# 

# 

# 

UNIVERSITY OF TECHNOLOGY, JAMAICA

Analysis of Programming Languages (CMP4011)

Jadan Jones

1701304

# 

# Project Outline

The aim of this assignment is to build an application program interface for speech synthesis for the Liz CRUZR robot using Natural Language Processing parser techniques. I created the API using Python using the following libraries/tools:

* Flask
* Imagemagick
* Pyttsx3
* Nltk
* PyPdf2
* Docx2txt
* Python-pptx

Flask was used to build out the various endpoints for the API and was also used to serve the HTML/CSS frontend. The core functionalities of the speech synthesiser was built with the nltk library. The nltk library already comes fitted with a wide range of methods which allows to perform all the basic Natural Language Processing steps all within one library. The nltk library contains all the dictionaries I need including several parts of speech libraries. To perform the actual speech generation, pyttsx was used.

# The User Interface

Graphical user interface, application

Description automatically generated

The user is greeted with 2 options. They can enter plain text into the text box, or they can upload a file to have the text extracted from it. The files the program accepts are text files, Microsoft Word Documents, and PDFs. The uploaded files are temporarily saved on the server during the synthesis process. The docx2txt, Py2PDF2 and python-pptx library was used to extract text from docx files, pdfs and PowerPoint files respectively. When the text is extracted, it is passed to the language processing engine for lexical analysis. After the analysis is done, the output is passed to the pyttsx3 library which produces an audio output. A PDF of the parse tree generated from the language processing engine is also sent to the webpage.

**Project Structure**

Graphical user interface, application

Description automatically generated

The project has a simple layout. The application is served from **app.py** which import several modules.

# File Handling

The file Handler performs 2 functions. The first function is accepting, sanitizing, and checking uploaded files for validity. If a file does not belong to the accepted group, then an error is thrown. It also checks if the uploaded file is empty. The second duty of the file handler is to generate and save the generated parse tree pdf to the server. This pdf is sent to the client each time a synthesis is done.

# Speech Handling

The speech handler contains the text to speech function of the pyttsx3 library. It contains 1 function which accepts text and passes it to the speak function within ppttsx3. The function provides the configuration for the voice used such as volume, gender, and speed.

Text

Description automatically generated

# Lexical Analysis Engine

The Lexical Analysis Engine performs all lexical analysis steps on the string that is passed to it from the user. The steps include tokenization, stemming, lemmatization, parts of speech tagging and normalizing. The first step is to identify all invalid characters such as punctuations and whitespaces and remove them from the token list. After the string is tokenized, we also split the string into sentences using the sentence tokenization method from the nltk library. Another tokenization technique that is performed is tri-gram tokenization which splits the text into three consecutive words at a time. The next thing that is done is the removal of stop words. Stop word are words that add no meaningful value to a passed string. These are words such as ‘a’, ‘of’, ‘I’ etc. The stop words are found using the stop word dictionary already built into nltk. The next step is stemming. Stemming is a process that normalizes the words or tokens into their base root form. Stemming works by cutting of the end or beginning of the tokens while taking into account a list or dictionary of common prefixes and suffixes that can be found in words. The stemmer that is used in this program is the Porter stemmer. However, this technique can be highly inaccurate and can sometimes produce incorrect and truncated word. This is where lemmatization comes in. Lemmatization groups together different inflected forms of a word (called a lemma) and maps several words into one common root. The output of lemmatization will always be a proper word as opposed to stemming. The lemmatization function used the parts of speech tags to accurately lemmatize the tokens. The parts of speech tagger use a parts of speech dictionary to map the tokens to its respective parts of speech.

Text, letter

Description automatically generated

**Sample Result**

Text

Description automatically generated

**Parsing Handler**

## Named Entity Recognition

The process of within the parsing handler which detects named entities such as movies, monetary values, organizations, locations, quantities, and persons is called Named Entity Recognition (NER).

There is the noun phrase identification which aims to extract the noun phrases from the text using the already parts of speech tagged tokens. Secondly, we have the phrase classification in which the nouns are classified into different groups such as location, person, organization etc.

At the end of the lexical analysis, there is parsing. To parse the sentence tokens, a RegexpParser method from the nltk library is used to define a custom regular expression grammar. Using the parsing rules, the tagged sentences can then be used to generate a parse tree. For example, with the string “I am a big fish.”, the parse tree generated is shown below.

**Parse Tree Example**

Chart, radar chart

Description automatically generated