Toward the Realization of Automatic Spoken Language Acquisition Mechanism for Human-Symbiotic Robots

APSIPA 2021 Tutorial

Takahiro Shinozaki (Tokyo Institute of Technology)
Takuma Okamoto (NICT)
Shinsuke Mori (Kyoto University)

Latest version of this material and spoken language acquisition toolkit will be available at:
https://github.com/tttslab/spolacq

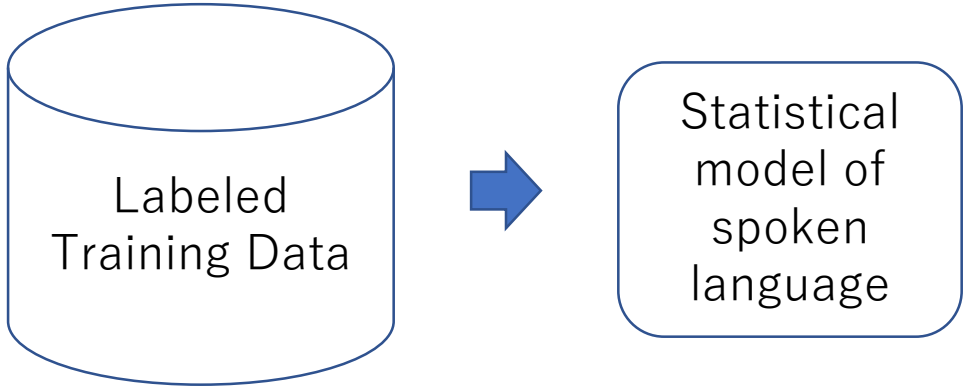
Ver. 2021 12/14

Table of Contents

PART 1

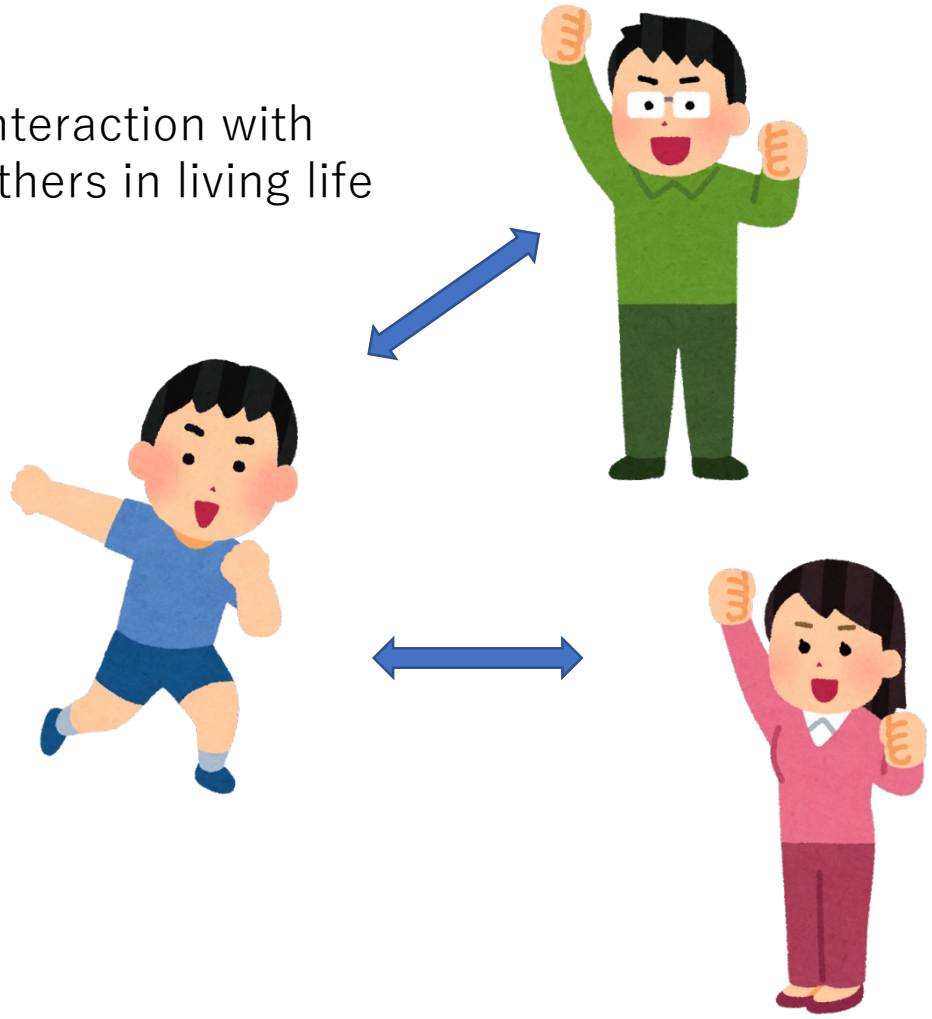
Motivation

Human and Machine Language Learning



Data set development + supervised learning

Interaction with
others in living life



Advantage of Supervised Learning

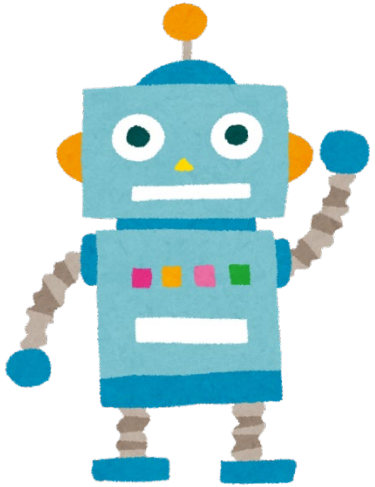


Image is by irasutoya

Train

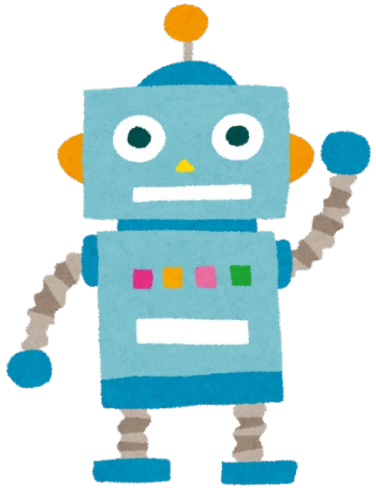
Engineer/Labeler

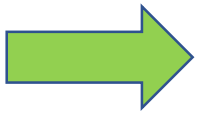
Interpret

The training is efficient given the label

Limitation of Supervised Learning

- Bounded by engineer/labeler's (teacher's) interpretation of the world
- Always needs engineer's intervention to work on new domain





Engineer/Labeler

Cannot be autonomous

Human Language Learning

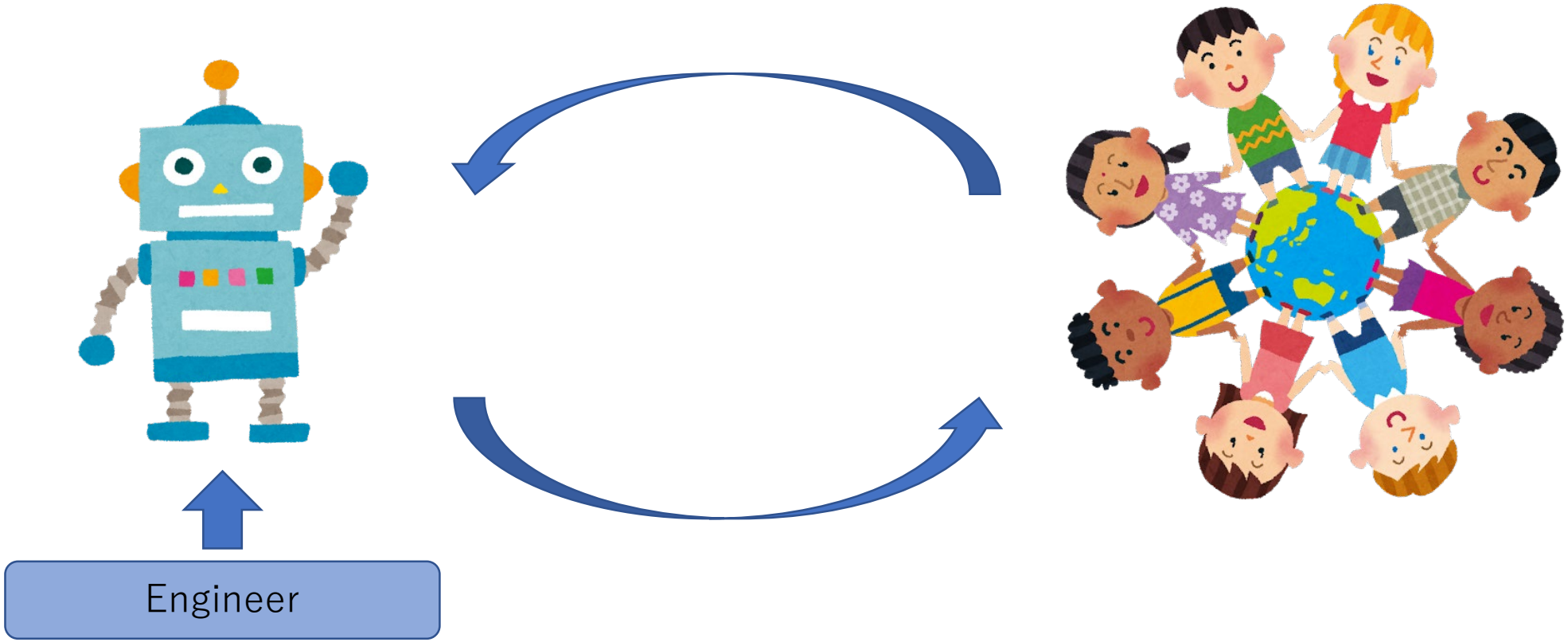
- Not bounded by teacher
- Flexible, adaptable, and creative

Autonomous

Direct interaction with the world

Automatic Spoken Language Acquisition

- Equips self-supervised learning ability to agents
- Realizes closed learning loop of spoken language learning

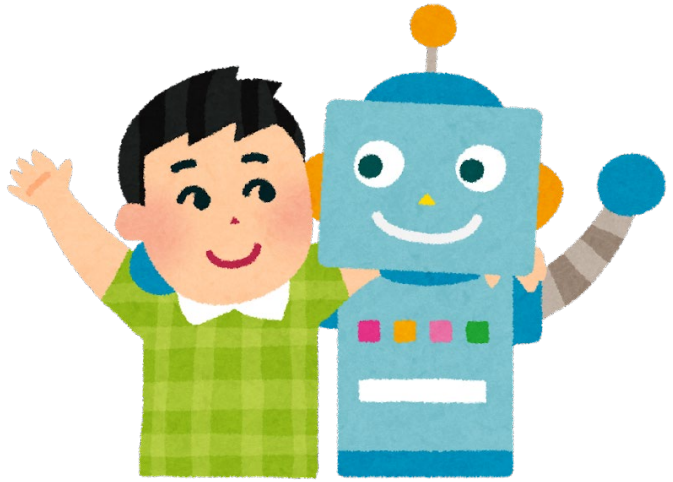


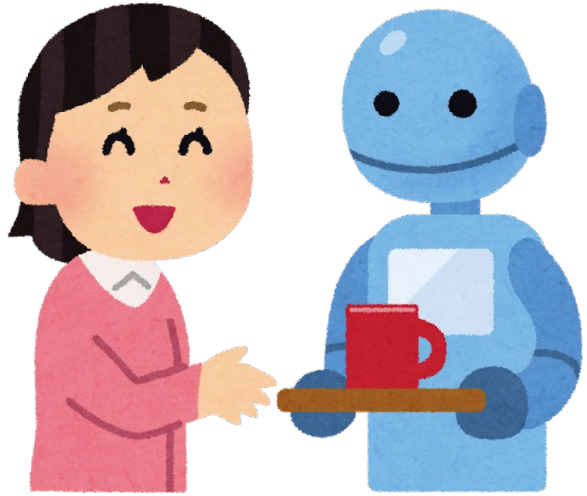
Application to Human-Symbiotic Robots

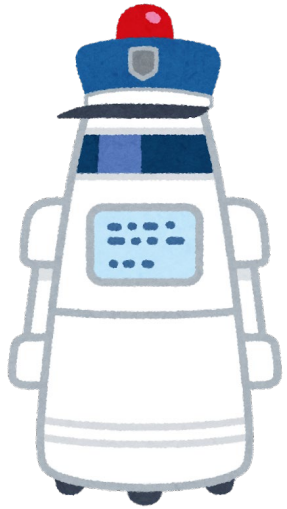
In the near future, robots will play an active role in various situations in human society

Such robots are required to have flexible and adaptive language ability based on meaning understanding

Automatic spoken language acquisition technology is indispensable for realizing such robots





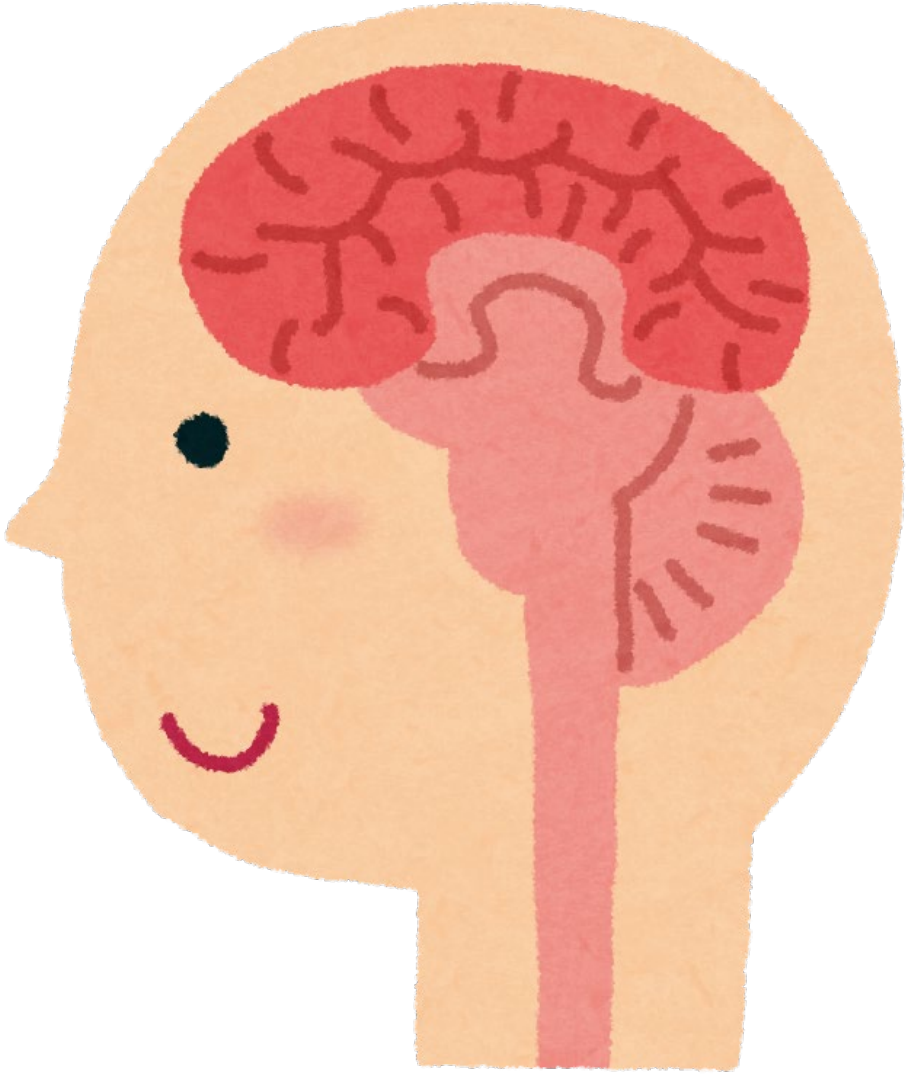


Application to Cognitive Science

How humans learn and understand language is still a mistily

By realizing spoken language acquisition agent, we get a mathematical model of human language acquisition that contributes to cognitive science

Educational and Medical applications would also be possible



E. Dupoux, "Cognitive science in the era of artificial intelligence: A roadmap for reverse-engineering the infant language-learner," Cognition, 2018.

