Ollscoil Chathair Bhaile Átha Cliath
Dublin City University

FUNCTIONAL SPECIFICATION

Loqui: A Voice Powered Python Assistant

CA400

NOV 2022

Student Information

Name	Katie Clancy	Niall Egan
Student No.	19452724	19378906

Table of Contents:

1. Introduction

1.1 Overview

Loqui will be a voice powered programming assistant for use with Python. It will be developed as a desktop application, with built in speech recognition functions so the user can program by speaking into a speech input device (such as a headset). The speech-to-text element will be built using Rasa ("*a framework for developing AI powered chatbots*")[1] which will aid the integration of speech recognition stages; Automated Speech Recognition (ASR)[a], Natural Language Processing (NLP)[b] and Text-To-Speech (TTS)[c], while the desktop app will use the Electron framework (an opensource framework used to create desktop apps) in it's creation while also incorporating API elements.

Our target users are beginner programmers, however this tool could be useful to many user groups including lecturers for teaching programming, and with further development could be useful in demonstrating the basic concepts of other languages.

2. General Description

2.1 Product / System Functions

Our app will allow users to dictate Python Code to create functioning Python programs. The Application will display the code created from the User's voice commands in an inbuilt editor. It will also provide assistance with syntax assistance. Users will not have to dictate every character. They can provide higher level commands and the application will take care of brackets, indenting, colons, etc.

2.2 User Characteristics and Objectives

2.2.1 User Characteristics

The expected features for are user community are:

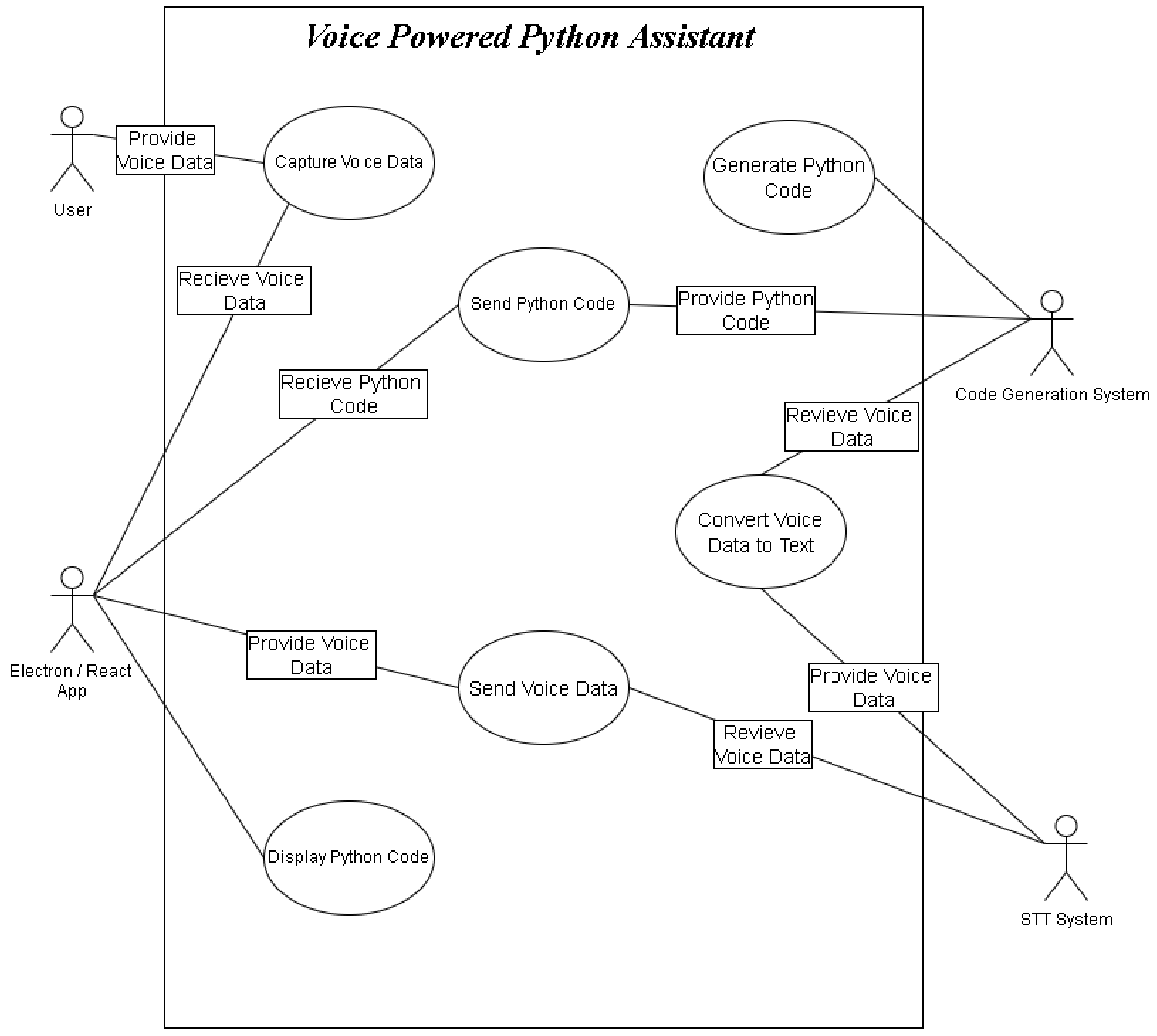
- A sufficient enough understanding of the Python Programming Language to produce and recognise valid code.
- An understanding of installing and launching desktop applications.

