IITB Summer Internship 2013

Project Report

Peer Evaluation System for MOOCs

Principal Investigator
Prof. D.B. Phatak

Project In-Charge
Mr. Nagesh Karmali

Project Mentors
- Miss. Birundha M.
- Miss. Firuza Aibara (PMO)
- Mr. Nilesh Birari
- Mr. Pawan Kumar Singh

Project Team Members
- Mr. Jaspal Singh
- Mr. Kshitji Jain
- Miss. Nikhita Vedula
- Miss. Priya Agrawal
- Miss. Priyanka Mathur
- Miss. Saumya Agrawal
Abstract

The importance of Massive Open Online Courses (MOOCs) has been growing over the years, and so has their role in distance education. This report explores in detail a suitable evaluation system which we have designed for such a MOOC, and also contrasts it with an existing open source assessment system, namely the edX Open Response Assessment. The database management systems employed by the edX have been studied, and various modules of the edX have been described as well. These include an Application Programming Interface wrapper for machine learning (Discern), an interface for the Learning Management System to communicate with external grader services (Xqueue), and a message broker software (RabbitMQ) which it makes use of. Another grading module XServer has been depicted, along with a few security mechanisms. In addition to this, various supplementary diagrams, and experiments which we have conducted during the course of our project have also been included.
List of Figures and Graphs

1.1 Massive Open Online Course ............................................................................................................. 7
2.1 Activity Flow for Calibration .............................................................................................................. 14
2.2 Activity Flow for Normalisation ....................................................................................................... 20
2.3 Activity Diagram for instructor ....................................................................................................... 23
2.4 Activity Diagram for Student ......................................................................................................... 24
4.1 Describing the PEAS Peer Grading Module through a block diagram. ......................................... 33
4.2 Block diagram for Self assessment in PEAS .................................................................................. 36
4.3 Block diagram for combination of self and peer assessment in PEAS ........................................... 41
5.1 Screenshot of terminal displaying a list of tables in mitx.db file .................................................. 42
5.2 Screenshot of terminal showing tables added on discern installation ........................................... 45
6.1 User interaction with discern ......................................................................................................... 47
6.2 Interaction of LMS with XQueue ..................................................................................................... 48
6.3 Passive graders ............................................................................................................................... 49
6.4 Active graders ............................................................................................................................... 51
6.5 Interaction of Xserver and XQueue ............................................................................................... 52
7.1 Graph showing that rate of publishing and receiving messages is almost equivalent ................. 55
7.2 Graph showing that 50000 messages got queued and ready to be sent ...................................... 55
7.3 Graph showing that rate of publishing is 220 messages per second ............................................. 56
7.4 50000 messages got queued in the queue .................................................................................... 56
7.5 Graph showing that queued messages got delivered in around 10 seconds ............................... 56

List of Tables

5.1 Database table for auth_user ............................................................................................................ 38
5.2 Database table for auth_userprofile .............................................................................................. 39
5.3 Database table for courseware_studentmodule ........................................................................... 39
5.4 Database table for generated certificates ...................................................................................... 40
5.5 Common fields for tracking logs ................................................................................................ 43
7.1 Calculation of normalization factor .............................................................................................. 55
7.2 Obtaining final marks of students ............................................................................................... 56
ACKNOWLEDGEMENT

It has been a great honour and privilege to undergo training at the Indian Institute of Technology, Bombay. We are deeply obliged to Prof. Deepak Phatak for giving us the opportunity to participate in the Summer Internship 2013. We would like to express our profound gratitude and deep regards to our Guide, Mr. Nagesh Karmali, for his exemplary guidance, monitoring and constant encouragement throughout the course of this internship. We are also obliged to our Mentor, Ms. Birundha Murugesan, for her cordial support and timely guidance.

Last but not the least, we would like to thank Ms. Firuza Aibara, Mr. Pawan Kumar Singh and Mr. Nilesh for their help and cooperation at various stages of this project.
Chapter 1
Massive Open Online Courses

1.1 Introduction
A massive open online course (MOOC) is an online course aimed at large-scale interactive participation and open access via the web. It provides interactive user forums in addition to the traditional course materials such as videos, readings and problem sets. MOOCs are a recent development in distance education.

![Fig 1.1: Massive Open Online Course](image)

1.2 Origin of MOOCs
MOOCs originated about 2008 within the open educational resources movement. The term MOOC was coined in 2008 during a course called “Connectivism and Connective Knowledge”
that was presented to 25 tuition paying students in Extended Education at the University of Manitoba in addition to 2300 other students from the general public who took the online course free of charge. The term was coined by Dave Cormier of the University of Prince Edward Island, and Senior Research Fellow Bryan Alexander of the National Institute for Technology in Liberal Education.

1.3 Recent Developments

In the fall of 2011 Stanford University launched three courses, each of which had an enrolment of about 100,000. The first of those courses, Introduction Into AI, was launched by Sebastian Thrun and Peter Norvig. Following the publicity and high enrolment numbers of these courses, Sebastian Thrun launched Udacity and Daphne Koller and Andrew Ng launched Coursera, both for-profit companies.

Concerned about the commercialization on online education, MIT launched the MITx not-for-profit later in the fall, an effort to develop a free and open online platform. Harvard joined the initiative, renamed edX, that spring, and University of California, Berkeley joined in the summer.

1.4 Exams and assessment

Assessment can be the most difficult activity to conduct online. The two most common methods of MOOC assessment are machine-graded multiple-choice quizzes or tests and peer-reviewed written assignments.

Peer review will often be based upon sample answers or rubrics, which guide the grader on how many points to award different answers. These rubrics cannot be as complex for peer grading as they can be for grading by teaching assistants, but students are expected to learn both by being the grader as well as by having their work graded.

