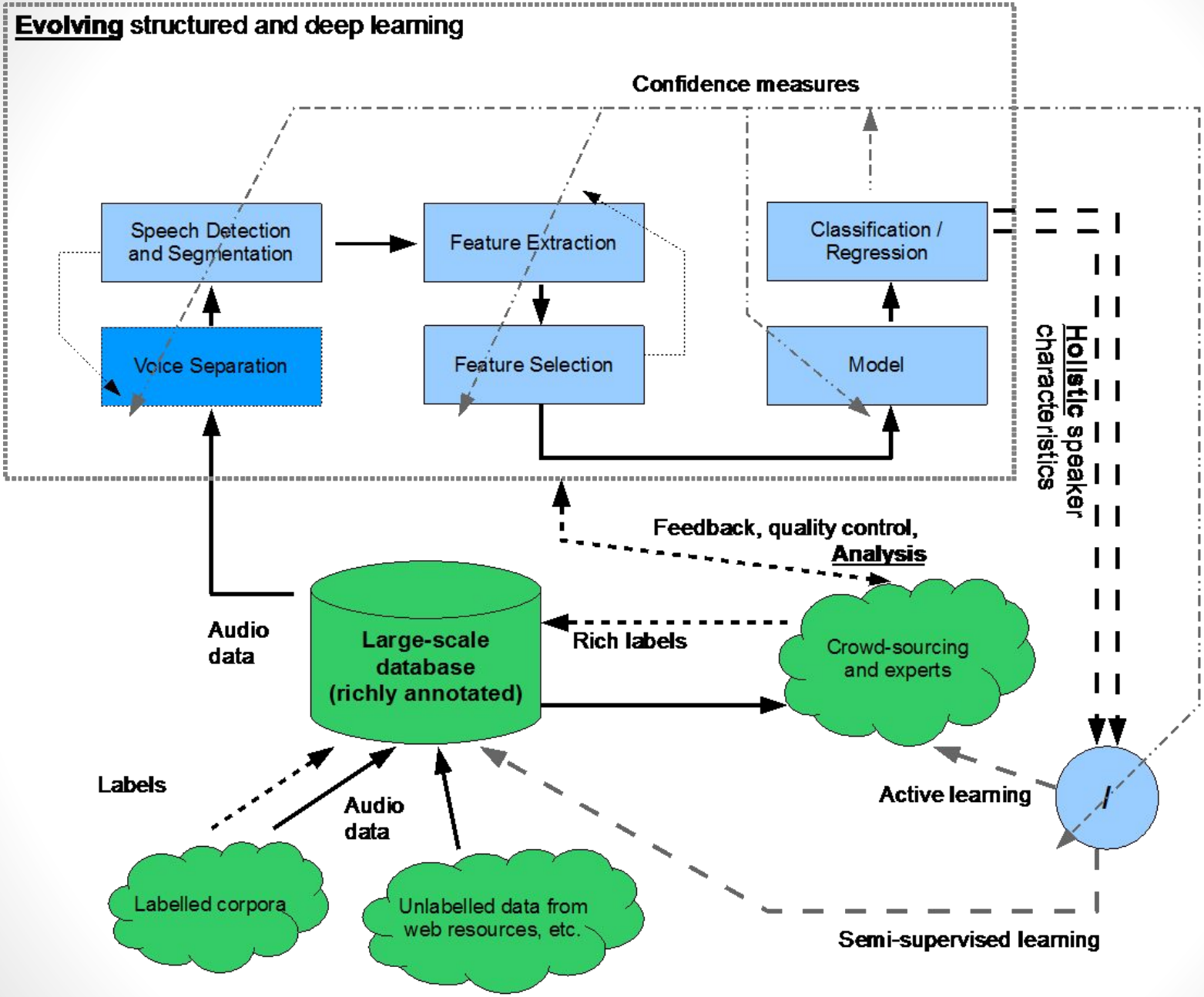
Topic 12 :
Speech Recognition

Speech Recognition

- Experts predict that 50% of all web searches will be made using voice by 2020
- Speech recognition is 1 of the 10 AI technologies that dominates in 2019
  - https://www.aitrends.com/ai-in-business/here-are-10-ai-technologies-that-will-dominate-in-2019/
- Algorithm trained on multiple speakers
- Incredibly complicated, computationally expensive, enormous vocabulary


Evolving structured and deep learning
Confidence measures
Speech Detection and Segmentation
Feature Extraction
Classification / Regression
Voice Separation
Feature Selection
Model
Holistic speaker characteristics
Feedback, quality control, Analysis
Audio data
Large-scale database (richly annotated)
Rich labels
Crowd-sourcing and experts
Labels
Audio data
Active learning
Labelled corpora
Unlabelled data from web resources, etc.
Semi-supervised learning

How does Speech Recognition works?

