Automated Formative Assessment and Comprehension Analysis: Teacher's Pet

Sameer Singh, Andrew Dertli, Monish Ramadoss, Kevin Norgaard, Nik Hammon

Abstract—**Teacher's Pet is a tool for automated formative assessment and comprehension analysis. Its design goal is to improve communication, understanding, and education in classroom or corporate training settings with a low-cost and innovative solution leveraging natural language processing (NLP) algorithms and serverless cloud computing. The results of our project are the completed product which is composed of the integration between several resources. We have created CloudFormation (CFN) templates specifying our distributed system. Included in these templates are the services such as an application programming interface (API), databases, queues, speech-to-text conversion, a container orchestration platform to handle NLP, and their respective Identity and Access Management (IAM) permissions. Our embedded system captures audio and controls the creation of Teacher's Pet sessions. Our user interface (UI) built using React with Redux pattern for state management allows users to answer the automated assessment in near real time. We have trained a machine learning model to preform natural language processing and question generation. Our methods involved the use of Amazon Web Services (AWS) CFN to enable repeatable deployments across our team members' accounts and using GitHub's organizations so we can collaborate seamlessly with our integration. Using AWS also simplified integration between the embedded system, the transcription service, the image used for machine learning, and the UI. This allowed for the distributed systems, the UI, the NLP algorithm and model, and the hardware component to be built in parallel.**

Index Terms – Amazon Web Services, CloudFormation, Comprehension Analysis, Distributed Systems, Embedded Systems, Formative Assessment, Natural Language Processing, Serverless Containers, Question Generation

I. Introduction

RESEARCH shows that effective teachers are the most important factor contributing to student achievement [1]. However, the current age of education requires proof to show true success [2]. Even good teachers face difficulty with knowing to what extent their students were able to understand their lectures. A common solution to determine student comprehension is to give summative assessments that test the student's understanding. However, after students have taken a summative assessment, it is usually too late for a teacher to do anything if the students failed to understand the material. As an alternative, formative assessment, which gathers feedback immediately on whether or not a class of students understood the material, should become more commonplace as a best practice in education systems. One of the most frequently cited works on formative assessment is the research review conducted by Black and William in 1998 who after synthesizing over 250 publications concluded that formative assessment is perhaps the most effective educational practice when it comes to improving academic achievement [3]. The research shows that formative assessment has numerous benefits including teachers achieving greater personal satisfaction and increased student engagement in learning. The multiplicity of practices that constitute formative assessment is limited to methods relying on teachers to put in additional work to ensure the assessments appropriately match their lesson plans. Provided this work could be automated and still cater to a personalized lesson plan, then teachers could still have the benefits of formative assessment but without the overhead of extra preparation ahead of time. There are other disadvantages associated with current methods of formative assessment such as teacher having to sacrifice time to assess during the lesson causing them to fear they might not finish it, or teachers may lack the proper training on how to implement formative assessment [4]. Our solution is to ameliorate these difficulties with formative assessment by introducing a comprehension analysis tool, Teacher's Pet, for use in classrooms that will automate formative assessment using audio capture and natural language processing and give teachers immediate feedback after a lecture on the extent to which their students were able to learn each topic.

Teacher's Pet was developed around the following goals. It should be able to deliver curated formative assessment in real time. It should function without requiring the teacher to sacrifice any of their lesson plan time. It should be compatible with existing microphone and sound recording technology in a classroom so that it may be easily setup and integrated with existing equipment. It should use low cost materials and serverless technology to reduce the price for consumers. Its hardware should appear aesthetically pleasing and classroom friendly for students of all ages. Lastly, the UI should be simple and easy to access from any electronic device with internet access.

While it is important to continue understanding student comprehension through summative assessment, we instead chose to implement deliberate formative assessment to hold students accountable for mastering material [2]. Some of the existing technology-powered formative assessment tools such as Edulastic analyze data instantly and track student understanding through delivering questions. Similarly, we

want to deliver a high standard of organized data visualization and to provide flexibility to teachers in the subject matter they can choose from. A limitation of Edulastic is it only contains material for English and math so far. Teacher's Pet will be able to deliver formative assessment that is directly extracted from the lecture that a speaker has given. This way, the speaker has the opportunity to improve how they teach a specific area as well as revisit any topics that were misunderstood before the session has finished.

Another formative assessment solution such as Nearpod allows teachers to create original multimedia presentations that they can embed multiple-choice questions into. When giving a presentation Nearpod allows teachers to interact with students and view their responses in real time. It is our goal to provide engagement, formative assessment, and content to a classroom the same way that Nearpod does. Unlike Nearpod, we want to save time and effort for teachers by keeping them from having to go through the preparing of embedded questions and have automated questions delivered to their students in real time. Clip, one activity of a suite from Spiral, an engaging and easy to use tool that support formative and summative assessment, allows teachers to create interactive videos that stop periodically to check understanding. Similarly, we want to allow all students, even the reluctant speakers to have a voice.

