SPEECH/VISION COGNITIVE SYSTEMS

Dr TIAN Jing
tianjing@nus.edu.sg

Module objective

Knowledge and understanding

- Understand the fundamentals of statistical speech recognition systems
- Understand basic concepts of vision cognitive systems

Key skills

- Design, build, implement and evaluate speech recognition approach in Python

Major reference

- [Introduction] CS131: ***Computer Vision: Foundations and Applications***, http://vision.stanford.edu/teaching/cs131_fall1718/syllabus.html
- [Comprehensive] ***Computer Vision Crash Course***, https://filebox.ece.vt.edu/~jbhuang/
- [Introduction] ***Automatic Speech Recognition***,
  https://github.com/ekapolc/ASR_course
- [Comprehensive] CS224S, ***Spoken Language Processing***,
  http://web.stanford.edu/class/cs224s/
- [Book] ***Speech and language processing***,
  https://web.stanford.edu/~jurafsky/slp3/

Topics

- Vision cognition systems
- Speech recognition systems
- Workshop: Design and build speech recognition system in Python

Vision cognition

The first computer vision project in 1966.

Abstract: The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system. The particular task was chosen because it can be segmented into sub-problems which allow individuals to work independently and yet participate in the construction of a system complex enough to be a real landmark in the development of “pattern recognition”.

Tasks

- Figure ground: Divide a picture into regions such as likely objects, likely background areas.
- Region description: Analysis of shape and surface properties.
- Object identification: Name objects by matching them with a vocabulary of known objects.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
PROJECT MAC

Artificial Intelligence Group July 7, 1966
Vision Memo. No. 100.

THE SUMMER VISION PROJECT

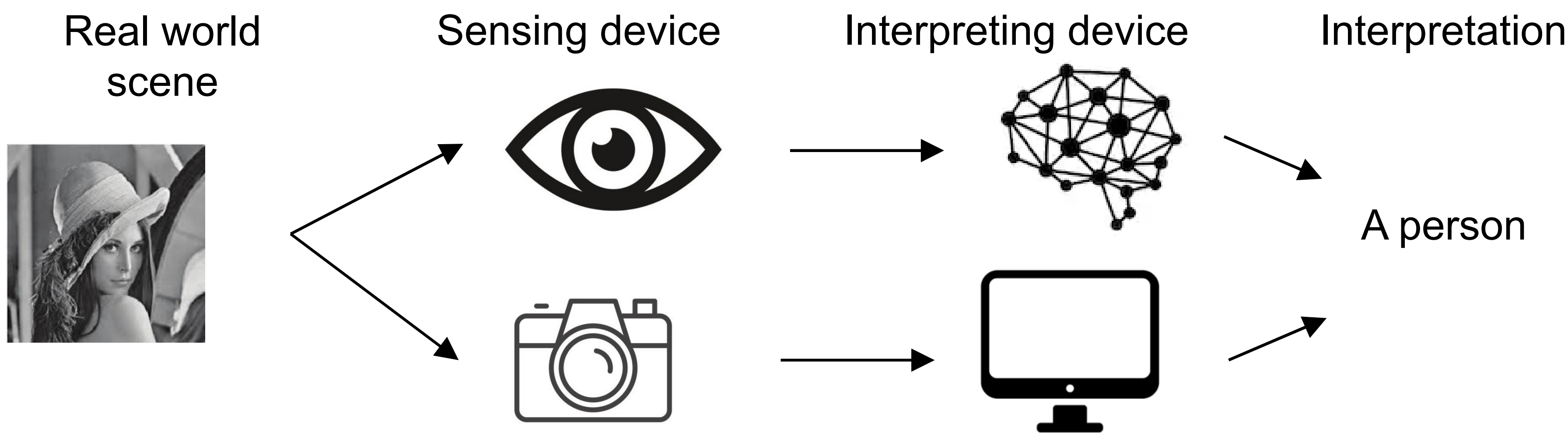
Seymour Papert

The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system. The particular task was chosen partly because it can be segmented into sub-problems which will allow individuals to work independently and yet participate in the construction of a system complex enough to be a real landmark in the development of "pattern recognition".

Reference: http://people.csail.mit.edu/brooks/idocs/AIM-100.pdf

Thinking humanly

- Humans use their eyes and brains to visually sense the world.
- Computers use their cameras and computation to visually sense the world.



Computers	Brains
Fixed architecture	Evolving architecture
Modular, (primarily) serial	Massively parallel
Separate hardware, software	No distinction between hardware and software
Separate computation, memory	No distinction between computation and memory

Reference: http://scienceblogs.com/developingintelligence/2007/03/27/why-the-brain-is-not-like-a-co/

Vision cognitive system pipeline

Application domain → Image acquisition → Image enhancement → Image processing and analysis → Feature representation & description → Object detection and recognition → Scene understanding → Decision

Reference:

- Mo Zi, 400 B. C., https://en.wikipedia.org/wiki/Camera_obscura
- Picture source, http://www.sohu.com/a/140776287_736731
- https://www.indiamart.com/proddetail/street-pole-cctv-camera-14049923862.html

Vision cognitive system pipeline

Application domain → Image acquisition → Image enhancement → Image processing and analysis → Feature representation & description → Object detection and recognition → Scene understanding → Decision

Online demo: http://ipolcore.ipol.im/demo/clientApp/demo.html?id=230

Vision cognitive system pipeline

Application domain → Image acquisition → Image enhancement → Image processing and analysis → Feature representation & description → Object detection and recognition → Scene understanding → Decision

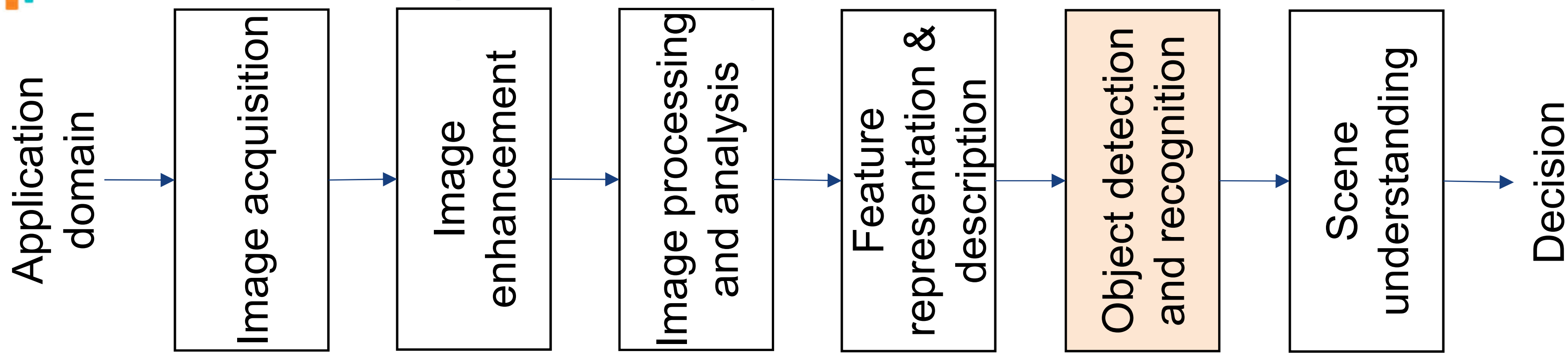
Online demo: http://bigwww.epfl.ch/demo/ip/demos/edgeDetector/

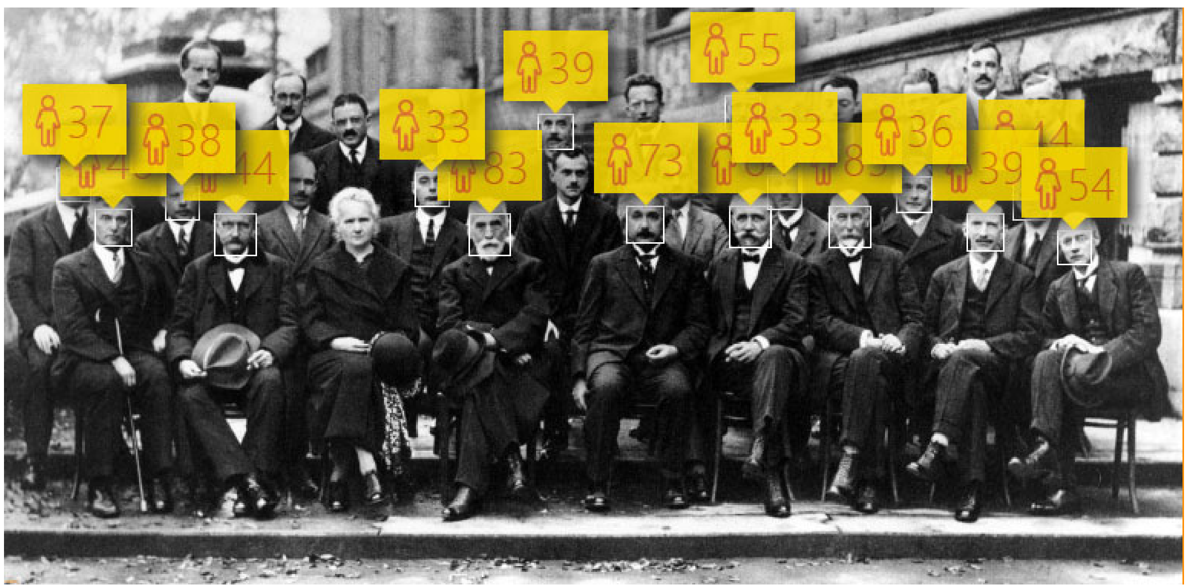
Vision cognitive system pipeline

Application domain → Image acquisition → Image enhancement → Image processing and analysis → **Feature representation & description** → Object detection and recognition → Scene understanding → Decision