1.5 Challenges and Criticisms

The 5 possible challenges for collaborative style MOOCs:
1. It feels chaotic as participants create their own content
2. It demands digital literacy
3. It demands time and effort from the participants
4. It is organic, which means the course will take on its own trajectory (you have got to let go).
5. As a participant you need to be able to self-regulate your learning and possibly give yourself a learning goal to achieve

1.6 Assessment Modules

1.6.1 Instructor Grading

This is the traditional grading technique in which the instructor examines the solution of the students. This grading method is difficult to implement in a massive open online course.

Advantages:
Correct Evaluation Of grades: Having an instructor grade your assignments assures correct evaluation.

Disadvantages:
Infeasible in MOOCs: It is not feasible for an expert to assess assignments in a MOOC

1.6.2 Self Grading

An effective self-assessment program allows students to meaningfully compare their success against that of their classmates, encourages academic honesty, and yet still gives each student considerable latitude when assigning their own grade.

Advantages:
Student Motivation: Student motivation is key to student success in and attitudes regarding education.

Students’ Responsible for their Own Learning: In addition to improved motivation, students felt self assessment increased responsibility for their own learning.

**Disadvantages:**

Grade Inflation: Self Assessment increased grade inflation. More students were found grading themselves higher than what they were supposed to get

**1.6.3 Peer Grading**

It is a process in which students or their peers’ grade assignments or tests based on a teacher’s benchmarks. The practice is employed to save teachers’ time and improve students’ understanding of course materials.

Peer assessment is an important form of collaboration that is used for quality control. It can improve students’ performance.

**Advantages**

Saves teachers’ time: Student grade assignments can save teacher’s time because an entire classroom can be graded together in the time that it would take a teacher to grade one paper. Moreover, rather than having a teacher rush through each paper, students are able to take their time to correct them.

Faster Feedback: It decreases the time taken for the students to get back their assignments.

Increases Learning Process: Students’ have more opportunity to view assignments of their peers’ and learn

**Disadvantages**

Reviewers are also our competitors: The students may intentionally rate their peers’ low so as to improve their relative performance

De-Motivate Students: Those who get low grades may regard peer assessment as inaccurate and get de-motivated.

Educators Hesitate to Adopt Peer Assessment: It was observed that educators hesitate in adopting peer assessment owing to the possibility of over marking or under marking.
1.6.4 Machine Grading

Machine Grading refers to getting the assignment graded by an auto-grader. People over the world are still working on the process of getting an open ended subjective question evaluated correctly by an auto grader. The concept of Artificial Intelligence is used in machine grading. Artificial Intelligence is a branch of computer science that studies and develops intelligent machine and software.
Chapter-2

PEAS EVALUATION SYSTEM

PEAS (Peer Expert Autograde Self) is the evaluation system designed by us. It is a web based tool that is to be used in association with a MOOC. The MOOC is used for uploading tutorials and enrolling students for various courses. The assignments for the course are to be provided on the Peer Evaluation System. Students will take up an assignment, and submit solutions. These solutions will then be evaluated by other students/instructor or himself and scores are given.

2.1 Features

The functionality of our system is fourfold i.e. it would consist of 4 different modules-

1. Peer evaluation.
2. Self assessment.
3. Auto-grading softwares.
4. Instructor evaluation.

A combination of the above mentioned assessment mechanisms can be used, providing weightage to scores from each method.

The instructor has the facility to select a grading mechanism for each assignment he uploads. He can choose from any of the above modules or their combination, specifying the weightage to each module.

2.2 Peer evaluation

In peer assessment, we develop a model that reliably measures the performance of students without the need for expert’s intervention. This system also gives the opportunity to students to learn by playing the role of both “student” and “teacher”. By reading other students’ answers every student learns alternate ways to solve a problem.

The following process is followed:
2.2.1 Calibration:

In our system to ensure proper grading, solution submitted by every student is evaluated by $n$ students (where $n$ is specified by instructor). The students are divided into $n$ groups based on their grading abilities decided by the incentive.

The review needs an incentive to evaluate the assignment correctly. If he evaluates the paper accurately, he will get a good incentive and not a good one otherwise. The calibration method is employed in the beginning when no user profile is maintained. Before the first assignment is to be evaluated, we need to divide the students in groups so that each paper is evaluated by a number of reviewers of different reviewing abilities. The ability of the reviewer to evaluate correctly is corresponding to the incentive he receives. The incentive is a number from 0 to 1.

For that, we need a calibration mechanism in the beginning of the course. There are three kinds of mechanisms, and the instructor of the course will make the decision on which method to use.

a) Peer Calibration:

In this method, the instructor provides an assignment and an arbitrary solution to that assignment to all the students enrolled in the course. Every student is asked to review the solution according to a set of questions specified in the feedback form, and also give a holistic rating on a scale of 1 to 10.

The average of all the ratings is calculated. Let it be $x$. Let the rating provided by a student be $x'$. Also, let the incentive he receives be denoted by $c$.

Then,

$$c = 1 - \frac{|x-x'|}{MM}, \text{ where } MM \text{ is the maximum marks viz 10.}$$

b) Expert Calibration:

Here, the instructor will provide an assignment and solution and will evaluate the paper himself and give a score on a scale of 1 to 10. He then asks all the students to evaluate the same paper and give their score. Let $x$ be the rating provided by instructor and $x'$ be the score provided by a student.

Then, the incentive $c$ received by each student is calculated in the same way,

$$c = 1 - \frac{|x-x'|}{MM}, \text{ where } MM \text{ is the maximum marks viz 10.}$$
c) No Calibration:

This mechanism is used if the instructor does not want to give anyone an incentive at the beginning of the course. Here, every student is considered to have 0 incentive and thus they are randomly divided into n groups in the beginning.

