Realtime Survey Audio Recording Processing

SM727 Final Project

Jay Kim

Github Link

Introduction

Web surveys are an incredibly convenient means of collecting data for the person administrating the survey from a logistical point of view. From the respondent's point of view, however, there can be multiple hurdles. For instance, they are less familiar with the user interface of the web survey, this might introduce measurement errors and even cause item non-response from the frustration of the interaction. On the other hand, if their primary means of communicating with other people over distances is over the phone, which is primarily covered by Computer Assisted Telephone Interviews (CATI), the respondent might avoid logging into web surveys through the distributed URL entirely and introduce non-response bias.

Assuming we have respondents who have made it onto the web survey, this project aims to implement and see if an alternative means of recording responses besides typing words into a text box is a feasible way of administrating surveys. Simply recording the responses and storing them would be a trivial problem. The actual challenge that comes with this would be that respondents would usually have a limited means of figuring out what actually got recorded should they wish to correct their answer. We would have to help them avoid the additional hassle of having to listen to what they said from start to finish and re-record their responses. On the other hand the survey administrator also has to listen to what was said for data processing after the survey data has been collected and this is also a laborious process. To help the respondent double check what they recorded in real time and the administrator streamline the transcribing process, we will utilize an API call to an external speech-to-text service to transcribe what was said and display the response in real time.

We believe this type of survey setup would be particularly advantageous for older respondents responding through their smartphones. They would not be as dexterous and their poorer eye sight would limit the space they could use to type their responses. If a on screen keyboard is being, further limiting the display space for the text box, the problem would be exacerbated. Instead with an audio recording setup, however, all they would need to do is press a single UI button to start recording and end recording.

Applications and Pipeline Setup

We will use Qualtrics XM for the web survey, the front end of the whole operation. Qualtrics is a commonly used application for setting up web surveys for people who do not have the time and resources program every single front end web UI element using JavaScript and the backend operations including but not limited to real time survey data loading, processing and storage, and export file generation.

We will use a cloud server hosted on Google Cloud Provider (GCP) to handle several real time operations starting from the API call from the Qualtrics application:

1. Storing the copy of audio recording to a cloud bucket. Qualtrics is able to handle audio recording at the front end, but it does not support storing audio files on its servers unless an additional costly subscription fee has been paid. Qualtrics will receive a URL that leads to the audio recording should the administrator wish to review the recording at a later date.
2. Making an API call and sending the audio file to GCP's speech-to-text application. This process can be replaced with other third party applications should the user feel other models might perform better. We will be keeping to native GCP applications to keep the service fee billing on one platform. The application will return the transcribed text and its confidence score for the transcription. This confidence score could be later used to isolate a few cases for quality control by the administrator.
3. Computation of a distance score converted to percentages (proximity score). Currently we are trying to see if the recorded response is close to a specific target word the respondent is trying to say. We are using a simple Levenshtein distance between the target word and the transcribed word divided by the maxLen = max(target word length, transcribed word length) to range bound it to 0 and 1, which is converted to a percentage. The equation is such: (1 - levDistance / maxLen) * 100. This step is where one might be able to make an API call to LLM model to instead summarize the transcribed response or feed a prompt to figure out the proximity to the target word instead of a hard coded rule based score. For instance, while the question we use asks for the name of the character being displayed, the respondent might say "I'm not sure, but I think the answer is X" instead of just saying "X", which we would sometime like to count as the correct answer if "X" was close to the target word. The problem would be that this could potentially add more seconds of latency as the LLM processes the prompt, and leave the user wondering whether they might have broken Qualtrics.

