanation: the first two tables show student and course recommendation results respectively(**data format** such as “9-7.476” means record id 9 in course.csv file is assigned similarity score 7.476). The third table shows the combination results from the first two tables. The algorithm follows previous introduction. For example, record “6(id:9)-5.8606” means record 9 which course ID is course 6 is assigned similarity score 5.8606(7.4674*0.4+4.7895*0.6=5.8606). The last table displays the top three courses which get the higher score to new student.

How to implement Bonus task Part

The rule for prerequisites is that a recommended subject cannot have a lower id than the highest completed course id. For example:

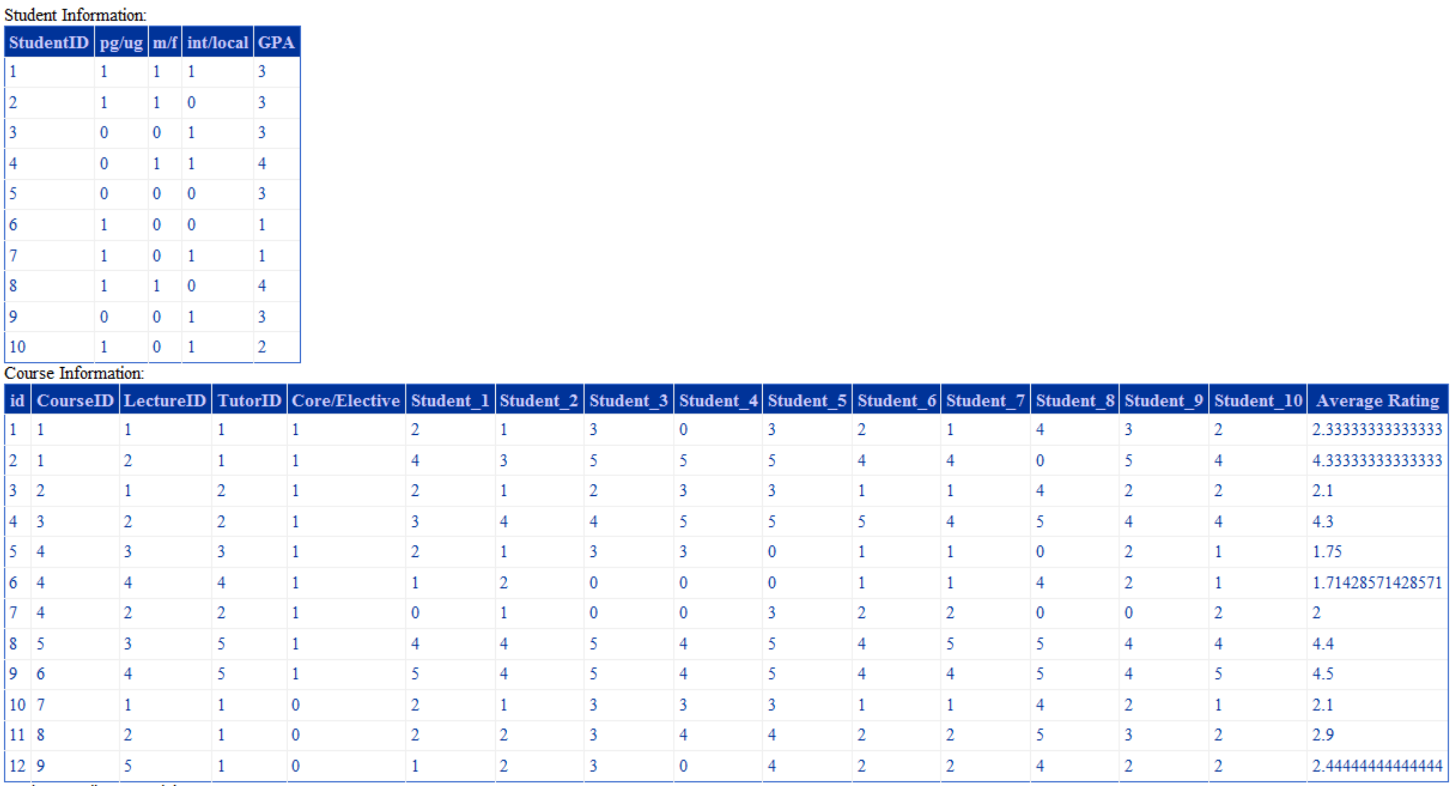
a. If a student has completed courses 1 and 4, then they cannot be recommended courses 2 or 3 as these are both lower than 4 (the highest completed course)
b. If they have done 1 and 3 then they cannot be recommended course 2 as this is lower than 3 (the highest completed course)
c. Of course a student cannot be recommended a course that they have previously completed.

The implementation for this part is:

a. The program will store the highest course ID from user inputs course information.
b. Program works out the final recommendation result which is stored in a sorted dictionary object.
c. Each element in sorted dictionary will be compared with the highest input course ID, only the course which has larger course ID can be added into final recommendation result list and completed courses cannot be added into final recommendation result list (we choose only three courses for this implementation).

3. Performance analysis

In this part, we will use a group of testing cases to demonstrate the performance for our system. The following two tables show the information read from training data(student.csv and course.csv). All other intermediate tables will provide later:



Student Information:

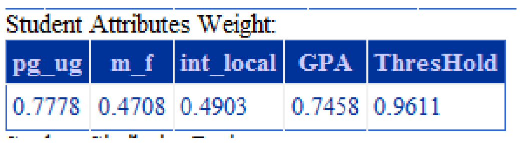
StudentID	pg/ug	m/f	int/local	GPA
1	1	1	1	3
2	1	1	0	3
3	0	0	1	3
4	0	1	1	4
5	0	0	0	3
6	1	0	0	1
7	1	0	1	1
8	1	1	0	4
9	0	0	1	3
10	1	0	1	2

Course Information:

id	CourseID	LectureID	TutorID	Core/Elective	Student_1	Student_2	Student_3	Student_4	Student_5	Student_6	Student_7	Student_8	Student_9	Student_10	Average Rating
1	1	1	1	1	2	1	3	0	3	2	1	4	3	2	2.33333333333333
2	1	2	1	1	4	3	5	5	5	4	4	0	5	4	4.33333333333333
3	2	1	2	1	2	1	2	3	3	1	1	4	2	2	2.1
4	3	2	2	1	3	4	4	5	5	5	4	5	4	4	4.3
5	4	3	3	1	2	1	3	3	0	1	1	0	2	1	1.75
6	4	4	4	1	1	2	0	0	0	1	1	4	2	1	1.71428571428571
7	4	2	2	1	0	1	0	0	3	2	2	0	0	2	2
8	5	3	5	1	4	4	5	4	5	4	5	5	4	4	4.4
9	6	4	5	1	5	4	5	4	5	4	4	5	4	5	4.5
10	7	1	1	0	2	1	3	3	3	1	1	4	2	1	2.1
11	8	2	1	0	2	2	3	4	4	2	2	5	3	2	2.9
12	9	5	1	0	1	2	3	0	4	2	2	4	2	2	2.44444444444444

Figure-7 Explanation: these two tables show the information read from student.csv and course.csv file. In the course information table, we add one column which names Average Rating, it shows average score for each record.

Based on these two tables, we can calculate student attributes weight which shows below:



Student Attributes Weight:

pg_ug	m_f	int_local	GPA	ThresHold
0.7778	0.4708	0.4903	0.7458	0.9611

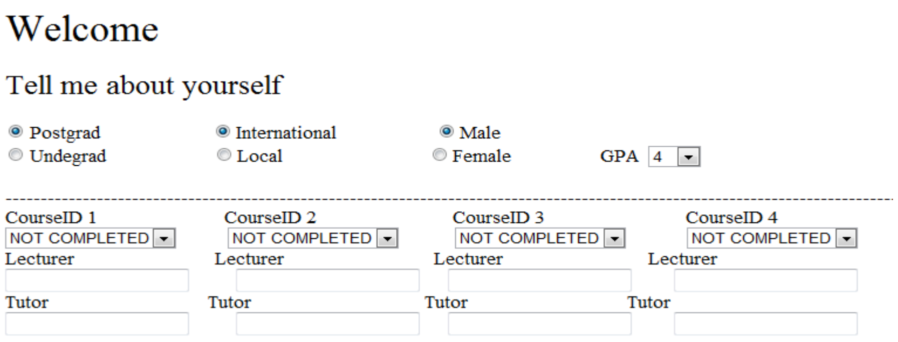
All the further testing calculation is based on these tables.

