AUTOMATED QUIZ GENERATION AND EVALUATION

Dr Sujit Kumar Chakrabarti
Ph.D. (IISc Bangalore)
Prof IIIT Bangalore
sujitkc@iiitb.ac.in

Keshav Gupta
M.Tech CSE
IIIT Bangalore

Eshita Sharma
M.Tech CSE
IIIT Bangalore

Lubaina Machinewala
M.Tech CSE
IIIT Bangalore

CONTEXT

It is an automated quiz generation and evaluation tool which is used to conduct a quiz with different formats such as match the following, multiple choice questions etc. Instructors can input all the details in one json file to set the quiz for the entire class and folders can be generated for each student as specified by the instructor. This software includes generation of assessment instrument from item bank mapped randomly for each student in pdf format, graphical user interface developed to input responses from the student which can be done without using any external server and automated evaluation of quiz by fetching the previous mapping and generating the report

IMPLEMENTATION

This software supports two types of objective question quizzes:

1. Simple quiz
   In this quiz, the students are supposed to answer all questions. All students answer the same question paper
2. Jumbled quiz
   In this form of quiz, each student gets a different question paper to answer, which contains questions sampled out of a larger item bank and jumbled.

Question Types

There are two types of questions that can be included in the question papers: Multiple Choice Questions (MCQ) and Match The Following (MTF). An MCQ is a question with two or more possible options out of which one or more may be correct choices. For example:
Question - Name two dynamically typed programming languages:

1. Java
2. Python
3. Haskell
4. Ruby

In the above, options 2 and 4 are correct answers.

The second question type is Match The Following(MTF), For example:
Match the following programming languages on the left column with properties on the right.

1. Python	A. Static typing
2. Java	B. Dynamic typing
3. C++	C. Untyped
4. OCaml	D. Implicit typing
	E. Explicit typing
	F. Functional
	G. Object oriented

In the above example, the matches are as follows
1. Python matches with B., D., F. and G.
2. Java matches with A., E. and G.
3. C++ matches with A., E. and G.
4. OCaml matches with A., D., F., and G.

A few points to note here:
• The mapping can be many to many, even allowing some options on the either side to have no matches at all (e.g. C.)
• There is no restriction on the number of options on either columns. For example, in the above example, we have 4 options on the LHS and 7 on the RHS.

Simple Quiz

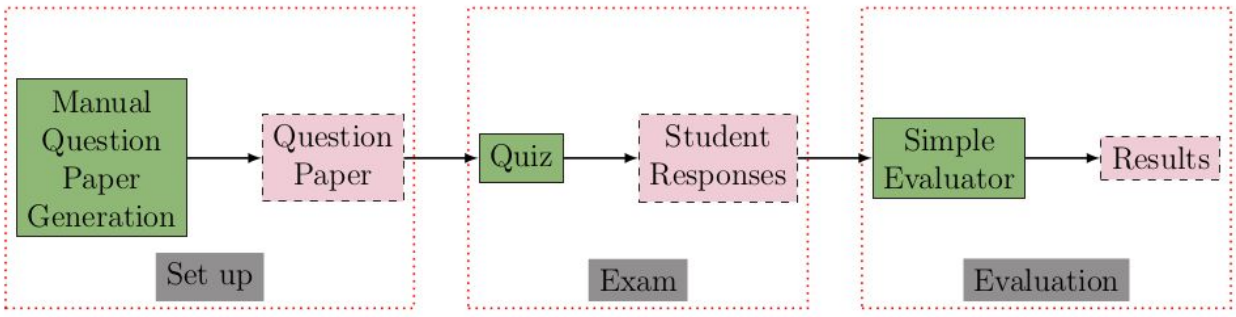

Fig 1: Simple Quiz Workflow

The question paper creation for a simple quiz is manual. All students solve an identical question paper. The evaluation step directly runs on the student responses using the SimpleEvaluator module.

