Comprehension Analysis Tool [Teacher's Pet]

Team - KAMN
Department of Electrical Engineering and Computer Science

Project Goal

Background: During communication between a speaker and an audience it is difficult for the speaker to know how well the audience understood their speech.

Goal: Design a system that can provide a speaker with a detailed breakdown of how well they are communicating. Speakers will receive feedback on guest comprehension.

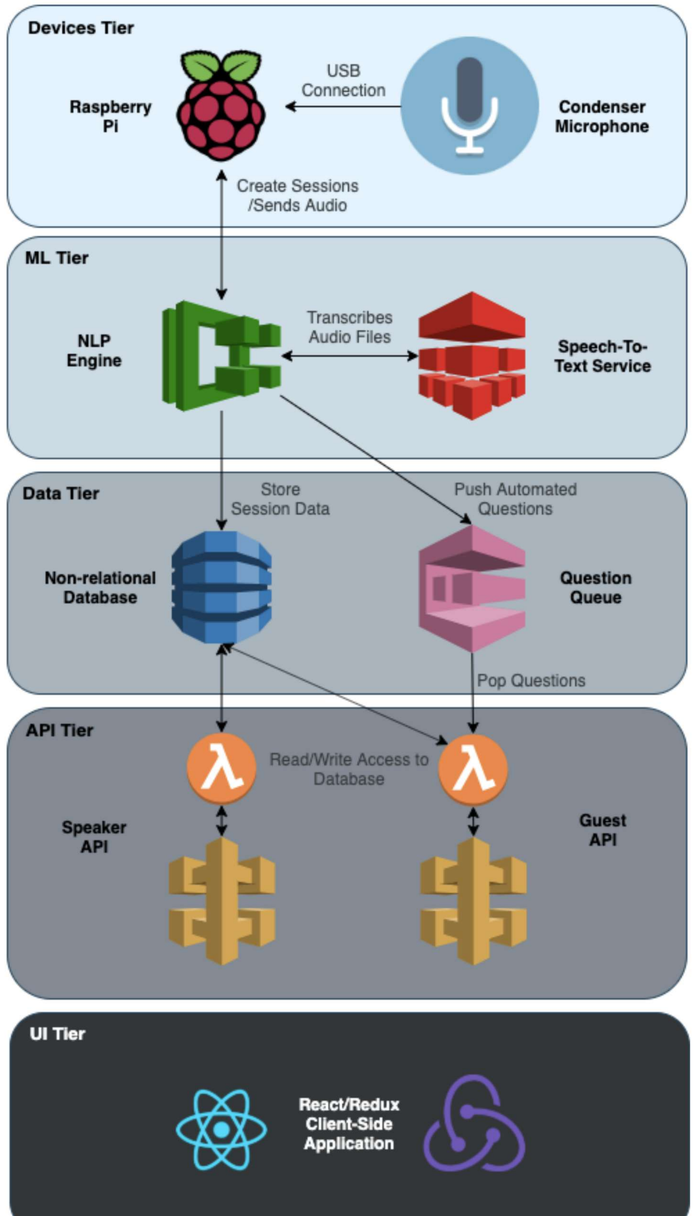
The techniques we will use to understand guest comprehension:

- **Questions** – guests answer questions during lecture
  - Template based - Predefined questions
  - Auto-generated - Relevant questions created autonomously during lectures by **natural language processing** algorithms

The information the speaker will receive as feedback:

- **Topics** that were covered in their talk
  - Which topics guests were struggling in (% of correct responses)

System Architecture



Current Progress

	Task	Nik	Monish	Dertli	Kevin
DESIGN DOCS	NLP Design Doc		✓		
	System Design Doc	✓			
	UI Mock-Ups	✓			
ML TIER	Cfn Template for ECS	✓			
	UI Mock-Ups	✓			
	NLP Enginer Docker Image		✓		
	Question Generation Algorithm		✓		
UI / API / DATA TIER	Backend Lambdas	✓			
	Web Hosting on S3	✓			
	React/Redux App Logic				✓
	Cfn Template for Databases	✓			
	Cfn Template for API Gateway	✓			
	App CSS Styling				✓
DEVICES TIER	3D Modeling		✓		
	State Machine Implementation			✓	
	Full Assembly			✓	
	Audio Capture				✓
	Buttons, LED, LCD Configured			✓	

Results: Our end-to-end system functionality is as follows, (1) An embeded system records and sends audio to transcription service (2) NLP algorithm in a container reads transcripts and posts autonomously generated questions to a queue service (3) Thin-clint application reads questions from queue and simulates a question and answer game for the user
Improvements: (1) To reach more users, we can scale our project by allowing multiple sessions to be run at once (2) To ensure the reliability of our implementation , we can add integration/unit tests (3) To improve the HCI of our project, we can increase instructional information (logging) on the hardware and user interface

UI Guest/Speaker

References

[1] Y.-H. Chan and Y.-C. Fan, "A Recurrent BERT-based Model for Question Generation," pp. 154–162, 2019, doi: 10.18653/v1/d19-5821.

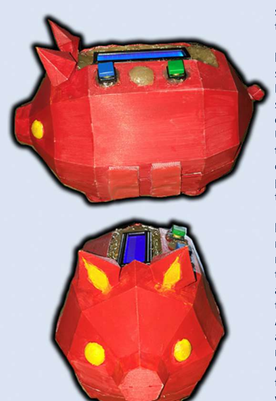
[2] T. Klein and M. Nabi, "Learning to Answer by Learning to Ask: Getting the Best of GPT-2 and BERT Worlds," 2019.

[3] K. Kriangchaivech and A. Wangperawong, "Question Generation by Transformers," 2019.

[4] K. Krishna and M. Iyyer, "Generating Question-Answer Hierarchies," pp. 2321–2334, 2019,z doi: 10.18653/v1/p19-1224.

[5] T. Kwiatkowski et al., "Natural Questions: A Benchmark for Question Answering Research," Trans. Assoc. Comput. Linguist., vol. 7, pp. 453–466, 2019, doi: 10.1162/tacl_a_00276.

Hardware



Implementation: The embedded system (left) is used to help illustrate the process.

First, a speaker will create a session by pressing the blue button. The session ID will be displayed on the LCD and an API call will add the session ID to a database and create a queue for the specific session ID. Users will log on to the session through a thin-client application by entering the session ID and begin to poll this queue for questions related to the lecture.

When the speaker presses the green button, the session will start. A recording device attached to a raspberry pi will continuously chunk the lecture into audio files. These files are sent to an AWS S3 where the transcription process is initiated by AWS Transcribe. The trasnscriptions are processed by NLP algorithms running in AWS ECS. The ECS containers will output questions to the session queue so that the users will recieve questions periodically throughout the lecture.

Team