Online demo: http://demo.ipol.im/demo/my_affine_sift/

Vision cognitive system pipeline



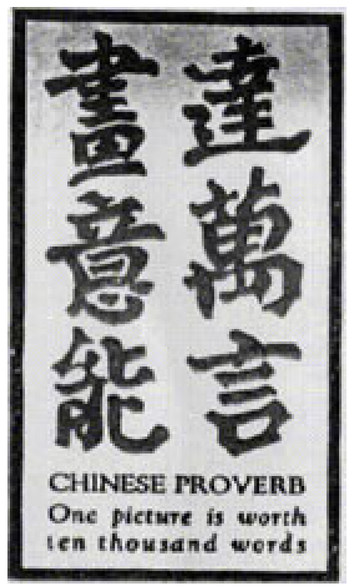


Three fundamental tasks
- Classification
- Detection
- Segmentation

Demo website: https://www.how-old.net

Vision cognitive system pipeline

Application domain → Image acquisition → Image enhancement → Image processing and analysis → Feature representation & description → Object detection and recognition → Scene understanding → Decision



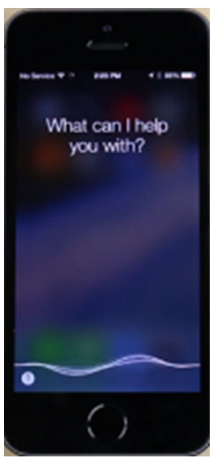
Reference
- https://en.wikipedia.org/wiki/A_picture_is_worth_a_thousand_words
- https://www.phrases.org.uk/meanings/a-picture-is-worth-a-thousand-words.html

Automatic speech recognition

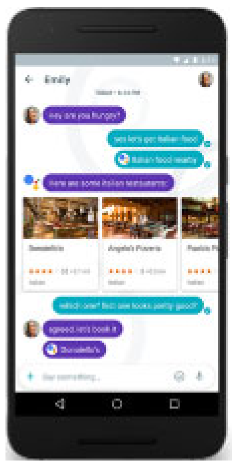
Amazon Echo
2015

Google Home
2016

Facebook M
2015

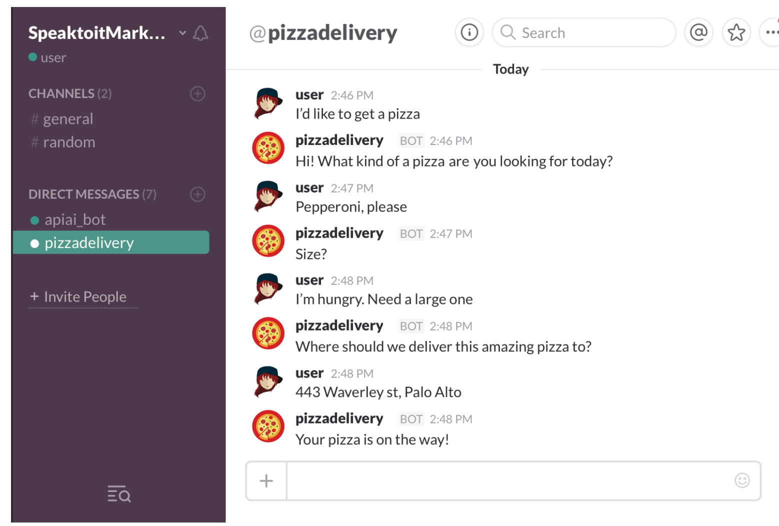


Apple
Siri
2011



Google
Assistant
2016

Microsoft
Cortana
2014



Slack Bot API
2015

Source: CS224S Spoken Language Processing, http://web.stanford.edu/class/cs224s/

Automatic speech recognition

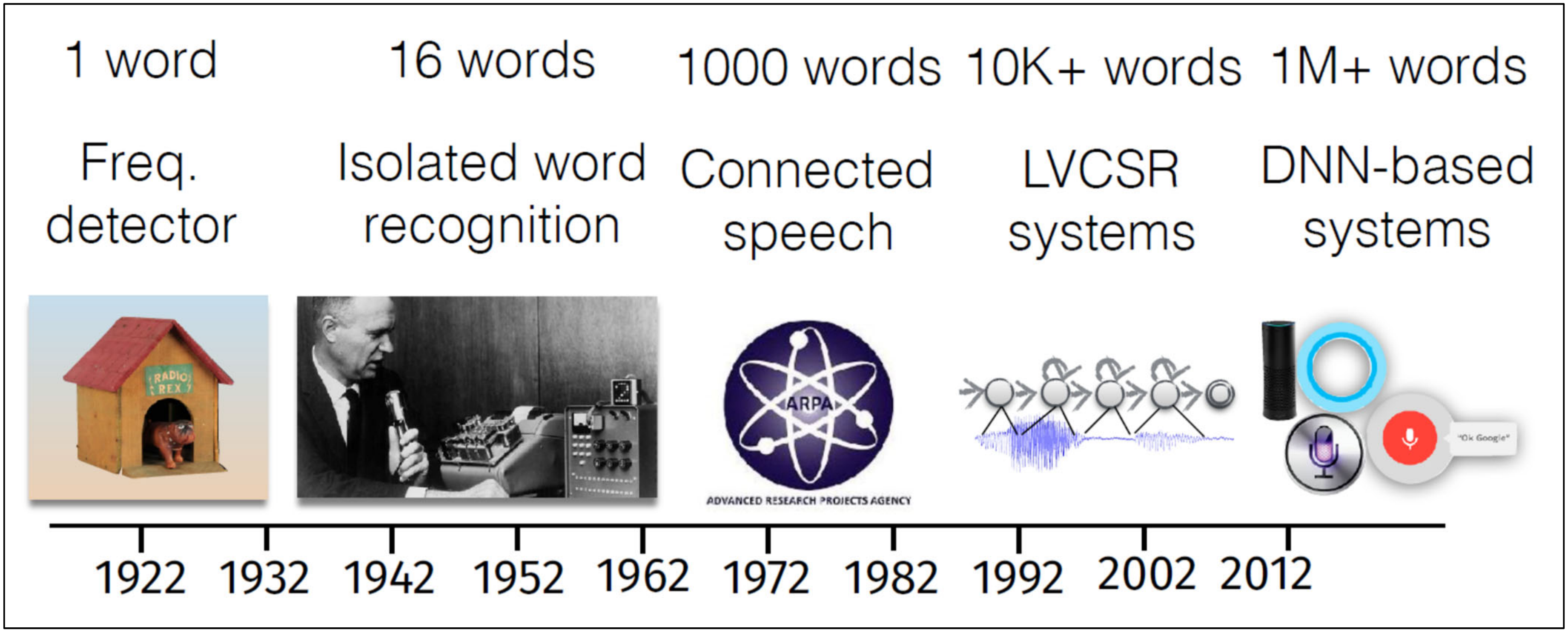
- Human-machine Interaction
  - Automatic Speech Recognition
  - Speech Synthesis / Text-to-Speech (TTS)
  - Natural Language Generation (NLG)
- Language
  - Statistical Machine Translation (SMT)
- Language Acquisition
  - Pronunciation Training
- Security/Forensics
  - Speaker ID
  - Speaker Verification
- Medical Applications
  - Diagnosis of Diseases
- Information Retrieval
  - Video/Audio Transcribing
  - Audio/Text Summarizing
- Speech Manipulation
  - Speaking Rate Adjusting

Automatic speech recognition

Challenges of speech recognition

- Style: Read speech or spontaneous (conversational) speech?
- Continuous natural speech or command & control?
- Speaker characteristics: Rate of speech, accent, prosody (stress, intonation), speaker age, pronunciation variability even when the same speaker speaks the same word
- Channel characteristics: Background noise, room acoustics, microphone properties, interfering speakers
- Task specifics: Vocabulary size (the number of words to be recognized), language-specific complexity, computational resource limitations