Weaker than Supervised Learning?

Currently, YES, automatic spoken language acquisition is weaker than supervised learning.
However. . .

In some applications, AI has already become stronger than human by breaking away from supervised learning

Figure is cited from [1]

Dilaudid, CC BY-SA 3.0, via Wikimedia Commons

Figure is cited from [3]

[1] V. Mnih+, "Playing Atari with Deep Reinforcement Learning," NIPS Deep Learning Workshop, 2013

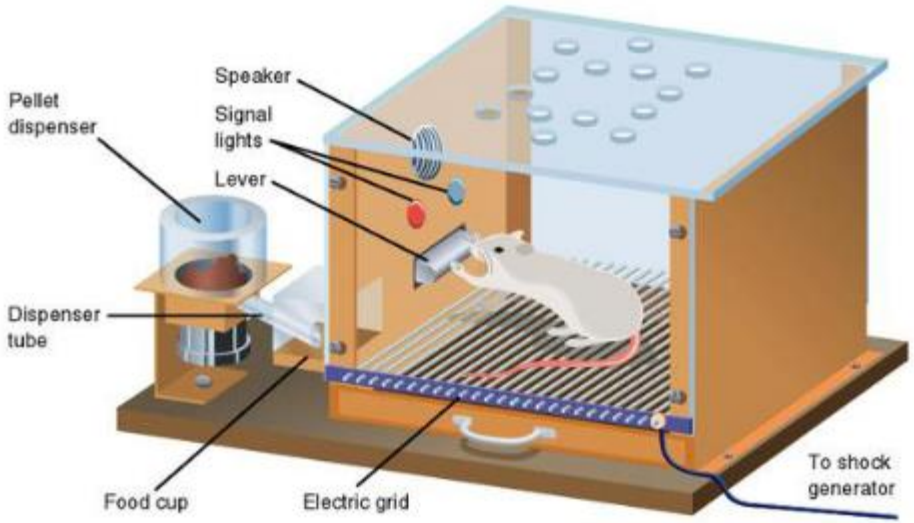
[2] D. Silver+, "Mastering the game of Go without human knowledge," Nature 2017

[3] N. Bhonker+, "Playing SNES in the Retro Learning Environment," arXiv, 2018

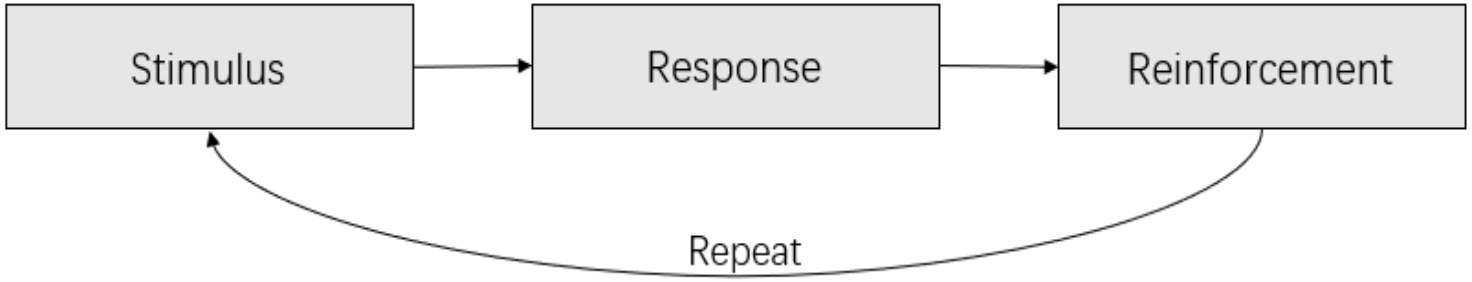
Cognitive Science Findings

Formulation as Operant Conditioning

Skinner[1] gave a widely-accepted explanation: children learn the language based on behaviorist reinforcement principles by associating words with meanings



Skinner box [2]

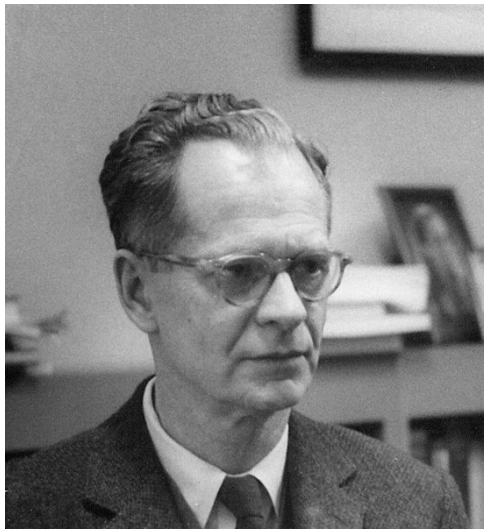


Skinner's language acquisition theory

[1] B. F. Skinner. "Verbal behavior," New York: Appleton-Century-Crofts, 1957.
[2] S. McLeod, "Skinner - Operant Conditioning," 2007 Retrieved from http://www.simplypsychology.org/operant-conditioning.html
[3] C. Sturdy+ How Much of Language Acquisition Does Operant Conditioning Explain?. Front Psychol, 2017.

Chomsky Rebutted Skinner's Argument

language acquisition could be explained by mechanisms of operant conditioning (OC)



Silly rabbit, CC BY 3.0, via Wikimedia Commons

B.F. Skinner

it was unlikely that parents were doing the slow and careful shaping of children's vocalizations and that there are grammatical regularities that cannot be discerned from the surface features of language alone

Augusto Starita / Ministerio de Cultura de la Nación, CC BY-SA, via Wikimedia Commons

N. Chomsky

[1] B. F. Skinner. "Verbal behavior," New York: Appleton-Century-Crofts, 1957.
[2] N. Chomsky, "A review of Skinner's verbal behavior," *Language*, 1959.
[3] C. Sturdy+ How Much of Language Acquisition Does Operant Conditioning Explain?. Front Psychol, 2017.

Infant's Language Development

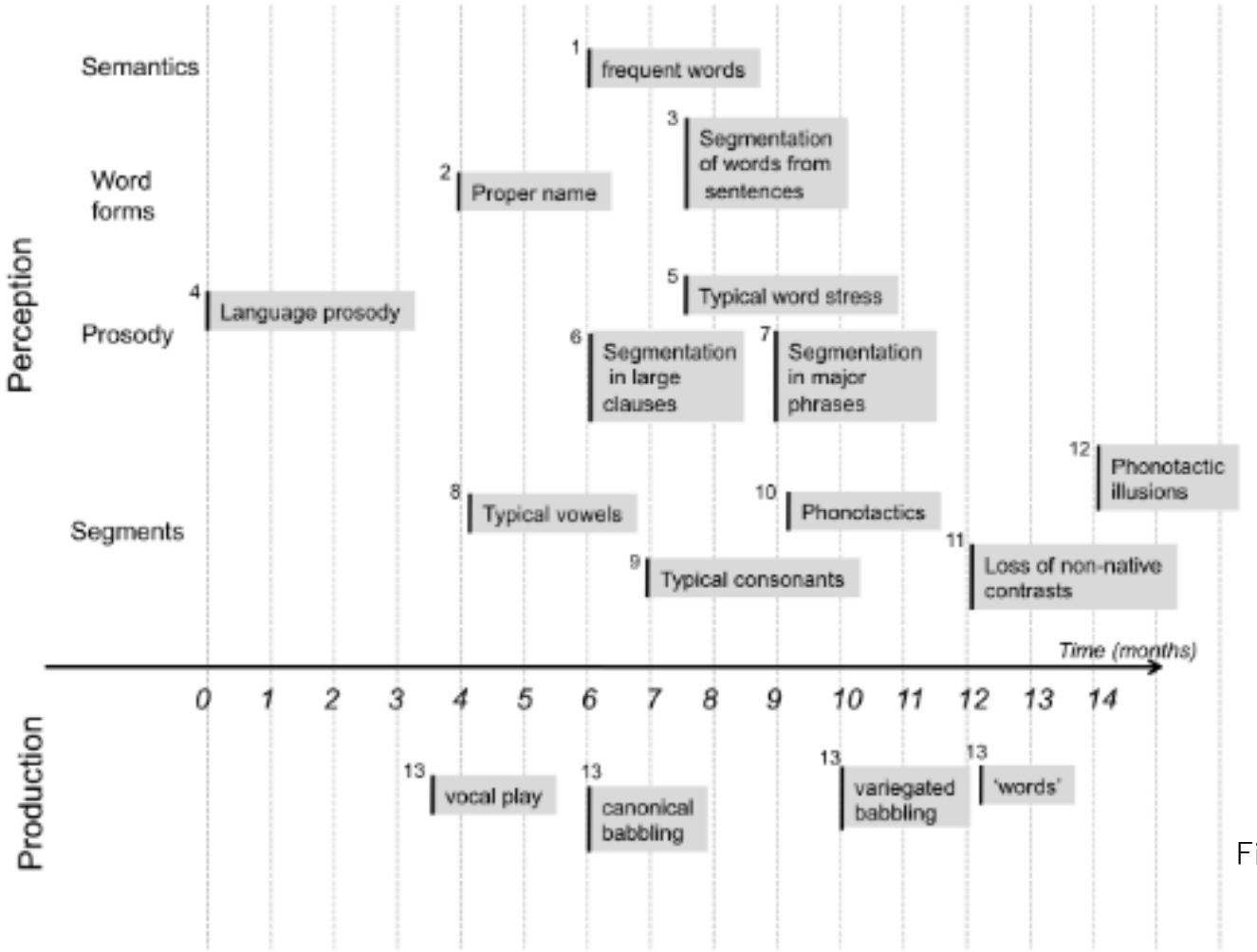


Figure is cited form [1]

[1] E. Dupoux, "Cognitive science in the era of artificial in-telligence: A roadmap for reverse-engineering the infant language-learner," Cognition, 2018.

How Do Infants Find Words?

Unlike written language, spoken language has no reliable markers to indicate word boundaries. Acoustic analysis of speech shows that there are no obvious breaks between syllables or words in the phrase: 'There are no silences between words' (a).

Word segmentation in printed text would be equally difficult if the spaces between words were removed. The continuous string of letters below could be broken up in two different ways, as shown in b.

Figure is cited form [1]

[1] P. Kuhl, "Early language acquisition: Cracking the speech code," Nature reviews. Neuroscience, 2004.

Consideration on Engineering Formulation

Why We Speak?

We have internal desires and need to work with the outside world to satisfy them

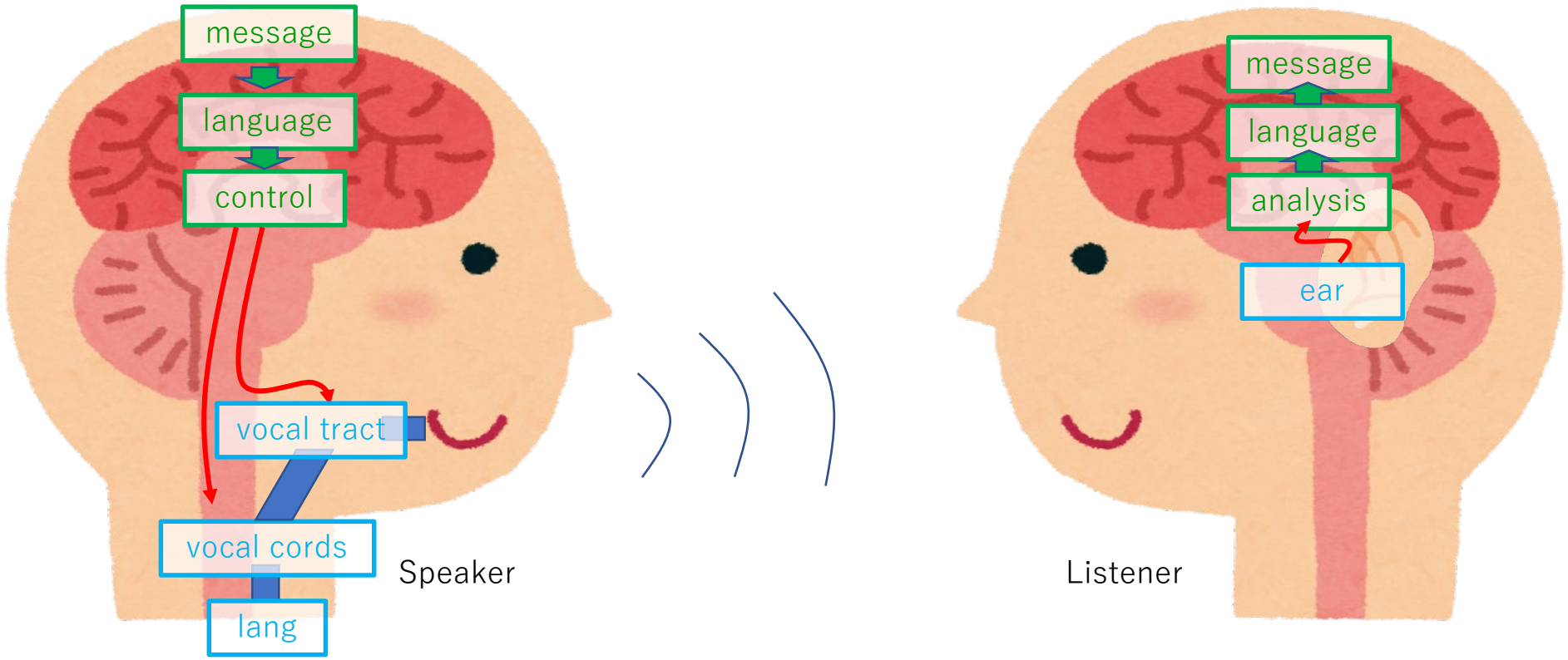
Speaking is an action like placing a stone in Go

Why We Listen?