Setup

Suppose you intend to conduct a simple quiz. As mentioned above, in a simple quiz, all students solve the same question paper. Such a quiz is ideal when there is small class and enough assurance that cheating is not a possibility. Of course, all questions are assumed to be either multiple choice questions (MCQ) or match the following questions (MTF). The assessment project can be set up conveniently using the setup utility. The setup utility automatically generates the skeletal infrastructure for conducting such a quiz.

Jumbled Quiz

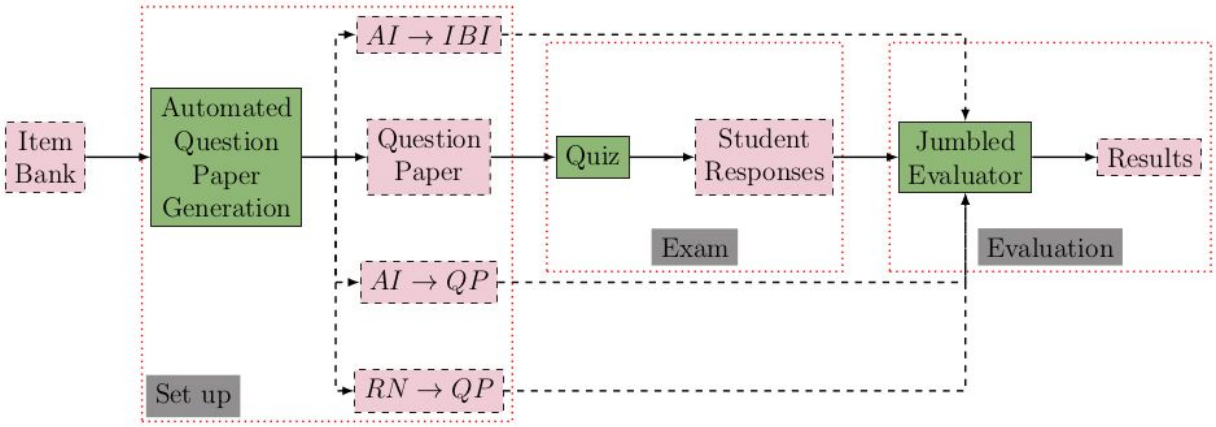

Fig 2: Jumbled Quiz Workflow

The question paper creation step uses the genAIs (generate assessment items) and genQPs (generate question papers) modules. Once the question papers are generated, a quiz is conducted. Finally, the automated evaluation process takes place using the JumbledEvaluator module.

TECHNICAL DETAILS

Python is used to construct this software along with Tinker for graphical user interface and Pdf Latex module for generating the question papers.

To discourage cheating in the class, we generate a set of question papers by randomly selecting n questions out of an item-bank of N questions. A set K of distinct assessment instruments are generated, numbered 0, 1, ..., |K − 1|, we call them assessment instruments.

The question paper generator module G generates a set C of codes each of which can be mapped to any one of the assessment instruments of K. Each of the code c in C is finally mapped to one

distinct question paper with c printed on it. This way, the students will not be able to identify which assessment instrument k ∈ K their copy of the question paper belongs to. Each question paper will have an empty table called the response table on page one which will be used by the student to fill in his responses.
G also generates a map from assessment instrument to question order. This tells us the original question number of each item in the given assessment instrument.

0	3	5	1	4	9	12	15	2	10	2
1	4	1	2	11	6	7	5	14	8	12
	…									
9	5	12	2	1	6	7	3	11	8	10

Figure 3: Assessment Instrument Item to Item Bank Item map AII→ IBI

For example: Figure 3 shows a possible mapping from assessment instruments to item bank items. The table can be interpreted as follows: There are 10 assessment instruments numbered 0 through 9. For each assessment instrument AI ∈ K (here |K| = 10), there is a row in the table. Each cell in that row has the item bank item number for that item. For instance, for AI = K[0], AI[0] = 3, AI[1] = 5 and so on. This module will generate a map – called QP → AI between question paper code to assessment instrument. For example: QP → AI = [0, 1, 2, ..., 9, 0, 1, 2, ...] could be one such mapping. It says that QP→ AI[0] = 0 (i.e. the question paper with code 0 maps to assessment instrument number 0). Similarly, QP → AI[11] = 1 (i.e. the question paper with code 11 maps to assessment instrument number 1) and so on.