Automatic speech recognition



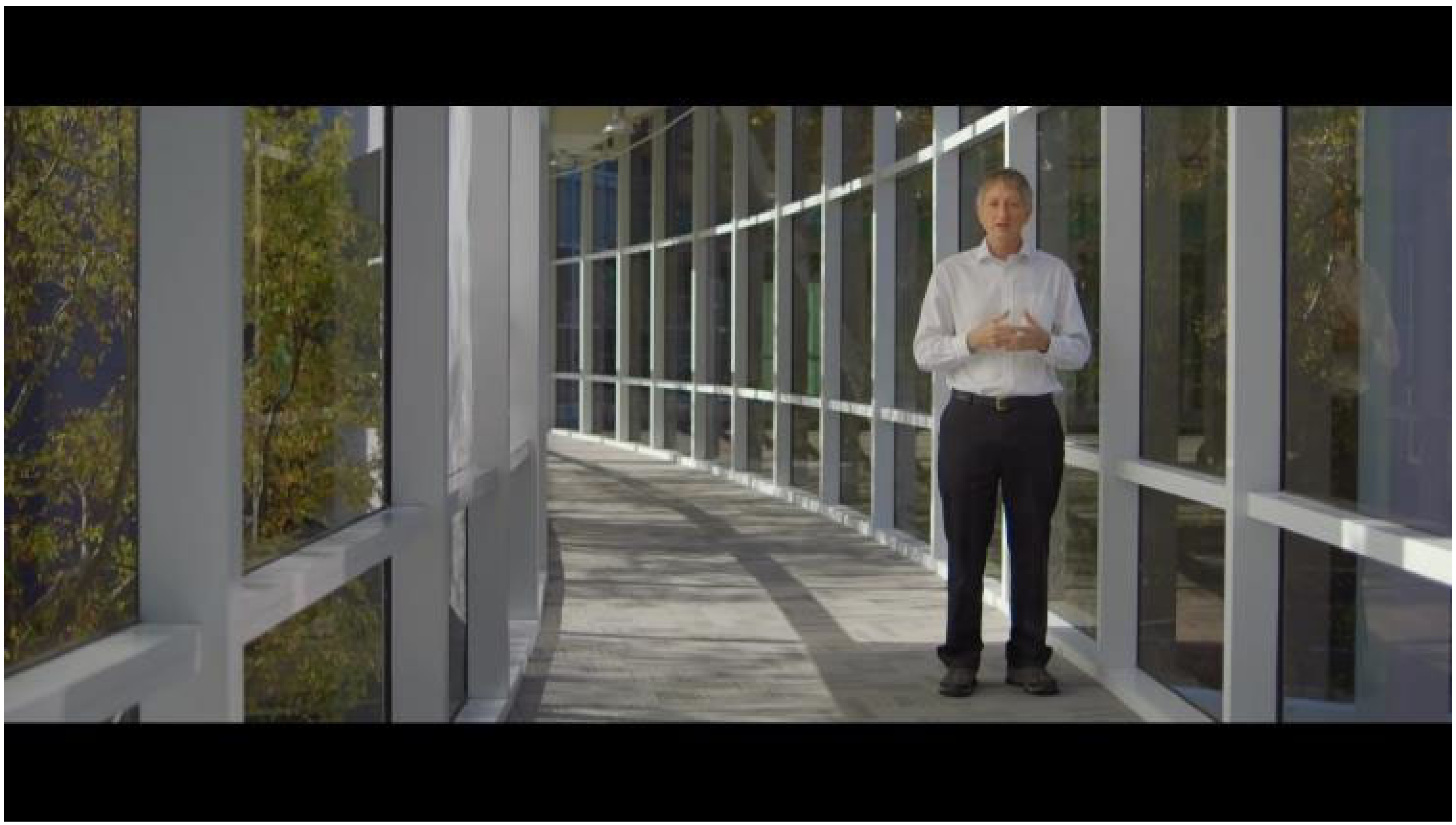
More introductions to history of automatic speech recognition can be found at

- https://ileriseviye.wordpress.com/2011/02/17/speech-recognition-in-1920s-radio-rex-the-first-speech-recognition-machine/
- https://machinelearning-blog.com/2018/09/07/a-brief-history-of-asr-automatic-speech-recognition/

Source: Automatic Speech Recognition (CS753), Lecture 1: Introduction to Statistical Speech Recognition, https://www.cse.iitb.ac.in/~pjyothi/cs753/

Automatic speech recognition

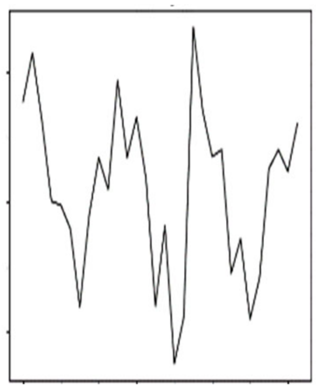
Behind the Mic: The Science of Talking with Computers, (6 minutes)
https://www.youtube.com/watch?v=yxxRAHVtafI
Language is easy for humans to understand (most of the time), but not so easy for computers. This video talks about speech recognition, language understanding, neural nets, and using our voices to communicate with the technology around us.



Speech recognition pipeline (1)

Objective: Recognize the word sequence given the input audio sequence.

Input audio sequence $\mathbf{A}$



Speech recognition

Word sequence $\mathbf{W} = \{\text{I stay in Singapore.}\}$

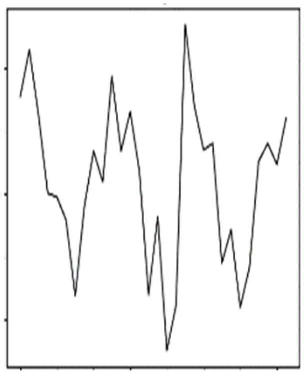
Let $\mathbf{A}$ represent an audio sequence and $\mathbf{W}$ denote a word sequence, then the speech recognizer decodes $\mathbf{W}^*$ as

$$\boxed{\mathbf{w}^* = \underset{\mathbf{W}}{\operatorname{argmax}}\, P(\mathbf{W}|\mathbf{A}) = \underset{\mathbf{W}}{\operatorname{argmax}} \frac{P(\mathbf{A}|\mathbf{W})P(\mathbf{W})}{P(\mathbf{A})} \propto \underset{\mathbf{W}}{\operatorname{argmax}}\, P(\mathbf{A}|\mathbf{W})\mathrm{P}(\mathbf{W})}$$

Further introduce acoustic features $\mathbf{O}$, phoneme $\mathbf{L}$, the optimization problem statement can be rewritten to be

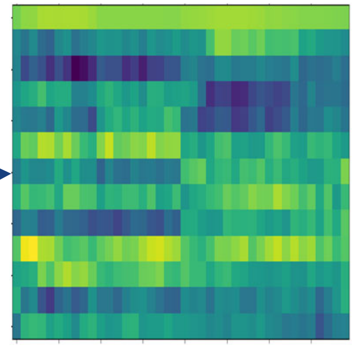
$$\boxed{\mathbf{w}^* = \underset{\mathbf{W}}{\operatorname{argmax}}\, P(\mathbf{A}|\mathbf{W})\mathrm{P}(\mathbf{W}) = \underset{\mathbf{W}}{\operatorname{argmax}}\, P(\mathbf{A}|\mathbf{O})P(\mathbf{O}|\boldsymbol{L})P(\mathbf{L}|\mathbf{W})P(\mathbf{W})}$$

Speech recognition pipeline (2)



