LABORATORY 5: SPEECH RECOGNITION AI

Learning Outcomes

By the end of this laboratory, student should be able to

- Understand how speech recognition works.
- Create speech recognition using python.

Activities

- Setting up a microphone
- Create a python script for speech recognition.
- Control a turtle sim using speech recognition.

Equipment

- Microphone /Phone
- Laptop

Introduction

Speech recognition

Speech recognition is like a smart translator for computers. It helps computers understand what we say by turning spoken words into text. This cool tech allows devices to listen to us,and some can even respond to our words. Behind the scenes, computer programs use microphones to capture what we say, process it, and turn it into something the computer can understand.

Even though it might sound high-tech, we use speech recognition every day. Have you ever called a company and spoken to a computer that helps you? That's one example. And those virtual assistants like Siri or Alexa? They chat with us using speech recognition magic!

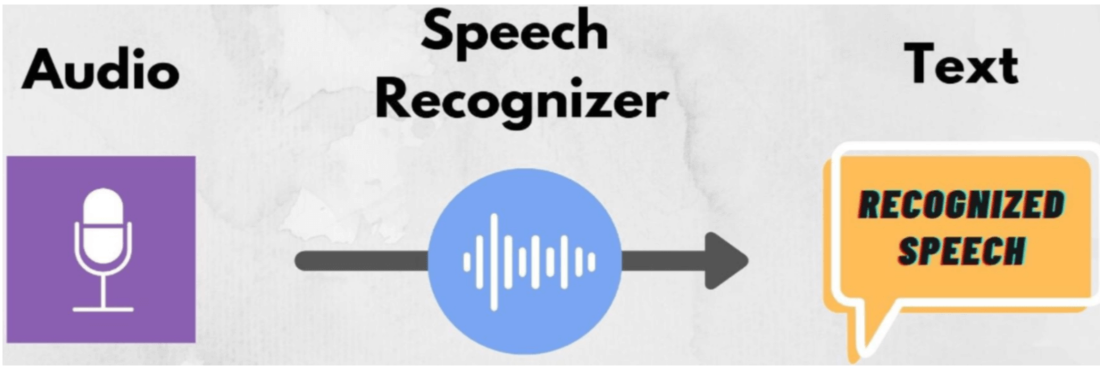
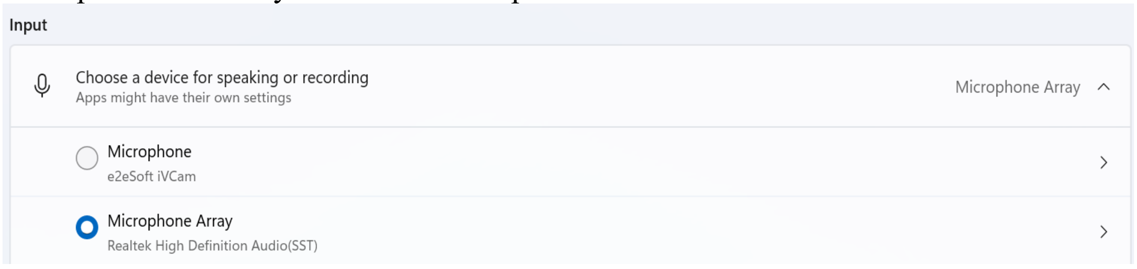


Figure 1: Showing how speech is converted to text

A. Setting up the microphone

If your laptop has a microphone, go to settings and search for sound settings. Select Input microphone deice as your default microphone.