2.2.2 User Objectives

From the Users perspective, our application should:

- Allow them to produce Python code of an acceptable standard that can be successfully run.
- Allow them to continue to create Python code while turning their attention away from the screen or multitasking.
- Provide an accessible environment to create Python code for Users with mobility issues.

2.3 Operational Scenarios



2.4 Constraints

- We are limiting the scope of the Application to only be compatible with the Python Programming Language. Trying to make the App compatible with multiple languages with different designs, e.g.: statically typed vs. dynamically typed, would not be practical in the development time. Python has a simple structure with very little boilerplate code. Focusing on this more straightforward language allows us to ensure that the language covers all its aspects.

- The Application will only be compatible with machines that can capture audio. Without any source of audio input, the application will not be able to function.

- The Application will only be able to receive and process voice commands in the English language. Setting up a Speech-to-Text system that can correctly handle multiple languages would not be practical in the development time.

-

3. Functional Requirements

(a)

Description – The Application must be able to receive voice commands from the User.

Criticality – This requirement is critical to the function of the Application. If this requirement is not met then the app is non-functional.

Technical issues – There may be an implementation issue when requesting access to a user's microphone to receive the voice commands. Sending the permission alerts from the React App to the Electron App.

Dependencies with other requirements – This requirement has no dependencies with other requirements.

(b)

Description – The Application must be able to send voice data to the Speech-To-Text System through the API.

Criticality – This requirement is critical to the function of the Application. If this requirement is not met then the app is non-functional.

Technical issues – Potential Issues with sending voice data through an API endpoint.

Dependencies with other requirements – This requirement is dependent on requirement (a). If the Application cannot record voice data then it can't be sent.

(c)

Description – The Application must be able to process the voice data into valid text.

Criticality – This requirement is critical to the function of the Application. If this requirement is not met then the app is non-functional.

Technical issues – No potential technical issues expected with this requirement.

Dependencies with other requirements – This requirement is dependent on requirement (b). The text can't be generated without the voice data from the user.

(d)

Description – The Application must be able to convert the text commands into valid Python code.

Criticality – This requirement is critical to the function of the Application. If this requirement is not met then the app is non-functional.

Technical issues – No potential technical issues expected with this requirement.