- Speech converted from physical sound to an electrical signal with a microphone, then to digital data with an analog-to-digital converter.
- Once digitized, several models can be used to transcribe the audio to text.
- Analyze, filter, digitize
  - Analyze sound
  - Filter what people say
  - Digitize into readable format.

How does Speech Recognition works (cont)?

- Neural networks are used to simplify speech signal using techniques for feature transformation and dimensionality reduction *before* HMM recognition.
- Voice activity detectors (VADs) are used to reduce an audio signal to only portions containing speech to prevent unnecessary analysis of parts of the signal.
- Most modern speech recognition systems rely on what is known as a Hidden Markov Model (HMM).

HMM Model for Speech Recognition

- Assume that a speech signal viewed on a short enough timescale (say, ten milliseconds), can be approximated as a stationary process— in which statistical properties do not change over time.
- Speech signal is divided into 10-millisecond fragments. The power spectrum of each fragment, which is essentially a plot of the signal's power as a function of frequency, is mapped to a vector of real numbers known as cepstral coefficients.
- Dimension of this vector is usually small—sometimes as low as 10, although more accurate systems may have dimension 32 or more. The final output of the HMM is a sequence of these vectors.

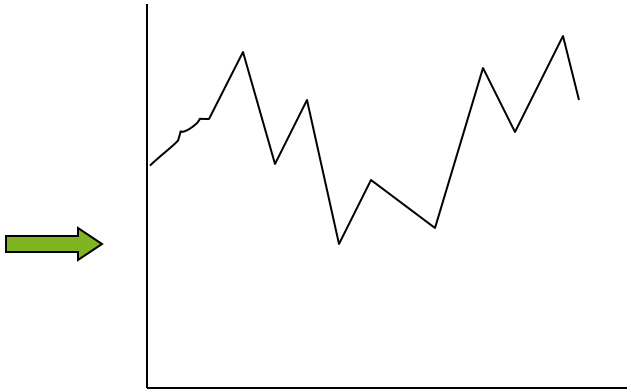
HMM Model for Speech Recognition

- To decode the speech into text, groups of vectors are matched to one or more phonemes—a fundamental unit of speech.
- calculation requires training, since the sound of a phoneme varies from speaker to speaker, and even varies from one utterance to another by the same speaker.
- A special algorithm is then applied to determine the most likely word (or words) that produce the given sequence of phonemes.

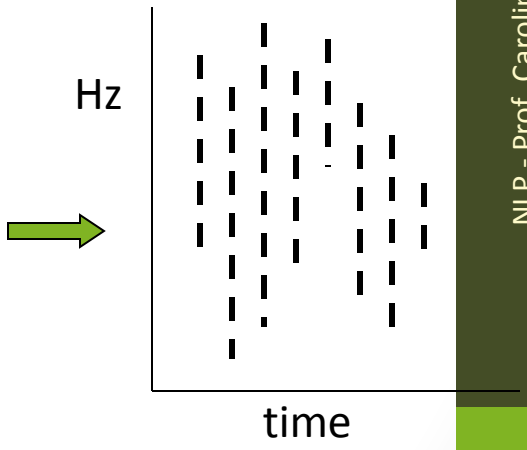
Speech Recognition

Input
(microphone records voice)

Analog Signal



Freq. spectrogram
(e.g. Fourier transform)



Speech Recognition (cont…)

- Frequency spectrogram
  - Basic sounds in the signal (40-50 **phonemes)** (e.g. “a” in “cat”)
    - Template matching against db of phonemes
      - Using dynamic time warping (speech speed)
  - Constructing words from phonemes (e.g. “th”+”i”+”ng”=thing)
    - Unreliable/probabilistic phonemes (e.g. “th” 50%, “f” 30%, …)
    - Non-unique pronunciations (e.g. tomato),
    - statistics of transitions phonemes/words (hidden Markov models)
- Words

Speech Recognition - Complications

- No simple mapping between sounds and words
  - Variance in pronunciation due to gender, dialect, …
    - Restriction to handle just one speaker
  - Same sound corresponding to diff. words
    - e.g. bear, bare
  - Finding gaps between words
    - “how to recognize speech”
    - “how to wreck a nice beach”
  - Noise

Packages Available in Python PyPI

- apiai (include NLP – identify speaker's intent)
- assemblyai
- google-cloud-speech (speech-to-text)
- pocketsphinx
- SpeechRecognition (wrapper for several popular speech APIs, e.g: Google Web Speech API)
- watson-developer-cloud
- wit (include NLP – identify speaker's intent)