We have internal states that are observable by listening

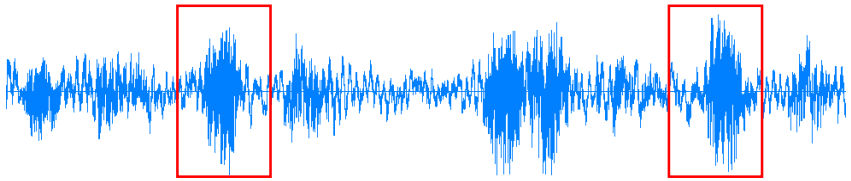
Conversation

Conversation is a chain of speaking and listening
built on the mechanism of sensorimotor and information processing

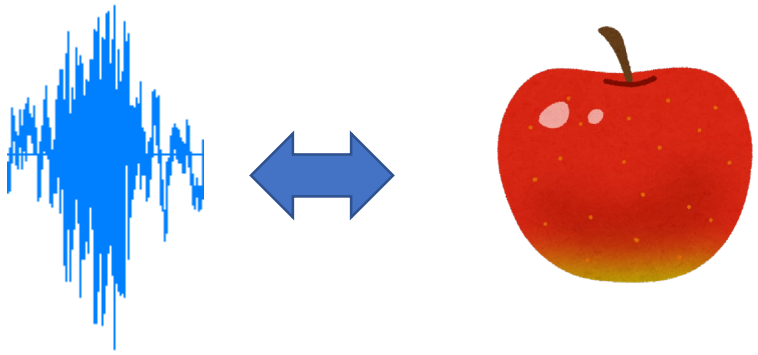


Functional Elements of Spoken Word Learning

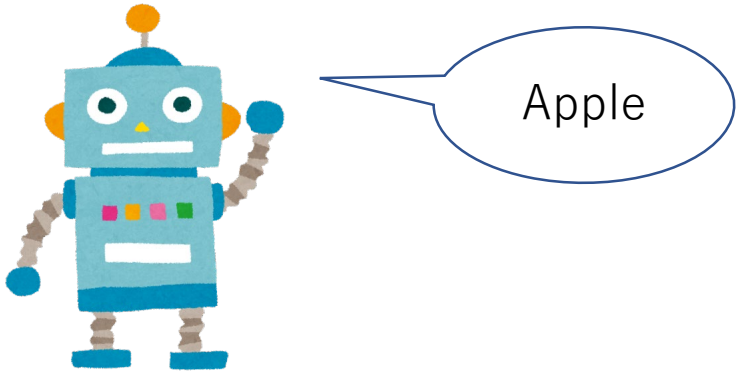
① Word discovery:
Discover repeating patterns (words) in audio signals



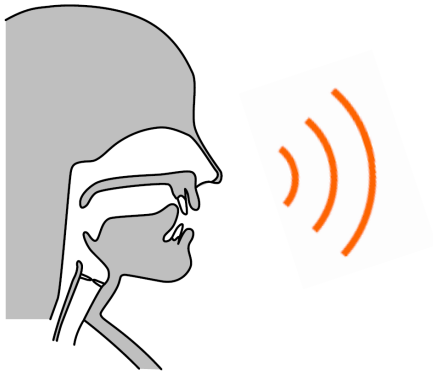
② Grounding:
Grounding the words to the world



③ Action learning:
Modeling the effects of word utterance



④ Pronunciation learning
Learns to generate waveform utterance



Language Ability of Dogs

Dogs can recognize words (Have the ability of ① and ②, but do not have ③ and ④)

① Word discovery
② Grounding
③ Action learning
④ Pronunciation learning

J. Pilley+, "Border collie comprehends object names as verbal referents," Behavioural Processes, 2011

Language Ability of Birds

Some birds can imitate human voice (Have the ability of ④, but do not have ①, ②, and ③)

① Word discovery
② Grounding
③ Action learning
④ Pronunciation learning

C. Cate+, "Vocal imitations and production learning by Australian musk ducks (Biziura lobata)," Phil. Trans. R. Soc. B, 2020

Language Ability of Humans

We have all ability of ① to ④, and thus can have speech conversation

Our conversation ability makes us unique and intelligent among animals

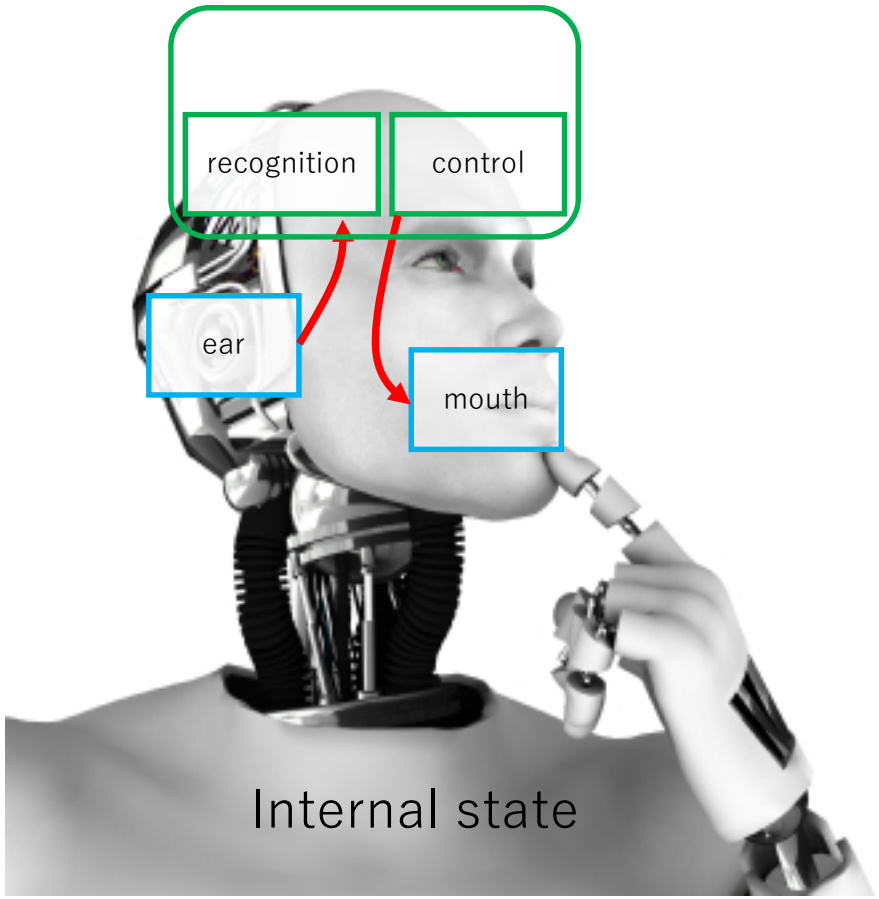
① Word discovery

② Grounding

③ Action learning

④ Pronunciation learning

Specification of the Agent

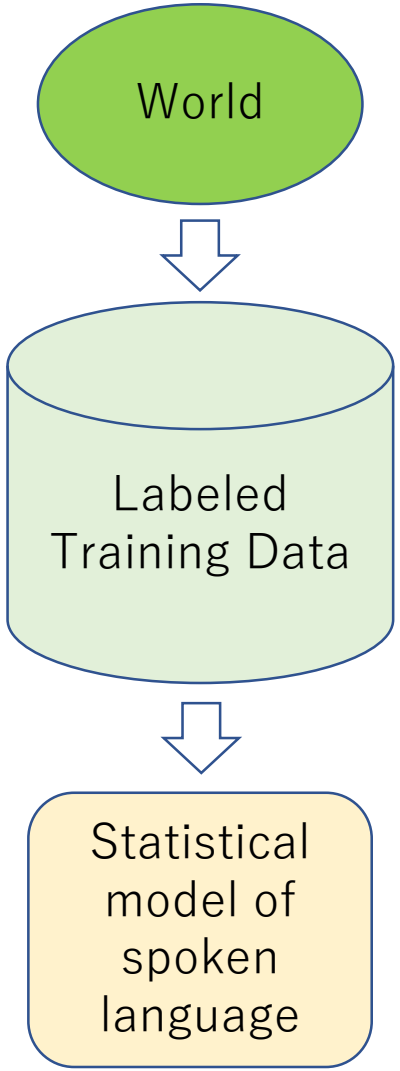


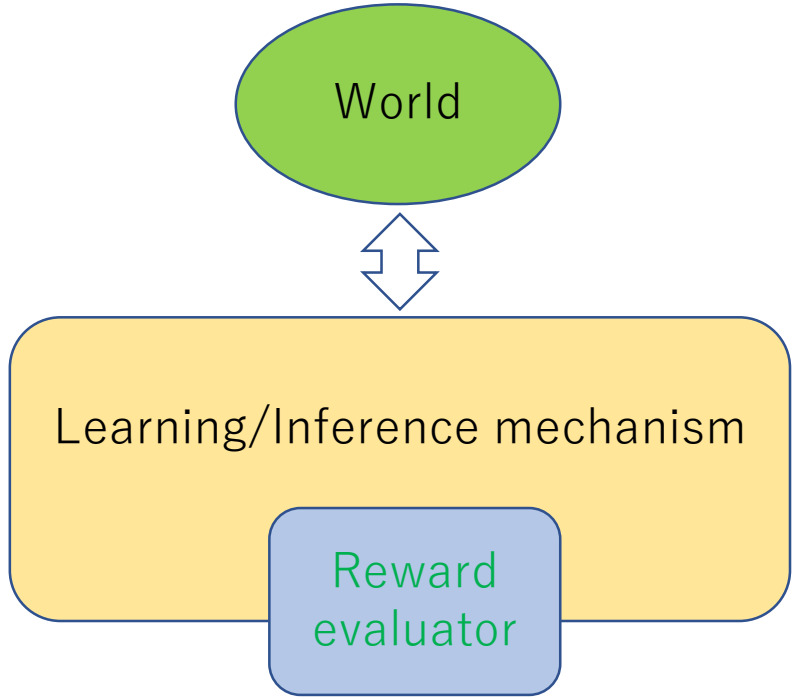
- To be autonomous, agents should equip with a reward evaluator (a sort of desire) inside
- It should support the four spoken word learning ability (①②③④)
- Agents should be able to act on external (i.e. hearing, sight, etc.) and internal (i.e. knowledge, preference, battery level, etc.) sensations
- As the problem setting, the outside world can get information about the agent's internal state only through the agent's verbal expression

Paradigm Shift in System Development

Gather language/task dependent labeled data

Train model. Devise the model structure and learning conditions





1. Develop language/task independent learning mechanism
2. Design reward function
3. Let the agent learn and work in real world

Engineer's role is to develop self-learning mechanism

Survey and Classification

Gorin+'s Call Routing System

Builds associations between words which occur in the input message and machine actions.
Initially, there is no vocabulary, no grammar, and no semantic associations

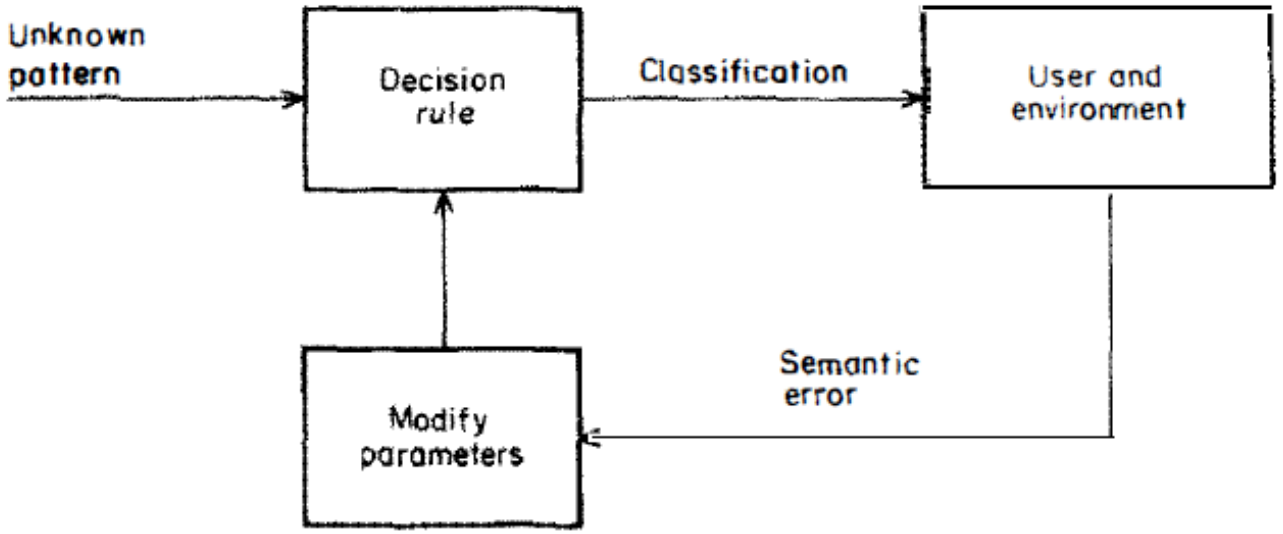


Figure 1. Adaptation and learning. (a) Traditional supervision. (b) Semantic-level error feedback.