In order to complete this project, we divided the project into multiple tiers, each containing services that performs specific functions and that teammates worked on in parallel. The use of GitHub Organizations provided us with a way to keep all of our repositories together and grant us all equal access privileges as contributors. The use of AWS CFN templates and GitHub was especially powerful as each member of our team could deploy stacks to CFN and test the entire system at their leisure. Together these methods helped us reach our result, made collaboration seamless, and made finishing within our timeframe possible. Our goal was twofold, first we designed and created the infrastructure necessary to run our project. This was centered around running an NLP algorithm in a container. We wanted to make sure that we could build a distributed system that all the subsystem we developed could easily integrate with. The integration of the individual subsystems was the second goal. This goal was reflected in the methods we stuck with during the development of the project. These methods were to design for a long period of time and implement gradually, building only what we needed to establish a proof of concept as well as an infrastructure we could develop on top of. As a result, our finished project achieves a fully integrated backend infrastructure powered by services described in AWS CFN templates, an embedded system with the capabilities to maintain state and capture audio, a trained model used to generate questions based on NLP algorithms, a docker image to run the NLP algorithms, and a UI styled using cascading style sheets (CSS) for user to submit responses to.

II. Main Body

In order to build Teacher's Pet, we divided the functionality necessary to achieve this product into multiple tiers, each containing orthogonal services that serve a specific function.

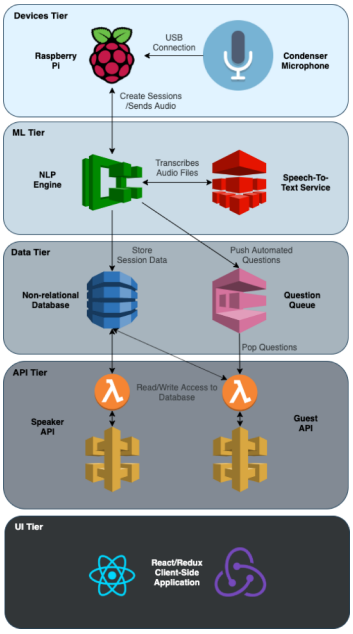

Fig. 1. Tier Hierarchy for Teacher's Pet.

Each member of our team specializes in a different engineering area and therefore worked to develop their own subsystem in an individual tier. Tiers contain multiple services that all communicate with one another in a distributed manner.

A. *Materials Used*

Teacher's Pet is comprised of an embedded system and a distributed cloud system. The hardware components are a Raspberry Pi, a liquid-crystal display (LCD), two buttons, and a light-emitting diode (LED). Additionally, a Prusa i3 MK3 was used to print the animal shaped cases out of polylactic acid, a polymer used for 3D printing. Velcro hold the 3D printed case together and the electronics are hot glued to inside. Our distributed cloud system is built using AWS. This requires the ownership of an AWS account and CFN Template to specify the services that are run on Amazon's hardware. The services used are Dynamo Database (DDB) Tables, Simple Queue Service (SQS), API Gateway (APIG), AWS Lambda, EC2 Container Service (ECS), AWS Transcribe, Virtual Private Cloud (VPC), Simple Storage Service (S3).

B. *High-level Hardware and Software System*

1) *Audio Capture*

Speech audio is captured by a USB-microphone in Advanced Linux Sound Architecture (ALSA). Gstreamer is an open-source, multimedia framework used to pipeline the audio components. The input from the microphone is encoded into Free Lossless Audio Codec (FLAC) format, chunked into 1 MB sequentially named files, and stored in a folder.

2) *Embedded Devices*

The embedded system involves a raspberry pi and auxiliary circuitry components including buttons, LED, and an LCD. The system's logic is implemented as a state machine on the raspberry pi that allows the user to preform actions such as starting, stopping, and creating a Teacher's Pet session with the press of a button. The 3D printed case for the device to

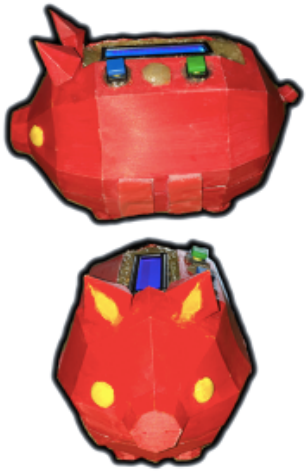

Fig. 2. Completed Embedded System Assembly for Teachers Pet.

house all the electronics inside is designed to resemble a pig, giving our project its namesake.