Installing Speech Recognition Package

- Go to command prompt and search for the directory containing the file pip.py
- If you do not have pip yet or it is not working, download get-pip.py and place it on your python directory. Run it :
- Example :

  C:\python\Scripts>python get-pip.py
- Next

  C:\python\Scripts>pip install SpeechRecognition

Testing the Speech Recognition Package

- From your Python interpreter/shell :

```
>>> import speech_recognition as sr
>>> sr.__version__
'3.8.1'
>>>
```

- Each `Recognizer` instance has seven methods for recognizing speech from an audio source using various APIs:
  - recognize_bing(): Microsoft Bing Speech
  - recognize_google(): Google Web Speech API
  - recognize_google_cloud(): Google Cloud Speech
    - requires installation of the google-cloud-speech package
  - recognize_houndify(): Houndify by SoundHound
  - recognize_ibm(): IBM Speech to Text
  - recognize_sphinx(): CMU Sphinx - requires installing PocketSphinx
  - recognize_wit(): Wit.ai

Testing the Speech Recognition Package

- From your Python interpreter/shell :

```
>>> import speech_recognition as sr
>>> sr.__version__
'3.8.1'
>>>
```

- Each `Recognizer` instance has seven methods for recognizing speech from an audio source using various APIs:
  - recognize_bing(): Microsoft Bing Speech
  - recognize_google(): Google Web Speech API
  - recognize_google_cloud(): Google Cloud Speech
    - requires installation of the google-cloud-speech package
  - recognize_houndify(): Houndify by SoundHound
  - recognize_ibm(): IBM Speech to Text
  - recognize_sphinx(): CMU Sphinx - requires installing PocketSphinx
  - recognize_wit(): Wit.ai

Transcribing Audio File

- Using `record()` to Capture Data From a File

```
>>> import os
>>> os.chdir("C:/Users/HP/Documents/ummi/CSC4309/sem1_1920/data")
>>> harvard = sr.AudioFile('harvard.wav')
>>> with harvard as source:
        audio = r.record(source)
```

```
>>> type(audio)
<class 'speech_recognition.AudioData'>
>>> r.recognize_google(audio)
'the stale smell of old beer lingers it takes heat to bring out the oder a cold
dip restores health and zest a salt pickle tastes fine with ham tacos Al pastor
are my favourite exist for food is the hot cross bun'
```

Transcribing Audio File

- Capturing segments of speech with `offset()` and `duration()`

```
>>> with harvard as source:
        audio = r.record(source, duration=5)

>>> r.recognize_google(audio)
'the stale smell of old beer lingers'
```

- The `record()` method, when used inside a with block, always moves ahead in the file stream.
- Specifying `duration()` might stop mid-phrase, or mid-word, affecting accuracy of transcription.

```
>>> with harvard as source:
        audio1 = r.record(source, duration=3)
        audio2 = r.record(source, duration=5)

>>> r.recognize_google(audio1)
'the stale smell of all'
>>> r.recognize_google(audio2)
'lingers it takes heat to bring out the oder a cold dip'
```

Transcribing Audio File

- Using `offset()` and `duration()` to ignore beginning of audio and last for certain duration.

```
>>> with harvard as source:
        audio3 = r.record(source, offset=4.5, duration=3.2)

>>> r.recognize_google(audio3)
'it takes me to bring out the oder a call'
```

Removing Noise from Audio File

```
>>> jackhammer = sr.AudioFile('jackhammer.wav')

>>> with jackhammer as noise:
        audioN = r.record(noise)

>>> r.recognize_google(audioN)
'smell fingers'

>>> with jackhammer as noise:
        r.adjust_for_ambient_noise(noise)
        audio = r.record(noise)

>>> r.recognize_google(audio)
'smell all gear lingers'

>>> with jackhammer as noise:
        r.adjust_for_ambient_noise(noise,duration=0.5)
        audio2 = r.record(noise)


>>> r.recognize_google(audio2)
'smell lingers'
```

Sample Guess Word Game

- Sample project : https://realpython.com/python-speech-recognition/#

- Sample speech:
  - https://www.voiptroubleshooter.com/open_speech/american.html
  - http://www.cstr.ed.ac.uk/projects/eustace/download.html

Group Exercise
Due in next class

Based on the values in Slide 33 :

i. Calculate the probabilities of $v_t(j)$ at each time *t* using the ***viterbi*** algorithm.

ii. Subsequently, modify the program in viterbi2.py to calculate the probabilities of **"The cook prepares a lovely drink"** using the ***viterbi*** algorithm in Python to verify your answer in (i)