① Word discovery *1
② Grounding
③ Action learning *2
④ Pronunciation learning

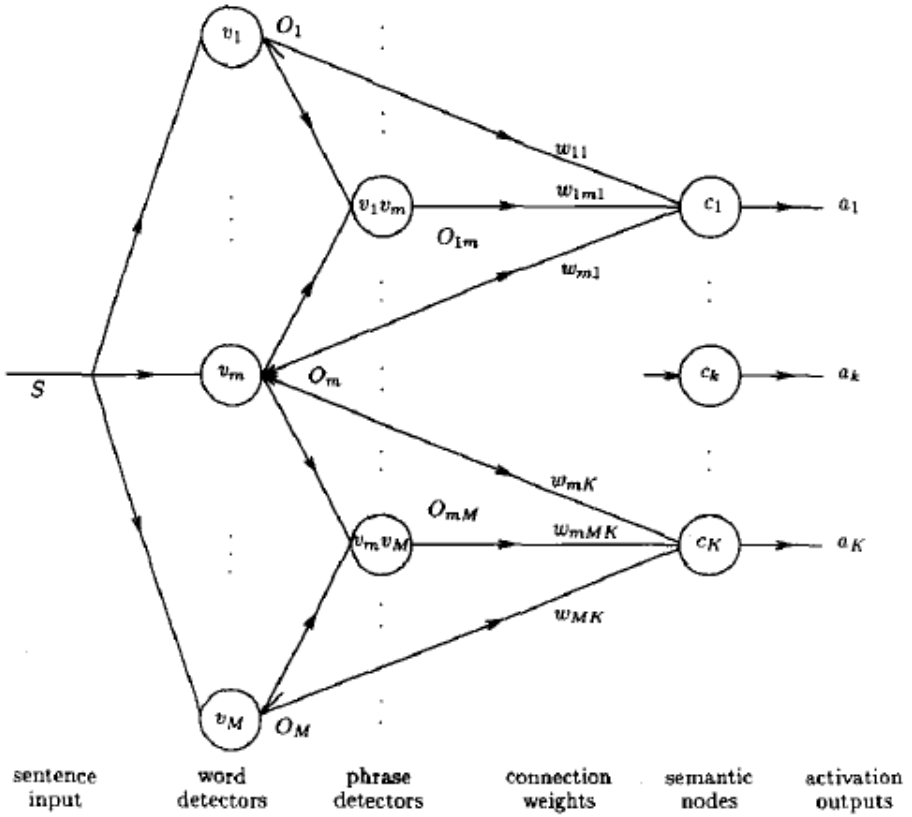
*1 Input is text

*2 Action space is pre-defined

A. Gorin+ "Adaptive acquisition of language," Computer Speech & Language, 1991

Extension to Speech Input

Later, Gorin+ extended their system to work on speech input of isolated word sequences by adding a DTW-based vocabulary learning module



① Word discovery

② Grounding

③ Action learning *1

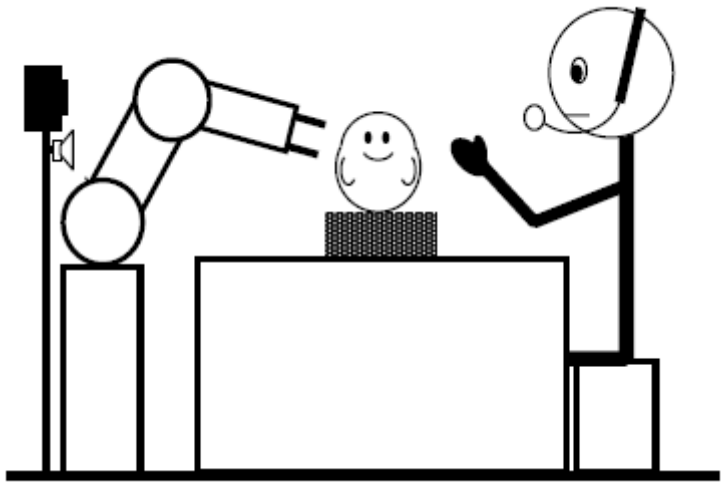
④ Pronunciation learning

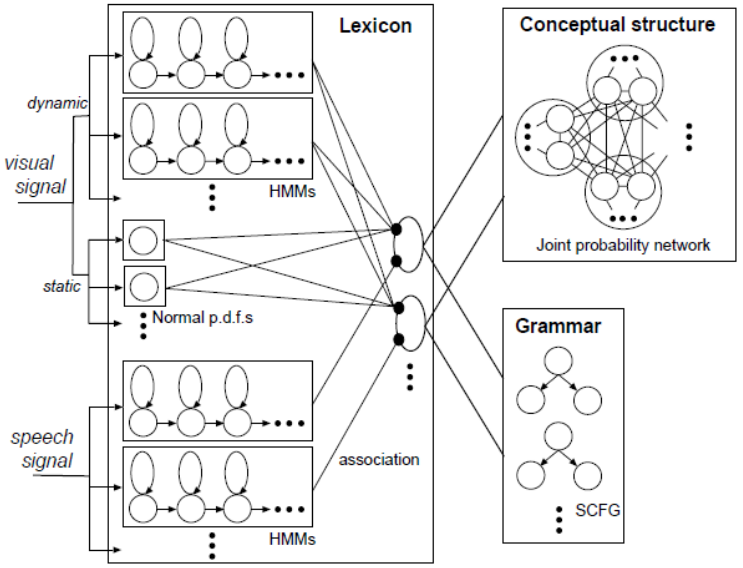
*1 Action space is pre-defined

A. Gorin+ "An Experiment in Spoken Language Acquisition," IEEE Trans. Speech and Audio Processing, 1994

Iwahashi's Human-Robot Interface System

- Supports sound and image inputs
- Works on isolated word sequences
- Action generation is based on imitation and does not support spontaneous trial and error



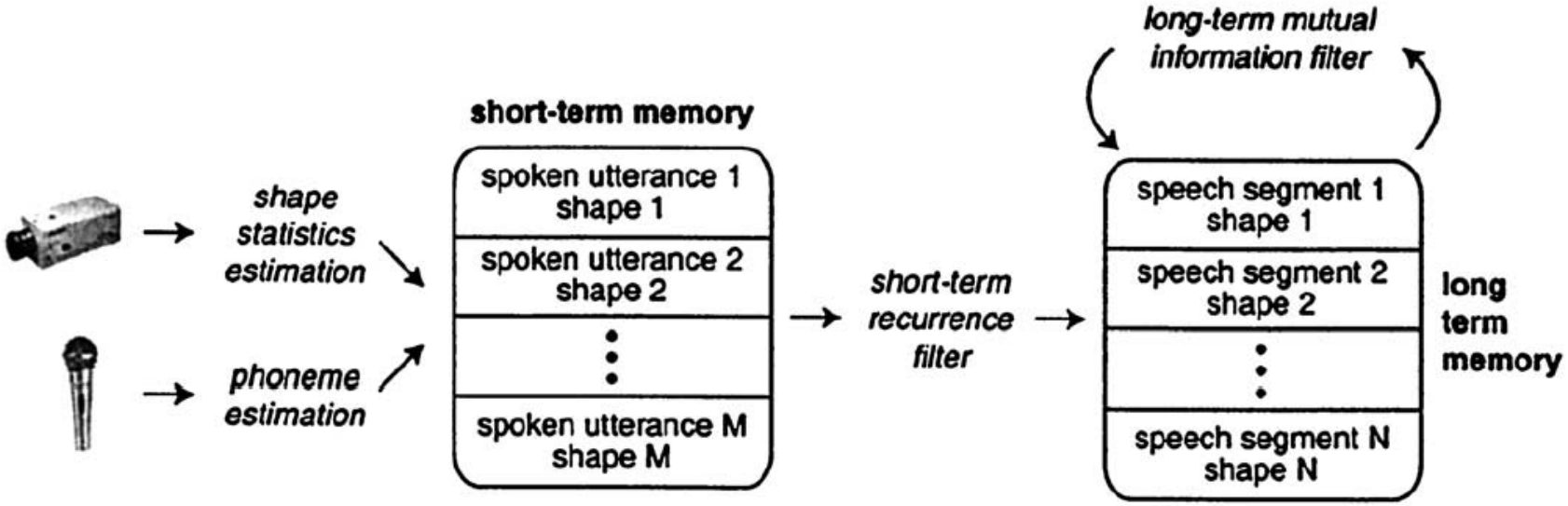


① Word discovery
② Grounding
③ Action learning
④ Pronunciation learning

N. Iwahashi, "Language acquisition through a human-robot interface," ICSLP, 2000

Roy+'s Vocabulary Learning System

- Learns vocabulary from a continuously spoken utterance
- Learns associations between wordsand image objects
- Needs a pre-trained phonetic speech recognizer
- No action generation



① Word discovery

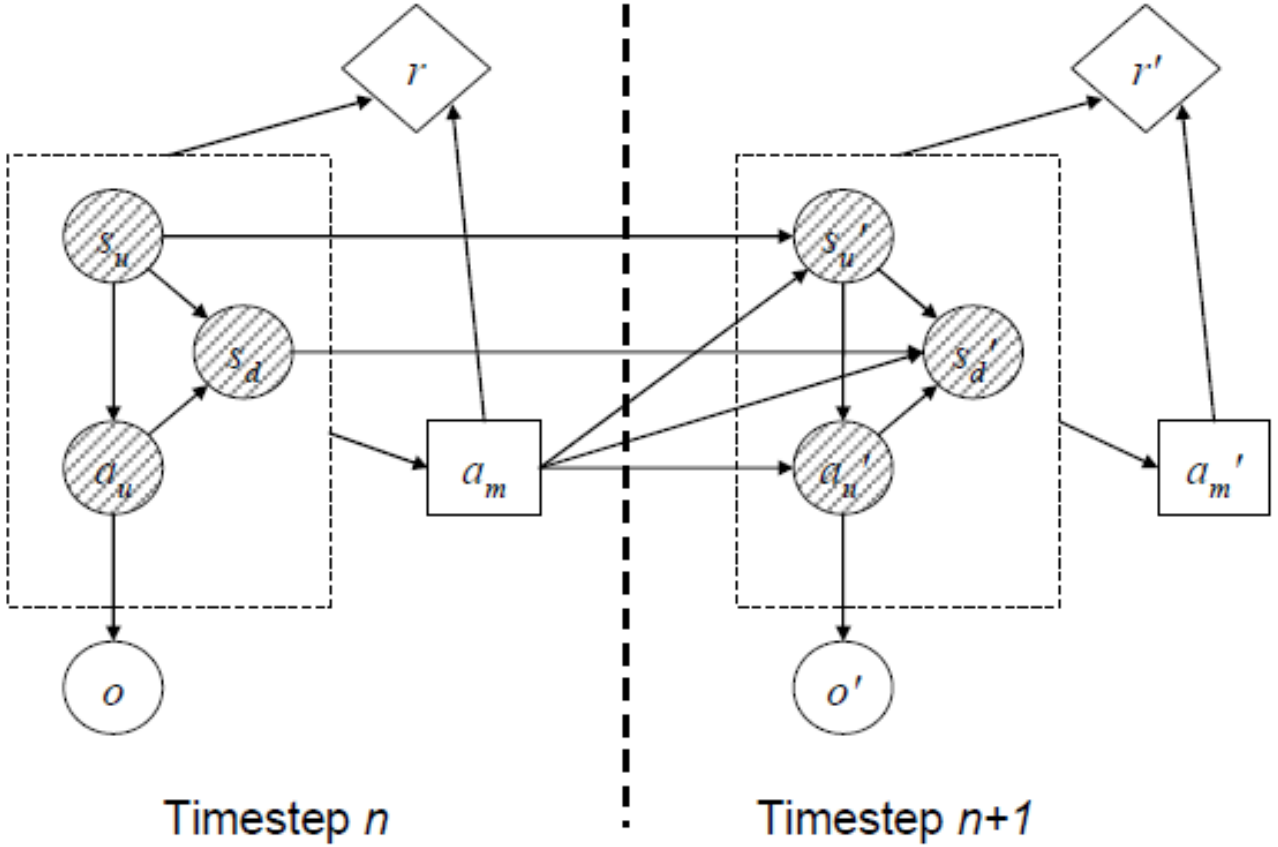
② Grounding

③ Action learning

④ Pronunciation learning

D. Roy+, "Learning Words from Sights and Sounds : A Computational Model," Cognitive Science, 2002

Williams+'s POMDP based Dialogue System



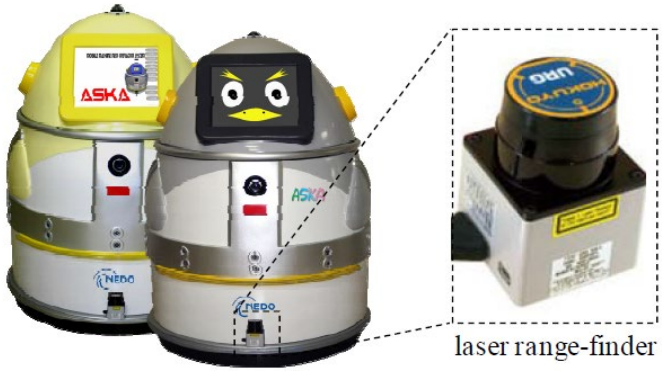
① Word discovery
② Grounding
③ Action learning*1
④ Pronunciation learning

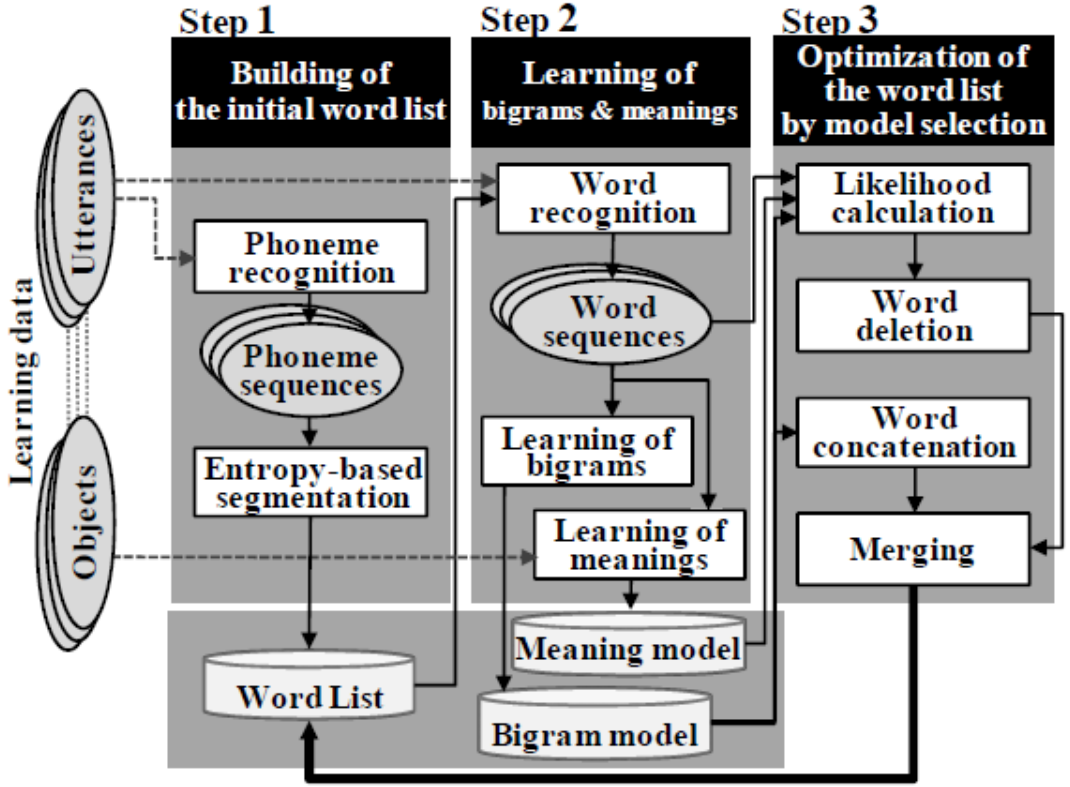
*1 Action space is pre-defined

J. Williams+, "Factored Partially Observable Markov Decision Processes for Dialogue Management," Proc. Knowledge and Reasoning in Practical Dialog Systems, 2005