Fig 2.1: Activity flow for calibration.
2.2.2 Division in groups:

The students are distributed into groups so that there is uniformity in the different reviews that each student gets. The list of all students, ordered by the incentive that they received (from high to low), is then divided into as many groups as there should be number of reviewers for each assignment (n). If there is more than one student with the same incentive, they are ordered alphabetically. For example, if there are 100 students and we need 4 reviewers for each assignment, the students are ranked according to the incentive they received. Then 4 (number of reviewers) groups are created, and first 25 students are placed in Group 1, next 25 in Group 2, next 25 in Group 3 and the last 25 in Group 4.

Number of groups = Number of reviewers for each assignment

2.2.3 Review Process:

For every student’s assignment, a student reviewer is selected at random from all the groups. The selection is at random, and makes sure that the student himself is not selected from his own group to evaluate his paper. Each student is informed on his page of the assignments about the assignments that he has to evaluate. A feedback form is provided which states the characteristics on the basis of which evaluation is done. A holistic rating is also provided on a scale of 1 to 10.

2.2.4 Normalisation:

To ensure fair assessment, we need a mechanism that takes into account that every student might not correctly grade his peer. We use a normalisation method that which is used to eliminate problems like student being generous and biased in giving scores. Here, mainly 4 cases are taken into consideration:

a) Normal Case:

In this case, the marks given by peers are considered to be fair.

b) Over-generous Case:

A student taking up peer assessment can award more marks than necessary to his peers, and a quid-pro-quo arrangement can be made.
c) Creative-accounting Case:

Here, under this case a student gives less marks to all his peers, in order to get a higher rank.

d) Penalised Case:

Another peer assessment scenario is where the majority of the group members decide to go against one person. In this case all the peers decide to give low marks to a specific student. This could be a valid case when the member did not contribute much to the project or when he actually deserves low marks.

A few terms are explained further which are used to calculate normalised marks of a student.

**Individual Effort Rating (IER):**

This is the summation of the marks received by a student from all his peers.

\[ \text{IER} = \sum_{i} M_i \text{, where } M_i = \text{marks given by student } i. \]

**Average Effort Rating (AER):**

This is the summation of the IER of each student divided by number of students in consideration.

\[ \text{AER} = \frac{\text{IER}}{i}, \text{ where } i \text{ is the number of students} \]

**Individual Weighting Factor (IWF):**

\[ \text{IWF} = \frac{\text{IER}}{\text{AER}}. \]

**Bias Factor:**

This factor tells us how much bias has been done by a particular student.

\[ \text{Bias Factor} = \frac{\text{Rating given to others}}{\text{AER}}. \]
**Normalisation Factor:**

If bias factor (calculated for each student individually) doesn’t come out to be in the range [0.98, 1.02], then the normalisation factor (for each student) is calculated and multiplied with marks awarded by that student.

Normalisation Factor = 1 / Bias Factor.

We now have a table with updated marks and this process is repeated until all the bias factors fall in the required range.
Fig. 2.2 Activity flow for normalization process.
2.2.5 Exceptional case

In case a student does not review the assignments allocated to him, the assignments will be send for rechecking. 
But this time these assignments would be forwarded only to good reviewers i.e. students with high incentives.

2.3 Self Assessment:

Self assessment is employed by the instructor when the main purpose of the course is learning. Students are asked to evaluate their own assignments.

Here, in this assessment technique a student firstly completes the assignment and then he himself evaluates his assessment as a reviewer based on the evaluation sheet given by the instructor. There is no provision for incentive in this mechanism. The main purpose of this assessment technique is to motivate students to learn. It focuses on the development of student’s judgment skills. It encourages students to reflect on their work.

2.4 Auto graders:

This will be done by special softwares for checking program codes, essays and multiple choice questions (MCQs). Here, the system will automatically evaluate with these existing online softwares and give the result.

2.5 Instructor assessment:

The instructor has the option to review the papers himself. He will score each assignment on a scale of 1 to 10. This is a useful option when there are less number of students enrolled in a course.
2.6 Combination of above methods (hybrid):

The instructor can adopt a combination of any of the above mentioned methods for evaluation for every assignment that he uploads. He has to provide the weightage for scores from each type of evaluation to be accounted in the final assignment score.

2.7 Activity flow in PEAS

There are two different users of this software:

- Instructor who can create courses and upload assignments.
- Students who can write assignments and evaluate others’ solutions.

The various activities performed by the instructor are:-

- Upload assignments.
- View the list of enrolled students, their assignments and grades.
- Design the course choosing different evaluation schemes.
- Evaluate reviews and change the grades if he deems fit.
Fig. 2.3 Activity diagram for instructor
Fig 2.4: Activity diagram for student
3.1 Introduction to Edx-Open Response Assessment

Open response assessment allows for multiple grader types to be used in a single problem. So you write one problem and one rubric, and the student writes one response. The multiple grader types are then fed this response. You can set "thresholds" for moving between graders. For example, if you decide that you problem should be self-assessed, then AI assessed, and then peer assessed, you can say that if the AI assessment judges the student response incorrect, then do not move on to peer grading. Or, if the student judges their response incorrect, then do not move on to AI assessment. This allows course staff to flexibly and relatively easily define complex student workflows.

Workflow

- Teaching team writes a question.
- He writes a clear and easy to follow rubric.
- Teaching team Decide what kinds of open response assessment types (self assessment, peer, etc), they would like in the problem.
- Take specific action related to the types of assessment that they want to incorporate.

Strengths

- Utilizes multiple open response assessment types, and allows for complex workflows (i.e. student self assesses, if they are correct, then move on to AI assessment. If the AI says that the response is correct, then move on to peer grading.)
- Allows students to get aggregated feedback from multiple grader types.