Testing 1: only provide student basic information

Input:

Student Information: 0,0,0,4 (undergrad, female, local, GPA)

Input shows below:



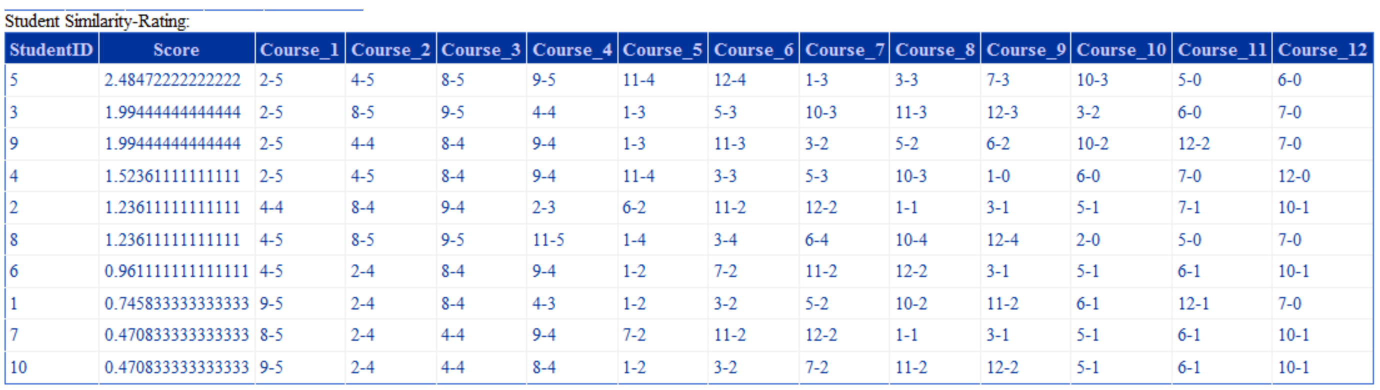
Welcome

Tell me about yourself

● Postgrad	● International	● Male	
○ Undegrad	○ Local	○ Female	GPA 4

CourseID 1	CourseID 2	CourseID 3	CourseID 4
NOT COMPLETED	NOT COMPLETED	NOT COMPLETED	NOT COMPLETED
Lecturer	Lecturer	Lecturer	Lecturer
Tutor	Tutor	Tutor	Tutor

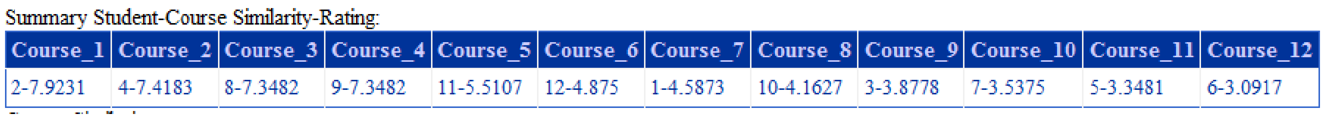
Intermediate results show below:



Student Similarity-Rating:

StudentID	Score	Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10	Course_11	Course_12
5	2.48472222222222	2-5	4-5	8-5	9-5	11-4	12-4	1-3	3-3	7-3	10-3	5-0	6-0
3	1.99444444444444	2-5	8-5	9-5	4-4	1-3	5-3	10-3	11-3	12-3	3-2	6-0	7-0
9	1.99444444444444	2-5	4-4	8-4	9-4	1-3	11-3	3-2	5-2	6-2	10-2	12-2	7-0
4	1.52361111111111	2-5	4-5	8-4	9-4	11-4	3-3	5-3	10-3	1-0	6-0	7-0	12-0
2	1.23611111111111	4-4	8-4	9-4	2-3	6-2	11-2	12-2	1-1	3-1	5-1	7-1	10-1
8	1.23611111111111	4-5	8-5	9-5	11-5	1-4	3-4	6-4	10-4	12-4	2-0	5-0	7-0
6	0.961111111111111	4-5	2-4	8-4	9-4	1-2	7-2	11-2	12-2	3-1	5-1	6-1	10-1
1	0.745833333333333	9-5	2-4	8-4	4-3	1-2	3-2	5-2	10-2	11-2	6-1	12-1	7-0
7	0.470833333333333	8-5	2-4	4-4	9-4	7-2	11-2	12-2	1-1	3-1	5-1	6-1	10-1
10	0.470833333333333	9-5	2-4	4-4	8-4	1-2	3-2	7-2	11-2	12-2	5-1	6-1	10-1

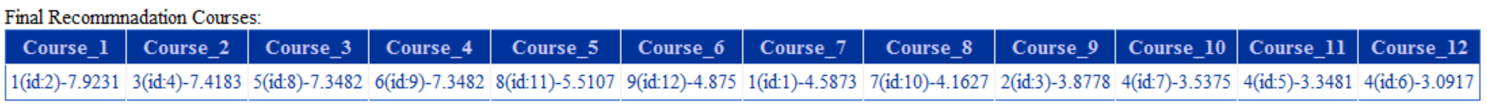
This table shows sorted similarity score for each student compared with new student by descending order. It also provides the rating value for each course for specific student.



Summary Student-Course Similarity-Rating:

Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10	Course_11	Course_12
2-7.9231	4-7.4183	8-7.3482	9-7.3482	11-5.5107	12-4.875	1-4.5873	10-4.1627	3-3.8778	7-3.5375	5-3.3481	6-3.0917

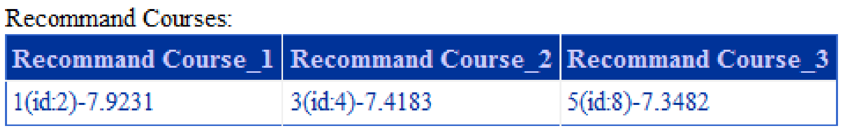
This table shows the combined calculation results from previous table.



Final Recommnadation Courses:

Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10	Course_11	Course_12
1(id:2)-7.9231	3(id:4)-7.4183	5(id:8)-7.3482	6(id:9)-7.3482	8(id:11)-5.5107	9(id:12)-4.875	1(id:1)-4.5873	7(id:10)-4.1627	2(id:3)-3.8778	4(id:7)-3.5375	4(id:5)-3.3481	4(id:6)-3.0917

This table provides the sorted final recommendation course with its score. In this situation, the user only provides student information, so we don’t need consider course information.


Recommand Courses:

Recommand Course_1	Recommand Course_2	Recommand Course_3
1(id:2)-7.9231	3(id:4)-7.4183	5(id:8)-7.3482

The final results select the top three courses from previous table. In this situation, we don’t need consider course rating history or the rule for prerequisites. Because new student only provides student information. We can see these three course are the most popular courses(look the average score) and they are also get the higher rating from the most similar students. So the algorithm makes good recommendations for this test.

Testing 2: provide both student and courses information

Testing 2.1

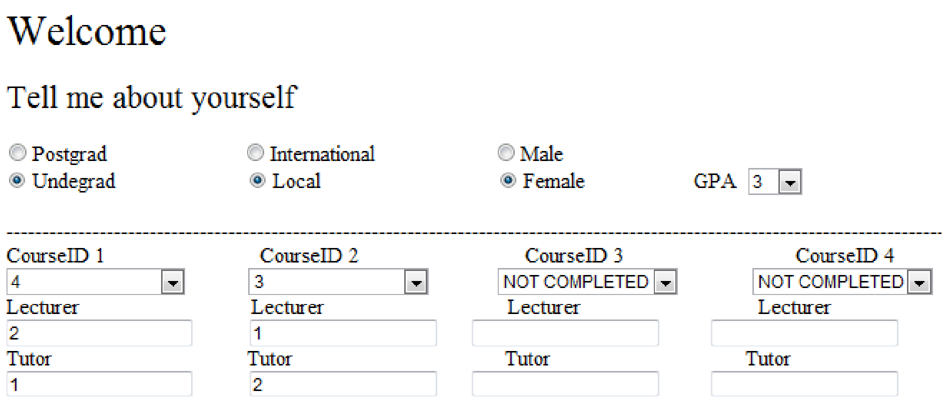
Input:

Student Information: 1,1,1,3 (undergrad, female, local, GPA)

Course information : 1,2,1-4 (course ID, lecturer ID, tutor ID, Rating)

2,1,2-3

Input shows below:

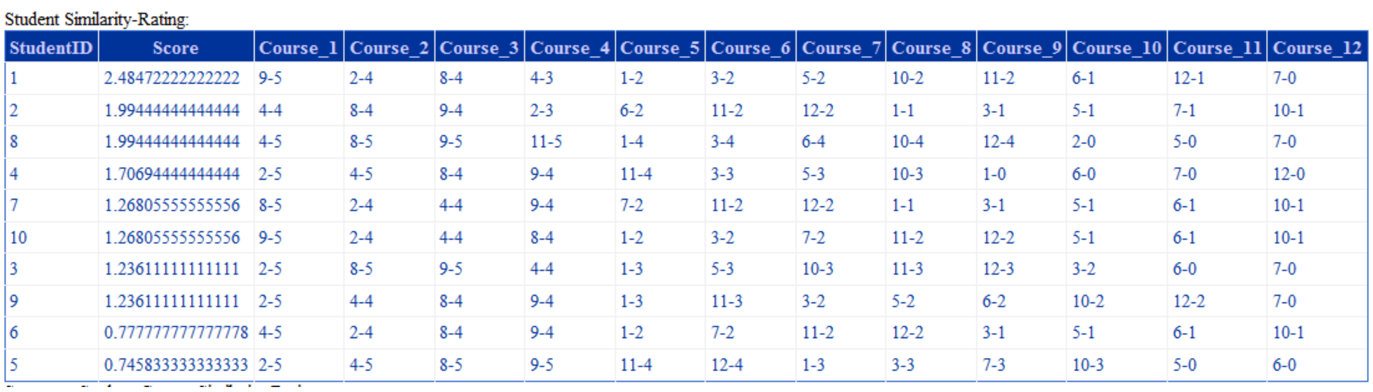

Welcome

Tell me about yourself

Postgrad [✓ Undegrad]   International [✓ Local]   Male [✓ Female]   GPA 3

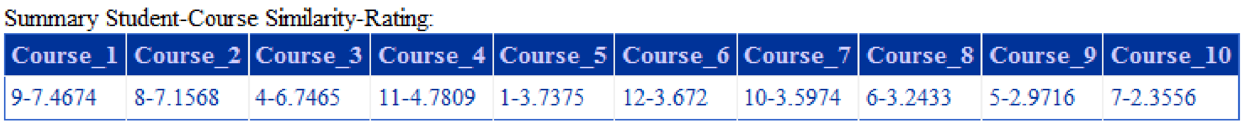
CourseID 1	CourseID 2	CourseID 3	CourseID 4
4	3	NOT COMPLETED	NOT COMPLETED
Lecturer	Lecturer	Lecturer	Lecturer
2	1		
Tutor	Tutor	Tutor	Tutor
1	2		

Intermediate results show below:


Student Similarity-Rating:

StudentID	Score	Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10	Course_11	Course_12
1	2.48472222222222	9-5	2-4	8-4	4-3	1-2	3-2	5-2	10-2	11-2	6-1	12-1	7-0
2	1.99444444444444	4-4	8-4	9-4	2-3	6-2	11-2	12-2	1-1	3-1	5-1	7-1	10-1
8	1.99444444444444	4-5	8-5	9-5	11-5	1-4	3-4	6-4	10-4	12-4	2-0	5-0	7-0
4	1.70694444444444	2-5	4-5	8-4	9-4	11-4	3-3	5-3	10-3	1-0	6-0	7-0	12-0
7	1.26805555555556	8-5	2-4	4-4	9-4	7-2	11-2	12-2	1-1	3-1	5-1	6-1	10-1
10	1.26805555555556	9-5	2-4	4-4	8-4	1-2	3-2	7-2	11-2	12-2	5-1	6-1	10-1
3	1.23611111111111	2-5	8-5	9-5	4-4	1-3	5-3	10-3	11-3	12-3	3-2	6-0	7-0
9	1.23611111111111	2-5	4-4	8-4	9-4	1-3	11-3	3-2	5-2	6-2	10-2	12-2	7-0
6	0.777777777777778	4-5	2-4	8-4	9-4	1-2	7-2	11-2	12-2	3-1	5-1	6-1	10-1
5	0.745833333333333	2-5	4-5	8-5	9-5	11-4	12-4	1-3	3-3	7-3	10-3	5-0	6-0

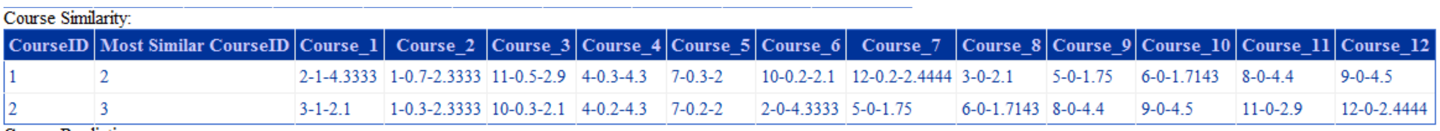
This table shows sorted similarity score for each student compared with new student by descending order. It also provides the rating value for each course for specific student.



Summary Student-Course Similarity-Rating:

Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10
9-7.4674	8-7.1568	4-6.7465	11-4.7809	1-3.7375	12-3.672	10-3.5974	6-3.2433	5-2.9716	7-2.3556

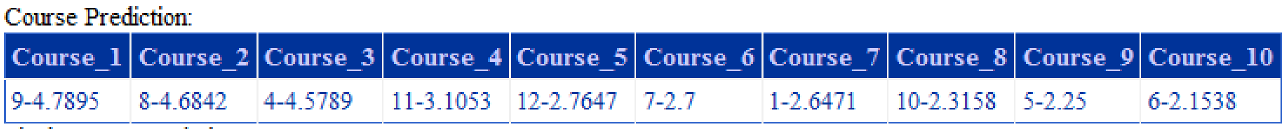
This table shows the combined calculation results from previous table.



Course Similarity:

CourseID	Most Similar CourseID	Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10	Course_11	Course_12
1	2	2-1-4.3333	1-0.7-2.3333	11-0.5-2.9	4-0.3-4.3	7-0.3-2	10-0.2-2.1	12-0.2-2.4444	3-0-2.1	5-0-1.75	6-0-1.7143	8-0-4.4	9-0-4.5
2	3	3-1-2.1	1-0.3-2.3333	10-0.3-2.1	4-0.2-4.3	7-0.2-2	2-0-4.3333	5-0-1.75	6-0-1.7143	8-0-4.4	9-0-4.5	11-0-2.9	12-0-2.4444

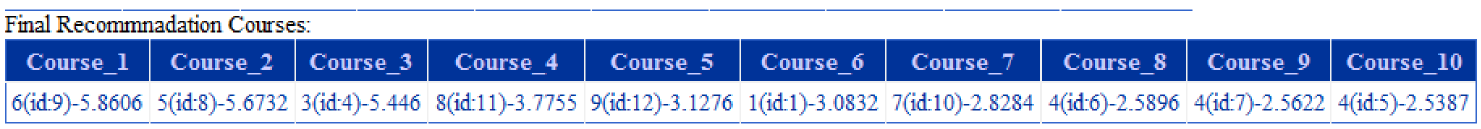
In this table, the first column **Course ID** means the input course ID from interface. The second column **Most Similar Course ID** means the program advises the most similar record id from course.csv file. For example, the record which id equals 2 in course.csv file is the most similar course for course 1. And the rest of the columns show the similarity score and average rating score for each course based on current user input course ID.



Course Prediction:

Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10
9-4.7895	8-4.6842	4-4.5789	11-3.1053	12-2.7647	7-2.7	1-2.6471	10-2.3158	5-2.25	6-2.1538

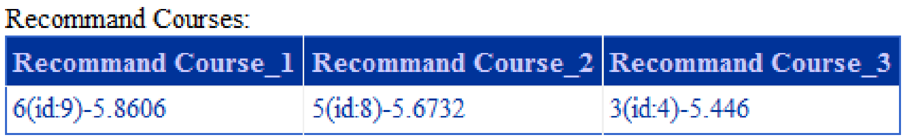
This table shows the sorted course prediction results after applying **Slope One algorithm** on training data.



Final Recommnadation Courses:

Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10
6(id:9)-5.8606	5(id:8)-5.6732	3(id:4)-5.446	8(id:11)-3.7755	9(id:12)-3.1276	1(id:1)-3.0832	7(id:10)-2.8284	4(id:6)-2.5896	4(id:7)-2.5622	4(id:5)-2.5387

This table provides the sorted final recommendation course with its score. In this situation, the program will combine both student recommendation results and course prediction results to generate this table.



Recommand Courses:

Recommand Course_1	Recommand Course_2	Recommand Course_3
6(id:9)-5.8606	5(id:8)-5.6732	3(id:4)-5.446

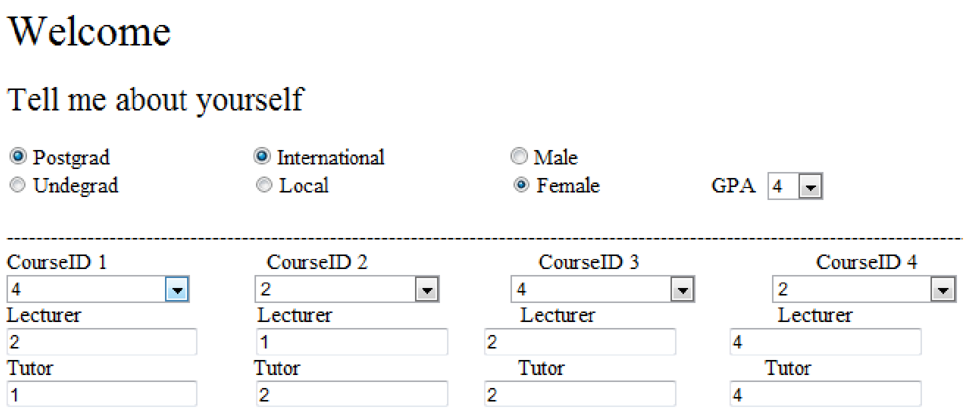
The final results select the top three courses which meet the needs of the rule for prerequisites from previous table. We can see these three courses are the most popular courses(look the average score) and they are also get the higher rating from the most similar students. And course prediction algorithm also gives the higher score for these courses. So the algorithm makes good recommendations for this test.