Input audio sequence $\mathbf{A}$

→ Feature extraction →

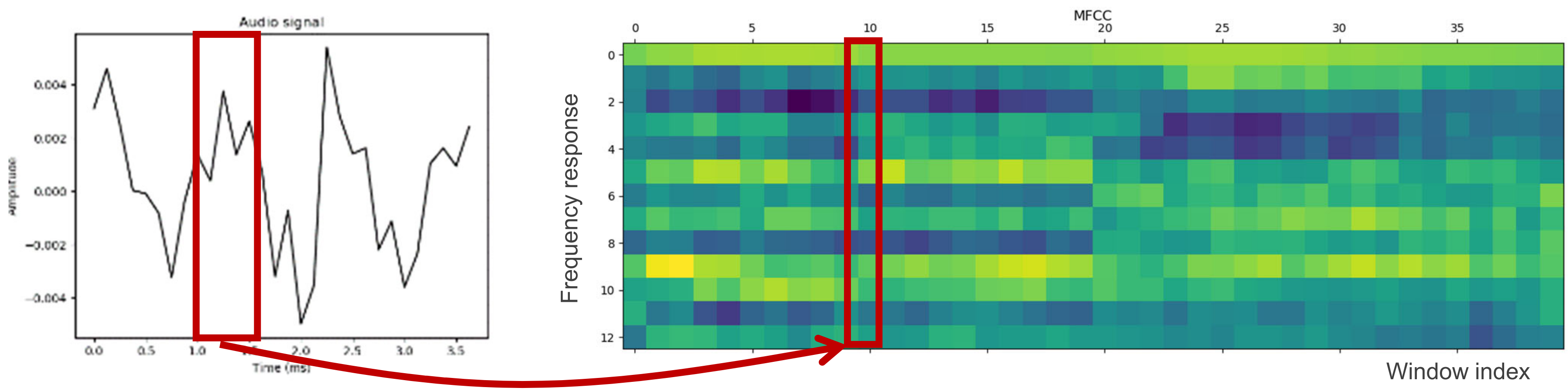


Extracted frequency-domain features $\mathbf{O}$

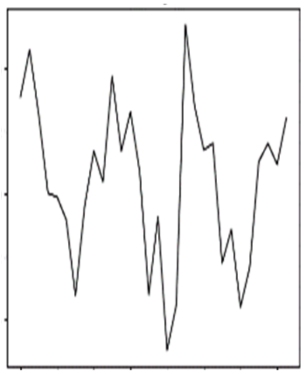
← Acoustic model ← Pronunciation model ← Language model

Word sequence $\mathbf{W}$: “I stay in Singapore”

Speech signal represented in time-domain (left figure) and frequency-domain (right figure), e.g., *Mel frequency cepstral coefficient* (MFCC), where a sliding window is applied to select a short interval signal then apply a frequency transformation (e.g., Fourier transform) to generate the response (one column in right figure).

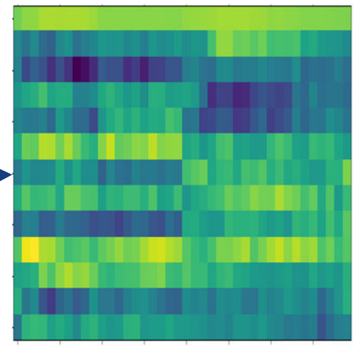


Speech recognition pipeline (3)



Input audio sequence **A**

Feature extraction



Extracted frequency-domain features **O**

Acoustic model

Pronunciation model

Language model

Word sequence **W**: "I stay in Singapore"

N = 1 : This | is | a | sentence — *unigrams:* this, is, a, sentence

N = 2 : This is | a sentence — *bigrams:* this is, is a, a sentence

N = 3 : This | is a | sentence — *trigrams:* this is a, is a sentence

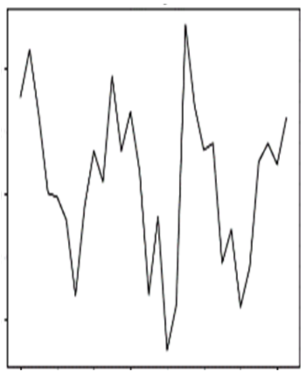
N-gram models: Build the language model by calculating probabilities from text training corpus: How likely is one word to follow another.

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Example: Bi-gram in the Berkeley Restaurant Project corpus of 9332 sentences.

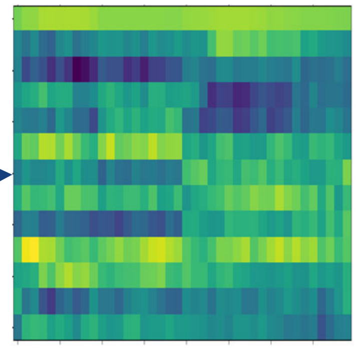
Reference: https://deepai.org/machine-learning-glossary-and-terms/n-gram

Speech recognition pipeline (4)



Input audio sequence **A**

Feature extraction



Extracted frequency-domain features **O**

Acoustic model

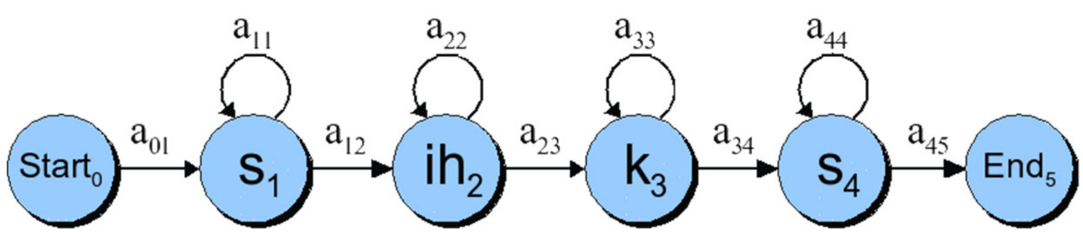
Pronunciation model

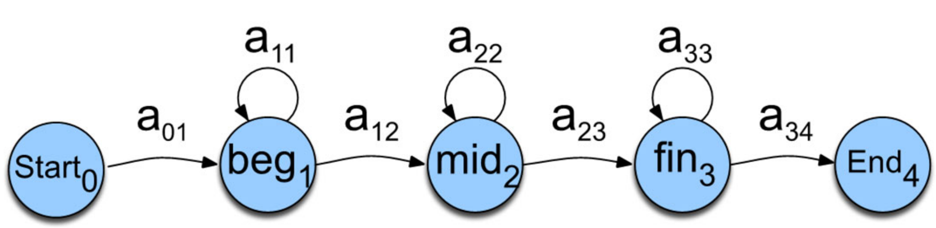
Phoneme **L**

Language model

Word sequence **W**: "I stay in Singapore"

I	AY
STAY	S T EY
IN	IH N
SINGAPORE	S IH NG AH P AO R

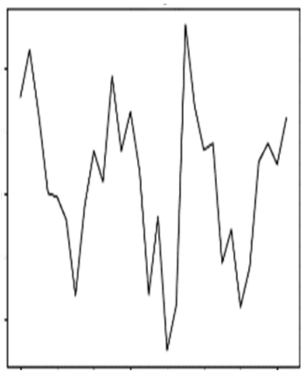




The Carnegie Mellon University Pronouncing Dictionary is an open-source machine-readable pronunciation dictionary for North American English that contains over 134,000 words and their pronunciations.

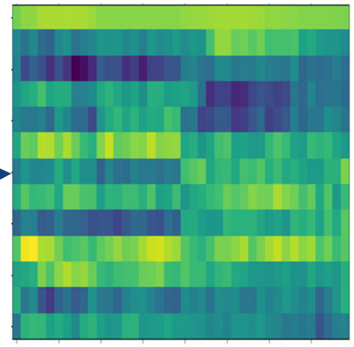
Reference: http://www.speech.cs.cmu.edu/cgi-bin/cmudict

Speech recognition pipeline (5)



Input audio sequence $\mathbf{A}$

→ Feature extraction →



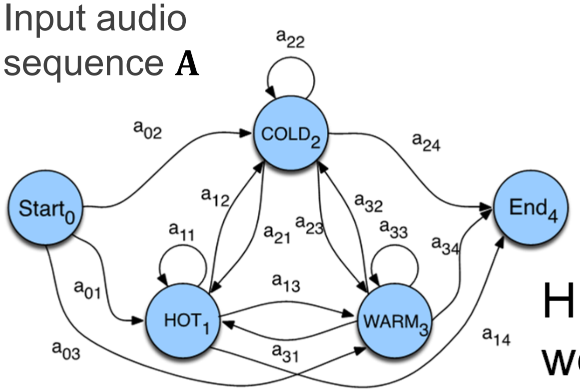
Extracted frequency-domain features $\mathbf{O}$

← Acoustic model ← Pronunciation model ← Language model

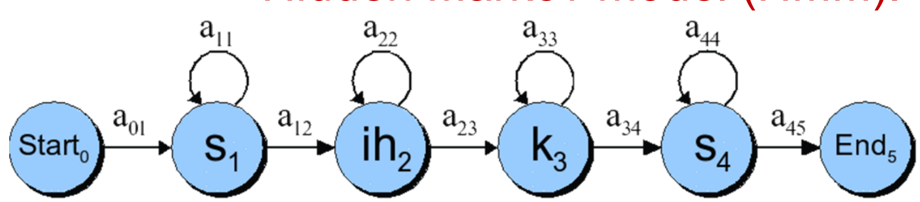
Phoneme $\mathbf{L}$

Word sequence $\mathbf{W}$: "I stay in Singapore"

Key challenge is how to estimate sequence.
→ Hidden Markov model (HMM).