Weaknesses

- As it utilizes multiple grader types, it could potentially require more staff grading.
- Could require more student effort to check on progress and respond to grader feedback.
3.2 Rubric design

Having a good rubric is key to helping students learn from self or peer evaluation.

- The Rubrics should be concrete
- It should be well illustrated with ample of examples
- Parallel language should be use in each axis of evaluation.
- Course Staff should try to anticipate the level of their students—e.g. don't ask students who are struggling to form complete sentences to evaluate the nuances of literary writing style.

3.3 Self-assessment

Workflow

- Teaching team writes a question.
- Teaching team writes an evaluation rubric—how can someone (in this case, the student themselves) assess the response in a consistent way
- Write a prompt for a hint—this will be shown to students whose self-assessed score is less than the max, and should ask them to reflect on what they missed, and what hint could help a student with similar misconceptions.
- Write a "you're done" message—this might prompt the student to further reflect on the experience.

Strengths

- Allows students to hopefully think critically about their performance.
- Students are able to instantly see the "correct" answer in the form of a rubric.
- Students can repeat the question as many times as they want.
Weaknesses

- Ora Self Assessment Module can be only used for learning purpose
- It doesn’t help in the actual grades of the students.
- There is no mechanism which forces student to evaluate himself and that too honestly
- Students are not the best at analyzing their own answers
- Answers may not be long enough to provide useful information
- Student may skip the question if it is not graded

Next steps

- Find a way to display metrics
- Allow students to use a rubric to self-assess themselves.
- Edx should incorporate ways to include self grading module for evaluation purpose also.

3.4 Instructor Grading

Workflow

- Teaching team writes a question.
- Teaching team writes an evaluation rubric–how can someone (in this case, the student themselves) assess the response in a consistent way
- Teaching team visit the Staff Grading page
- Team grades each submitted problem

Strengths

- Students get very reliable score and feedback
- Allows instructor to know firsthand how students are performing in the class and tailor class accordingly
The instructor grading is used for initial training data set for the AI Assessment using the Machine Learning Algorithms.

**Weaknesses**

- Instructor Grading is very difficult to scale on a large scale for MOOC

**Next steps**

- Allow instructors to 'check off' specific rubric items for quicker and more explicit grading

### 3.5 AI assessment

At a high level, the program computes which features of the manually-assessed student responses correlated with you giving them good and bad scores, and then applies that grading policy to the new submissions. The AI is incapable of making its own judgments about what is "good" and what is "bad," and thus mimics a human scorer. The difficult part is extracting good features—the system looks at the individual words, sequences of words, grammar and spelling, tries to assess how on-topic the responses are, and has other heuristics. It can certainly be fooled, though we typically find that in order to fool it, one need to know enough to be able to write a good answer in the first place.

The ML repo allows anyone to use machine learning based automated classification. This automated classification can work on both free text (essays, content, etc), and on numeric values.

Let's say that you have 10000 user reviews for 15 books (ie “I loved this!”, “I didn’t like it.”, and so on). What you really want to do is use the user reviews to get an aggregate score for each book that indicates how well-received it is. But, in your haste to collect the data, you forgot to get scores from the users. In this case, the text of the user reviews is your predictor, and the score that you want to collect from each user for each book is the target variable.

So, how do you turn the text into numbers? One very straightforward way is to just label each of the reviews by hand on a scale from 0 (the user didn’t like it at all) to 5 (they really loved it). But, somewhere around review 200 you are going to start to get very sick of the whole process. A less labor intensive way is to use automated classification.
If you choose to use automated classification for this task, you will score some reasonable subset of the reviews (if you score more, the classification will be more accurate, but 200 should be fine as a baseline). Once you have your subset, which can also be called a “training” set, you will be able to “train” a machine learning model that learns how to map your scores to the text of the reviews. It will then be able to automatically score the rest of the 9800 reviews. Let’s say you also want to take the user’s activity level into account in order to weight the score. You can add in a numeric predictor in addition to your existing text predictor (the review text itself) in order to predict the target variable (score).

This repo gives you a nice, clean way to do that via convenience functions grade, grade_generic, create, and create_generic.

**Workflow**

- Teaching team writes a question
- They make a plan for evaluation, typically using a rubric, and include it in the problem definition.
- When students submit, the submissions will be queued until the machine learning algorithm has enough training data
- Course staff will go to the Staff Grading page in the courseware, and grade a set of submissions (The model expect that ~100 are needed to get reasonable baseline results, but scoring more than this will improve performance)
- Once there is enough training data, the machine learning algorithm will grade the rest of the submissions.
- Course Staff can check on how the assessment model is doing, and re-train.

**Strengths**

- Good for assessing students.
- Much faster than peer grading.
- Can give unlimited feedback on spelling and grammar.

**Weaknesses**

- Has not been fully tested on live data
• Cannot give human-quality feedback

Next steps

• More feedback to students.
• Integrate with the rubric so students can clearly see why they received a certain score.

3.6 Peer assessment

Workflow

• Course Staff writes a question
• Course Staff writes an evaluation rubric that is broken up point by point, and highlights the components of a good answer.
• When a student begins to peer grade, they will first be asked to "calibrate" on instructor graded essays.
• Calibration ensures that the student understands how the problem should be graded, and shows them essays of varying quality.
• If students calibrate properly (come close to instructor scores on calibration essays), then they can peer grade.
• Currently each essay is peer graded by 3 students, and once peer grading is complete, the essay is returned to the student with their feedback.