Taguchi's Place Name Learning Robot

- Learns the place names from utterance and location coordinates
- Needs a pre-trained phoneme recognizer
- Needs a pre-defined grammar that models the dialogue story





① Word discovery

② Grounding

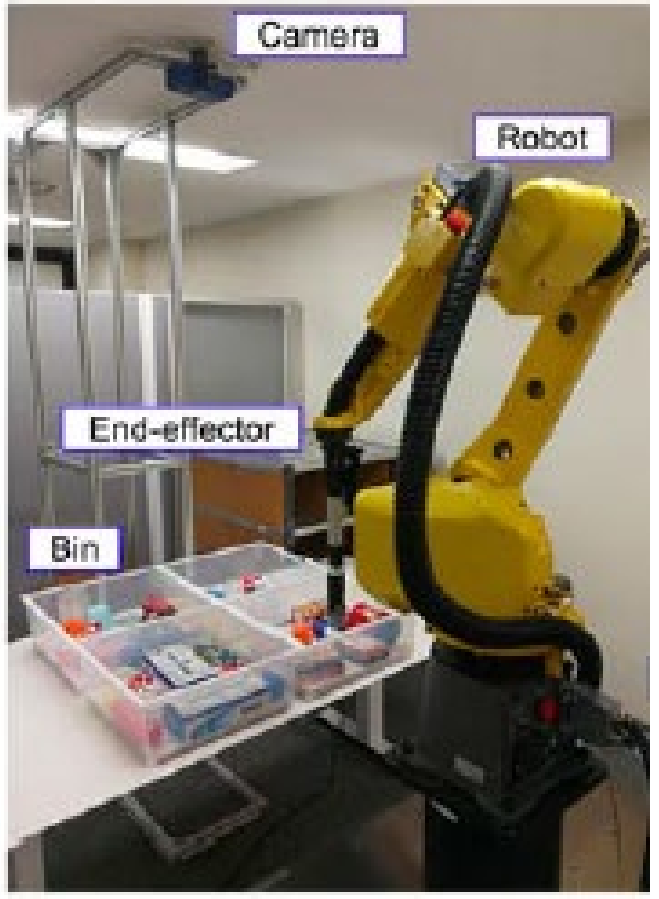
③ Action learning

④ Pronunciation learning

Taguchi+, "Learning Place-Names from Spoken Utterances and Localization Results by Mobile Robot," Interspeech, 2011

Hatori’s Robot Arm System

- Learns to move a robot arm following spoken instructions by reinforcement learning
- Needs a pre-trained speech recognition system to transcribe speech utterances

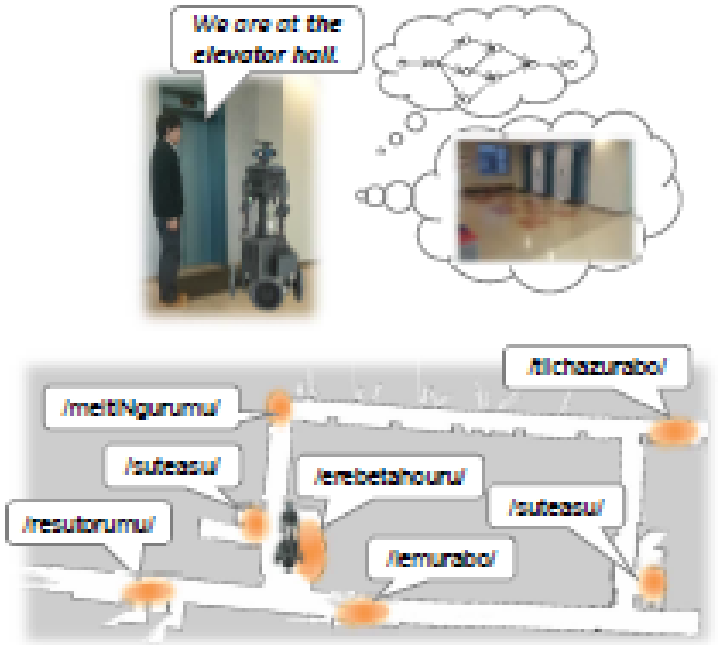


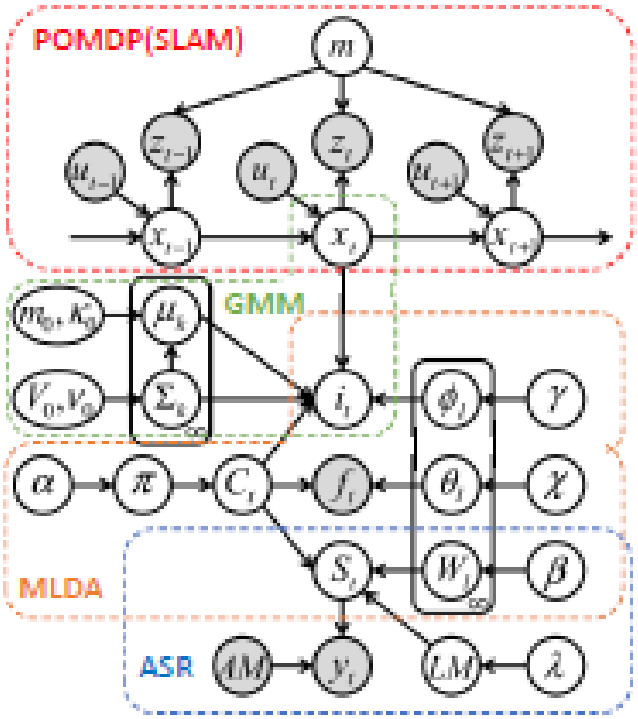
① Word discovery
② Grounding
③ Action learning
④ Pronunciation learning

Hatori+, "Interactively picking real-world objects with unconstrained spoken language instructions," IEEE International Conference on Robotics and Automation, 2018

Taniguchi+'s Neuro-SERKET

- Uses hierarchical Bayes model to have an integrated representation of language knowledge
- Needs a pre-trained phoneme recognizer





① Word discovery

② Grounding

③ Action learning (?)

④ Pronunciation learning

Taniguchi+, "Neuro-SERKET: Development of Integrative Cognitive System through the Composition of Deep Probabilistic Generative Models," New Generation Computing, 2020

PART 2

Reinforcement Learning

Markov Process

- A Markov Process is a tuple $\langle S, T \rangle$
  - S: set of states
  - T: state transition matrix $T_{ss'} = P(S_{t+1} = s' | S_t = s)$
  - States satisfy Markov Property :

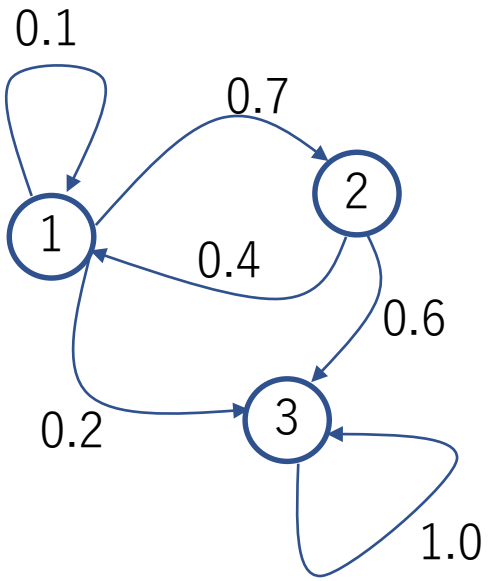
$$P(S_{t+1}|S_1, S_2, \cdots, S_t) = P(S_{t+1}|S_t)$$

*$T_{ss'}$ is ss' element of matrix T

Example:

$$S = \{1,2,3\}$$

$$T = \begin{bmatrix} 0.1 & 0.7 & 0.2 \\ 0.4 & 0 & 0.6 \\ 0 & 0 & 1.0 \end{bmatrix}$$



Probability of State Sequences

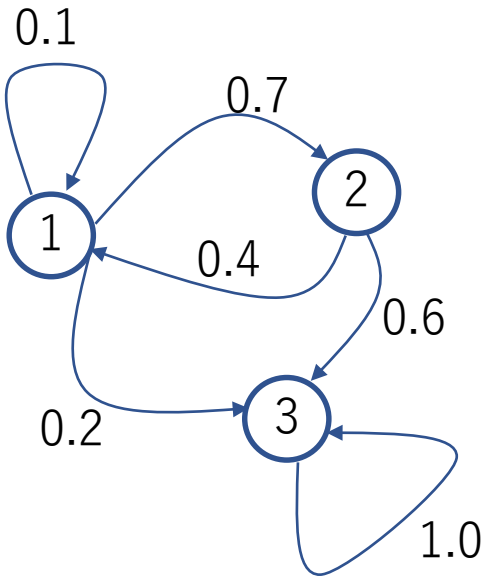
$$P(S_1, S_2, \cdots, S_t) = P(S_1)P(S_2|S_1)P(S_3|S_1, S_2)\cdots P(S_t|S_1, S_2, \cdots, S_{t-1}) = P(S_1)P(S_2|S_1)\cdots P(S_t|S_{t-1})$$

Let $\mathbf{u}(t) = \begin{bmatrix} P(S_t = 1) \\ P(S_t = 2) \\ \vdots \\ P(S_t = N) \end{bmatrix}$, then $\mathbf{u}(t+1) = T^T\mathbf{u}(t)$ because $P(S_t) = \sum_{S_{t-1}} P(S_t|S_{t-1})P(S_{t-1})$

Example:

$S = \{1,2,3\}$

$$T = \begin{bmatrix} 0.1 & 0.7 & 0.2 \\ 0.4 & 0 & 0.6 \\ 0 & 0 & 1.0 \end{bmatrix}$$



Assume $P(S_1 = 1) = 1.0$. Then

$P(S_1 = 1, S_2 = 2, S_3 = 3)$
$= P(S_1 = 1)P(S_2 = 2|S_1 = 1)P(S_3 = 3|S_2 = 2)$
$= 1.0 * 0.7 * 0.6 = 0.42$

$$\mathbf{u}(2) = \begin{bmatrix} 0.1 & 0.7 & 0.2 \\ 0.4 & 0 & 0.6 \\ 0 & 0 & 1.0 \end{bmatrix}^T \begin{bmatrix} 1.0 \\ 0.0 \\ 0.0 \end{bmatrix}$$

$$= \begin{bmatrix} 0.1 & 0.4 & 0 \\ 0.7 & 0 & 0 \\ 0.2 & 0.6 & 1.0 \end{bmatrix} \begin{bmatrix} 1.0 \\ 0.0 \\ 0.0 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.7 \\ 0.2 \end{bmatrix}$$

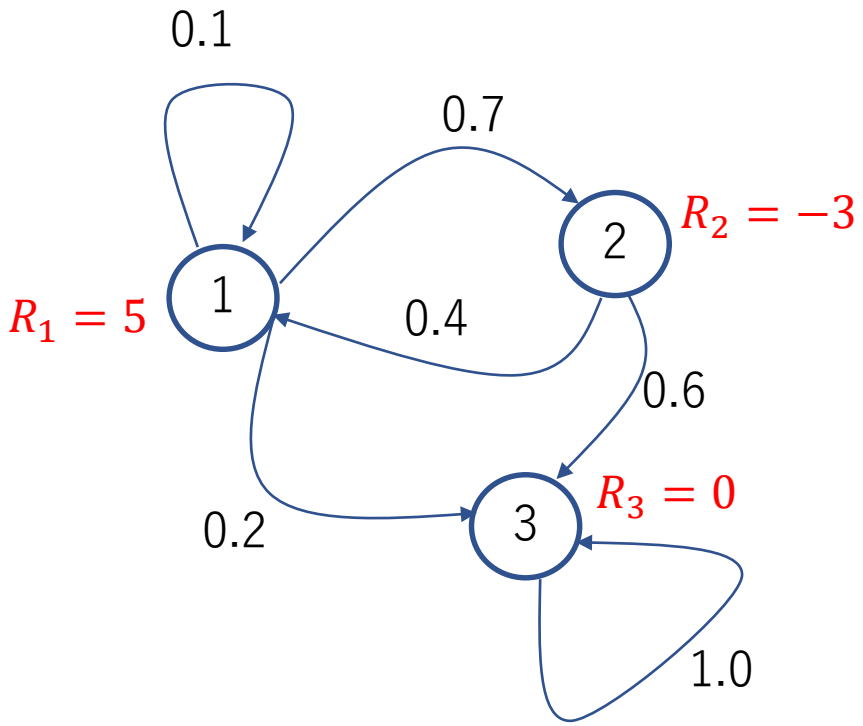
Markov Reward Process

- A Markov Reward Process is a tuple $\langle S, T, R, \gamma \rangle$
  - S: set of states
  - T: state transition matrix $T_{ss'} = P(S_{t+1} = s' | S_t = s)$
  - R: reward function $R_s = R(s)$
  - γ: discount factor $\gamma \in [0, 1]$

Example:

$$S = \{1,2,3\}$$

$$T = \begin{bmatrix} 0.1 & 0.7 & 0.2 \\ 0.4 & 0 & 0.6 \\ 0 & 0 & 1.0 \end{bmatrix}$$