HMM for weather



HMM for words

$Q = \{q_1, q_2, \cdots, q_t\}$	A set of N states for observations	Each observation has one state
$A = \{a_{11}, a_{12}, \cdots, a_{nn}\}$	A state transition probability matrix A, each a_{ij} representing the probability of moving from the state i to the state j, $s.t. \sum_{j=1}^{n} a_{ij} = 1$	Learned from speech training dataset
$O = \{o_1, o_2, \cdots, o_t\}$	A sequence of T observations	Observed speech data
$B = b_i(o_t)$	An observation likelihood, called emission probability, representing the probability of an observation o_t being generated from a state i	Learned from speech training dataset
S	A set of states (e.g., HOT_1, $COLD_2$, $WARM_3$, S_1, ih_2, k_3, etc), a special start state $Start_0$ and an end state End_4 that are not associated with observations, together with their transition probabilities out of the start state and into the end state.	

HMM: A toy example

State pransition probability				Observation likelihood		
Today weather	Tomorrow weather			Weather	Probability of	
	Sunny (S)	Raining (R)	Cloudy (C)		Umbrella (U)	No umbrella (N)
Sunny (S)	0.8	0.05	0.15	Sunny (S)	0.1	0.9
Raining (R)	0.2	0.6	0.2	Raining (R)	0.8	0.2
Cloudy (C)	0.2	0.3	0.5	Cloudy (C)	0.3	0.7

Q: Given that today weather is S, what is the probability that tomorrow is S and the day after is R?

Markov assumption

$$P(q_2 = S, q_3 = R | q_1 = S) = P(q_3 = R | q_2 = S, q_1 = S) P(q_2 = S | q_1 = S)$$
$$= P(q_3 = R | q_2 = S) P(q_2 = S | q_1 = S) = 0.05 \times 0.8 = 0.04$$

Q: Given that you don't use umbrella (N) for three days, calculate the probability for the weather on these three days to be $\{q_1 = S, q_2 = C, q_3 = S\}$. Note that the prior probability for the start state as sunny (S) on day one is assumed to be 1/3 (three weather has the same probability).

$$P(q_1 = S, q_2 = C, q_3 = S | o_1 = N, o_2 = N, o_3 = N)$$
$$= P(o_1 = N | q_1 = S) P(o_2 = N | q_2 = C) P(o_3 = N | q_3 = S) P(q_1 = S) P(q_2 = C | q_1 = S) P(q_3 = S | q_2 = C)$$
$$= 0.9 \times 0.7 \times 0.9 \times 1/3 \times 0.15 \times 0.2 = 0.0057$$

Reference: http://www.iitg.ac.in/samudravijaya/tutorials/hmmTutorialBarbaraExercises.pdf

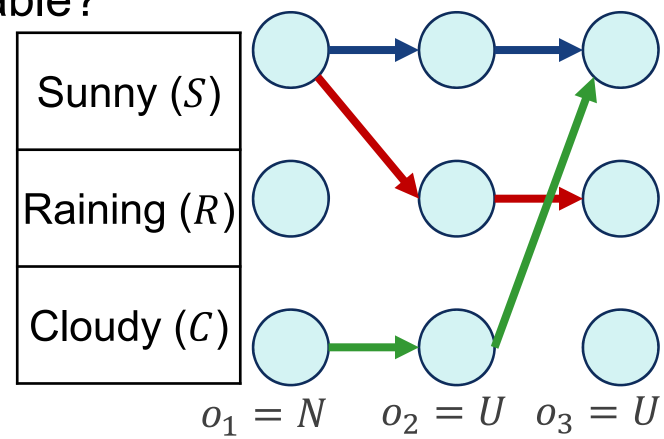
HMM: Sequence estimation

Q: Given that three days your umbrella observations are: {no umbrella (N), umbrella (U), umbrella (U)}, find the most probable weather-sequence.

Idea 1: If we ignore the weather as a 'sequence' and treat each day weather separately, the most probable weather are Sunny (S), Raining (R), Raining (R).

Idea 2: Exhaustively evaluate probability of each sequence. For example, consider following three possible sequences, which is most probable?

- Blue sequence: Sunny (S), Sunny (S), Sunny (S)
- Red sequence: Sunny (S), Raining (R), Raining (R)
- Green sequence: Cloudy (C), Cloudy (C), Sunny (S)



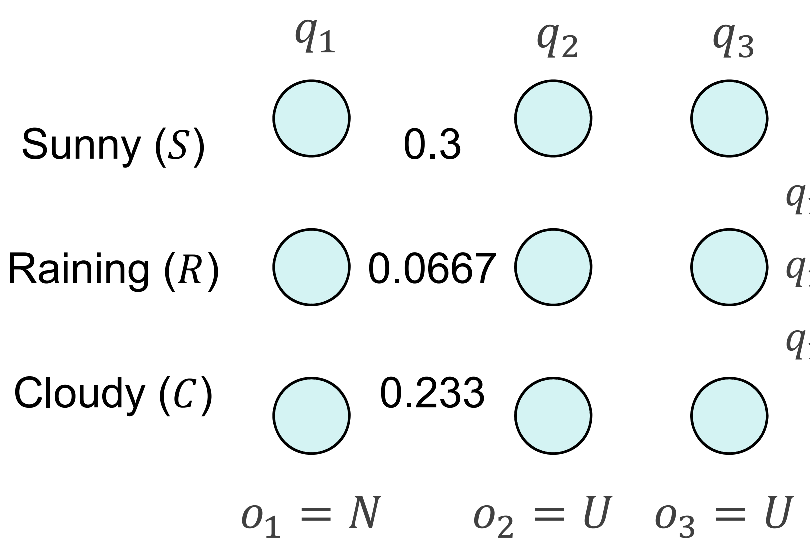
Idea 3: Design an efficient method to evaluate all possible sequence and find the most probable one.
→ We will study Viterbi algorithm in next few slides.

Viterbi: A single-line $.predict(O)$ function in hmmlearn library

HMM: Viterbi algorithm

Key idea: “Optimal policy is composed of optimal sub-policies”.

1. Initialization: Calculate probability of the first day state based on first day observation and (assumed to be equal) prior probability starting from all possible states.
2. Recursion: For all following days, calculate probability of each state based on current observation and the largest (previous state probability × transition probability) from the previous day. Record the ‘best path’ ending at current state from the previous day.
3. Termination and back tracing: For the last day, choose the state with the highest probability. Trace back according to the recorded most probable path.



q_1 is S: $P(q_1 = S) \times P(o_1 = N | q_1 = S) = \frac{1}{3} \times 0.9 = 0.3$

q_1 is R: $P(q_1 = R) \times P(o_1 = N | q_1 = R) = \frac{1}{3} \times 0.2 = 0.0667$

q_1 is C: $P(q_1 = C) \times P(o_1 = N | q_1 = C) = \frac{1}{3} \times 0.7 = 0.233$

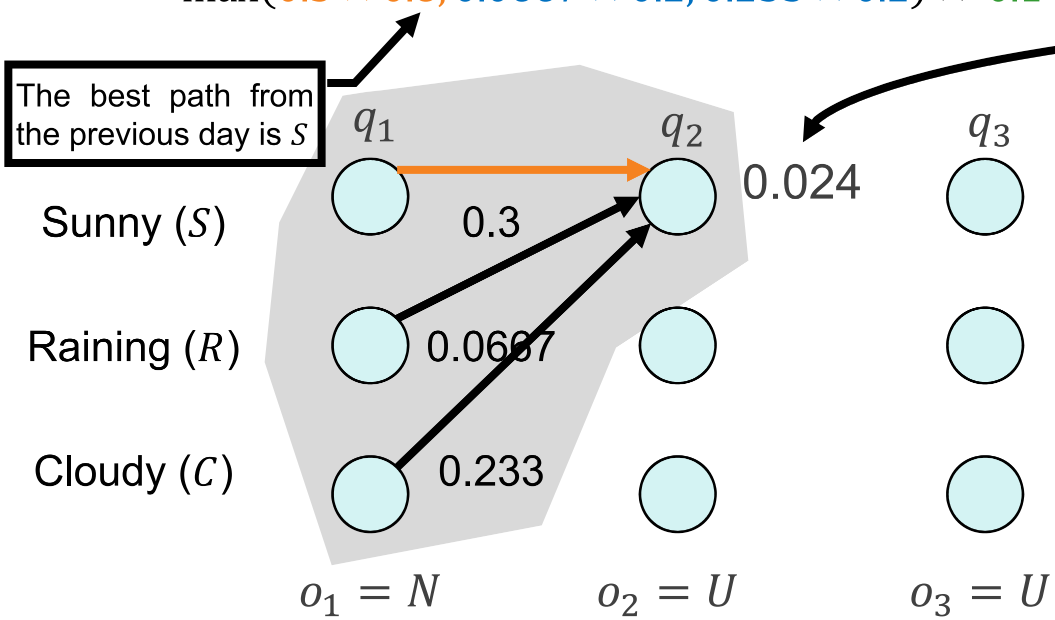
HMM: Viterbi algorithm

Key idea: "Optimal policy is composed of optimal sub-policies".