Testing 2.2

Input:

Student Information:	0,1,0,4 (undergrad, female, local, GPA)
Course information :	1,2,1-4 (course ID, lecturer ID, tutor ID, Rating)
	2,1,2-2
	3,2,2-4
	4,4,4-2

Input shows below:



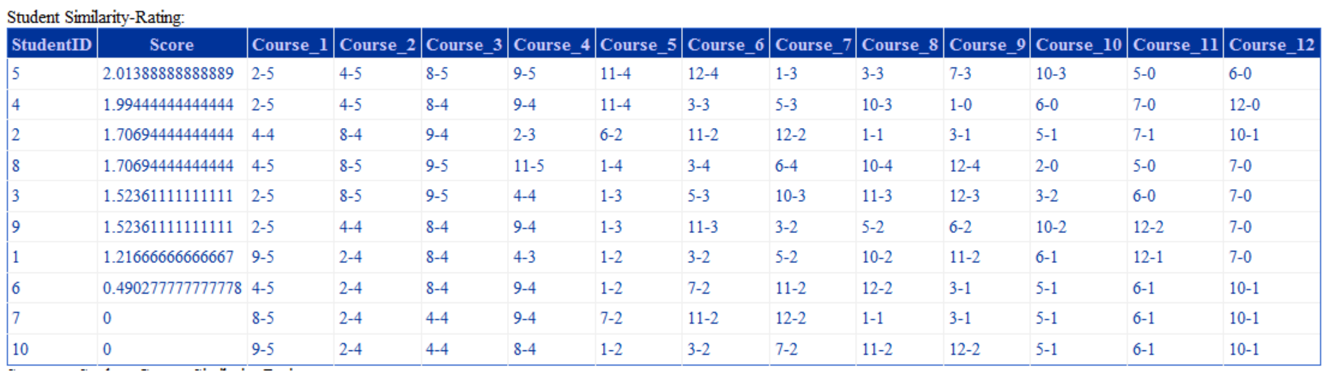
Welcome

Tell me about yourself

[✓ Postgrad]	[✓ International]	Male	
Undegrad	Local	[✓ Female]	GPA 4

CourseID 1	CourseID 2	CourseID 3	CourseID 4
4	2	4	2
Lecturer	Lecturer	Lecturer	Lecturer
2	1	2	4
Tutor	Tutor	Tutor	Tutor
1	2	2	4

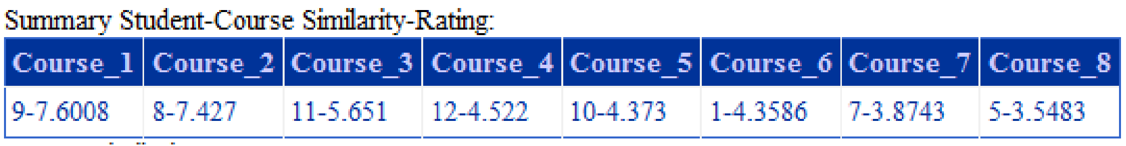
Intermediate results show below:



Student Similarity-Rating:

StudentID	Score	Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10	Course_11	Course_12
5	2.01388888888889	2-5	4-5	8-5	9-5	11-4	12-4	1-3	3-3	7-3	10-3	5-0	6-0
4	1.99444444444444	2-5	4-5	8-4	9-4	11-4	3-3	5-3	10-3	1-0	6-0	7-0	12-0
2	1.70694444444444	4-4	8-4	9-4	2-3	6-2	11-2	12-2	1-1	3-1	5-1	7-1	10-1
8	1.70694444444444	4-5	8-5	9-5	11-5	1-4	3-4	6-4	10-4	12-4	2-0	5-0	7-0
3	1.52361111111111	2-5	8-5	9-5	4-4	1-3	5-3	10-3	11-3	12-3	3-2	6-0	7-0
9	1.52361111111111	2-5	4-4	8-4	9-4	1-3	11-3	3-2	5-2	6-2	10-2	12-2	7-0
1	1.21666666666667	9-5	2-4	8-4	4-3	1-2	3-2	5-2	10-2	11-2	6-1	12-1	7-0
6	0.490277777777778	4-5	2-4	8-4	9-4	1-2	7-2	11-2	12-2	3-1	5-1	6-1	10-1
7	0	8-5	2-4	4-4	9-4	7-2	11-2	12-2	1-1	3-1	5-1	6-1	10-1
10	0	9-5	2-4	4-4	8-4	1-2	3-2	7-2	11-2	12-2	5-1	6-1	10-1

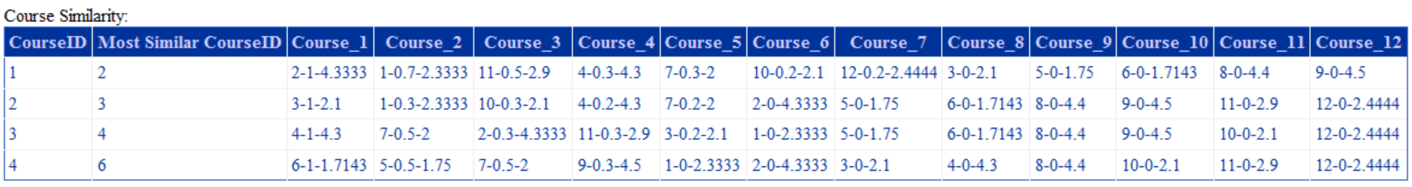
This table shows sorted similarity score for each student compared with new student by descending order. It also provides the rating value for each course for specific student.



Summary Student-Course Similarity-Rating:

Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8
9-7.6008	8-7.427	11-5.651	12-4.522	10-4.373	1-4.3586	7-3.8743	5-3.5483

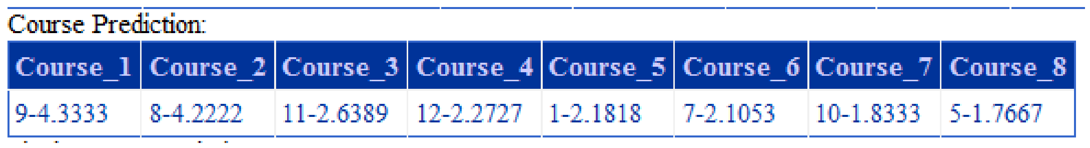
This table shows the combined calculation results from previous table.



Course Similarity:

CourseID	Most Similar CourseID	Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10	Course_11	Course_12
1	2	2-1-4.3333	1-0.7-2.3333	11-0.5-2.9	4-0.3-4.3	7-0.3-2	10-0.2-2.1	12-0.2-2.4444	3-0-2.1	5-0-1.75	6-0-1.7143	8-0-4.4	9-0-4.5
2	3	3-1-2.1	1-0.3-2.3333	10-0.3-2.1	4-0.2-4.3	7-0.2-2	2-0-4.3333	5-0-1.75	6-0-1.7143	8-0-4.4	9-0-4.5	11-0-2.9	12-0-2.4444
3	4	4-1-4.3	7-0.5-2	2-0.3-4.3333	11-0.3-2.9	3-0.2-2.1	1-0-2.3333	5-0-1.75	6-0-1.7143	8-0-4.4	9-0-4.5	10-0-2.1	12-0-2.4444
4	6	6-1-1.7143	5-0.5-1.75	7-0.5-2	9-0.3-4.5	1-0-2.3333	2-0-4.3333	3-0-2.1	4-0-4.3	8-0-4.4	10-0-2.1	11-0-2.9	12-0-2.4444

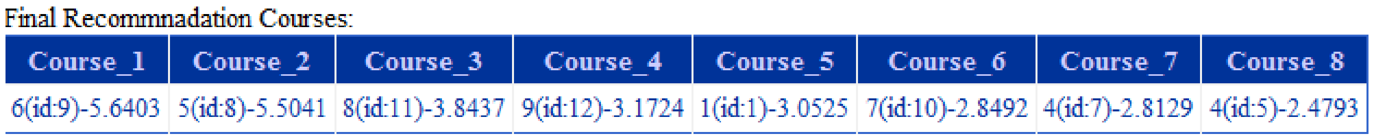
In this table, the first column **Course ID** means the input course ID from interface. The second column **Most Similar Course ID** means the program advises the most similar record id from course.csv file. For example, the record which id equals 2 in course.csv file is the most similar course for course 1. And the rest of the columns show the similarity score and average rating score for each course based on current user input course ID.