Input

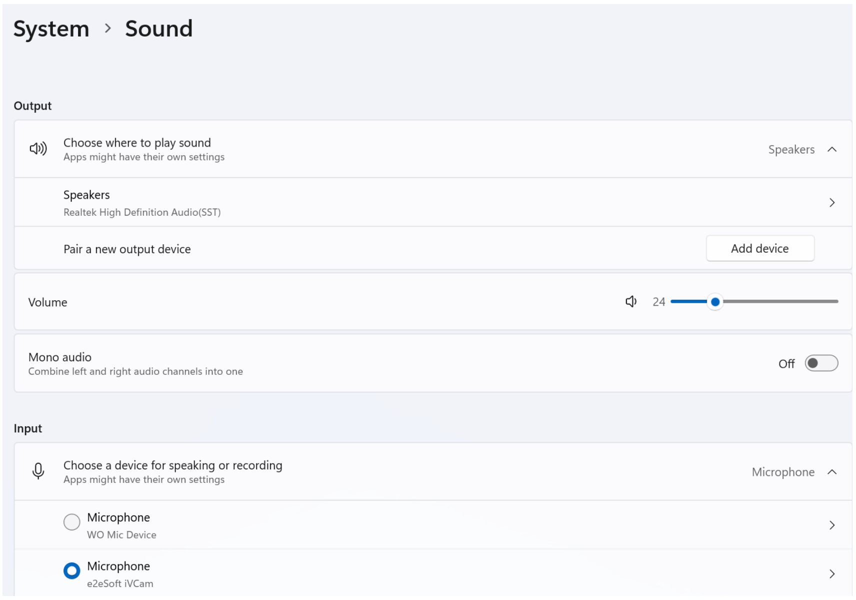
Choose a device for speaking or recording
Apps might have their own settings — Microphone Array

Microphone
e2eSoft iVCam

[✓ Microphone Array]
Realtek High Definition Audio(SST)

If you do not have a microphone on your laptop, there are many ways to set up your microphone. We will be converting our phone into a microphone.

1. First, download and install e2esoft iVCam on laptop from this link: https://ivcam.en.softonic.com/
2. Download iVCam app on your phone in Google Play store or App store.
3. On laptop, go to sound settings and select input device for microphone as e2esoft iVCam.


System › Sound

Output

Choose where to play sound
Apps might have their own settings — Speakers

Speakers
Realtek High Definition Audio(SST)

Pair a new output device — Add device

Volume — 24

Mono audio
Combine left and right audio channels into one — Off

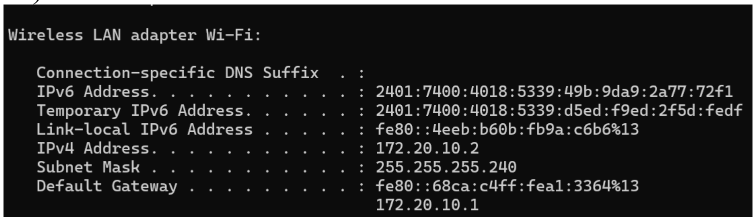
Input

Choose a device for speaking or recording
Apps might have their own settings — Microphone