Dependencies with other requirements – This requirement is dependent on requirement (c). The text output from the Speech-to-Text System is needed for this requirement.

(e)

Description – The Application must be able to display the generated Python code in the Electron / React App.

Criticality – This requirement is a high priority for the Application but not critical. It can still be able to function without this feature, but it will lower the quality of the user experience significantly.

Technical issues – Potential issues with display animations for the text.

Dependencies with other requirements – This requirement is on all previous requirements. The whole system must be functional for this requirement to be met.

4. System Architecture

A system's interpretation of speech takes place in several stages.

The voice recognition system captures speech via a device (i.e. headset, microphone) and is filtered to remove unwanted noise and extract relevant & required information. During this stage of processing, the speech is normalized as not all users will speak at the same volume and pace.

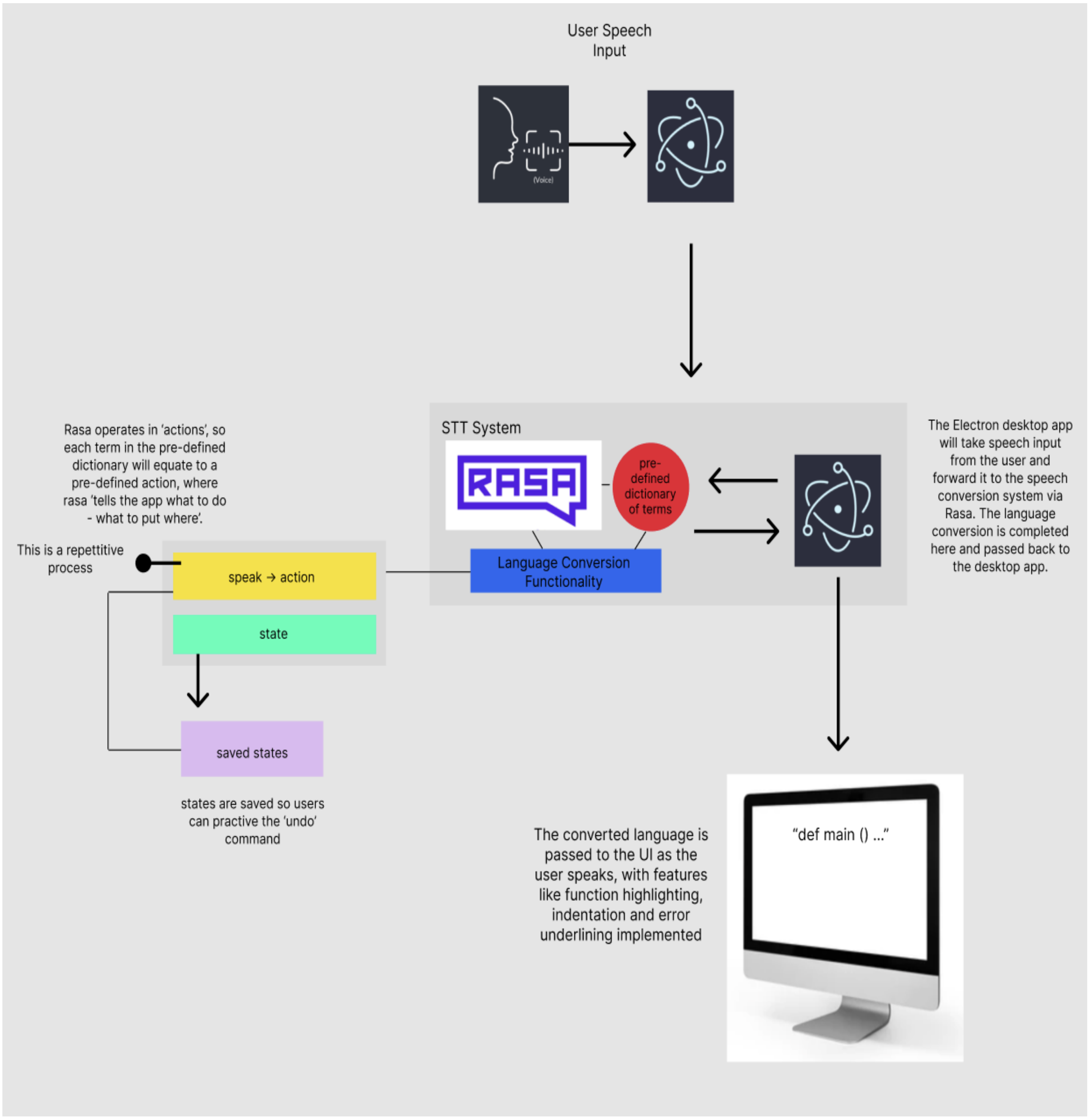
Segmentation[j] is performed for sentences that are longer to break them up for easier processing.

