OS Project Report.

Title: Voice Assistant for Linux

Team members:
Hriday Kondru (B20CS021)
Rohit Bhanudas Kote (B20CS056)
Perumalla Aasrish Vinay (B20CS042)
Ribhav Manish (B20CS054)

Required dependencies:

pip install time
pip install beautyfulsoup4
pip install google
pip install speechrecognition
pip install pipwin
pip install pyaudio
Pip install transformers
Pip install tensorflow

Problem Statement:

Today almost every Operating System has some sort of voice assistant. For example, Siri, google(the google assistant), and Bixby. Even the windows system has one called Cortona. But we wanted to build a robust voice assistant for windows and Linux which could perform all the rudimentary tasks in the OS

Methodology:

We first created a voice-to-text algorithm on python using the library Speech_Recognition.
For accessing the microphone we used the module speech_recognition. Microphone and to convert the speech to text, we use google_recognize(), it uses google's free web search API and does not require any API key.

The output of the algorithm is a string of the user's speech.

For text segmentation, we required a model which can produce good results without any prior training. This is because we do not have a dataset for voice commands specifically for Linux Voice assistants.
So the solution to the problem was Zero-Shot Classification(ZSL).
ZSL is a model which can learn from labeled data and predict the classes which were never seen during the training of the model.
This model is most useful in our case as we just need to define certain classes of output and the model will label the input with the given classes.

We used the ZSL model from an online repository- “Hugging face”.
Hugging Face is an online community that provides models based on open-source code technologies.

The model used for this project is:facebook/bart-large-mnli

Lastly we mapped all the labels of ZSL outputs to the Linux functions using python and the OS library of python.
The OS library is used to implement functionalities like showing the contents of the folder and make a new directory.

Outcomes:

The voice assistant can run the following commands:
Show contents of a file
Show today's date
Search for a given keyword on google
Make a new directory
Simple arithmetic equations

Problems faced:

- As we are using a pre-trained Machine learning model for the text segmentation, and we are running it locally on the CPU of our laptops. Thus the process is time taking. The table depicting all values is shown below:

Computing resource	Time taken (for only text segmentation)	Voice recognition software 1	Voice Recognition software 2
Local CPU (without power source)	24.914 seconds	28.7412 seconds	35.6747 seconds
CPU with power source	21.1363 seconds	21.1556 seconds	13.8402 seconds
Tesla K80 GPU (Collab free GPUs)	9.7338 seconds	Did not work	Did not work

- The first voice recognition software relatively slower than the second one, but our experiments suggest that the second one is not reliable in terms of accuracy thus we are using the 1st voice recognition software.
- As the collab does not have microphone access, we were not able to find the time taken by the different speech-to-text recognizers on GPU
- Another issue faced while creating the mapping of ZSL labels to Linux functions was creating the "cd" command. The actual problem with the "cd" command is that when we run a python shell, it does enter the required directory, yet as the shell execution completes it just moves automatically to the home directory. We have verified that the shell actually moved into the directory by running the "show contents" command in the function itself; this printed only the contents of the required directory but not the whole system. Hence it is not maintainable to stay in the given directory while running the python script.