Course Prediction:

Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8
9-4.3333	8-4.2222	11-2.6389	12-2.2727	1-2.1818	7-2.1053	10-1.8333	5-1.7667

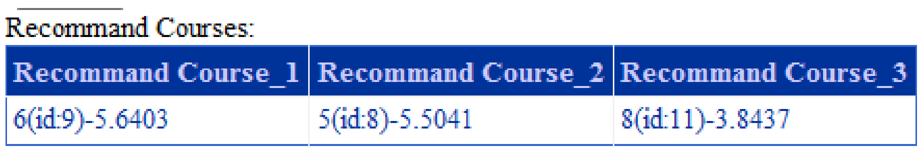
This table shows the sorted course prediction results after applying **Slope One algorithm** on training data.



Final Recommnadation Courses:

Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8
6(id:9)-5.6403	5(id:8)-5.5041	8(id:11)-3.8437	9(id:12)-3.1724	1(id:1)-3.0525	7(id:10)-2.8492	4(id:7)-2.8129	4(id:5)-2.4793

This table provides the sorted final recommendation course with its score. In this situation, the program will combine both student recommendation results and course prediction results to generate this table.



Recommand Courses:

Recommand Course_1	Recommand Course_2	Recommand Course_3
6(id:9)-5.6403	5(id:8)-5.5041	8(id:11)-3.8437

The final results select the top three courses which meet the needs of the rule for prerequisites from previous table. We can see these three courses are the most popular courses(look the average score) and they are also get the higher rating from the most similar students. And course prediction algorithm also gives the higher score for these courses. So the algorithm makes good recommendations for this test.

Testing 2.3

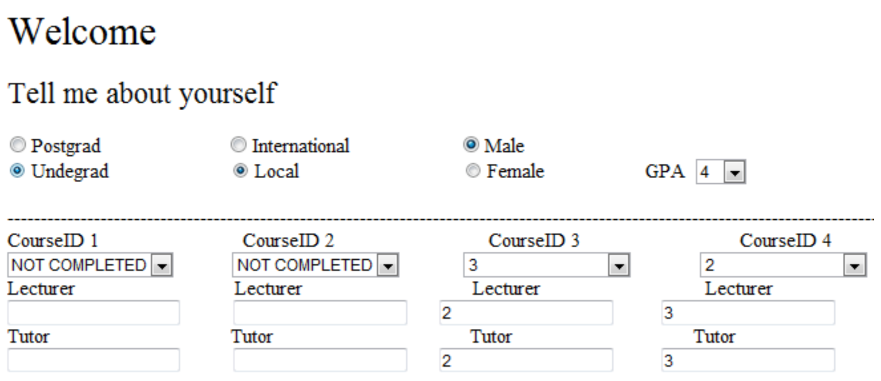
Input:

Student Information: 1,0,1,4 (undergrad, female, local, GPA)

Course information : 3,2,2-3 (course ID, lecturer ID, tutor ID, Rating)

4,3,3-2

Input shows below:



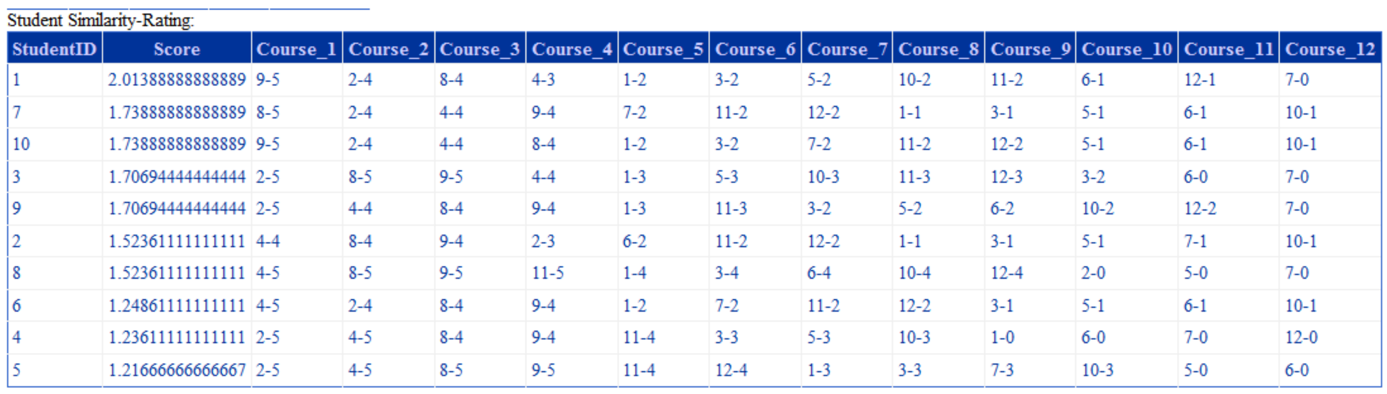
Welcome

Tell me about yourself

Postgrad	International	[✓ Male]	
[✓ Undegrad]	[✓ Local]	Female	GPA 4

CourseID 1	CourseID 2	CourseID 3	CourseID 4
NOT COMPLETED	NOT COMPLETED	3	2
Lecturer	Lecturer	Lecturer	Lecturer
		2	3
Tutor	Tutor	Tutor	Tutor
		2	3

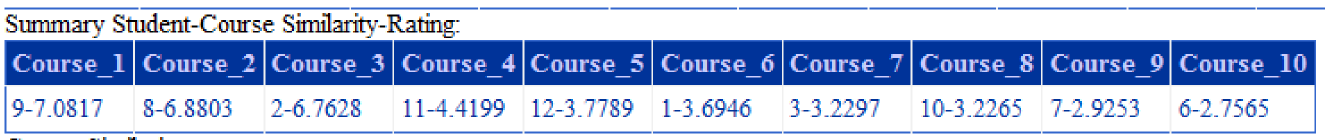
Intermediate results show below:



Student Similarity-Rating:

StudentID	Score	Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10	Course_11	Course_12
1	2.01388888888889	9-5	2-4	8-4	4-3	1-2	3-2	5-2	10-2	11-2	6-1	12-1	7-0
7	1.73888888888889	8-5	2-4	4-4	9-4	7-2	11-2	12-2	1-1	3-1	5-1	6-1	10-1
10	1.73888888888889	9-5	2-4	4-4	8-4	1-2	3-2	7-2	11-2	12-2	5-1	6-1	10-1
3	1.70694444444444	2-5	8-5	9-5	4-4	1-3	5-3	10-3	11-3	12-3	3-2	6-0	7-0
9	1.70694444444444	2-5	4-4	8-4	9-4	1-3	11-3	3-2	5-2	6-2	10-2	12-2	7-0
2	1.52361111111111	4-4	8-4	9-4	2-3	6-2	11-2	12-2	1-1	3-1	5-1	7-1	10-1
8	1.52361111111111	4-5	8-5	9-5	11-5	1-4	3-4	6-4	10-4	12-4	2-0	5-0	7-0
6	1.24861111111111	4-5	2-4	8-4	9-4	1-2	7-2	11-2	12-2	3-1	5-1	6-1	10-1
4	1.23611111111111	2-5	4-5	8-4	9-4	11-4	3-3	5-3	10-3	1-0	6-0	7-0	12-0
5	1.21666666666667	2-5	4-5	8-5	9-5	11-4	12-4	1-3	3-3	7-3	10-3	5-0	6-0

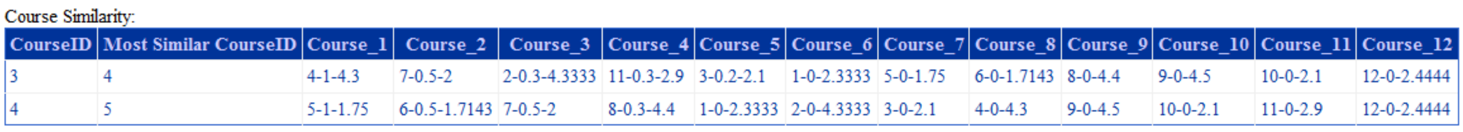
This table shows sorted similarity score for each student compared with new student by descending order. It also provides the rating value for each course for specific student.