There is an element called the Acoustic Model[k] that is fundamental to a speech recognition system. This is the part that defines the relationship between the captured speech and the pre-defined vocabulary of terms. When the speech has been processed, it is stored.

The assistant model to the system is the linguistic model[l], whose main function is to rectify the inaccuracies generated in the Acoustic Model's processing. Probability distributions are used in this stage of STT to compute the likelihood of the next wordbased on the pre-defined set of vocabulary.

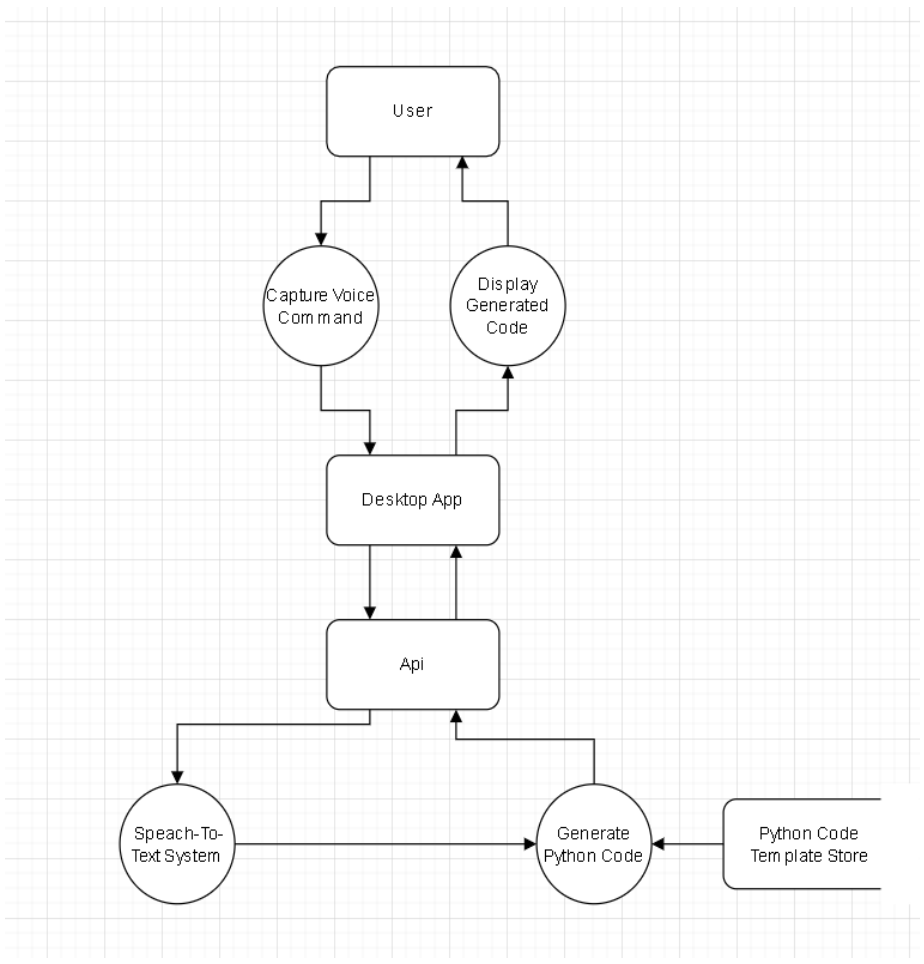
Once all processing of the language is complete, the data is passed to text display.