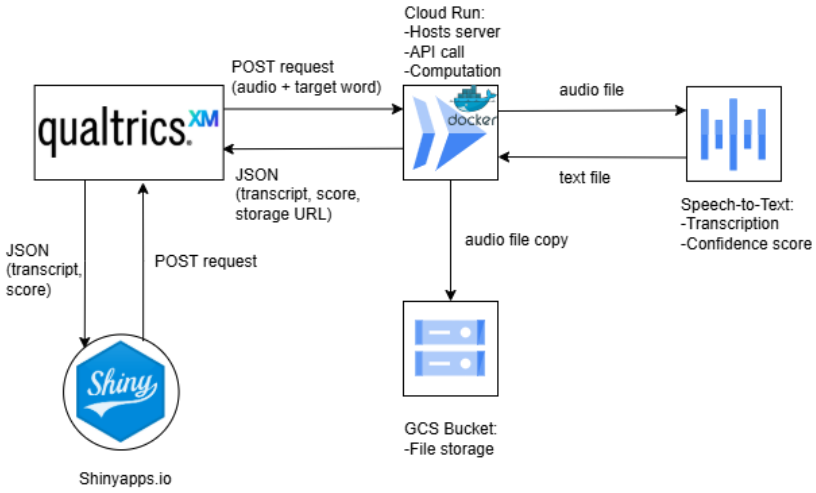

Figure 1: Data Pipeline Flowchart

Qualtrics Link

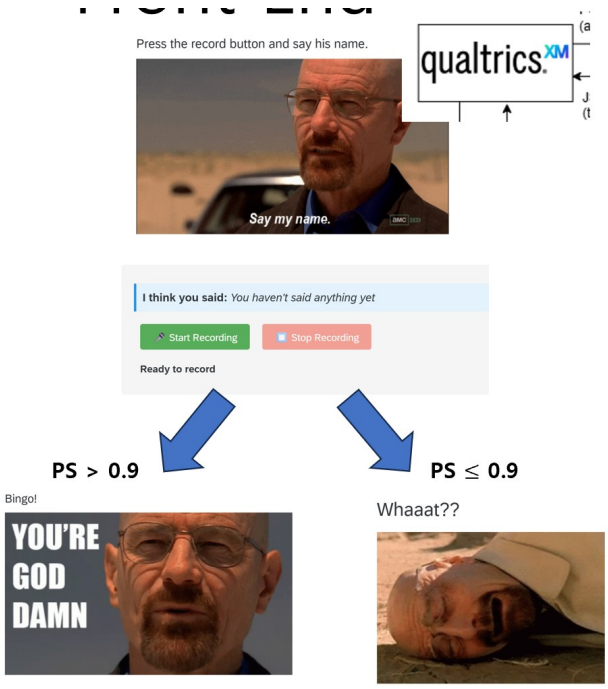

Figure 2: Qualtrics question flow diagram

We also add one more component to the data pipeline that leads to the Shiny dashboard. This

part simulating the part where the survey administrator needs to monitor the responses at a quick glance. The Shiny application will make a API call for the recorded data and Qualtrics API will export a JSON object containing the responses.

While Qualtrics does provide some visualization tools for the collected data, the application is rather clunky to interface with and the options are limited. The Shiny application is hosted for free through Shinyapps.io. We made this choice because it is quite difficult to implement a Shiny server by oneself and for our purposes this simple setup will do. The inherent limitation of a Shiny application is that it will not be running unless someone visits the URL, so there will be some delay expected for the application to fetch the data and process it.

The application displays a word cloud and the distribution of proximity scores which are the most relevant details for our current Qualtrics survey.

Shiny Application Link

Figure 3: Shiny Application

Conclusion

Recording audio recordings and transcribing them for the respondent and survey administrator was a feasible solution. With the current tools available, however, there are a few seconds of latency expected even with simple one word answers and we expect further complexities, including but not limited to transcription latency and accuracy, to arise once the answers go beyond 10 seconds.

Therefore, while answers that fall between one or two sentences might be fine for real time transcription, lest we risk enormous latency, we would not recommend involving transcription for more complex answers into the full survey experience of the respondent nor would we have it take part in influencing any part of the survey flow. Instead we would limit it to help streamline processing the audio files for analysis for the survey administrator after all the data has been collected to the server.