Summary Student-Course Similarity-Rating:

Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10
9-7.0817	8-6.8803	2-6.7628	11-4.4199	12-3.7789	1-3.6946	3-3.2297	10-3.2265	7-2.9253	6-2.7565

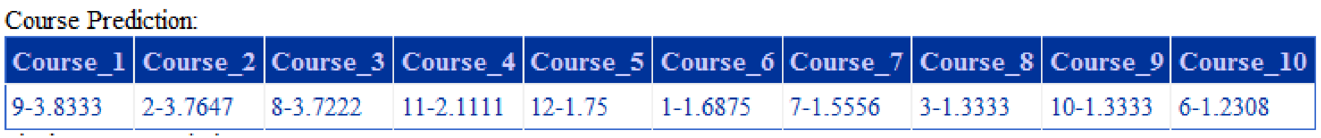
This table shows the combined calculation results from previous table.



Course Similarity:

CourseID	Most Similar CourseID	Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10	Course_11	Course_12
3	4	4-1-4.3	7-0.5-2	2-0.3-4.3333	11-0.3-2.9	3-0.2-2.1	1-0-2.3333	5-0-1.75	6-0-1.7143	8-0-4.4	9-0-4.5	10-0-2.1	12-0-2.4444
4	5	5-1-1.75	6-0.5-1.7143	7-0.5-2	8-0.3-4.4	1-0-2.3333	2-0-4.3333	3-0-2.1	4-0-4.3	9-0-4.5	10-0-2.1	11-0-2.9	12-0-2.4444

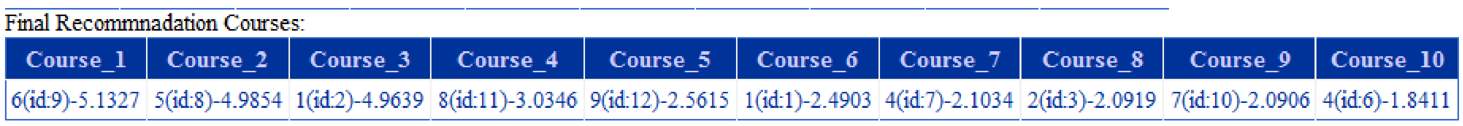
In this table, the first column **Course ID** means the input course ID from interface. The second column **Most Similar Course ID** means the program advises the most similar record id from course.csv file. For example, the record which id equals 5 in course.csv file is the most similar course for course 4. And the rest of the columns show the similarity score and average rating score for each course based on current user input course ID.



Course Prediction:

Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10
9-3.8333	2-3.7647	8-3.7222	11-2.1111	12-1.75	1-1.6875	7-1.5556	3-1.3333	10-1.3333	6-1.2308

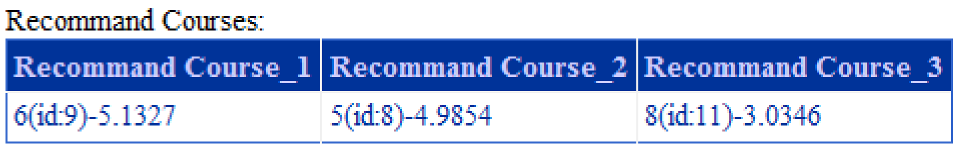
This table shows the sorted course prediction results after applying **Slope One algorithm** on training data.



Final Recommnadation Courses:

Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10
6(id:9)-5.1327	5(id:8)-4.9854	1(id:2)-4.9639	8(id:11)-3.0346	9(id:12)-2.5615	1(id:1)-2.4903	4(id:7)-2.1034	2(id:3)-2.0919	7(id:10)-2.0906	4(id:6)-1.8411

This table provides the sorted final recommendation course with its score. In this situation, the program will combine both student recommendation results and course prediction results to generate this table.



Recommand Courses:

Recommand Course_1	Recommand Course_2	Recommand Course_3
6(id:9)-5.1327	5(id:8)-4.9854	8(id:11)-3.0346

The final results select the top three courses which meet the needs of the rule for prerequisites from previous table(so the program ignores Course_3 from previous table). We can see these three courses are popular courses(look the average score and ignore the courses which ID less than course 4) and they are also get the higher rating from the most similar students. And course prediction algorithm also gives the higher score for these courses. So the algorithm makes good recommendations for this test.

Testing 2.4

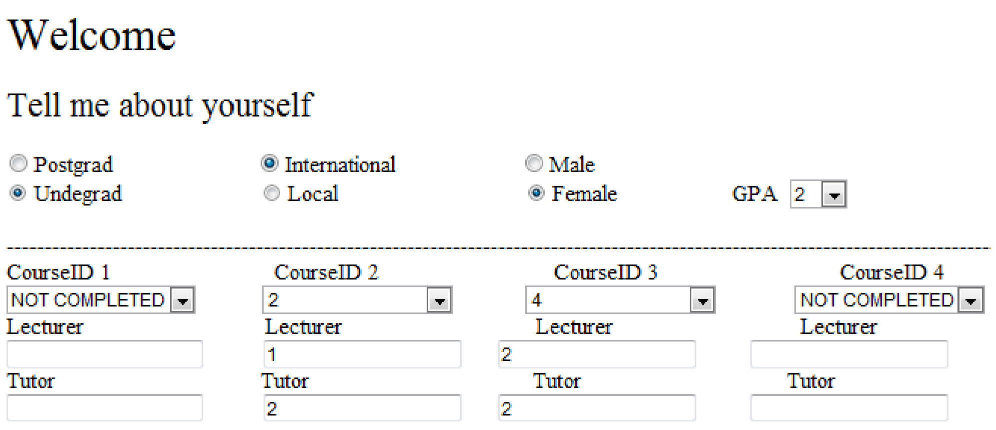
Input:

Student Information: 1,1,0,2 (undergrad, female, local, GPA)

Course information : 2,1,2-2 (course ID, lecturer ID, tutor ID, Rating)

3,2,2-4

Input shows below:



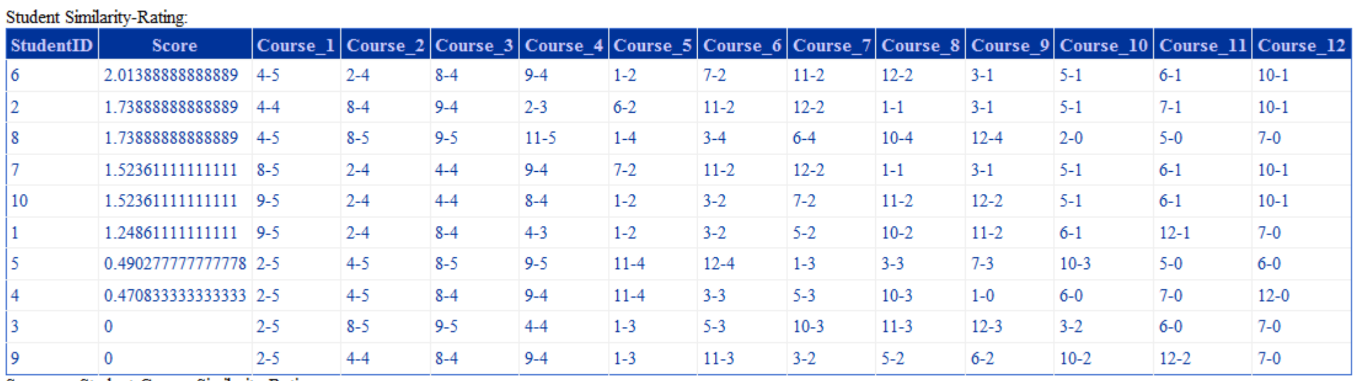
Welcome

Tell me about yourself

○ Postgrad	● International	○ Male	
● Undegrad	○ Local	● Female	GPA 2

CourseID 1	CourseID 2	CourseID 3	CourseID 4
NOT COMPLETED	2	4	NOT COMPLETED
Lecturer	Lecturer	Lecturer	Lecturer
	1	2	
Tutor	Tutor	Tutor	Tutor
	2	2	

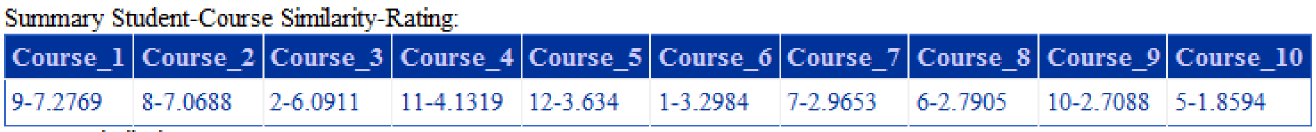
Intermediate results show below:



Student Similarity-Rating:

StudentID	Score	Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10	Course_11	Course_12
6	2.01388888888889	4-5	2-4	8-4	9-4	1-2	7-2	11-2	12-2	3-1	5-1	6-1	10-1
2	1.73888888888889	4-4	8-4	9-4	2-3	6-2	11-2	12-2	1-1	3-1	5-1	7-1	10-1
8	1.73888888888889	4-5	8-5	9-5	11-5	1-4	3-4	6-4	10-4	12-4	2-0	5-0	7-0
7	1.52361111111111	8-5	2-4	4-4	9-4	7-2	11-2	12-2	1-1	3-1	5-1	6-1	10-1
10	1.52361111111111	9-5	2-4	4-4	8-4	1-2	3-2	7-2	11-2	12-2	5-1	6-1	10-1
1	1.24861111111111	9-5	2-4	8-4	4-3	1-2	3-2	5-2	10-2	11-2	6-1	12-1	7-0
5	0.490277777777778	2-5	4-5	8-5	9-5	11-4	12-4	1-3	3-3	7-3	10-3	5-0	6-0
4	0.470833333333333	2-5	4-5	8-4	9-4	11-4	3-3	5-3	10-3	1-0	6-0	7-0	12-0
3	0	2-5	8-5	9-5	4-4	1-3	5-3	10-3	11-3	12-3	3-2	6-0	7-0
9	0	2-5	4-4	8-4	9-4	1-3	11-3	3-2	5-2	6-2	10-2	12-2	7-0

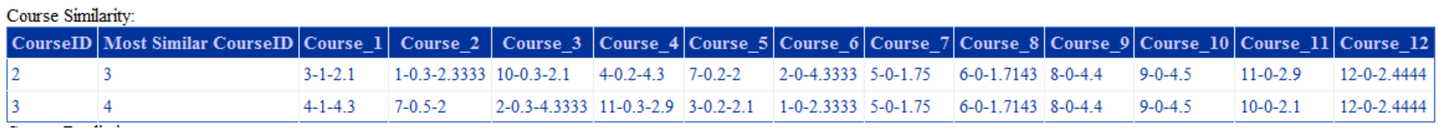
This table shows sorted similarity score for each student compared with new student by descending order. It also provides the rating value for each course for specific student.



Summary Student-Course Similarity-Rating:

Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10
9-7.2769	8-7.0688	2-6.0911	11-4.1319	12-3.634	1-3.2984	7-2.9653	6-2.7905	10-2.7088	5-1.8594

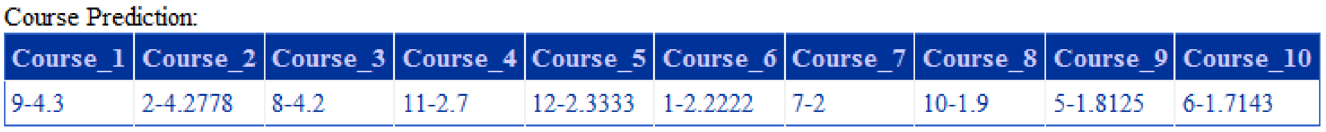
This table shows the combined calculation results from previous table.



Course Similarity:

CourseID	Most Similar CourseID	Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10	Course_11	Course_12
2	3	3-1-2.1	1-0.3-2.3333	10-0.3-2.1	4-0.2-4.3	7-0.2-2	2-0-4.3333	5-0-1.75	6-0-1.7143	8-0-4.4	9-0-4.5	11-0-2.9	12-0-2.4444
3	4	4-1-4.3	7-0.5-2	2-0.3-4.3333	11-0.3-2.9	3-0.2-2.1	1-0-2.3333	5-0-1.75	6-0-1.7143	8-0-4.4	9-0-4.5	10-0-2.1	12-0-2.4444

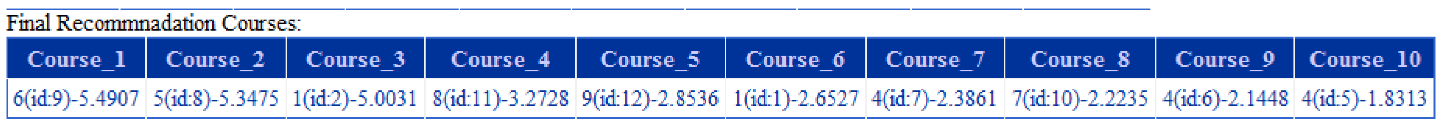
In this table, the first column **Course ID** means the input course ID from interface. The second column **Most Similar Course ID** means the program advises the most similar record id from course.csv file. For example, the record which id equals 3 in course.csv file is the most similar course for course 2. And the rest of the columns show the similarity score and average rating score for each course based on current user input course ID.



Course Prediction:

Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10
9-4.3	2-4.2778	8-4.2	11-2.7	12-2.3333	1-2.2222	7-2	10-1.9	5-1.8125	6-1.7143

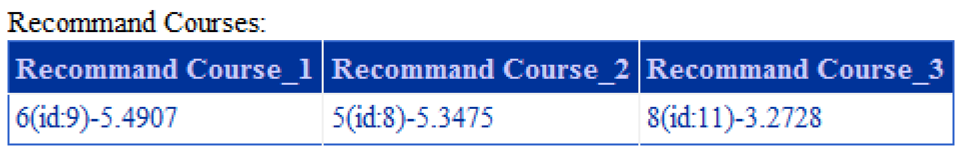
This table shows the sorted course prediction results after applying **Slope One algorithm** on training data.



Final Recommnadation Courses:

Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10
6(id:9)-5.4907	5(id:8)-5.3475	1(id:2)-5.0031	8(id:11)-3.2728	9(id:12)-2.8536	1(id:1)-2.6527	4(id:7)-2.3861	7(id:10)-2.2235	4(id:6)-2.1448	4(id:5)-1.8313

This table provides the sorted final recommendation course with its score. In this situation, the program will combine both student recommendation results and course prediction results to generate this table.


Recommand Courses:

Recommand Course_1	Recommand Course_2	Recommand Course_3
6(id:9)-5.4907	5(id:8)-5.3475	8(id:11)-3.2728

The final results select the top three courses which meet the needs of the rule for prerequisites from previous table(so the program ignores Course_3 from previous table). We can see these three courses are popular courses(look the average score and ignore the courses which ID less than course 3) and they are also get the higher rating from the most similar students. And course prediction algorithm also gives the higher score for these courses. So the algorithm makes good recommendations for this test.

Testing 2.5

Input:

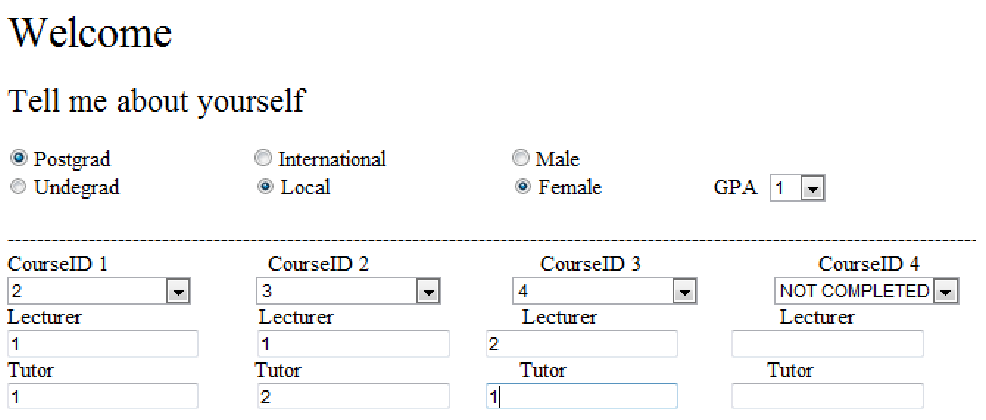
Student Information: 0,1,1,1 (undergrad, female, local, GPA)

Course information : 1,1,1-2 (course ID, lecturer ID, tutor ID, Rating)

2,1,2-3

3,2,1-4

Input shows below:


Welcome

Tell me about yourself

[✓ Postgrad] Undegrad

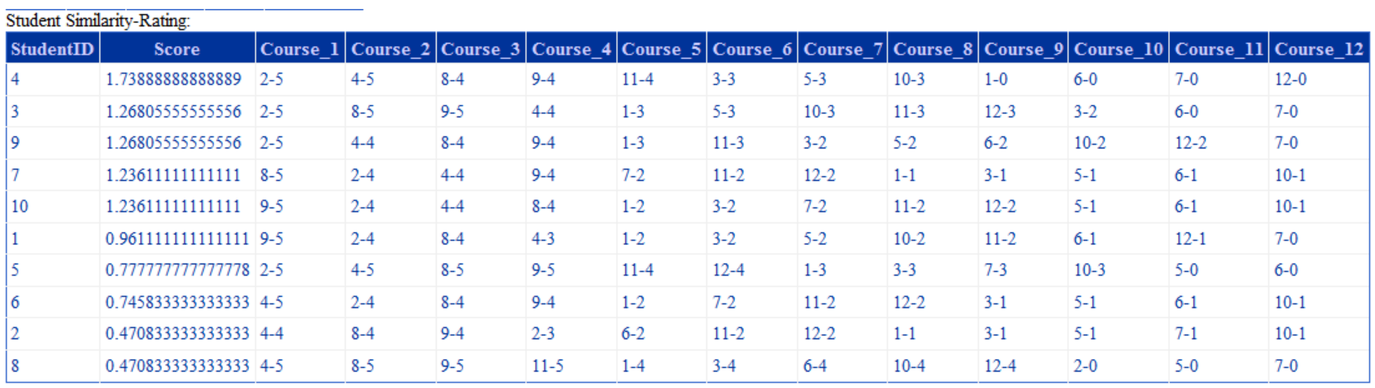
International [✓ Local]