To relate this process to our system and the tools we will use, when the speech is taken in by Electron and passed to Rasa, term recognition for the Python programming language takes place. This will be made possible by the construction of a dictionary containing all operable commands. Terms are matched to actions (e.g. def main () -> highlighted in the relevant colors via the color code). 'States' - determined by a pre-defined pause period in the user's speech - are saved when the pause threshold is speech, and temporarily saved. This will enable the user to make use of the *"undo"* command for mistakes; the previous state will be restored. All elements of speech-to-text conversion will take place in and around the Rasa-built framework (such as parsing, NLP (Natural Language Processing) etc). This data will be passed between Rasa and the API & Electron system to ultimately display the contents of the conversion (including built-in system corrections of incorrectly spoken words) on the desktop app interface. Users will also be able to nmake use of commands like *"run"*, which will output the results of their program.


User Speech Input
(Voice)
STT System
RASA
pre-defined dictionary of terms
Language Conversion Functionality
The Electron desktop app will take speech input from the user and forward it to the speech conversion system via Rasa. The language conversion is completed here and passed back to the desktop app.
Rasa operates in 'actions', so each term in the pre-defined dictionary will equate to a pre-defined action, where rasa 'tells the app what to do - what to put where'.
This is a repettitive process
speak → action
state
saved states
states are saved so users can practive the 'undo' command
The converted language is passed to the UI as the user speaks, with features like function highlighting, indentation and error underlining implemented
"def main () ..."

5. High-Level Design

This is the Data-Flow Diagram for our Application:



The steps in our data flow are:

1. The voice command provided by the User is captured by the Desktop App.

2. The Desktop App passes the voice data to our Api.

3. The voice data is processed by the Speech-To-Text system.

4. The outputted text is passed to the process to generate Python code along with our generalised Python templates.

5. The generated text is passed back to the Desktop App through the Api.

6. The generated code is displayed to the user in the Desktop App.

6. Preliminary Schedule

We have split up our project into four major parts; *Idea Formation & approval*[f] (which has already been completed), *Deliverables*[g], *App & API*[h] and *TTS & Code Assistance System*[i]. This way we could divide the work into parts to be assigned to each of us individually and parts we can both complete together against a timeline.

The following GANTT Chart is an accurate representation of the dividing of our work displayed over a timeline to keep us on track for full delivery:

CA400 Schedule 2022/2023

Color Code: Niall Katie Both Members

PROJECT TITLE	*Loqui*: Voice Powered Python Assistant	**PROJECT SUPERVISOR**	Michael Scriney
PROJECT MEMBERS	Niall Egan & Katie Clancy	**DEADLINE**	28/04/2023 @ 17:00

