Oheneba Kwaku Addo Dade

Wepea Adamwaba Buntugu

Cohort B

Group 2

CS254 Introduction to Artificial Intelligence

Project Report

Monday 28th November 2022

Page Break

INTRODUCTION

This project aims to develop a program that applies Artificial Intelligence methods and concepts taught in class to create social connections between people tailored explicitly to the Ashesi community. We explored a possible use case in the Buddy Up program, where first-year students are paired with continuing students. With the aid of this program, we believe that pairings would be given a sense of meaning and reasoning and would not just be randomized.

APPROACH

- **THE K-NEAREST NEIGHBOR (KNN) ALGORITHM**

  Serving as the core component of the program, this algorithm is what we opted to use to make the pairings. The *scikit-learn* machine learning library contained the different functions that enabled us to implement our program. Just like any other machine learning process, however, we needed to obtain some data on which our functions would be called. Because the KNN algorithm is a lazy learning algorithm, we did not have to obtain any prior data for testing or training, as this algorithm postpones work until receiving records and only performs the work necessary to produce the intended output. As a result, we could adequately curate a well-thought-out strategy for collecting data, which we believed would be relevant to our project. As mentioned already, this project was aimed at the Ashesi community, so this served as the source

for our data. We selected five questions that would be used to obtain the levels of five different interests among various students. After obtaining the data, which we stored in a CSV file, the next step was to feed it into the library, performing the necessary tweaks and modifications to obtain the desired output.

- **THE NLP COMPONENT**

The program incorporated NLP to improve the user experience of our program (and also to give it some extra razzle-dazzle). The NLP task that was most prominent was speech recognition and this was used to collect user input at different points in the program. A text-to-speech library was also used to convert information that would have been printed to the console into audio data that the user can hear. Text-to-speech systems are also known as engines and these engines usually have two parts: the front-end and the back-end. The front-end is in charge of processing text data that it receives. This is done by converting text that may contain abbreviations, figures and other non-word data into words. These words are then matched to their respective phonetic symbols. These symbols are the representation of how the word is meant to be pronounced according to the rules of the language in consideration. The word "that" for example, has a phonetic transcription of ðæt. This phonetic data is combined with prosodic data, now prosodic data refers to how words, phrases and sentences are combined together, concepts like stressing, inflection and others come into play here. Now, prosodic data is important to help generated speech sound more human. The back-end of the text-to-speech engine synthesizes speech using the phonetic and prosodic data that has been provided by the front-end. The text-to-speech library we used for the project was pyttxs3. Within the pyttxs3 library, we used the windows speech API which

is used for speech synthesis in windows applications and powered the backend of the text-to-speech system.

For the speech-to-text aspect of the program, we used Google's speech to text API which makes calls to the Google trained model whenever the speech recognition function is called. The Google Speech to Text model can be broken down into three main parts. The acoustic model takes audio data and chops it up into small chunks for analysis. It then outputs a list of probabilities for all the different chunks that have been created which show how likely a definite number of letters is to be in that chunk. The pronunciation model then restricts the possible letter combinations based on how likely they are to occur in a given language. Finally, the language model selects the most suitable sequence of words from the information provided by the pronunciation model. For our project, we used speech to text to take user input which was mainly numerical answers to a set of questions – users were required to say their name as well. The speech to text model function quite well with recognising letters in spite of the variety of accents that were used, even though it did struggle with the number 8 quite a bit and for this reason we had to implement a safeguard around instances were users said the number 8. Apart from that, the NLP aspect of the project helped to create a more usable program and allowed us to explore text-to-speech(TTS) and speech-to-text(STT) concepts along the way.