3) Machine Learning Infrastructure

To deploy a machine learning application at scale we used containerization. ECS is the container orchestration tool that AWS provides. It runs inside of a private subnet inside a VPC. Our network configuration that our containers are deployed includes this VPC along with two subnets, a security group, a routing table, and a public load balancer. Our ECS compute instance type is Fargate due to its ability to deploy serverless container keeping our entire stack off of servers. Our NLP algorithm is built into an image and stored in the EC2 Container Repository (ECR) for the Fargate Containers to pull and run it. While writing our own NLP algorithm and model, some of the existing research done on this topic we will be drawing inspiration from has shown that a machine learning model can automatically generate questions from Wikipedia passages using transformers [5]. In addition, the result of a neural question generation from text on the SQuAD dataset shows another method that can produce semantically correct and diverse questions [6].

This tier is also responsible for tracking when an audio file has been uploaded through the use of triggering Lambda functions with an S3 event. To transcribe the text, we used AWS Transcribe to save us time developing an algorithm that already exists. Another S3 event triggers a Lambda that pushes the transcription to a SQS parameter queue and initiates an ECS task that will start by polling the message from the queue and conclude by pushing an autonomously generated question into another SQS.

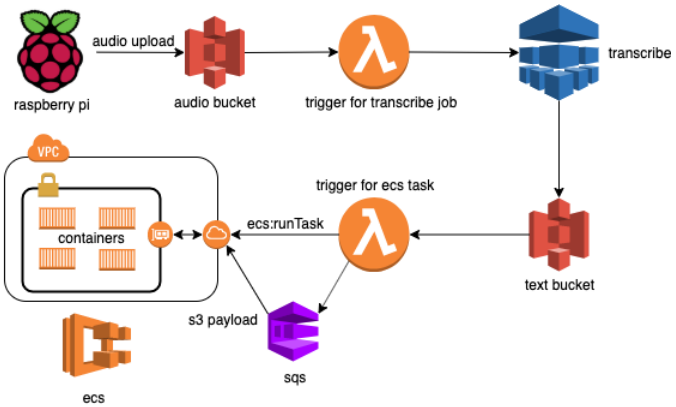

Fig. 3. Machine Learning Tier Cloud Infrastructure.

4) Client-Side Game Mechanics and Polling

The data tier includes a database for storing session statistics and generated questions. It uses Amazon's Dynamo Database, a non-relational database where a simple table can be deployed serverless. The data tier also includes a FIFO queue for each session, powered by Amazon SQS, for storing questions generated by the NLP engine so that they may be polled by our API and delivered to our frontend application. The API tier provides backend logic with Amazon's Lambda functions, serverless functions triggered by events such as RESTful calls to Amazon's API Gateway. All of these services will be deployed via Amazon CFN, a tool for deploying Infrastructure as Code. The UI tier includes a React/Redux application that is hosted through static storage in an Amazon S3 bucket and communicates with our backend by making ajax calls to the API Gateway. The game itself shows listeners questions that they can submit answers for, and it shows speakers the responses to those questions so they can get immediate feedback after a session.

C. Methods

In order to complete our design project, we worked collaboratively, developed individual systems and integrated these systems together to communicate end-to-end. Each teammate designed a different component in the system. Our team used design documents to collaborate and share knowledge between group members. Our design documents were thoroughly detailed discussing the business logic, usage cases for the system, system architecture, and distributed failure cases. We spent four to five weeks solely working on the design phase for our Teacher's Pet. During implementation, it was our goal to reach a stable point at which we could assure the feasibility of our design and continue to embellish features of each individual component in parallel by adding to our codebases. This method of software development is slightly different from building prototypes because the code that we perform the end-to-end testing on will not be thrown out, but instead will form the

Fig. 4. UI design for Teacher's Pet Website. The listeners see questions on the left and the lecturers see the results on the right.

foundation for more code to follow. By implementing this way, we were addressing our concern of having a distributed system designed by individual group members head-on. Once we had all built and understood our subsystems, we began to meet with each other to discuss and implement integration functionality to the project. Finally, we preformed end-to-end testing on the fully integrated system by reproducing a typical use case in which audio is recorded by the embedded system and a question is generated that appears on our thin-client UI.

D. Results

The use of design documents greatly helped get all team members to a better understanding of the work that they would be carrying out as well as the work being done by the rest of the team. Across all tiers in our project, the results were significant. In the devices tier, the state machine was implemented to control when the device would start a session, when it would start recording the session and when it would stop. The devices tier transfers audio files it records and chunks to the machine learning tier with a function for uploading files to an AWS S3 bucket. This bucket triggers a lambda function and the execution is identical to what is described above in machine learning infrastructure. We have fully integrated the NLP algorithm with the infrastructure by building a docker image and publishing it to a repository within ECR, the infrastructure is complete and saved on a CFN template which is under version control in our team's GitHub. Currently, we have developed a thin-client application that is hosted in an AWS S3. Users can make APIs calls that poll questions from a SQS queue that is populated with questions from the NLP engine running in AWS ECS containers. With the individual subsystems integrated across all tiers, Teacher's Pet is capable of recording lectures and autonomously generating questions for users in a styled UI.