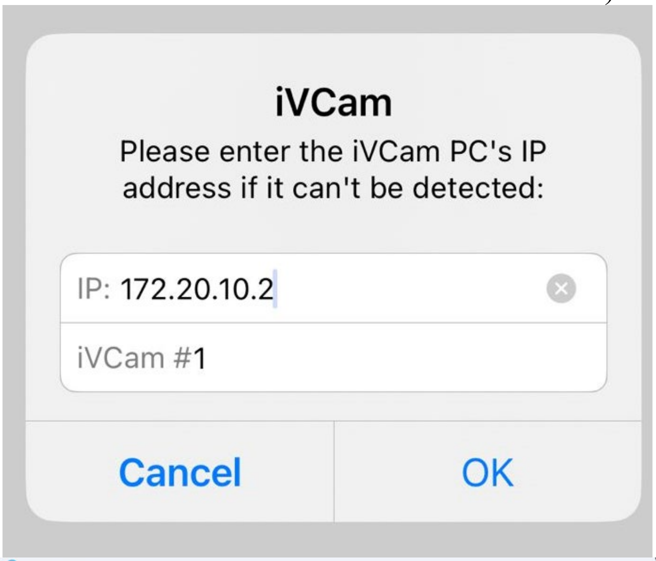
Microphone
WO Mic Device

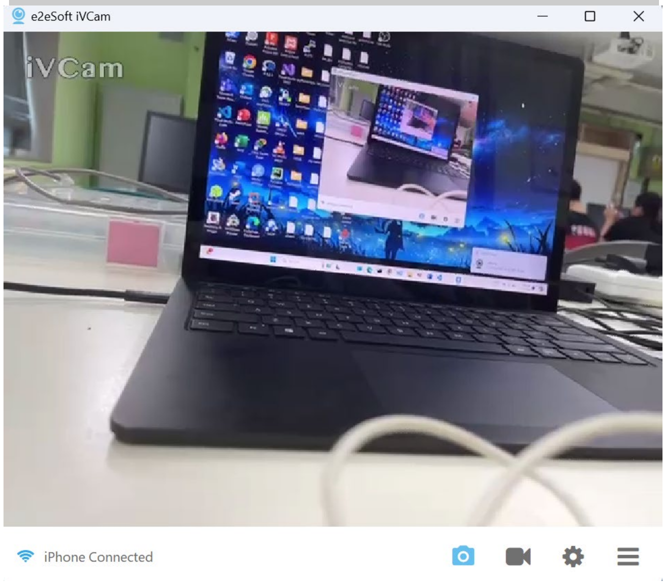
[✓ Microphone]
e2eSoft iVCam

4. Next, turn on your mobile data and enable hotspot. Connect your laptop's Wifi to your phone's hotspot. Then, connect your phone to your laptop physically using a USB cables (use phone's charging cable)
5. Open command prompt and type in ipconfig. Look for the laptop Wifi's IP address (e.g.172.20.10.2).


```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2401:7400:4018:5339:49b:9da9:2a77:72f1
   Temporary IPv6 Address. . . . . . : 2401:7400:4018:5339:d5ed:f9ed:2f5d:fedf
   Link-local IPv6 Address . . . . . : fe80::4eeb:b60b:fb9a:c6b6%13
   IPv4 Address. . . . . . . . . . . : 172.20.10.2
   Subnet Mask . . . . . . . . . . . : 255.255.255.240
   Default Gateway . . . . . . . . . : fe80::68ca:c4ff:fea1:3364%13
                                       172.20.10.1
```

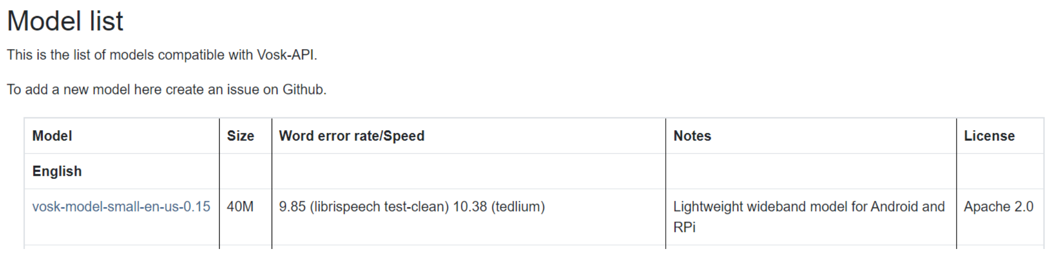
6. On your phone, open iVCam and click the add “+” button, input the laptop Wifi’s IP address. (Make sure that both are connected to the same Wifi network (SPStudent won’t work) and connected via a USB cable at the same time)


iVCam
Please enter the iVCam PC's IP address if it can't be detected:
IP: 172.20.10.2
iVCam #1



B. Downloading libraries for python

1. Download lab 5 folder and open the **Lab5_Code** folder inside VScode.
2. Use the existing environment created in Lab 4 or set up a new **virtual environment** with python interpreter 3.9/3.10/3.11 and launch it. (Refer to Lab 4 if you have forgotten)
3. Pip installs vosk and pyaudio python libraries:
   - First update your pip using
     python -m pip install --upgrade pip
   - Pip install vosk
     pip install vosk
   - Pip install Pyaudio.
     pip install pyaudio
4. So, once you have done with downloading these libraries you need to download a speech recognition model for using vosk. Here is the link: https://alphacephei.com/vosk/models



Model list

This is the list of models compatible with Vosk-API.

To add a new model here create an issue on Github.