					OCTOBER				NOVEMBER				DECEMBER				JANUARY				FEBRUARY				MARCH				APRIL			
					1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
IDEA FORMATION & APPROVAL	**Start Date**	**End/Due Date**	**Member Assigned**																													
Idea Formation	05/10/22	26/10/22	🟥	100%	■	■	■																									
Supervisor Scouting	05/10/22	19/10/22	🟥	100%			■	■																								
Research	05/10/22	26/10/22	🟥	100%	■	■	■	■	■																							
Project proposal	22/10/22	26/10/22	🟥	100%			■	■																								
Proposal Approval	26/10/22	26/10/22	🟥	100%				■																								
Idea Panel Pitch	20/10/22	20/10/22	🟥	100%			■																									
DELIVERABLES																																
Functional Spec	04/11/22	13/11/22	🟥	52%					■	■																						
Ethical Approval Form Completion	Dec Week 1	Dec Week 1	🟥	0%									■																			
Ethical Approval Form Approval	Dec Week 1	Dec Week 12	🟥	0%									■	■																		
User Testing Group Compilation & Study Completion	Jan Week 3	March Week 2	🟥	52%															■	■	■	■	■	■	■	■						
APP & API																																
Electron | React App & FastApi Mockup	28/10/22	Nov Week 4	🟩	70%				■	■	■	■	■																				
Electron | React App Backend	Nov Week 4	Jan Week 2	🟩	0%								■	■	■	■	■	■	■														
Electron | React App User Interface	Jan Week 1	Feb Week 1	🟩	0%													■	■	■	■	■											
FastApi Setup & Endpoints	Jan Week 3	Feb Week 3	🟩	0%															■	■	■	■	■									
Unit Testing	Feb Week 4	March Week 3	🟩	0%																				■	■	■	■					
App-To-Api Testing	Feb Week 4	March Week 1	🟩	0%																				■	■							
TTS & CODE ASSISTANCE SYSTEM																																
Rasa Env Setup	Oct Week 4	Nov Week 1	🟦	90%				■	■																							
Speech-To-Text Conversion Mockup	Nov Week 1	Nov Week 3	🟦	25%					■	■	■																					
STT Testing	Nov Week 3	Dec Week 2	🟦	10%							■	■	■	■																		
STT Further Development	Dec Week 1	Jan Week 2	🟦	0%									■	■	■	■	■	■														
Recognizeable Vocab Library Assembly	Jan Week 3	Feb Week 3	🟦	0%															■	■	■	■	■									
System Testing	Feb Week 3	March Week 2	🟦	0%																			■	■	■	■						
User Testing	March Week 4	April Week 3	🟥	0%																								■	■	■	■	
Front-To-Back End Integration	March week 3	April Week 3	🟥	0%																							■	■	■	■	■	

7. Glossary

[a] *Automated Speech Recognition (ASR)*; the first step in speech recognition which transcribes the audio; in simpler terms it transforms the spoken language into text.

[b] *Natural Language Processing (NLP)*; this is the (second) stage that helps the computer understand spoken language in a human-like way. It's the stage that makes sense of the speech.

[c] *Text-To-Speech (TTS)*; the final stage in speech recognition which implements Speech Recognition[d], which is where the sounds taken in by the recogntiton system are used to build words.

[d] *Speech Synthesis*; this is where text is converted to speech (in broader terms), which is not an element we will incorporate in our speech recognition system, but we will require part of the speech synthesis process in order to have our system recognize words spoken by the user. This process follows a generalized version of lexical and syntax analysis[e] to do this.

[e] *Lexical & Syntax Analysis*; these are stages of compiler actions that break down characters into recognizeable tokens to then place them back together into a string to be analyzed.

[f] *Idea Formation & approval*; this is the stage of project formation where we were paird as a team and had to come up with our idea, research & develop the concept, scout for project supervisors, write a proposal & present it to a panel for approval.

[g] *Deliverables*; this is the section of our project that includes documents that need to we written up and submitted to the relevant personnel for approval also (such as ethical approval & functional specification).

[h] *App & API*; this is mainly the front-end of the project; the desktop app and API portion of development.

[i] *TTS & Code Assistance System*; this is mainly the back end of the application, its functionality in terms of the code demonstrating and animation and speech recognition aspects.

[j] *Segmentation;* in the speech recognition process, longer sentences are divided up into parts that are more manageable for processing.

[k] *Acoustic Model;* this model is the most important part of STT system, that inhabits a sequence-to-sequence like structure, where sound input is processed and matched with suitable terms in the vocabulary library.

[l] *Linguistic Model;* this is the part that converts speech to text. It acts as an assistant model to the system.

8. References

[1] *Rasa description:* "*a framework for developing AI powered chatbots*"; https://www.coursera.org/projects/chatbot-rasa-python