Abstract

Studying quizzes without a strategy can lead to unsuccessful and frustrating results. The goal of this capstone project is to use artificial intelligence techniques to generate quizzes tailored to the user. A chosen topic will contain a database filled with an abundance of questions and information relevant to each question. The user will be given a quiz on the topic, and the results of the quiz will be sent to an artificial intelligence engine. This engine will use different techniques to generate new quizzes that focus on certain aspects based on received data and targeted characteristics. The end results of this project should be endlessly generating quizzes that assist the user in learning efficiently.

Cody Perdue, Dr. Gary Newell

Dr. James Buss

Honors Capstone Project

7 May 2021

Using Artificial Intelligence Techniques to Generate Quizzes Tailored to the User

Today, students often have a hard time memorizing everything there is that they need to learn in order to succeed in their classes and further yet out into the real world. With all the info they are cramming into their heads, it can make recollection difficult to accomplish. To assist in helping memorization, many different methods have been formed. From flashcards to quizzes to spaced repetition, many different ideas attempt to ease the struggle of learning. At the same time, artificial intelligence is being improved constantly in the technological field. This type of technology aims to simulate the intelligence of a human brain through a machine. With this project, my mentor, Dr. Gary Newell, and I wanted to attempt combining these two concepts of enhanced learning and artificial intelligence by creating quizzes that are optimized for the user to learn most efficiently.

The first component necessary for this project is the quiz that the user will be taking. We wanted to craft this quiz similarly to other learning devices such as Quizlet. The quiz will have a main topic, and each question must have three extra components along with the question and answer: the subtopic, the level, and the type of question. The subtopic is simply breaking down the main topic into smaller parts. The level can be seen as the difficulty of the contents of the question. The type of question refers to the format of the question, whether it be a multiple choice, matching, or other type of problem. To have our project for any topic, we set the structure so that the user supplies the total number of questions, subtopics, levels, and types as well as the lists of all the questions containing the required parameters. The input for the program will read in all the questions from the database that have been passed to it, along with a file that gives the number of problems per subtopic, level, and type to be put into the quiz. On the first run of the quiz, these numbers should be equal between each subtopic, level, and type to create a neutral all-rounded quiz. After the quiz has been generated following the given parameters, the user will then proceed to take the quiz. Once they have answered all the questions, their results will be shown on screen as well as sent to a new output file that will be used to generate the next quiz.

The output file is sent to the other major part of the project: the artificial intelligence (AI) algorithm. This algorithm computes the number of questions that will be passed to the next quiz for efficient studying. This algorithm can be adjusted to target specific categories; for example, subtopics that the user is having difficulties with can be focused on in the next quiz, or the average difficulty of the questions can be adjusted depending on the user’s score. Through this targeting, we can attempt to maximize the efficiency of a user’s studying. The algorithm performs its analysis on the data and produces the next set of input for the quiz using statistical methods. We performed research on different artificial intelligence algorithms and planned to use different methods, but due to time and complexity constraints, we had to focus on simpler means.

The AI algorithm techniques we researched and initially planned to use together were Bayesian analysis, genetic algorithms, and artificial neural networks. The first technique used would be Bayesian analysis, a probabilistic AI technique which uses previous results to predict future behavior. The next step would be genetic algorithms, which aim to simulate genetics by mimicking ideas such as populations, mutations, and evolution to produce and choose new output. The last step would be artificial neural networks, which are modeled after the brain and learn via training neurons with strengthening and weakening connections. These techniques would operate in tandem to form a powerful artificial intelligence algorithm that would lead us closer to maximizing the efficiency of student learning through our quizzes.

To create this project, we programmed everything from the ground up. To do so, we used the Java programming language, as that is one of the most familiar programming languages to most programmers and is a primary focus of our computer science education here at Northern Kentucky University. Java is an object-oriented programming language, which means ideas and concepts that one wants to program is created through what are known as objects; this lets us easily structure our ideas into organized code. The “Quiz” class is the file that operates inputting quiz information, delivering the quiz to the user, and outputting results to the user and a new file. In the beginning, two arrays are created: one that holds all the questions from the input files and one that holds all the categories (subtopics/levels/types) that the questions will have. These arrays hold “Question” objects and “Category” objects respectively; these objects are managed from the “Question” and “Category” class and allow us to define, store, and access important information to and from each. The “ImportNumbers” and “ImportQuestions” functions are passed the file names and grab the relevant info from said files. The “CheckParameters” function will ensure that the values passed from the input file are correct and will not cause any errors, then the quiz is generated by the “GenerateQuiz” function. This function will grab questions from the complete pool that fit the quiz criteria, add them to a new quiz array, and will pass the array to the “TakeQuiz” function. This final function will give the questions from the array one at a time to the user, and after the questions are all answered through the command line interface, results will be presented and exported. A few helper functions such as “contains” and “getCategory” ensure clean and efficient code.

The other main component, the “Algorithm” class, takes the results and intelligently creates the next set of input for a future quiz. The file will compute the difficulty of the previous quiz and present it along with the percentage the user scored correct through the “getdifficulty” and “getdifficultyresults” functions. Next, “getpercents” and “testpercs” will get the percentages for the number of problems and print it to the screen. Finally, the program will then generate the quiz in “genquiz”, using the calculated percentages to judge how many questions total and how many per type should be added; this is influenced by the user’s weak points. The “Quiz” class and the “Algorithm” class can be called alternatively to continue the process of creating user-tailored quizzes.

To give an example of how this whole project works, I will explain how we did everything during our building and testing. For our chosen topic, we picked Japanese, as that has been a recent interest of mine that I have been independently studying for a while now. I created a list of 480 questions split equally into five subtopics: hiragana, katakana, radicals, kanji, and vocabulary. I equally split these questions into four separate levels for computing difficulty, simply labeled “Level1” through “Level4”. Finally, I split them into four different question types – true or false, matching, multiple choice, and fill in the blank – while making sure the questions and answers were also formatted similarly. An input file was also created that gives the number of subtopics, levels, and types as well as the number of questions for each, initially set to be all equal for a total of 48 questions. I had the quiz program read these files and store the data, generate the quiz, and give the quiz to the user. These results were shown to the user and saved to the output file, which was then ran through our algorithm program and output new numbers for the next quiz. Our algorithm is currently calculated towards targeting user difficulties, so weak topics are focused on and the quiz is set to adjust to suit the user (a lower score should lower the difficulty). Certain types of questions, such as fill in the blank, are also noted as being more difficult and accounted for in our algorithm. The user can continue to take these personally generated quizzes unlimited times, the goal being to continue until they are comfortable with the subject matter.

All in all, I am proud of what we were able to create for this capstone project. Due to a mixture of time and complexity constraints, and mostly due to a late start, a lot of planned ideas were unable to be executed. Given more time, I believe we could have created a more advanced algorithm with interesting and useful mechanisms that would both improve the effectiveness of our project while simultaneously giving me more skills and experience to take with me on future endeavors in my career. Nevertheless, I believe what has been made shows a solid first step at a useful process utilizing artificial intelligence that can be expanded upon to improve a user’s studying habits and learning capabilities.