Model	Size	Word error rate/Speed	Notes	License
English				
vosk-model-small-en-us-0.15	40M	9.85 (librispeech test-clean) 10.38 (tedlium)	Lightweight wideband model for Android and RPi	Apache 2.0

Unzip the file into your local disk or somewhere easy to access. (Make sure that the directory the file is in does not have any spaces)

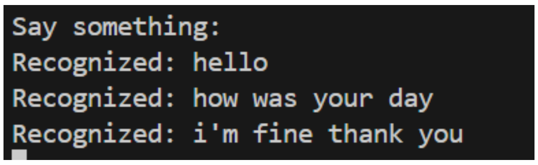
C. Code for speech recognition model

Open **main.py** and replace the highlighted part with the directory where your download speech recognition model is located. E.g. “C:\vosk-model-small-en-us-0.15\vosk-model-small-en-us-0.15”

```
from vosk import Model, KaldiRecognizer
import pyaudio

# Load the Vosk model and recognizer
#Change your directory to where the downloaded speech recognition model zip
file is located (Directory must not have spaces!!!)
model = Model(r"C:\vosk-model-small-en-us-0.15\vosk-model-small-en-us-0.15")
recognizer = KaldiRecognizer(model, 16000)

# Initialize PyAudio and the microphone stream
mic = pyaudio.PyAudio()
stream = mic.open(format=pyaudio.paInt16, channels=1, rate=16000, input=True,
frames_per_buffer=8192)
stream.start_stream()

try:
    print("Say something:")
    while True:
        data = stream.read(4096)
        if recognizer.AcceptWaveform(data):
            result = recognizer.Result()
            recognized_text = result[14:-3].strip().lower()
            if recognized_text:
                print("Recognized:", recognized_text)


except KeyboardInterrupt:
    pass

# Close the microphone stream and exit
stream.stop_stream()
stream.close()
mic.terminate()
```

When you are done, execute this command in a terminal “**.\.venv\Scripts\python main.py**”. Then, turn on the mic on your laptop or phone and say something.

Output:



```
Say something:
Recognized: hello
Recognized: how was your day
Recognized: i'm fine thank you
```

D. Code explanation

```
from vosk import Model, KaldiRecognizer
import pyaudio
```

This line of code we import the library vosk for speech recognition and pyaudio to use the input device.

```
model = Model(r"C:\vosk-model-small-en-us-0.15\vosk-model-small-en-us-0.15")
recognizer = KaldiRecognizer(model, 16000)
```

The first line of code is to load the pretrained model. Then the second the second line of code is to load the recognizer. Basically, using a Kaldi speech recognition engine with a specified pre-trained model and a sample rate of 16,000 Hz.

```
# Initialize PyAudio and the microphone stream
mic = pyaudio.PyAudio()
stream = mic.open(format=pyaudio.paInt16, channels=1, rate=16000, input=True,
frames_per_buffer=8192)
stream.start_stream()
```

The first line The mic variable becomes an object that allows you to interact with audio-related functionalities.
Then the next line is important as this are your stream parameters. The parameters are as follows:

- Format =pyaudio.paInt16 . Each sample is a 16-bit signed integer.There are other formats such as paint24, paint32 , paFloat32, paFloat64.These formats provide higher precision but have taken higher space than paInt16. So paInt16 is a good balance.
- Channels can basically be 1 which is mono and 2 is stereo.
- Rate is basically the sample rate of the audio stream. (Make this same as model sample rate)
- Input =True basically make the mic an input device, change to false and will make it an output device
- Frame_per_buffer: Is the chuck of data that is being processed per frame by pyaudio(usually twice the **stream.read()** size)

```
try:
    print("Say something:")
    while True:
        data = stream.read(4096)
        if recognizer.AcceptWaveform(data):
            result = recognizer.Result()
            recognized_text = result[14:-3].strip().lower()
            if recognized_text:
                print("Recognized:", recognized_text)
```

This code above is for creating an output.

- data = stream.read(4096): This line reads 4096 frames of audio data from the PyAudio stream (stream). The size of the chunk (4096 frames) is determined by the frames_per_buffer parameter specified when opening the stream.

- if recognizer.AcceptWaveform(data):: This condition checks whether the speech recognizer (presumably from the Vosk library) accepts the waveform data. The AcceptWaveform method is used to pass the audio data to the recognizer for processing. If the recognizer successfully processes the waveform, the condition is true.

- result = recognizer.Result(): This line retrieves the result from the recognizer. It assumes that the recognizer has successfully accepted the waveform.

- recognized_text = result[14:-3].strip().lower(): This line extracts the recognized text from the result. It removes some prefix and suffix (possibly related to the specific format of the result) and then converts the recognized text to lowercase for uniformity.