Strengths

• Students are able to learn by looking at other people's papers (when they are peer grading)
• Can generate good feedback and scoring if students are engaged.
• Calibration essays can serve as learning tools

Weaknesses
• Need to avoid some known problems with peer grading systems

• There is no normalization process in Ora Peer Grading Module which ensures that student get fair grades.

• There is no incentive mechanism which may promote students to do fair grading and timely grading. If no incentive is provided student will not opt for peer grading.

• The Edx-Calibration lacks lot of fundamental problems such as presently if students exceed 6 times the calibration, despite having low quality in calibration he is permitted to the peer grading.

• Also the calibration is used just for training students, it can also be used for giving incentive/marks on the basis of calibration also to distribute assessment evaluation to evaluators using the calibration score.

• Need to make sure peer-graders are calibrated properly

• Good filtering required to prevent very low-quality responses from being fed into the peer grading system

Next steps

• Allow students to see submissions from the students who graded them (Edx-Future Goals but according to us the identity of evaluator should remain anonymous).

• Match peer graders together in a smart way that optimizes learning.

• Integrate rubrics into peer grading to allow students to give feedback by clicking the appropriate rubric boxes.
Chapter 4
Comparison between EDX-Ora and PEAS

4.1 Edx-ORA : Open Response Assessment

Open response assessment allows for multiple grader types to be used in a single problem.

The different grading mechanism that ORA supports are:

- Grading using Machine Learning Algorithms
- Instructor and course staff grading
- Peer grading
- Self grading
- None
- Basic Check

4.2 PEAS : Peer Expert AutoGrader Self Evaluation

The basic modules of our PEAS system are:

- Peer grading
- Self grading
- Auto grading
- Expert grading
4.3 Comparison of Peer Grading Module of ORA and PEAS

The features common to the both modules are:

a) Both ORA and PEAS use calibration for peer grading.
b) ORA has presently fixed grading per individual assessment to be 3 while PEAS offer flexible number (recommends 3 to 5)

Exclusive features

a) ORA don’t have any mechanism to do normalization of grades while PEAS do normalization of grades so that students receive fair grades.
b) Clearing Calibration Test in ORA is a compulsory feature while in PEAS it is just a onetime process.
c) PEAS use Incentive Mechanism to promote Honest Grading and ensure that students should get fair grades.
d) PEAS Uses optimize distribution algorithm for evaluator to ensure fairness of the grading process.

Fig 4.1: Describing the PEAS Peer Grading Module through a block diagram.
4.4 Comparison of Calibration Module of ORA and PEAS

The features common to the both modules are:

  a) Both ORA and PEAS use expert calibration for calibration.
  b) Both uses calibration for initial training of students and giving them feel of the peer grading.

Exclusive features

  a) PEAS uses both expert and peer calibration as the model for calibrating before the peer assessment. It also gives flexibility to choose no calibration as one of the starting model.
  b) PEAS Uses Calibration for
      a. Initial training of Students for their peer evaluation
      b. It rates their grading skills based on their calibration score.
      c. It uses this score for initial evaluator distribution.
      d. If no calibration is chosen the evaluators are distributed on the random basis, but if calibration model is chosen the distribution is based on the calibration score.
  c) ORA use Calibration for training students how to peer grade and students must clear calibration test before grading peers.
  d) In ORA students must calibrate for minimum number of times (presently 3).
  e) In ORA if Students calibrate higher than the maximum times (presently 6), he is given permission to grade peers despite the quality of his calibration.

4.5 Comparison of Self Grading Module of ORA and PEAS

The features common to the both modules are:

  a) Both ORA and PEAS emphasize on self grading for the learning and understanding of the student. Research’s have well shown that self assessment is best for learning own mistakes and own faults.
  b) Both use Rubrics Design for the self assessment guidelines.

Exclusive features

  a) ORA uses self assessment only for learning purpose
  b) It doesn’t contributing in grading
  c) It shows hint if your score is less than max score.
d) There is immediate display of answers to the students.
e) PEAS also normalize score to prevent the self over-grading.
f) There is Incentive to students to ensure that they follow the rubrics guidelines and do honest self-grading.
g) Two different self grading models available in PEAS only:
   a. Pure Self (for learning purpose only)
   b. Hybrid Self (for learning as well as grading)

---

**PEAS PURE SELF ASSESSMENT**

Fig 4.2: Block diagram for Self Assessment in PEAS
4.6 Comparison of Auto Grading Module of ORA and PEAS

a) PEAS will be using external auto grader service for doing auto grading
b) PEAS will be using Discern API of EDX to use its EASE auto grading library for essays.
c) Also PEAS will incorporate programming language compilers to auto grade programming codes.
EdX mainly makes use of the following two database management systems:

1. **SQLite**: a lightweight, relational database management system that implements most of the SQL standard.
2. **MongoDB**: an open source, document oriented, “No-SQL” database management system.

There are 9 tables in the django legacy database initially, which are inherited by the edX database, in addition to its own database tables. The SQLite database of the edX platform itself contains around 85 tables, whose number increases as additional modules are installed.

EdX uses an XML format to describe the structure and contents of its courses. This file gives all the information to uniquely identify a particular run of any course, for example, which organization is producing the course, what the course name is etc.

The second important aspect of the database is the use of Policy files. These files specify metadata about the content elements – things which are not inherent to the definition of the content, but which describe how the content is presented to the user and used in the course.

### 5.1 Internal Data Formats

It consists of details pertaining to how the course structure, student state/progress, and events are stored internally. There are three major components:

1. **Student Info and Progress Data**
2. **Discussion Forums Data**
3. **Tracking Logs**

#### 5.1.1 Student Info and Progress Data

This component includes details on how edX stores student state data internally. This information includes demographic information collected at signup, course enrollment, course progress, and certificate status. The following conventions are used for the same:
1. MySQL 5.1 is used with InnoDB tables.
2. All strings are stored as UTF-8.
3. All dates and times are stored as UTC.