1. Initialization: Calculate probability of the first day state based on first day observation and (equal) prior probability starting from all possible states.
2. Recursion: For all following days, calculate probability of each state based on current observation and the largest (previous state probability × transition probability) from the previous day. Record the 'best path' ending at current state from the previous day.
3. Termination and back tracing: For the last day, choose the state with the highest probability. Trace back according to the recorded most probable path.

q_2 is S: $\max(P(q_1 = S)\alpha_{ss}, P(q_1 = R)\alpha_{rs}, P(q_1 = C)\alpha_{cs})\ P(o_2 = U|q_2 = S)$

$= \max(0.3 \times 0.8, 0.0667 \times 0.2, 0.233 \times 0.2) \times\ 0.1\ = 0.24 \times 0.1 =\ 0.024$



'Best path' is defined as the best, that is, the largest (previous state probability × transition probability), among three possible paths highlighted in gray region.

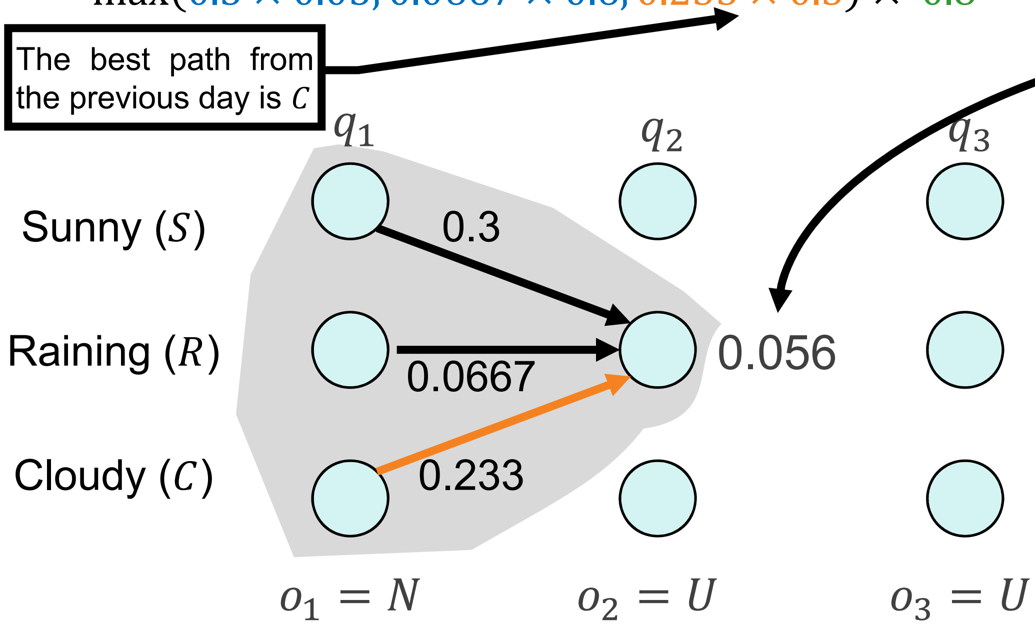
Note: α is transition probability, $\alpha_{ss} = 0.8$ from S to S

HMM: Viterbi algorithm

Key idea: “Optimal policy is composed of optimal sub-policies”.

1. Initialization: Calculate probability of the first day state based on first day observation and (equal) prior probability starting from all possible states.
2. Recursion: For all following days, calculate probability of each state based on current observation and the largest (previous state probability × transition probability) from the previous day. Record the ‘best path’ ending at current state from the previous day.
3. Termination and back tracing: For the last day, choose the state with the highest probability. Trace back according to the recorded most probable path.

q_2 is R: $\max(P(q_1 = S)\alpha_{sr}, P(q_1 = R)\alpha_{rr}, P(q_1 = C)\alpha_{cr})\ P(o_2 = U|q_2 = R)$
$= \max(0.3 \times 0.05, 0.0667 \times 0.6, 0.233 \times 0.3) \times\ 0.8\ = 0.233 \times 0.8 =\ 0.056$



‘Best path’ is defined as the best, that is, the largest (previous state probability × transition probability), among three possible paths highlighted in gray region.

Note: α is transition probability, $\alpha_{ss} = 0.8$ from S to S

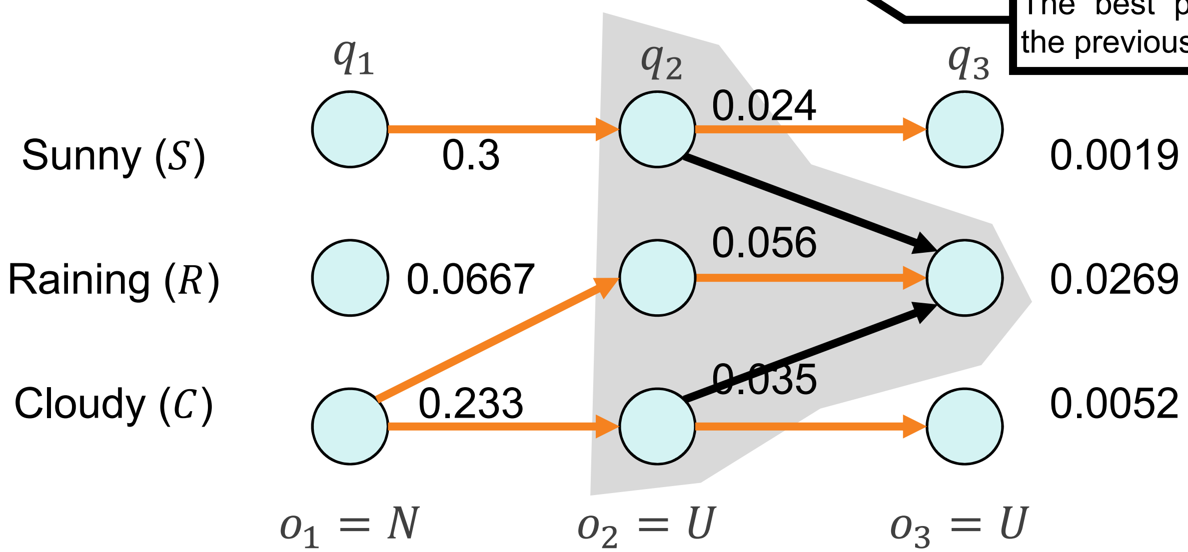
HMM: Viterbi algorithm

Key idea: "Optimal policy is composed of optimal sub-policies".

1. Initialization: Calculate probability of the first day state based on first day observation and (equal) prior probability starting from all possible states.
2. Recursion: For all following days, calculate probability of each state based on current observation and the largest (previous state probability × transition probability) from the previous day. Record the 'best path' ending at current state from the previous day.
3. Termination and back tracing: For the last day, choose the state with the highest probability. Trace back according to the recorded most probable path.

q_3 is R: $\max(P(q_2 = S)\alpha_{sr}, P(q_2 = R)\alpha_{rr}, P(q_2 = C)\alpha_{cr})\, P(o_3 = U | q_3 = R)$
$= \max(0.024 \times 0.05, 0.056 \times 0.6, 0.035 \times 0.3) \times 0.8 = 0.0336 \times 0.8 = 0.0269$

The best path from the previous day is R



'Best path' is defined as the best, that is, the largest (previous state probability × transition probability), among three possible paths highlighted in gray region.