Return and State-Value Function

- Return G_t is a sum of discounted rewards starting from time-step t

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- State-value function $v(s)$ is the expected return starting from state s

$$v(s) = \sum_{s_{t+1}, s_{t+2}, \cdots, s_\infty} P(s_{t+1}, s_{t+2}, \cdots, s_\infty | s_t = s)\, G_t = E[G_t | S_t = s]$$

Markov Decision Process (MDP)

- A Markov Decision Process is a tuple $\langle S, A, T, R, \gamma \rangle$
  - S: set of states
  - A: set of actions (action space)
  - T: state transition matrix $T^a_{ss'} = P(S_{t+1} = s' | S_t = s, A_t = a)$
  - R: reward function $R^a_s = R(s, a)$
  - γ: discount factor $\gamma \in [0, 1]$

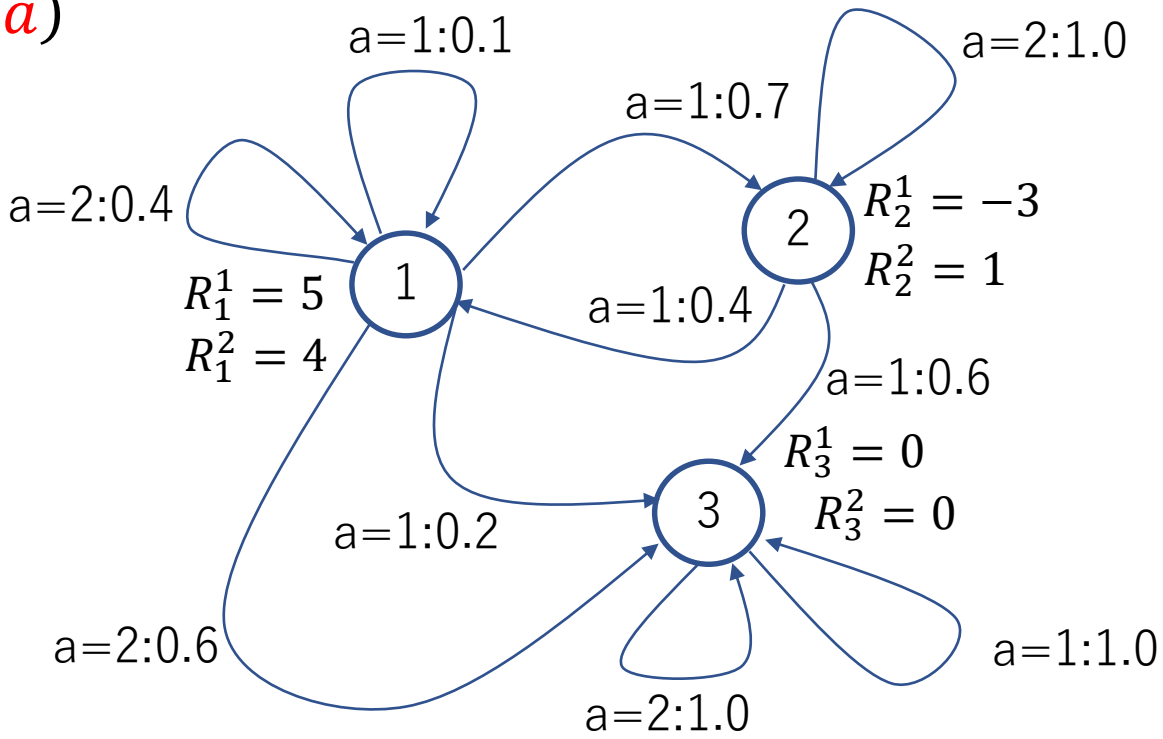
Example:

$$S = \{1,2,3\}$$

$$A = \{1,2\}$$

$$T_{a=1} = \begin{bmatrix} 0.1 & 0.7 & 0.2 \\ 0.4 & 0 & 0.6 \\ 0 & 0 & 1.0 \end{bmatrix}$$

$$T_{a=2} = \begin{bmatrix} 0.4 & 0 & 0.6 \\ 0 & 1.0 & 0 \\ 0 & 0 & 1.0 \end{bmatrix}$$



Model

- For a Markov Decision Process $\langle S, A, T, R, \gamma \rangle$, a model (in the context of reinforcement learning) is a parametrized representation of $\langle S, A, T, R \rangle$

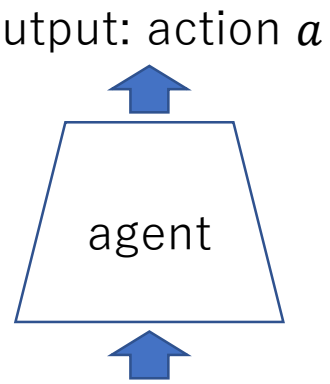
- Model based reinforcement learning
  - Learning algorithms that directly access T and R

- Model free reinforcement learning
  - Learning algorithms that do not directly access T and R
  - Instead, gets samples of actions and rewards by interacting with the environment

Policy

- A policy π is a distribution over actions given states

$$\pi(a|s) = P(A_t = a|S_t = s)$$



- A policy defines the behavior of an agent

- For an MDP $\langle S, A, T, R, \gamma \rangle$ and a policy π,
  let $T^{\pi}_{s,s'} = \sum_{a \in A} \pi(a|s) T^{a}_{ss'}$ and $R^{\pi}_{s} = \sum_{a \in A} \pi(a|s) R^{a}_{s}$
  - $\langle S, T^{\pi} \rangle$ is a Markov process
  - $\langle S, T^{\pi}, R^{\pi}, \gamma \rangle$ is a Markov reward process

Action-Value Function

- Action-value function $q_\pi(s,a)$ is the expected return starting from state s, taking action a, and then following policy π

$$q_\pi(s,a) = \sum_{s_{t+1},s_{t+2},\cdots,s_\infty} P_\pi(s_{t+1}, s_{t+2}, \cdots, s_\infty | s_t = s, A_t = a)\, G_t = E[G_t | S_t = s, A_t = a]$$

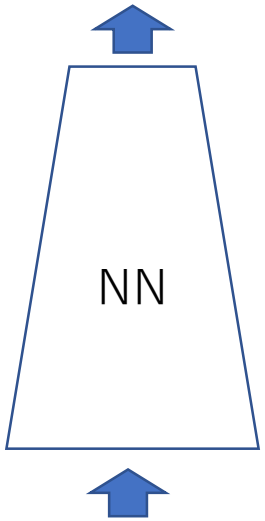
- Relationship with state-value function

$$v_\pi(s) = \sum_{a \in A} \pi(a|s)\, q_\pi(s,a)$$

$$q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a\, v_\pi(s')$$

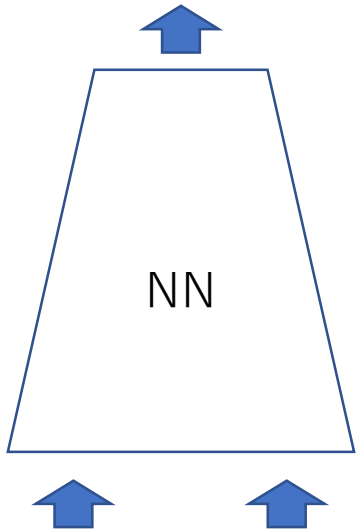
NN Implementation of Value Functions

Output state value $v(s)$



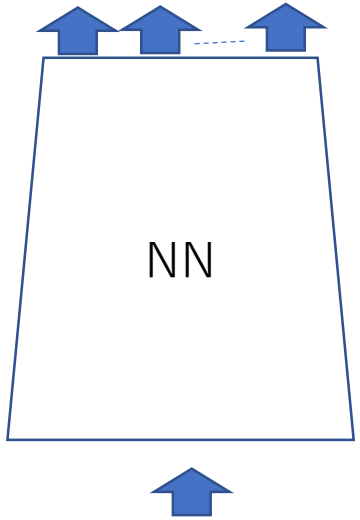
Input state s

Output action value $q_\pi(s,a)$



state s action a

Output $q_\pi(s,a=1), q_\pi(s,a=2), \cdots, q_\pi(s,a=N)$



state s

Bellman Expectation Equation

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \, v_\pi(s') \right)$$

$$q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a'|s') \, q_\pi(s',a')$$

ϵ-Greedy Exploration

- Given an action-value function $Q(s, a)$, defines the policy $\pi(a|s)$ as:

$$\pi(a|s) = \begin{cases} \frac{\epsilon}{m} + 1 - \epsilon & \text{If } a = argmax_{a' \in A} Q(s, a') \\ \frac{\epsilon}{m} & \text{Otherwise} \end{cases}$$

Sarsa

Sarsa is an iterative update method of $Q(s,a)$

Initialize $Q(s,a)$ and s.

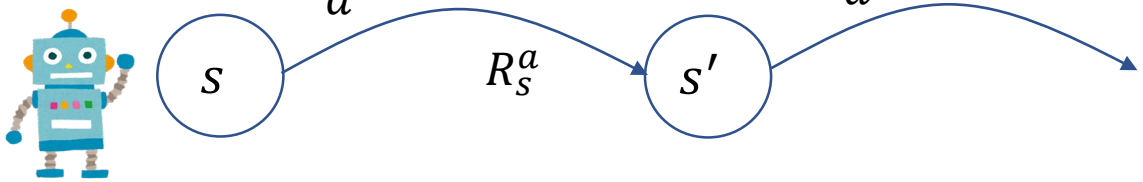
At each episode:

Choose an action a at state s based on a policy derived from $Q(s,a)$ using, e.g., ϵ-greedy

Execute the action a, and observe the reward R_s^a and the next state s'

Choose an action a' at state s' based on a policy derived from $Q(s',a')$ using, e.g., ϵ-greedy

Repeat



- Sarsa is model free
- Sarsa is on-policy

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(R + \gamma Q(s',a') - Q(s,a)\big)$$
$$s \leftarrow s'$$
$$a \leftarrow a'$$

Q-Learning

Q-Learning is an iterative update method of $Q(s, a)$

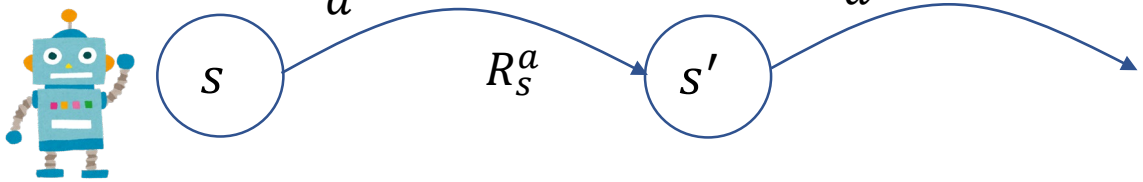
Initialize $Q(s, a)$ and s.

At each episode:

Choose an action a at state s based on a policy derived from $Q(s, a)$ using, e.g., ϵ-greedy

Repeat

Execute the action a, and observe the reward R_s^a and the next state s'



- Q-Learning is model free
- Q-Learning is off-policy

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left(R + \gamma \max_{a'} Q(s', a') - Q(s, a) \right)$$

$$s \leftarrow s'$$

Deep Q-Learning

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

For episode = 1, M **do**

  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

  For t = 1,T **do**

    With probability ε select a random action a_t

    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$

    Execute action a_t in emulator and observe reward r_t and image x_{t+1}

    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D

    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$

    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ

    Every C steps reset $\hat{Q} = Q$

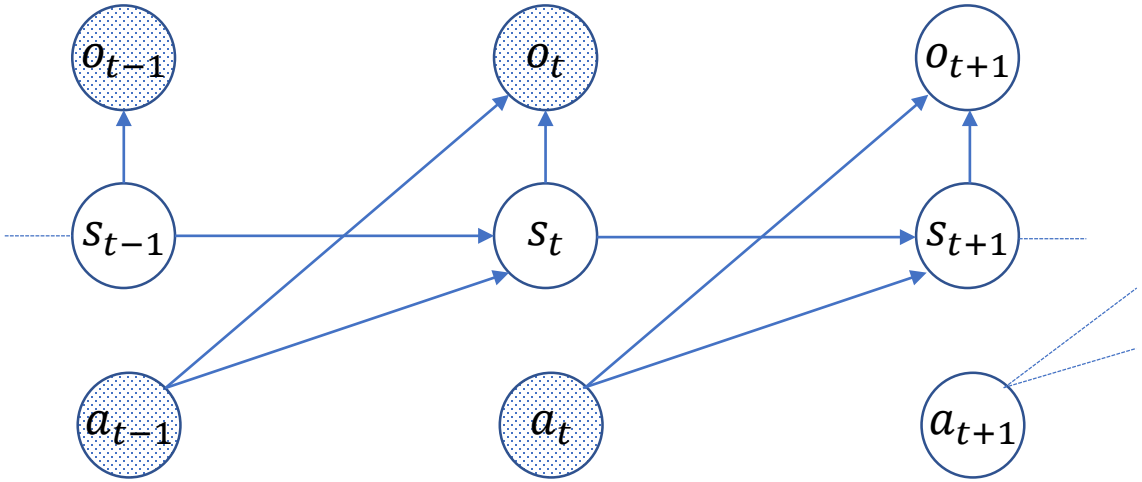
  End For

End For

V. Mnih+, "Human-level control through deep reinforcement learning," Nature, 2015

POMDP

- Partially Observable Markov Decision Process (POMDP) models an environment that is partially observable
- A POMDP is a tuple $\langle S, A, O, T, R, Z, \gamma \rangle$
  - $\langle S, A, T, R, \gamma \rangle$: Underlining MDP
  - O: set of observations
  - Z: observation function $Z(a_t, s_{t+1}, o_{t+1}) = P(o_{t+1}|a_t, s_{t+1})$



G. Monahan+, "A Survey of Partially Observable Markov Decision Processes: Theory, Models, and Algorithms," Management Science, 1982

G. Sani+, "A survey of point-based POMDP solvers," Auton Agent Multi-AG, 2012

Agent History and Belief

- Agent history
  - A history h_t is a sequence of actions and observations

$$h_t = \langle a_0, o_1, a_1, o_2, a_2, o_3, \cdots, a_{t-1}, o_t \rangle$$

  - History is a random variable having Markov Property

$$P(h_{t+1}|h_1, h_2, \cdots, h_t) = P(h_{t+1}|\, h_t)$$

- Belief
  - Belief $b = P(s|h)$ is a posterior distribution of state given history
  - Belief is a random variable having Markov Property

$$P(b_{t+1}|b_1, b_2, \cdots, b_t) = P(b_{t+1}|\, b_t)$$

Converting POMDP to MDP

For a POMDP $\langle S, A, O, T, R, Z, \gamma \rangle$:

- By considering history h as a state,

  $\langle \{h\}, A, P(h'|a,h), \sum_s R(s,a)P(s|h), \gamma \rangle$ is a MDP

- By considering belief b as a state,

  $\langle \{b\}, A, P(b'|a,b), \sum_s R(s,a)b(s), \gamma \rangle$ is a MDP

Update of Belief

$$b_{t+1} = P(s_{t+1}|h_{t+1}) = P(s_{t+1}|h_t, a_t, o_{t+1}) = \frac{P(o_{t+1}|h_t, a_t, s_{t+1})P(s_{t+1}|h_t, a_t)}{P(o_{t+1}|h_t, a_t)}$$

$$= \frac{P(o_{t+1}|a_t, s_{t+1}) \sum_{s_t} P(s_{t+1}|h_t, a_t, s_t)P(s_t|h_t, a_t)}{\sum_{s_t} P(s_t|h_t, a_t) \sum_{s_{t+1}} P(o_{t+1}|h_t, a_t, s_t, s_{t+1})P(s_{t+1}|h_t, a_t, s_t)}$$

$$= \frac{P(o_{t+1}|a_t, s_{t+1}) \sum_{s_t} P(s_{t+1}|a_t, s_t)P(s_t|h_t)}{\sum_{s_t} P(s_t|h_t) \sum_{s_{t+1}} P(o_{t+1}|a_t, s_{t+1})P(s_{t+1}|a_t, s_t)}$$

$$= \frac{Z(a_t, s_{t+1}, o_{t+1}) \sum_{s_t} T^{a_t}_{s_t s_{t+1}} b_t}{\sum_{s_t} b_t \sum_{s_{t+1}} Z(a_t, s_{t+1}, o_{t+1}) T^{a_t}_{s_t s_{t+1}}}$$

Use conditional independence c.f. d-separation [2]

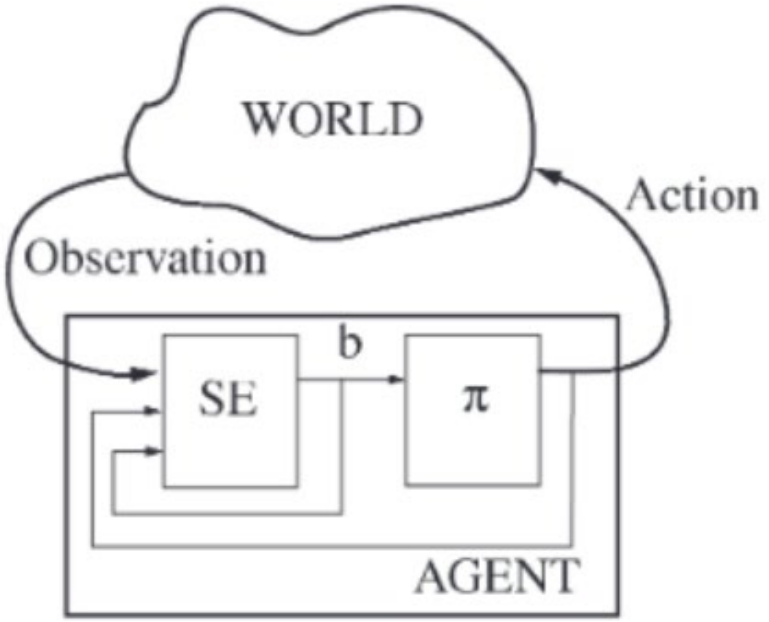


Figure is from [1]

[1] L. Kaelbling+, "Planning and acting in partially observable stochastic domains," Artificial Intelligence, 1998
[2] J. Perl, "Probabilistic Reasoning in Intelligent Systems," Morgan Kaufmann, 1988

Policy Gradient

- Directly parametrize the policy and optimize it [1]
- Policy performance objective (There are several variations [2])

$$J(\theta) = \sum_s d^{\pi_\theta}(s)\, V^{\pi_\theta}(s), \qquad d^{\pi_\theta}(s) = \lim_{t\to\infty} P(s_t = s | s_0, \pi_\theta)$$

- Policy gradient theorem

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta log \pi_\theta(s,a) Q^{\pi_\theta}(s,a)]$$

- REINFORCE[3]
  - Use return as an unbiased sample of $Q^{\pi_\theta}(s,a)$

$$\theta \leftarrow \theta + \alpha G_t \nabla_\theta \log \pi_\theta(a_t|s_t)$$

[1] R. Sutton+, "Policy Gradient Methods for Reinforcement Learning with Function Approximation," NISP, 1999
[2] R. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," Machine Learning, 1992
[3] J. Schulman+, "High-Dimensional Continuous Control Using Generalized Advantage Estimation," ICLR 2016

Actor-Critic

- Parameterize $Q^{\pi_\theta}(s,a)$ and learns it together with policy π_θ

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta log \pi_\theta(s,a) Q^{\pi_\theta}(s,a)] \approx E_{\pi_\theta}[\nabla_\theta log \pi_\theta(s,a) Q_w(s,a)]$$

- Actor: π_θ (with parameters θ)
- Critic:$Q_w(s,a)$(with parameters $\mathbf{w}$)

Deterministic Policy Gradient

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

[1] D. Silver+, "Deterministic Policy Gradient Algorithm," ICML 2014
[2] T. Lillicrap+, "Continuous Control With Deep Reinforcement Learning," ICLR 2016

Deterministic Policy Gradient

- Deterministic policy gradient models the policy as a deterministic decision
- Because a summation (or integration) over action is removed in the training, it is expected to be more efficient than stochastic policy especially in high dimensional action spaces

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

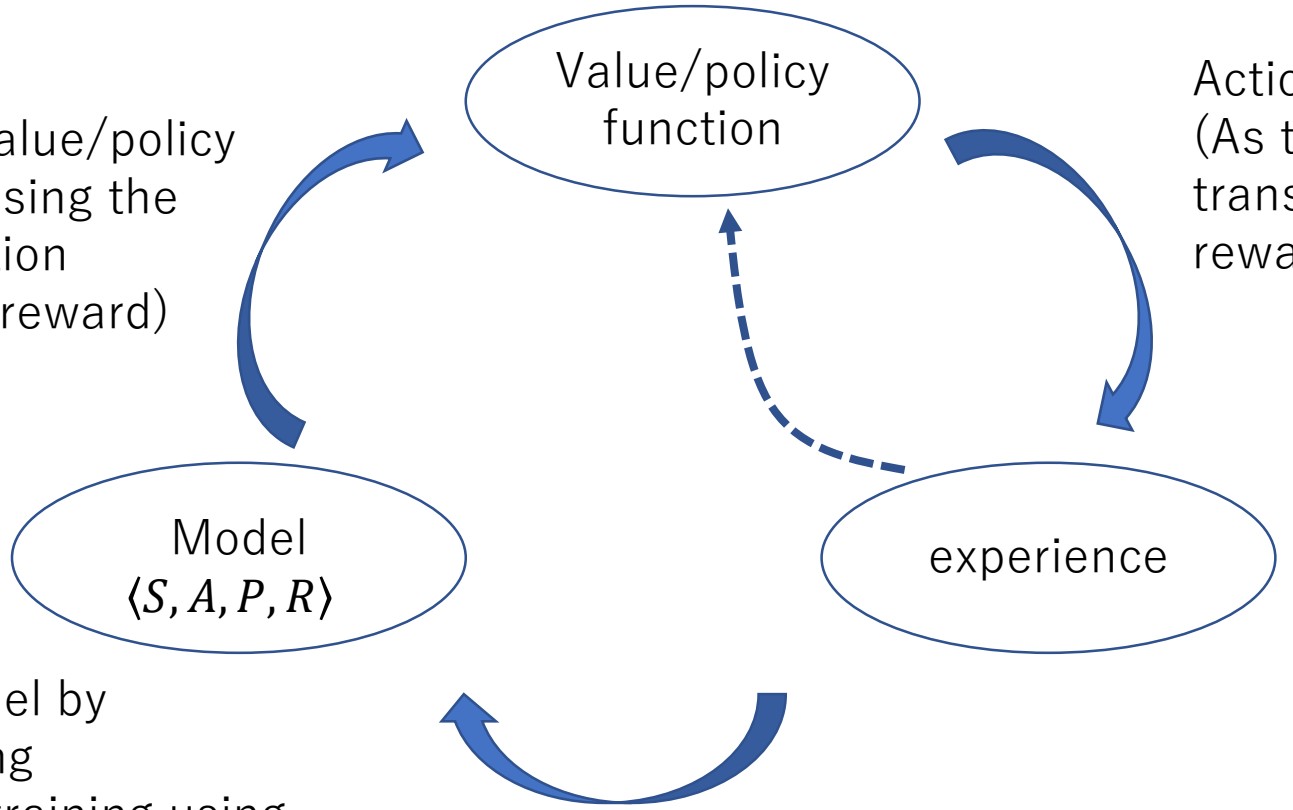
  end for
end for

[1] D. Silver+, "Deterministic Policy Gradient Algorithm," ICML 2014
[2] T. Lillicrap+, "Continuous Control With Deep Reinforcement Learning," ICLR 2016

Model-Based Reinforcement Learning

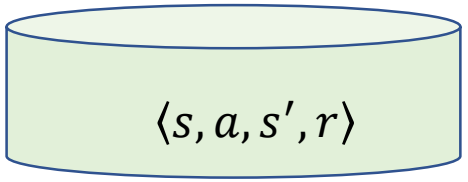
- Learns model from experience, and use it to construct a value or policy function

Planning
(Estimate the value/policy functions accessing the modeled transition probability and reward)



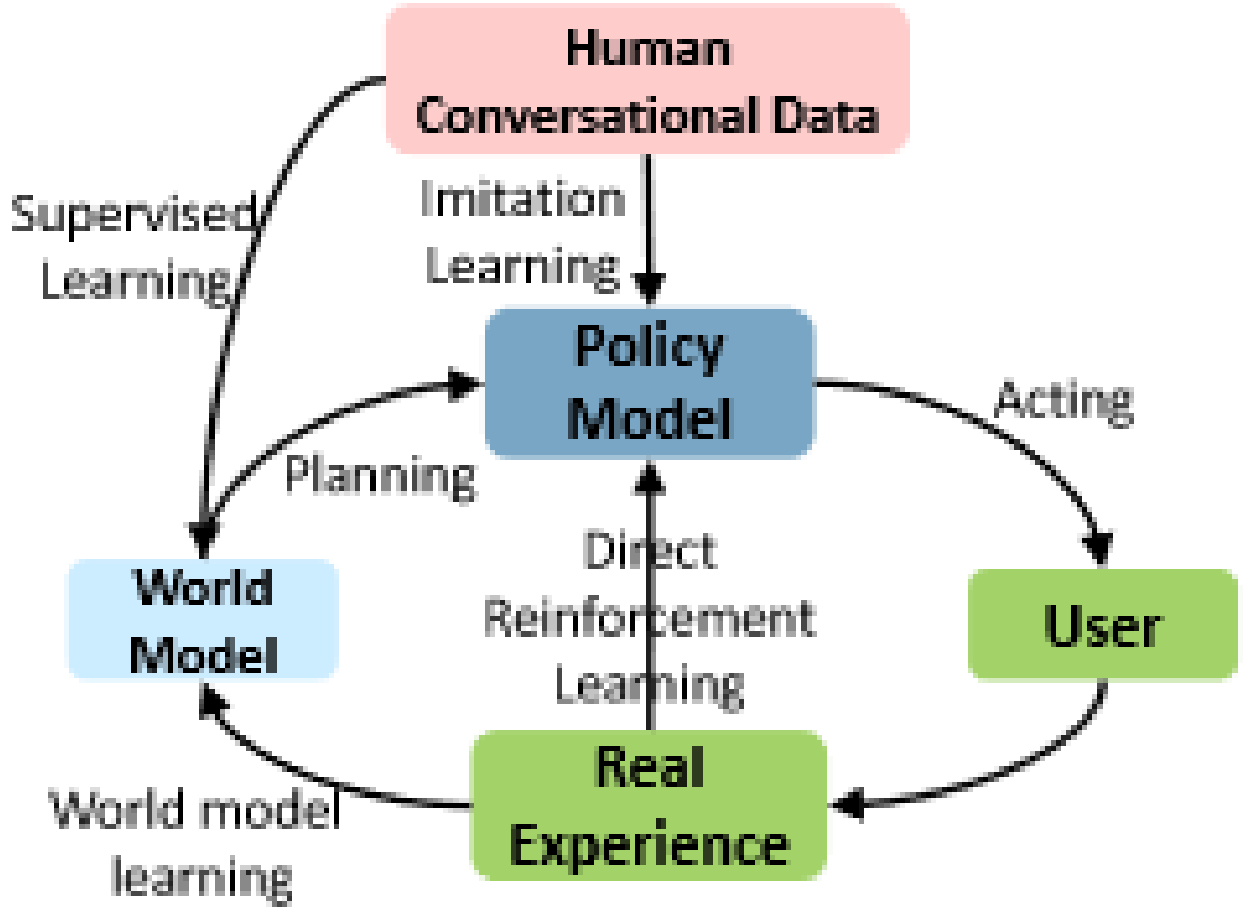
Action execution
(As the result, the state transitions to the next state and reward is obtained)

Accumulate tuples of action, state transition, and reward



Estimate the model by supervised training
(self-supervised training using the self-accumulated data)