A detailed description of the prominent database tables in this module is provided below.

1. **auth_user**

   The *auth_user* table holds generic information necessary for basic login and permissions information.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>int(11)</td>
<td>NO</td>
<td>PRI</td>
</tr>
<tr>
<td>username</td>
<td>varchar(30)</td>
<td>NO</td>
<td>UNI</td>
</tr>
<tr>
<td>first_name</td>
<td>varchar(30)</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>last_name</td>
<td>varchar(30)</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>email</td>
<td>varchar(75)</td>
<td>NO</td>
<td>UNI</td>
</tr>
<tr>
<td>password</td>
<td>varchar(128)</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>is_staff</td>
<td>tinyint(1)</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>is_active</td>
<td>tinyint(1)</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>is_superuser</td>
<td>tinyint(1)</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>last_login</td>
<td>datetime</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>date_joined</td>
<td>datetime</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>status</td>
<td>varchar(2)</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>email_key</td>
<td>varchar(32)</td>
<td>YES</td>
<td></td>
</tr>
</tbody>
</table>

   Table 5.1: Database table for auth_user

2. **auth_userprofile**

   The *auth_userprofile* table is mostly used to store user demographic information collected during the signup process, and also to store certain additional metadata relating to certificates. Every row in this table corresponds to one row in *auth_user*. 
3. courseware_studentmodule

The courseware_studentmodule table holds all courseware state for a given user. Every student has a separate row for every piece of content in the course.
4. certificates_generatedcertificate

This table tracks certificate state for students who have been graded after a course completes. Currently the table is only populated when a course ends and a script is run to grade students who have completed the course.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Key</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>int(11)</td>
<td>NO</td>
<td>PRI</td>
<td>NULL</td>
<td>auto_increment</td>
</tr>
<tr>
<td>user_id</td>
<td>int(11)</td>
<td>NO</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>download_url</td>
<td>varchar(128)</td>
<td>NO</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>grade</td>
<td>varchar(5)</td>
<td>NO</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>course_id</td>
<td>varchar(255)</td>
<td>NO</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>key</td>
<td>varchar(32)</td>
<td>NO</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>distinction</td>
<td>tinyint(1)</td>
<td>NO</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>status</td>
<td>varchar(32)</td>
<td>NO</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>verify_uuid</td>
<td>varchar(32)</td>
<td>NO</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>download_uuid</td>
<td>varchar(32)</td>
<td>NO</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>varchar(255)</td>
<td>NO</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>created_date</td>
<td>datetime</td>
<td>NO</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>modified_date</td>
<td>datetime</td>
<td>NO</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>error_reason</td>
<td>varchar(512)</td>
<td>NO</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.4: Database table for certificates_generatedcertificate
Fig 5.1: The screenshot of the terminal displaying a list of all prominent tables contained in the edX database, as part of the mitx.db file.
5.2 Discussion Forums Data

Discussions in edX are stored in a non-SQL, MongoDB database as collections of JSON documents. There are two types of objects stored here, namely a CommentThread and a Comment.

A CommentThread represents a comment that opens a new thread – usually a student question of some sort. A Comment is a reply in the conversation started by a CommentThread. The attributes that these two objects share are also included in this category.
5.3 Tracking Logs

Tracking logs are represented as JSON files that catalog all user interactions with the site. To avoid filename collisions, they are organized by server name, where each directory corresponds to a server where they were stored.

The table below depicts its common fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Details</th>
<th>type</th>
<th>values/format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>username of the user who triggered the event, empty string for anonymous events (not logged in)</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>Session</td>
<td>key identifying the user’s session, may be undefined</td>
<td>string</td>
<td>32 digits key</td>
</tr>
<tr>
<td>Time</td>
<td>GMT time the event was triggered</td>
<td>string</td>
<td>YYYY-MM-DDThh:mm:ss.xxxxxx</td>
</tr>
<tr>
<td>Ip</td>
<td>user ip address</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>Agent</td>
<td>users browser agent</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>Page</td>
<td>page the user was visiting when the event was generated</td>
<td>string</td>
<td>$URL</td>
</tr>
<tr>
<td>event_source</td>
<td>Identifies whether the event originated in the browser (via javascript) or on the server (during the processing of a request).</td>
<td>string</td>
<td>browser, server</td>
</tr>
<tr>
<td>event_type</td>
<td>type of event triggered, values depends on event_source</td>
<td>string</td>
<td>more details listed below</td>
</tr>
<tr>
<td>Event</td>
<td>specifics of the event (dependency of the event_type)</td>
<td>string/json</td>
<td>the event string may encode a JSON record</td>
</tr>
</tbody>
</table>

Table 5.5: Common fields of Tracking Logs
Chapter 6

Edx Modules

6.1 Discern

It is an ML service API wrapper that allows anyone to use machine learning based automated classification. This automated classification can work on both free text (essays, content, etc), and on numeric values.

Text and scores (training set) are added in to the API, and after the grading takes place, scores can be retrieved.

6.1.1 Working Mechanism

A user is created to interact with the various API resources provided by Discern. Once the user has logged in, an organisation object and a course object can be created. A fixed number of essay objects and instructor scored essay grade objects are then fed to the API, thus creating a model (training set). Hereafter, each additional essay added to the API, will automatically have an essay grade object associated with it that contains its machine evaluated score.

By default, each user is restricted to only seeing the objects they have created. The python request module has been used to interact with our Discern server, and the response is obtained in JSON, i.e. the Java Script Object Notation.
6.1.2 Structure of the API

The API is structured around models built in Django framework. It is abstracted into model resources by a web service API framework for Django, namely Tastypie. It provides a convenient, yet highly powerful and customisable abstraction for creating REST-style interfaces.