E. Controlling a turtlesim using speech recognition

Now that you know how the speech recognition works you can try to use to do task such as turning on led lights or other outputs that can be triggered by speech. So, for the lab we will do a simple turtle sim controlled by speech recognition.

Open main2.py and replace the highlighted part again with the directory where your download speech recognition model is located. Then, run the code with "**.\.venv\Scripts\python main2.py**"

```
from vosk import Model, KaldiRecognizer
import pyaudio
import turtle

# Load the Vosk model and recognizer
model = Model(r"C:\vosk-model-small-en-us-0.15\vosk-model-small-en-us-0.15")
recognizer = KaldiRecognizer(model, 16000)

# Initialize PyAudio and the microphone stream
mic = pyaudio.PyAudio()
stream = mic.open(format=pyaudio.paInt16, channels=1, rate=16000, input=True,
frames_per_buffer=1024)
stream.start_stream()

# Initialize the TurtleSim window
window = turtle.Screen()
window.title("Voice Control TurtleSim")
t = turtle.Turtle()

# Define function to move the turtle
def move_turtle(direction):
    if direction == "left":
        t.left(90)
    elif direction == "right":
        t.right(90)
    elif direction == "up":
        t.forward(10)
    elif direction == "down":
        t.backward(10)

try:
    while True:
        data = stream.read(16000)

        if recognizer.AcceptWaveform(data):
            result = recognizer.Result()
            recognized_text = result[14:-3].strip().lower()

            print("Recognized:", recognized_text)

            if recognized_text == "stop":
                break
            else:
                move_turtle(recognized_text)

except KeyboardInterrupt:
    pass

# Close the microphone stream and exit
stream.stop_stream()
stream.close()
mic.terminate()
turtle.bye()
```

With this code above, you should be able to control the turtlesim with voice commands.

Main3.py only display recognized words such as “left”, “right”, “up”, “down” on the turtlesim as the turtle moves. Open main3.py, replace the highlighted part and run the script “**.\.venv\Scripts\python main3.py**”.

```
from vosk import Model, KaldiRecognizer
import pyaudio
import turtle

# Load the Vosk model and recognizer
model = Model(r"C:\vosk-model-small-en-us-0.15\vosk-model-small-en-us-0.15")
recognizer = KaldiRecognizer(model, 16000)

# Initialize PyAudio and the microphone stream
mic = pyaudio.PyAudio()
stream = mic.open(format=pyaudio.paInt16, channels=1, rate=16000, input=True,
frames_per_buffer=1024)
```

```
stream.start_stream()

# Initialize the TurtleSim window
window = turtle.Screen()
window.title("Voice Control TurtleSim")
t = turtle.Turtle()

# Define function to move the turtle
def move_turtle(direction):
    if direction == "left":
        t.undo()
        t.left(90)
        t.write("left")
    elif direction == "right":
        t.undo()
        t.right(90)
        t.write("right")
    elif direction == "up":
        t.undo()
        t.forward(10)
        t.write("up")
    elif direction == "down":
        t.undo()
        t.backward(10)
        t.write("down")

try:
    while True:
        data = stream.read(16000)

        if recognizer.AcceptWaveform(data):
            result = recognizer.Result()
            recognized_text = result[14:-3].strip().lower()

            if recognized_text in ["up", "down", "left", "right"]:
                 move_turtle(recognized_text)
                 print("Recognized:", recognized_text)
            if recognized_text == "stop":
                break

except KeyboardInterrupt:
    pass
# Close the microphone stream and exit
stream.stop_stream()
stream.close()
mic.terminate()
turtle.bye()
```

t.write() is to write text at the current position and t.undo() is to undo the last turtle action when the function move_turtle() is called and the last action undo is t.write(). This line "if recognized_text in ["up", "down", "left", "right"]:" ensures that recognized words printed out are either up, down, left, or right.