E. Performance

The primary constraint that defines the effectiveness of our project is timing. When it comes to NLP question generation, initially we were concerned that the resources allocated by ECS for each container would be insufficient to handle the expensive algorithmic computation. When we tested our system's performance capabilities rigorously, we gained insight into the time consumption for our entire backend to run. The timing of our services is very near real time from the moment that an audio file is dropped into a S3 on the backend and by the time the question makes it into the queue. This shows that the performance of the services we are using will be sufficient to our needs. The containerized technology takes additional time to load and run an algorithm over its model but by launching several containers in parallel we are avoiding this bottleneck to some extent. The overall computation time of our backend was determined to be within our needs for the expected use cases.

III. Summary and Conclusion

Our team worked together through iterative phases of design and implementation and across multiple tiers of our software and hardware stack to complete this design project. By following our procedures, we were all able to gain a clear understanding of the tasks that each member was working on and we were enabled to give guidance among each other while we were designing. The procedure we followed during development focused on achieving end-to-end system integration across all of our distributed services before scaling each service with other features such as styling, distributed error handling, validation etc. This left us with an infrastructure that we could continuously expand upon while we tested that it was meeting our business case, the autonomous generation of questions. We opted for a longer design phase of four to five weeks in order to choose a solution that we hope will be sufficient to deliver our product.

Certain tradeoffs that we made such as using serverless containers could have had devastating results on the timing of our end product. Ultimately, we decided to use this in our project because it allowed us to decisively dive in on implementation, complete the project within our timeframe, and will ideally lead to lowering costs. In addition to decisions that we made due to our bias for action, we also chose to use technologies that were unfamiliar to us but are relevant in the field of software engineering. This is another reason we choose to use serverless containers instead of a dedicated server, and it greatly expanded our appreciation and knowledge of the field. The completion of this project has given each member of our team several new relevant skills in our fields. These skills include the creating serverless backend infrastructure in AWS CFN, building machine learning models, implementing concepts in NLP into an algorithm for question generation, embedded systems design, building and styling a React and Redux application, hosting a thin-client application on an AWS S3, building and deploying docker images in containers, and using a container orchestration platform ECS which requires specific network configurations.

IV. Acknowledgements

The success of Teacher's Pet has come from the hard work of our dedicated team and the advice and guidance of our faculty advisor. We would like to thank our faculty advisor Dr Sameer Singh, a professor at the University of California Irvine working on large-scale and interpretable machine learning applied to information extraction and NLP. Singh has worked with our group to share his experiences doing research of NLP algorithms at scale. He has advised us in what parts of our project are feasible and has reviewed our design documents to see if they were logical. The project was self-funded, with the help of Andrew Dertli, and Kevin Norgaard for providing, configuring, and assembling the hardware resources. The NLP research, model design, and model testing were performed by Monish Ramadoss. The infrastructure and CFN templates were created by Niklas Hammon. The AWS resources were provided with minimal cost due to the low billing for the first year. We are very grateful of each team member as we could not have completed this project without, our advisor, having the necessary resources provided through each other, and having reasonably inexpensive equipment sources.

References

[1] Admin, ERN. "Effective Teachers Are the Most Important Factor Contributing to Student Achievement." *Educational Research Newsletter*

and Webinars, 30 Sept. 2003, www.ernweb.com/educational-research-articles/effective-teachers-are-the-most-important-factor-contributing-to-student-achievement/.

[2] McCrann, John T. "So How Do You Know They Got It? Showing Evidence of Learning." *Education Week - Prove It: Math and Education Policy*, 8 Jan. 2016, blogs.edweek.org/teachers/prove-it-math-and-education-policy/2016/01/how-do-you-know-they-got-it.html.

[3] Andersson, Catarina, and Torulf Palm. "The Impact of Formative Assessment on Student Achievement: A Study of the Effects of Changes to Classroom Practice after a Comprehensive Professional Development Programme." *Learning and Instruction*, Pergamon, 26 Dec. 2016, www.sciencedirect.com/science/article/pii/S0959475216302900.

[4] Sasser, Nesa. "What Are the Advantages & Disadvantages of Formative Assessment?" *Synonym*, 27 June 2018

[5] Kriangchaivech, Kettip, and Artit Wangperawong. *Question Generation by Transformers*. 2019.

[6] Klein, Tassilo, and Moin Nabi. "Learning to Answer by Learning to Ask: Getting the Best of GPT-2 and BERT Worlds." ArXiv.org, 6 Nov. 2019, arxiv.org/abs/1911.02365v1.