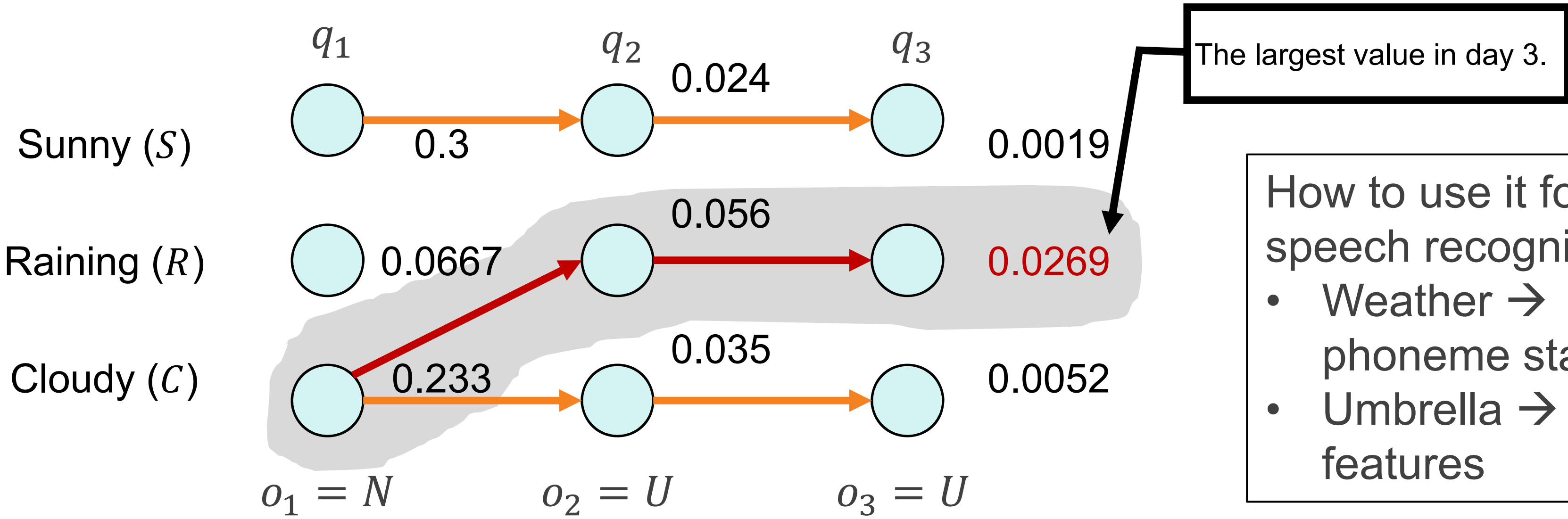
Note: α is transition probability, $\alpha_{ss} = 0.8$ from S to S

HMM: Viterbi algorithm

Key idea: "Optimal policy is composed of optimal sub-policies".

1. Initialization: Calculate probability of the first day state based on first day observation and (equal) prior probability starting from all possible states.
2. Recursion: For all following days, calculate probability of each state based on current observation and the largest (previous state probability × transition probability) from the previous day. Record the 'best path' ending at current state from the previous day.
3. Termination and back tracing: For the last day, choose the state with the highest probability. Trace back according to the recorded most probable path.

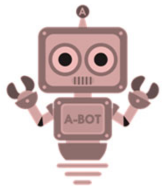
The optimal sequence: Cloudy (C), Raining (R), Raining (R).
Recall that the result (in previous Idea 1) is Sunny (S), Raining (R), Raining (R).



How to use it for speech recognition?

- Weather → phoneme state
- Umbrella → audio features

Use case: Language, vision and actions



Did anyone enter this room last week?

Yes, 127 instances logged on camera

Show me images of anyone carrying a black bag.

…

Is there smoke in any room around you?

Yes, in one room

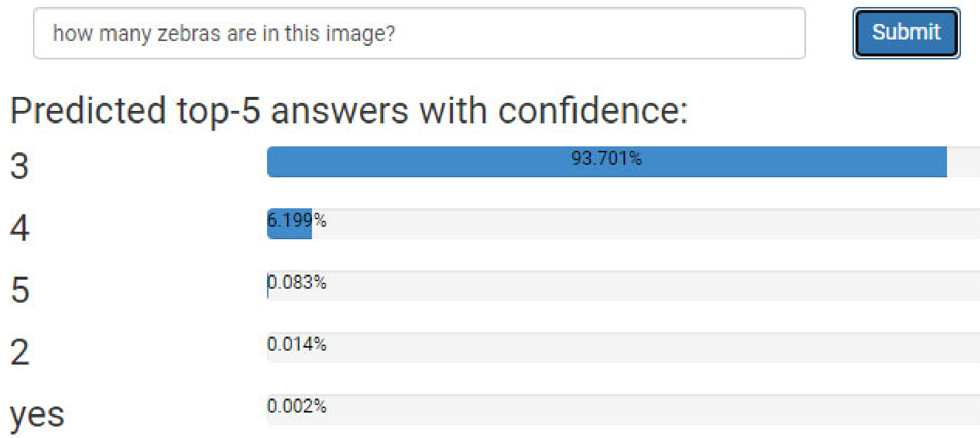
Go there and look for people

…

Reference: Connecting language and vision to actions, https://lvatutorial.github.io/

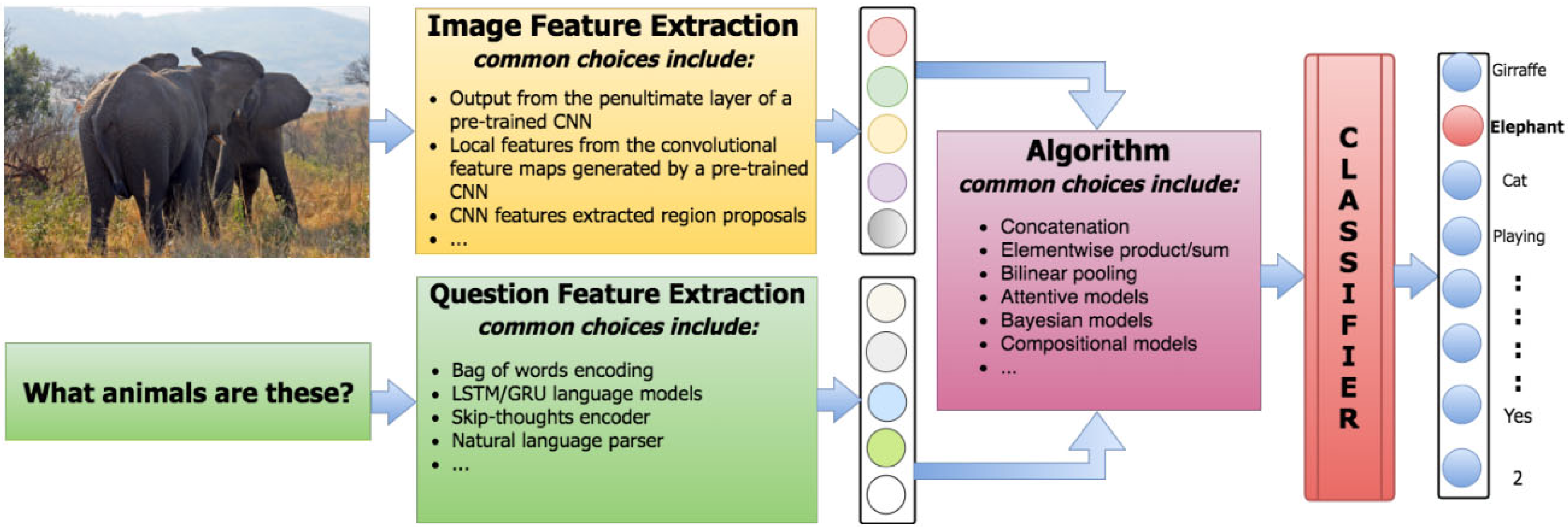
Use case: Visual question answering (VQA)

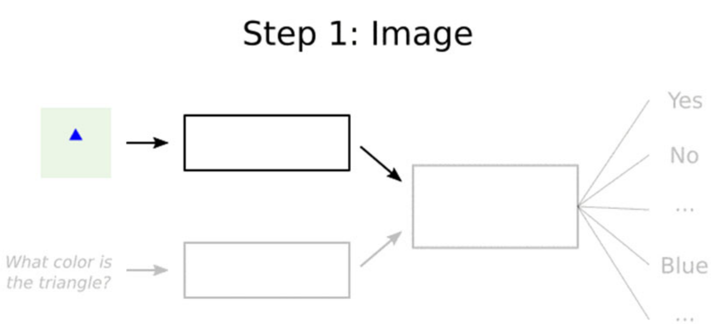
Objective: Given an image and a natural language open-ended question, generate a natural language answer. This is reasoning techniques using both language and vision knowledge.