The available models are listed below:

1. **Organization**

   An organization defines a group of users, like a school, university etc. Each organization contains multiple courses, and multiple users.

2. **User**

   A user is the basic unit of the application. Each user will belong to zero or many organizations, and will be a part of zero or many courses. Each user also will be associated with any essays that they have written, and any essay grades that they have done.

3. **Course**

   The course is essentially a container for problems. Each course can belong to zero or many organizations. Each course has zero to many users, and zero to many problems.

4. **Problem**
A problem contains meta-information for a problem, such as a prompt, maximum scores, etc. It contains zero to many essays, and is a part of zero to many courses.

5. Essay

The essay is the basic unit of written work. Each essay is associated with a single problem and a single user. It can have multiple essay grades.

6. EssayGrade

This is the basic unit that represents a single grader grading an essay. Graders can be of multiple types (peer, self, machine, etc), and can give varying scores and feedback. Each essaygrade is associated with a single user (if human graded), and a single essay.

7. Membership

This links between an organization and a user. It has a field to indicate the role of the user within the organization. Currently the roles are “student”, “teacher”, and “administrator.” The first user who creates an organization will automatically be made an administrator.

6.1.3 ML grading

This is an important functionality provided by Discern along with its utilities as an API. It has the following main components:

1. ML Grader

The ML grader calls on the machine learning algorithm to grade a given essay. It can load a model file which is created by a logged in user, and can also save a created model to a local file.

2. ML Model Creation

This module provides scripts to generate a machine learning model from input data, for a given problem. It can also save a machine learning model to file or upload results of the grading, as per the requirement.

3. ML Grading Utility Functions

This module performs many functions, such as:

- It gets the currently active model file for a given problem (django model) and target number problem.
- It can create a directory for a file if it does not exist.
- It can dump input data to a file, and can also generate a path from a problem and a target number problem.
4. **Grading Tasks**

This is a module used by celery to decide what tasks it needs to do.

### 6.2 XQUEUE

XQueue is an interface for the LMS to communicate with external grader services. It receives the submissions from the LMS and forwards it to the external grader.

For example, when a student submits a problem in the LMS, it gets sent to the XQueue. The XQueue then has the problem graded by an external service and sends the response back to the LMS.

**INTERACTION OF LMS WITH XQUEUE**

1. The LMS pushes student submissions to the XQueue with an HTTP POST request to the URL `/xqueue/submit`. The submission contains a callback URL indicating where the graded response should be sent.

2. When the submission has been graded, the XQueue pushes a response back to the LMS with an HTTP POST request to the callback URL.

The following figure describes the mechanism.

![Fig 6.2: Interaction of LMS with XQUEUE.](image-url)
INTERACTION OF EXTERNAL GRADERS WITH XQUEUE

There are two ways kinds of grading services: passive and active. These interact with the XQueue in different ways.

**Passive Graders**

Passive graders wait for the XQueue to send them submissions. They then respond synchronously with a graded response.

1. The LMS sends messages to a particular queue.

2. XQueue checks its settings and finds that the queue has a URL associated with it. XQueue forwards the message to that URL.

3. The passive grader receives a POST request from the XQueue and responds synchronously with the graded response.

4. XQueue forwards the graded response to the callback URL the LMS provided in its original message.

**Active Graders**

Active graders pull messages off the XQueue and push responses back to the XQueue.

1. The test client sends messages to a particular queue
2. The active grader polls the XQueue. When it receives a submission, it pushes a response back to the XQueue.

3. XQueue pushes the response back to the LMS.

Fig 6.4: Active Graders

6.2.1 RabbitMQ

Introduction

RabbitMQ is a message broker software which implements the Advanced Messaging Queuing Protocol 0.9.1. It is easy to use and runs on all major operating systems. The software is also open source and commercially supported. It also supports a huge number of developer platforms.

Advanced Messaging Queuing Protocol
It is an open standard for passing business messages between applications or organizations. It was designed for security, reliability, standard, open and interoperability.

It is a binary, application layer protocol, designed to efficiently support a wide variety of messaging applications and communications patterns. It provides flow controlled, message-oriented communication with message-delivery guarantees such as at-most-once (where each message is delivered once or never), at-least-once (where each message is certain to be delivered, but may do multiple times) and exactly-once (where the message will always certainly arrive and do so only once). It assumes an underlying reliable transport layer protocol such as Transmission Control Protocol.

**Common Terminologies Used In RabbitMQ**

**Producer**

A producer is a program that sends message to a broker. It is represented with a “P”.

**Consumer**

A consumer is a program that receives a sent message. It is represented by a “C”

**Queue**

A queue is the name for a mailbox. It is used inside RabbitMQ. Although messages flow through RabbitMQ and your applications, they can be stored only inside a queue. A queue is not bound by any limits, it can store as many messages as you like – it’s essentially an infinite buffer. Many producers can send messages that go to one queue, many consumers can try to receive date from one queue. It can be represented as
6.3 XSERVER

XServer accepts student code submissions from the LMS and runs the code using (separate) courseware graders.

1. It is the Python code grader subsystem.

2. It receives grading requests from xqueue, passes the grading through the grader logic, and returns the response to xqueue.

The grader logic is specific to each problem, and thus a substantial part of the python code grader.

Contains submitted assignments

---

6.3.1 JSON

- JSON or JavaScript Object Notation.
- The JSON format was originally specified by Douglas Crockford.
- JSON is syntax for storing and exchanging text information.
- JSON has been extended from the JavaScript scripting language.
- JSON is language independent.
- Similar to XML, rather easy ad smaller.