EVALUATION

The response rearranger refers to the RN → QP and QP → AI map to extract the assessment instrument for each roll number. Using this, the evaluator rearranges the responses in the order as per the item bank to create a rearranged response for the roll number n, R’ (n). This is given to the evaluator for final automated evaluation

This software was used to conduct a Python Quiz for M.Tech 2020 Batch in IIIT Bangalore. We created an item bank of 55 questions and each assessment item was mapped with random 20 questions from the item banks. Question paper and required files such as instructions, python script for graphical user interface (to record the response) were generated for each of 200 students, later the responses were recorded from the student and evaluated after fetching the previous mapping of assessment items.

LIMITATIONS

We are eliminating the use of external server to conduct the quiz so it leads to some limitations such as

- Since everything is happening on a local student's machine, we can't impose a sectional time limit in the quiz.

- It require certain system requirements to conduct the quiz

DEMONSTRATION

The only file here which is important to notice is the configuration file config.json.
Every quiz will have its own configuration file with all the relevant meta-data pertaining to the quiz. Its contents are as follows :

1. course name - Name of the course
2. course code - Course code as prescribed by the university
3. assessment name - The exam name. Here it's "demo", but in practice it could be something like "Quiz 1", "Mid-term Examination" etc.
4. assessment type - This is the quiz type. Currently, there are two types available: Simple and Jumbled.
5. roll number file - "path for class.csv "
6. assessment home "path"
7. Number of items per assessment instrument - Number of questions you want on your question paper. Its relevance is based on the quiz type. In particular, a jumbled quiz will generate question papers each having so many questions on it. Simple quiz will simply ignore this field**.**
8. items - This is the list of questions in the item bank. Here, we list all items named item1, item2, item3 and item4 and so on. It further consists of
    a. Name
    b. Properties
        i. Qtype - MCQ/MTF
        ii. Number of options
        iii. Range - Optional(required in case of MTF)
        iv. Marks

After filling this config.json the instructor can do following things such as:

Generate the quiz

This generates all the necessary files for the quiz. Let's take a look at the contents of the directory:

1. assessment-instruments: This is the directory created – currently empty – to contain the question papers once they are generated.
2. config.py: This is the file which contains the Python translation of the contents of the config.json file.
3. evaluation: This is the evaluation directory. The evaluate.py file is the evaluator file which will be run once the quiz is conducted and submissions have come in.
4. gen.py: This script needs to be run next to generate the quiz.
5. item-bank: This contains the item bank, in the form of stubs of LATEX text. These need to be filled up to create the question papers. Note that there are five item stubs created, since that's the number of items mentioned in config.json. For example, the contents of the item1.tex is:

   \ question
   \ label { q : SE101 : demo : item1 }
   \ begin { enumerate }
   \ item option 1
   \ item option 2
   \ item option 3
   \ item option 4
   \ end { enumerate }
6. packages: When the gen.py file is executed, it is this directory – currently empty where the question papers will be generated.

Generate the Question Papers

The main effect of running this script would be that the packages directory is no more empty. It contains one directory for each roll number. In there, among other files, you will find the question paper for that roll number: packages/rn1/rn1.pdf, packages/rn2/rn2.pdf, packages/rn3/rn3.pdf and packages/rn4/rn4.pdf.

Administer the Quiz

At the time of administering the quiz, these question papers can be shared with the students using an appropriate external method, e.g. email, LMS or a shared drive. This is outside the preview of this system.

At the end of administering the quiz, the submitted answers are provided in the submissions directory in a particular format. In this demo, we simulate the process of answer submission by running the backup script provided.

Evaluate the Quiz

The result of evaluation is stored in the result.csv file in the evaluation directory.

REFERENCES

[1]
[2]
[3]