Demo website: http://vqa.cloudcv.org

Use case: Visual question answering (VQA)





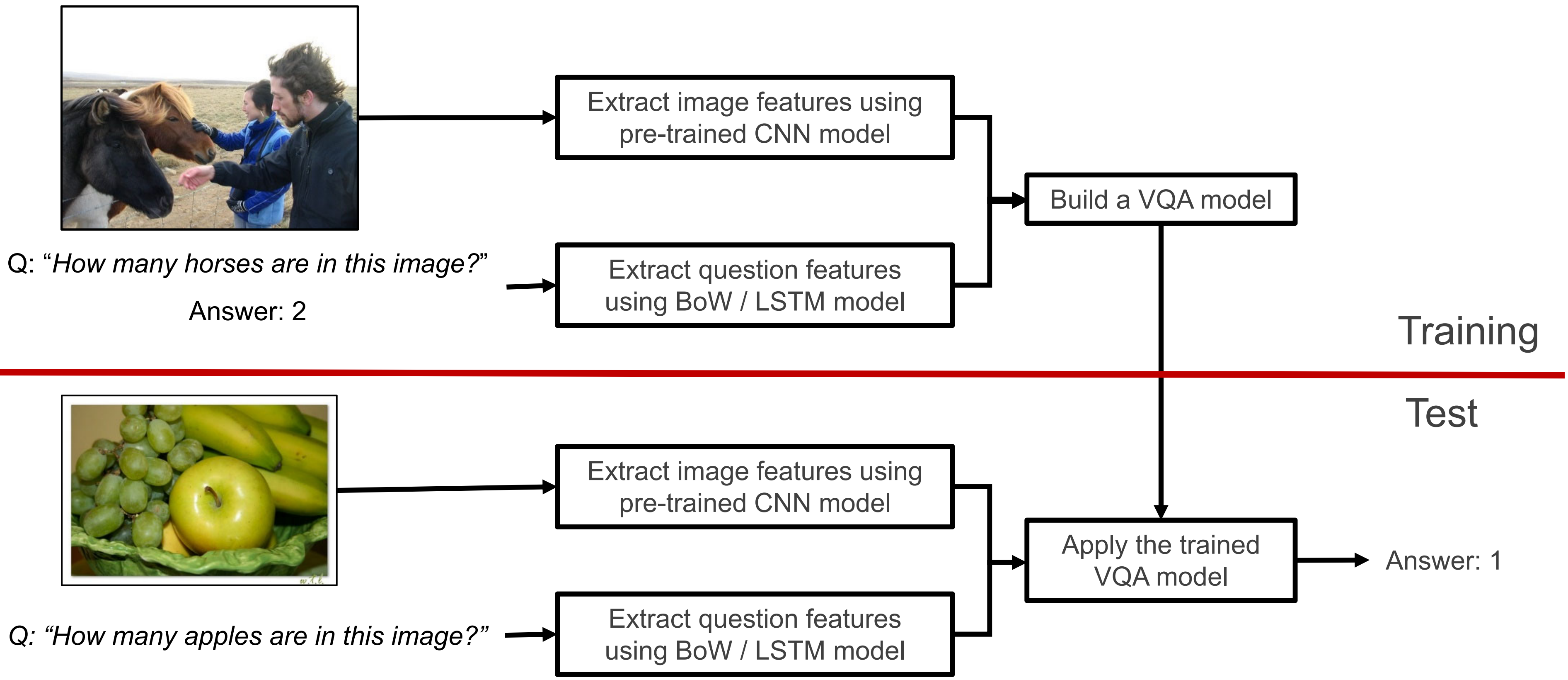
Tutorial: A gentle introduction to Visual Question Answering (VQA) using neural networks, https://victorzhou.com/blog/easy-vqa

Reference: Visual Question Answering: Datasets, Algorithms, and Future Challenges, https://arxiv.org/abs/1610.01465

Use case: Visual question answering (VQA)

The full VQA pipeline example

- Pre-process both image and question/answer text
- Design a model architecture and train the model
- Deploy the model and process the new test image and question input



Workshop: Speech cognitive systems

Objective

- Build a HMM-based speech recognition for single word (command and control)

Reference

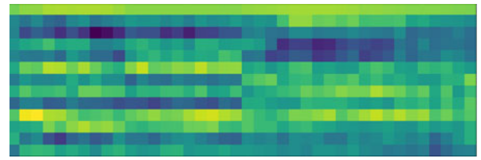
- Prateek Joshi, Python Machine Learning Cookbook, Packt Publishing, 2016, Code available at https://github.com/PacktPublishing/Python-Machine-Learning-Cookbook

Workshop: Speech cognitive systems

apple
banana
kiwi
lime
orange
peach
pineapple

Training data:
7 classes (isolated word), 15 samples (audio clip) each class

apple01.wav
apple02.wav
apple03.wav
apple04.wav
apple05.wav
apple06.wav
apple07.wav
apple08.wav
apple09.wav
apple10.wav
apple11.wav
apple12.wav
apple13.wav
apple14.wav
apple15.wav



Extract speech features

Build a HMM model for each class

A set of trained HMM models
(totally, 7 models for 7 classes)

Training

Test

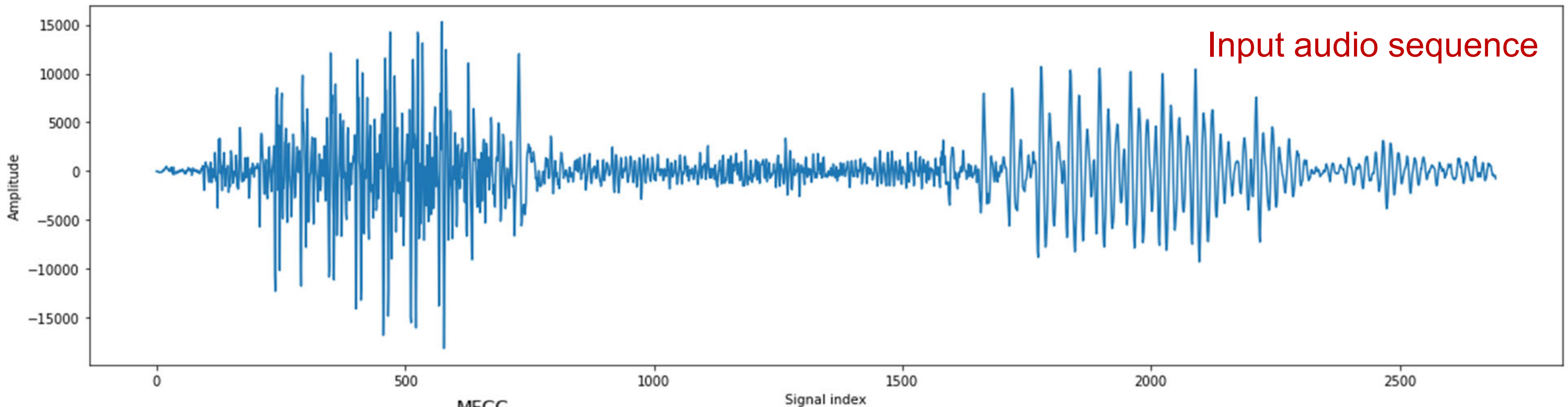
Test data

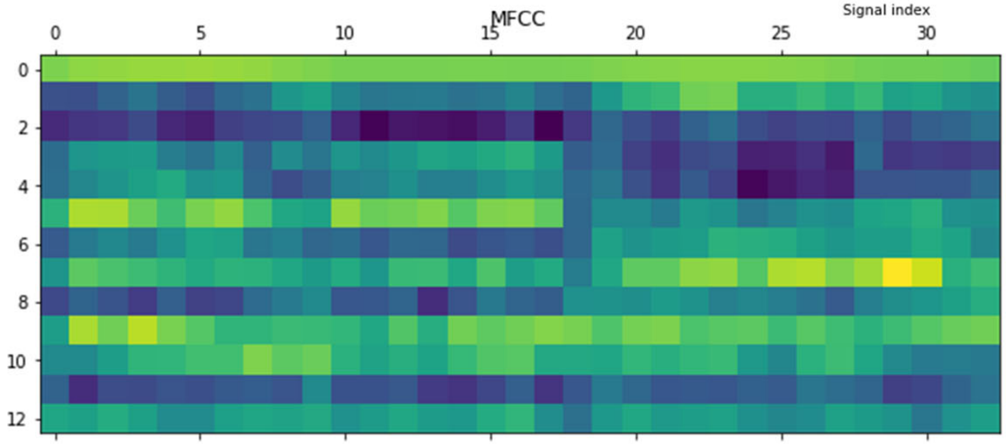
banana15.wav

Evaluate with seven models and choose the model with the largest confidence score output

Recognition result: *Banana*

Workshop: Speech cognitive systems





Variables	Dimensions
Input audio (apple01.wav)	(2694,)
MFCC feature of input signal (# windows, # features)	(33, 13)
HMM transition probability matrix with 3 components (user-defined)	(3, 3)
HMM state sequence (one state per window of input signal)	(33,)

```
[[9.32529578e-01, 4.36394206e-26, 6.74704225e-02]
 [1.53171049e-39, 9.05902521e-01, 9.40974787e-02]
 [1.24927288e-01, 1.02681348e-01, 7.72391365e-01]]
```

HMM transition probability matrix

```
[0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 2 2 1 1 1 1 1 1 1 1 1 1 1 2 2]
```

HMM state sequence

What we have learnt

- A typical vision cognitive system pipeline
- A statistical speech cognitive system framework
- Isolated word speech recognition using Hidden Markov model (HMM)

Thank you!

Dr TIAN Jing
Email: tianjing@nus.edu.sg