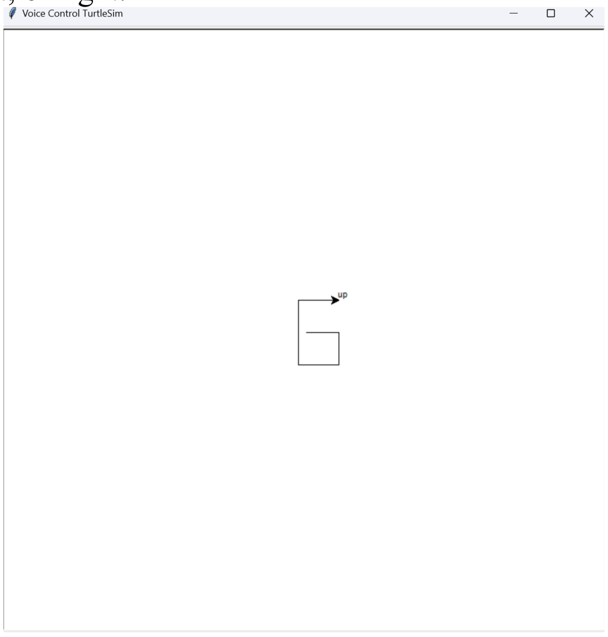
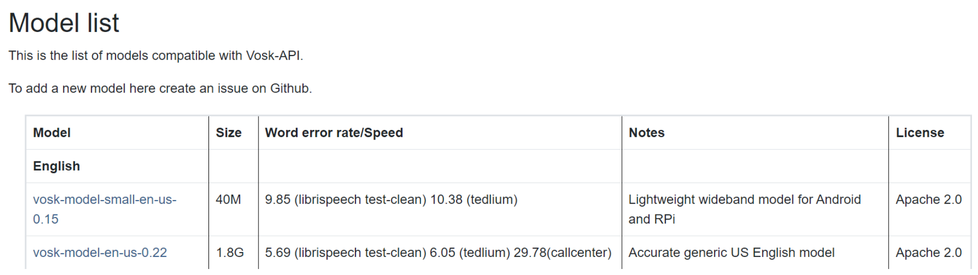


Figure 1: Number 6 made by turtle which is moved by speech recognition

F. Improving the accuracy of speech recognition

Normally, you train the speech recognition model to recognized certain words better like “right”, “left”, “up”, “down” by gathering a bigger dataset of audio recordings containing instances of the target words ("left," "up," "right," "down"). You could also train the pre-trained models to get the results you want but we are going to use a different speech recognition model with lower word error rate.

The ratio of transcription errors to total words uttered is known as word error rate (WER). In speech-to-text, a lower WER indicates improved speech recognition accuracy. A 20% WER, for instance, indicates that the transcript is 80% correct. For example, the small model used in the previous sessions has a WER of 9.85% which means that the model can recognized 90.15% words correctly.



Model list

This is the list of models compatible with Vosk-API.

To add a new model here create an issue on Github.

Model	Size	Word error rate/Speed	Notes	License
English				
vosk-model-small-en-us-0.15	40M	9.85 (librispeech test-clean) 10.38 (tedlium)	Lightweight wideband model for Android and RPi	Apache 2.0
vosk-model-en-us-0.22	1.8G	5.69 (librispeech test-clean) 6.05 (tedlium) 29.78(callcenter)	Accurate generic US English model	Apache 2.0

Figure 2: Vosk API Speech Recognition Models

Let’s look at the second vosk speech recognition model “vosk-model-en-us-0.22”. You can notice the WER for librispeech test-clean and tedium are lower as compared to the first model. Furthermore, the second model has a file size of 1.8GB compared which means it is a big model. Big vosk models are used for high-accuracy transcription on the server and apply advanced AI algorithms. For activity 2), you can choose to use this big vosk model or choose a new speech recognition model like Google Cloud Speech-to-Text etc. ****Beware that loading a big speech recognition model is very resource-intensive and you will need a high-end CPU**

Question 1: Can you tell the difference in the frames_per_buffer and stream.read?

Question 2: What should you change to save more space for your audio files?

Question 3: How to make main.py/main2.py recognize words more accurately?

Activity 1) Edit main2.py so that when the turtle moves, the words displayed has a front size of 20.

Activity 2) Use a different speech recognition model to improve the accuracy of speech recognition.

Reference

alphacephei (no date) Vosk models, VOSK Offline Speech Recognition API. Available at: https://alphacephei.com/vosk/models (Accessed: 19 April 2024).

Google (no date) Select a transcription model | cloud speech-to-text documentation | google cloud, Google. Available at: https://cloud.google.com/speech-to-text/docs/transcription-model (Accessed: 19 April 2024).