Male [✓ Female]

GPA 1

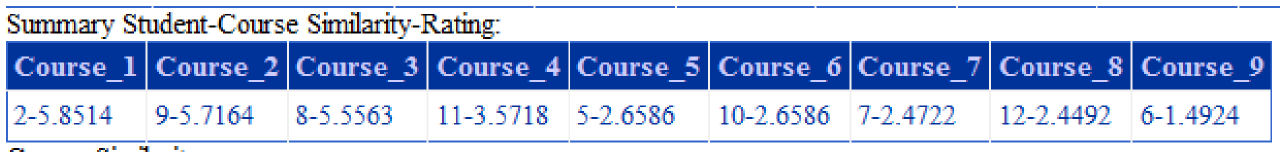
CourseID 1	CourseID 2	CourseID 3	CourseID 4
2	3	4	NOT COMPLETED
Lecturer	Lecturer	Lecturer	Lecturer
1	1	2	
Tutor	Tutor	Tutor	Tutor
1	2	1	

Intermediate results show below:


Student Similarity-Rating:

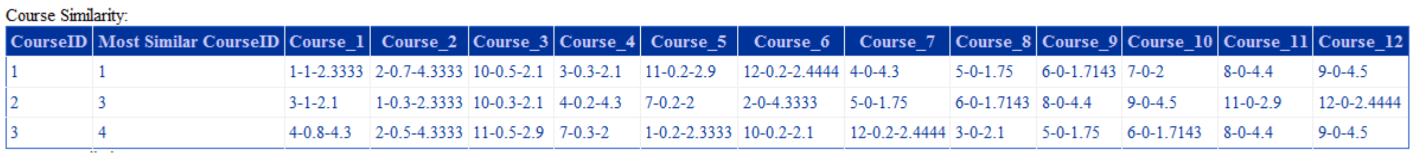
StudentID	Score	Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10	Course_11	Course_12
4	1.73888888888889	2-5	4-5	8-4	9-4	11-4	3-3	5-3	10-3	1-0	6-0	7-0	12-0
3	1.26805555555556	2-5	8-5	9-5	4-4	1-3	5-3	10-3	11-3	12-3	3-2	6-0	7-0
9	1.26805555555556	2-5	4-4	8-4	9-4	1-3	11-3	3-2	5-2	6-2	10-2	12-2	7-0
7	1.23611111111111	8-5	2-4	4-4	9-4	7-2	11-2	12-2	1-1	3-1	5-1	6-1	10-1
10	1.23611111111111	9-5	2-4	4-4	8-4	1-2	3-2	7-2	11-2	12-2	5-1	6-1	10-1
1	0.961111111111111	9-5	2-4	8-4	4-3	1-2	3-2	5-2	10-2	11-2	6-1	12-1	7-0
5	0.777777777777778	2-5	4-5	8-5	9-5	11-4	12-4	1-3	3-3	7-3	10-3	5-0	6-0
6	0.745833333333333	4-5	2-4	8-4	9-4	1-2	7-2	11-2	12-2	3-1	5-1	6-1	10-1
2	0.470833333333333	4-4	8-4	9-4	2-3	6-2	11-2	12-2	1-1	3-1	5-1	7-1	10-1
8	0.470833333333333	4-5	8-5	9-5	11-5	1-4	3-4	6-4	10-4	12-4	2-0	5-0	7-0

This table shows sorted similarity score for each student compared with new student by descending order. It also provides the rating value for each course for specific student.


Summary Student-Course Similarity-Rating:

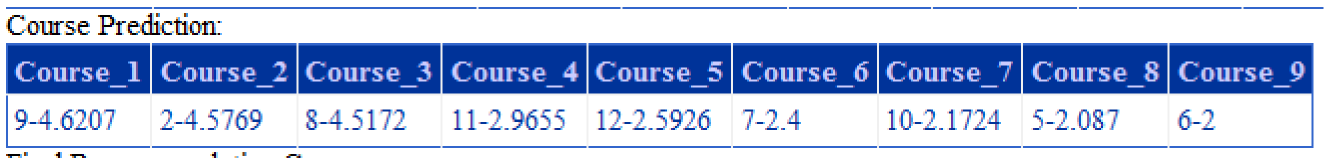
Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9
2-5.8514	9-5.7164	8-5.5563	11-3.5718	5-2.6586	10-2.6586	7-2.4722	12-2.4492	6-1.4924

This table shows the combined calculation results from previous table.


Course Similarity:

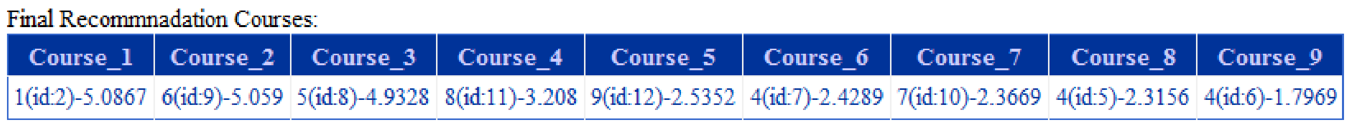
CourseID	Most Similar CourseID	Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9	Course_10	Course_11	Course_12
1	1	1-1-2.3333	2-0.7-4.3333	10-0.5-2.1	3-0.3-2.1	11-0.2-2.9	12-0.2-2.4444	4-0-4.3	5-0-1.75	6-0-1.7143	7-0-2	8-0-4.4	9-0-4.5
2	3	3-1-2.1	1-0.3-2.3333	10-0.3-2.1	4-0.2-4.3	7-0.2-2	2-0-4.3333	5-0-1.75	6-0-1.7143	8-0-4.4	9-0-4.5	11-0-2.9	12-0-2.4444
3	4	4-0.8-4.3	2-0.5-4.3333	11-0.5-2.9	7-0.3-2	1-0.2-2.3333	10-0.2-2.1	12-0.2-2.4444	3-0-2.1	5-0-1.75	6-0-1.7143	8-0-4.4	9-0-4.5

In this table, the first column **Course ID** means the input course ID from interface. The second column **Most Similar Course ID** means the program advises the most similar record id from course.csv file. For example, the record which id equals 3 in course.csv file is the most similar course for course 2. And the rest of the columns show the similarity score and average rating score for each course based on current user input course ID.


Course Prediction:

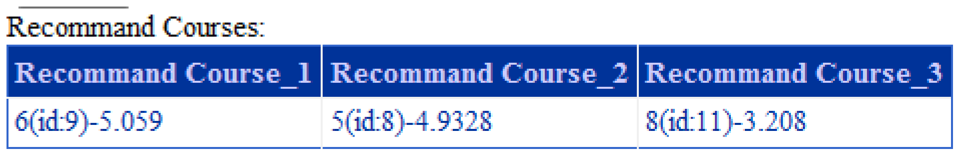
Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9
9-4.6207	2-4.5769	8-4.5172	11-2.9655	12-2.5926	7-2.4	10-2.1724	5-2.087	6-2

This table shows the sorted course prediction results after applying **Slope One algorithm** on training data.


Final Recommnadation Courses:

Course_1	Course_2	Course_3	Course_4	Course_5	Course_6	Course_7	Course_8	Course_9
1(id:2)-5.0867	6(id:9)-5.059	5(id:8)-4.9328	8(id:11)-3.208	9(id:12)-2.5352	4(id:7)-2.4289	7(id:10)-2.3669	4(id:5)-2.3156	4(id:6)-1.7969

This table provides the sorted final recommendation course with its score. In this situation, the program will combine both student recommendation results and course prediction results to generate this table.


Recommand Courses:

Recommand Course_1	Recommand Course_2	Recommand Course_3
6(id:9)-5.059	5(id:8)-4.9328	8(id:11)-3.208

The final results select the top three courses which meet the needs of the rule for prerequisites from previous table(so the program ignores Course_1 from previous table). We can see these three courses are popular courses(look the average score and ignore the courses which ID less than course 3) and they are also get the higher rating from the most similar students. And course prediction algorithm also gives the higher score for these courses. So the algorithm makes good recommendations for this test.

Reference:

1. Slope One Algorithm   http://en.wikipedia.org/wiki/Slope_One

2. Lecture Note 2.