6.3.2 JSON PROTOCOL
• **JSON-RPC** is a Remote procedural call protocol encoded in JSON.

• **JSON-PROTOCOL** allows for notifications and for multiple calls to be sent to the server which may be answered out of order.

• JSON data is used here, as an easy way to pass assignment to the grader.

---

**REQUEST**

![Diagram of REQUEST](image)

**RESPONSE**

![Diagram of RESPONSE](image)

**Fig 6.6: MECHANISM OF HTTP JSON PROTOCOL**

The **Client** is defined as the origin of Request objects and the handler of Response objects.

The **Server** is defined as the origin of Response objects and the handler of Request objects.
REQUEST AND RESPONSE

XQueue pushes student submissions to your server using an HTTP JSON protocol.

- The request will have a JSON object as a body, with these keys:
  "xqueue_body": A string, a JSON-encoded object with these keys:

- "student_response": A string, the student's response. The main task of the grader is to determine the correctness of this response.

- "grader_payload": A string. It will be taken from the content of the <grader_payload> XML tag in the problem.

Eg: {
   "xqueue_body":
     {
     "student_response": \"def double(x):\n         return 2*x\n     \\
     "grader_payload": \"anything_you_like\"\n     
   }
}

The response must have a JSON object as a body, with these keys:

"correct": true or false

"score": A numeric (int or float) value to assign to the answer, we often use 0 or 1.

"msg": An HTML string, will be shown to user

The sample response could look like this:

{ "correct": true, "score": 1, "msg": "<p>Great! You got the right answer!</p>" }
6.3.3 SECURITY MECHANISMS:

6.3.3.1 SANDBOX:

A sandbox is a testing environment that isolates untested code changes and outright experimentation from the production environment.

=>record_suspicious_submission()

=>code_has_suspicious_statements()
Chapter 7

Experimental Results

This chapter contains the different experiments conducted during our project

7.1 RabbitMQ

Experimental Setup

The experiment was conducted to test the efficiency of RabbitMQ. A program (Send.java) was written to send 50000 messages to a queue and another program (Recv.java) was written to receive those messages.

We observed that if the program Recv.java was executed before Send.java then the rate of sending was equivalent to the rate of receiving.

![Graph showing that the rate of publishing and receiving messages is almost equivalent](image)

**Fig 7.1:** Graph showing that the rate of publishing and receiving messages is almost equivalent

If the program Send.java was executed before Recv.java we observe that the messages got queued.

![Graph showing that 50000 messages got queued and are ready to be sent](image)

**Fig 7.2:** Graph showing that 50000 messages got queued and are ready to be sent
Fig 7.3: Graph showing that the rate of publishing is around 220 messages per second

![Graph showing the rate of publishing]

Fig 7.4: It can be shown that 50000 messages got queued in the queue

Now the program Recv.java was executed

It can be seen that all the queued messages got delivered

![Graph showing queued messages]

Fig 7.5: Graph showing that the queued messages got delivered in around 10 seconds
7.2 Normalisation Technique

Let us take an example of 6 students: A, B, C, D, E and F. Suppose that the instructor wants each assignment to be evaluated by 3 students, which means the students are to be divided into 3 groups (2 students in one group) after ordering them by incentive.

Assume that the groups are created as: A, C in Group 1; B, F in Group 2; D, F in Group 3.

<table>
<thead>
<tr>
<th></th>
<th>Reviewer</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>IER</th>
<th>IWF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>22</td>
<td></td>
<td></td>
<td>1.27</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>6</td>
<td>7</td>
<td>3</td>
<td>16</td>
<td></td>
<td></td>
<td>0.92</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>9</td>
<td>8</td>
<td>3</td>
<td>20</td>
<td></td>
<td></td>
<td>1.15</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td>4</td>
<td>9</td>
<td>3</td>
<td>16</td>
<td></td>
<td>0.92</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td>7</td>
<td>3</td>
<td>0</td>
<td>10</td>
<td></td>
<td></td>
<td>0.58</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td>5</td>
<td>9</td>
<td>6</td>
<td>20</td>
<td></td>
<td></td>
<td>1.15</td>
<td></td>
</tr>
<tr>
<td>Total rating given to others</td>
<td>22</td>
<td>19</td>
<td>16</td>
<td>19</td>
<td>16</td>
<td>12</td>
<td>104</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Since the bias factor for all the students does not fall in the required range, i.e. 0.98 to 1.08 we will multiply the marks given by A to others (B, C and E) with his normalization factor (0.78). This will be done for all the students, and a new table is created.

<table>
<thead>
<tr>
<th></th>
<th>Reviewer</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>IER</th>
<th>IWF</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.1: Calculation of normalization factor
Table 7.2: Obtaining final marks of students

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>5.4</td>
<td>8.28</td>
<td>8.7</td>
<td>22.38</td>
<td>1.29</td>
<td></td>
</tr>
<tr>
<td>Total rating given to others</td>
<td>17.16</td>
<td>17.48</td>
<td>17.28</td>
<td>17.48</td>
<td>17.44</td>
<td>17.4</td>
</tr>
<tr>
<td>AER(104.24/6) = 17.37</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bias Factor</td>
<td>0.98</td>
<td>1.006</td>
<td>0.99</td>
<td>1.006</td>
<td>1.004</td>
<td>1.001</td>
</tr>
<tr>
<td>Normalisation Factor</td>
<td>1.02</td>
<td>0.99</td>
<td>1.01</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
</tr>
</tbody>
</table>

Since all the bias factors fall in the required range, we stop our calculation here. The marks given to these students are the new normalised marks.
Chapter 8

Conclusion
Chapter 9

Future Work
References


[10] Peer Assessment in University Teaching: Evaluating seven course designs


[16] www.rabbitmq.com