Deep Dyna-Q



B. Peng+, "Deep Dyna-Q: Integrating Planning for Task-Completion Dialogue Policy Learning," ACL 2018

Textbooks and Online Materials

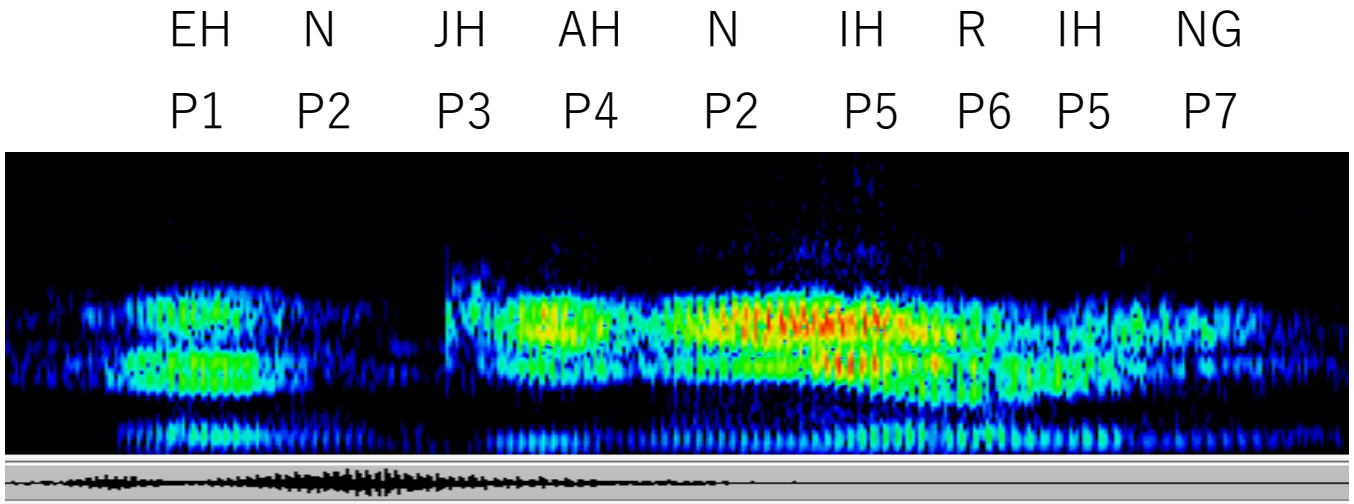
- Textbook
  - Richard Sutton and Andrew Barto, "Reinforcement Learning: An Introduction," A Bradford Book, 1998
  - C. Bishop, "Pattern Recognition and Machine Learning," Springer, 2006 (Chapter 8 describes Bayesian networks and d-separation)
- Online materials
  - Lecture note by D. Silver materials https://www.davidsilver.uk/teaching/
  - Documents of Stable-Baselines3 https://stable-baselines3.readthedocs.io/en/master/

PART 3

Subword/Word Discovery and Language Modeling

Unsupervised Subword Learning

- Find phonetically consistent units
  - Basic requirement
  - Differences of speakers should be eliminated
- Have one-to-one mapping with real units (phones, syllables, etc.)
  - May or may not be required depending on the applications



Syllable Segmentation by Envelope Minima Detection

Detects minima from smoothed signal amplitude envelopes

Black line: True word boundary

Blue line: Amplitude envelope

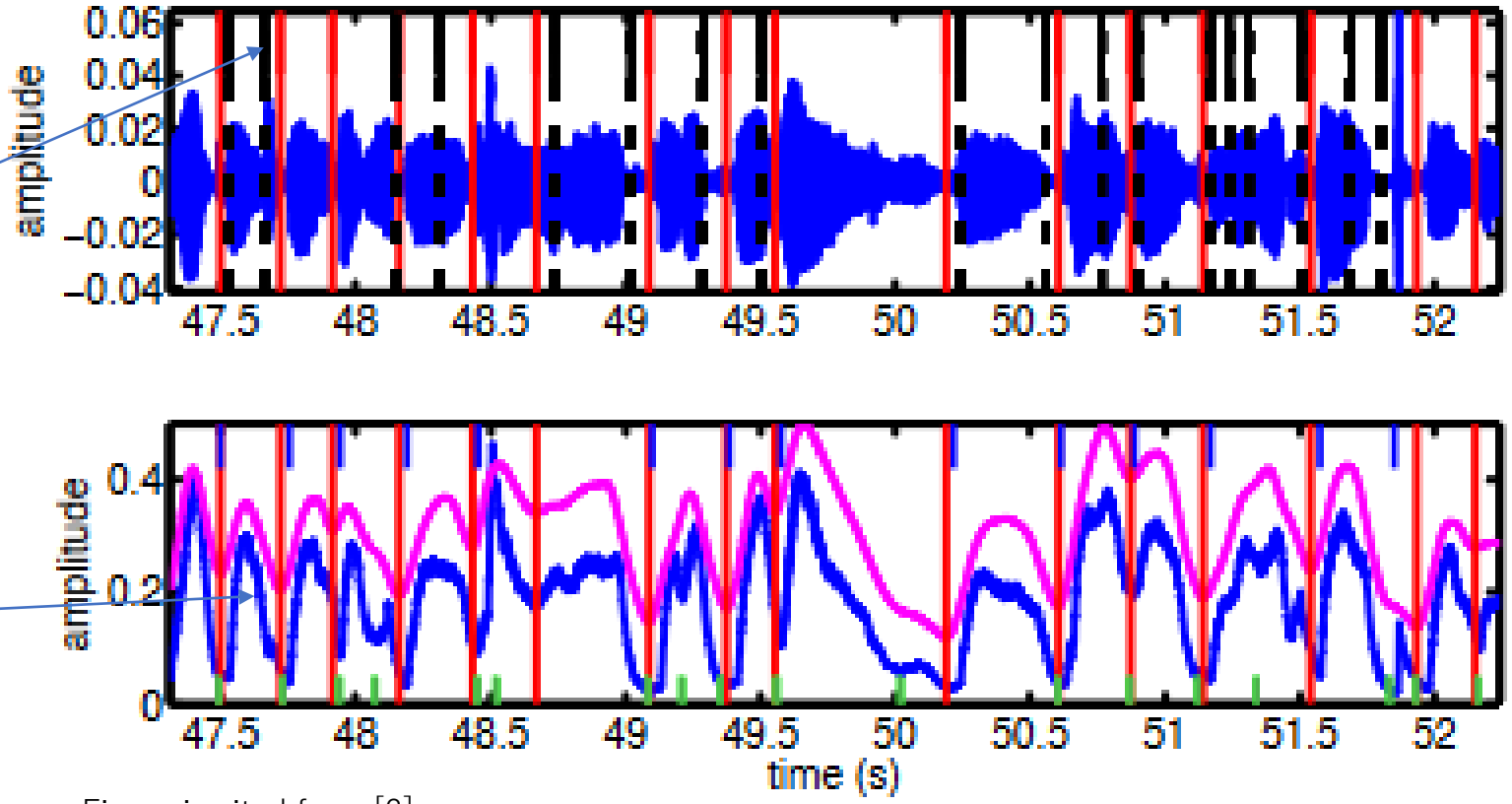


Figure is cited from [2]

[1] R. Villing+, "Performance limits for envelope-based automatic syllable segmentation," Proc. ISSC, 2006
[2] O. Rasanen+, "Unsupervised word discovery from speech using automatic segmentation into syllable-like units," Proc. Interspeech 2015

Wav2Vec2.0

Learns discrete code (pseudo phone) and segmentation from raw speech

Gumbel SoftMax based quantization +
BERT like mask based unsupervised learning

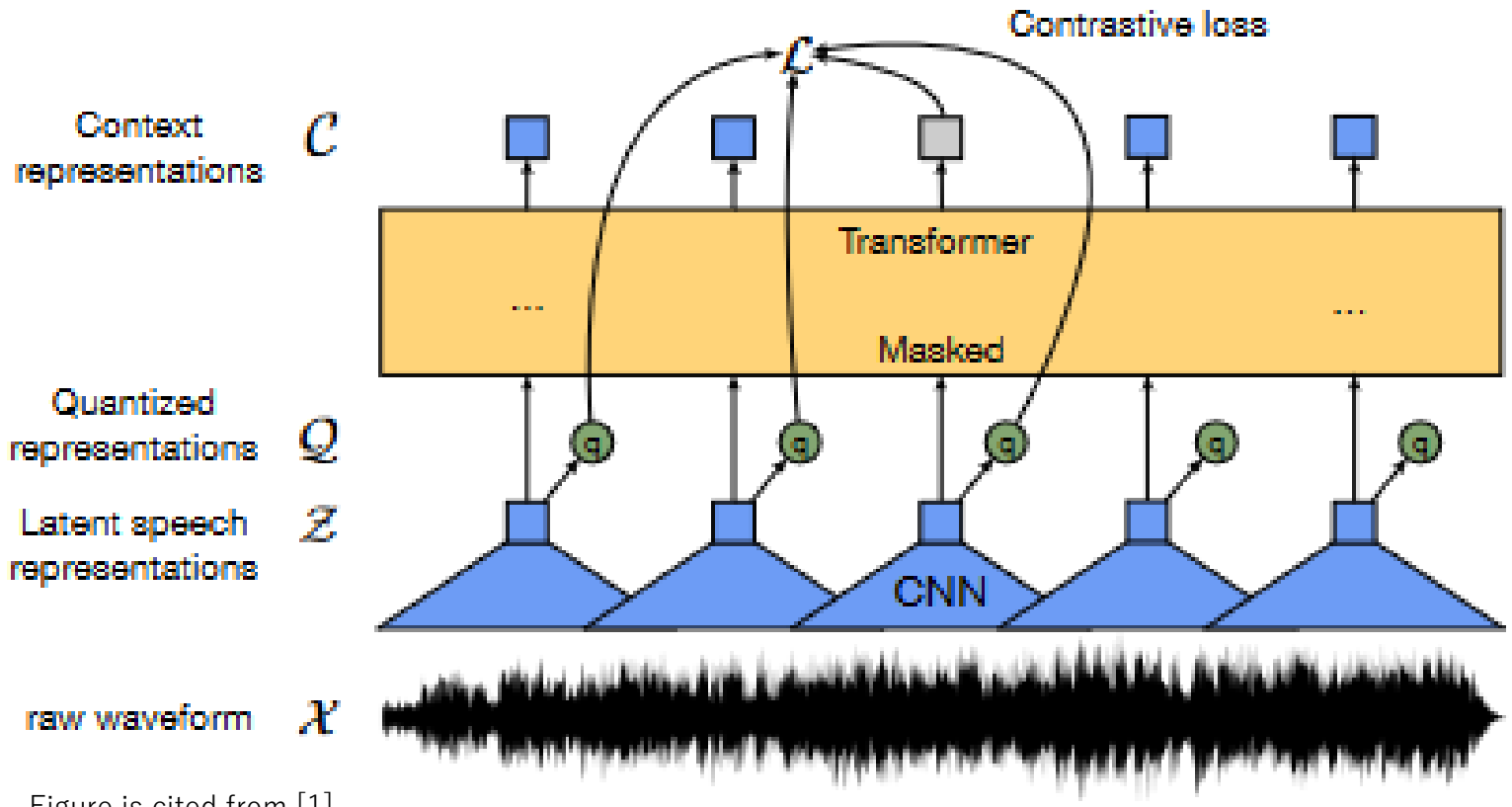


Figure is cited from [1]

[1] A. Baevski+, "wav2vec2.0: A Framework for Self-Supervised Learning of Speech Representations ," NeurIPS, 2020
[2] S. Schneider+, “wav2vec: Unsupervised Pre-training for Speech Recognition," Interspeech, 2019

Gumbel SoftMax

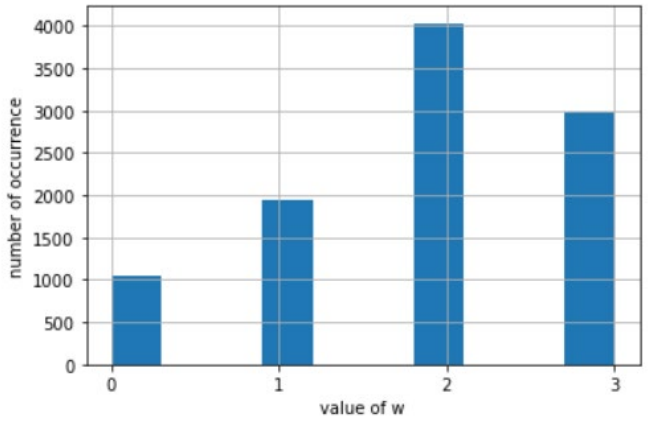
Gumbel-Max Trick

Let $\boldsymbol{u}$ be a sample from a V dimensional uniform distribution $\boldsymbol{u}\sim\mathbf{Uniform}(0,1)^V$ and $\boldsymbol{\pi}$ is a categorical distribution with V categories. Let $w(\boldsymbol{u})$ be a transformation where

$$w = argmax_k \frac{\exp(\log(\boldsymbol{\pi}) + \boldsymbol{n})}{\sum_{i=1}^{V} \exp(\log(\pi_i) + n_i)}, \boldsymbol{n} = -\log(-\log(\boldsymbol{u}))$$

Then w follows $\boldsymbol{\pi}$.

Example $\boldsymbol{V} = 4, \quad \boldsymbol{\pi} = [0.1, 0.2, 0.4, 0.3]$

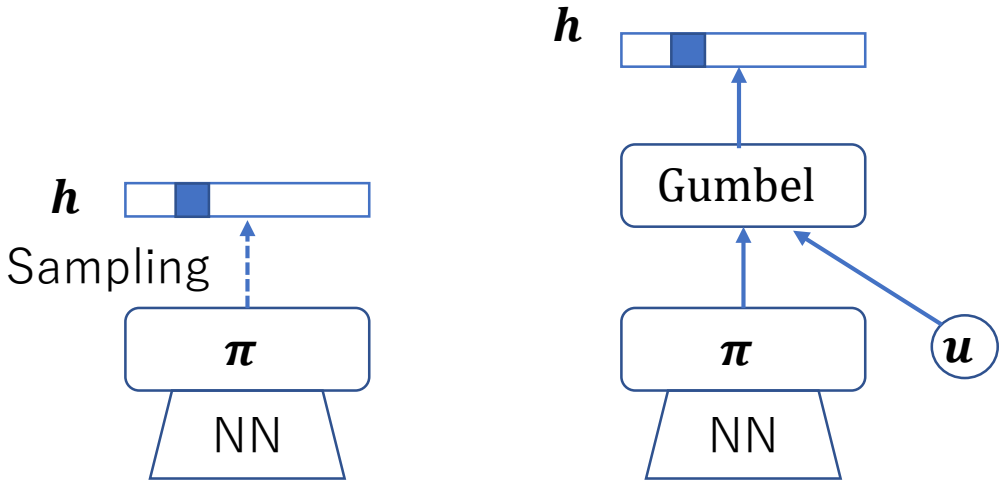


Gumbel SoftMax

Categorical distribution $\boldsymbol{p}$ by Gumbel SoftMax approaches to one-hot vector $\boldsymbol{h}$ as temperature τ becomes smaller.

$$\boldsymbol{p} = \frac{\exp(\frac{\log(\boldsymbol{\pi}) + \boldsymbol{n}}{\tau})}{\sum_{i=1}^{V} \exp(\frac{\log(\pi_i) + n_i}{\tau})}$$

Since $w = argmax_i \boldsymbol{p} = argmax_i \boldsymbol{h}$, we can implement the sampling from $\boldsymbol{\pi}$ in a differentiable way



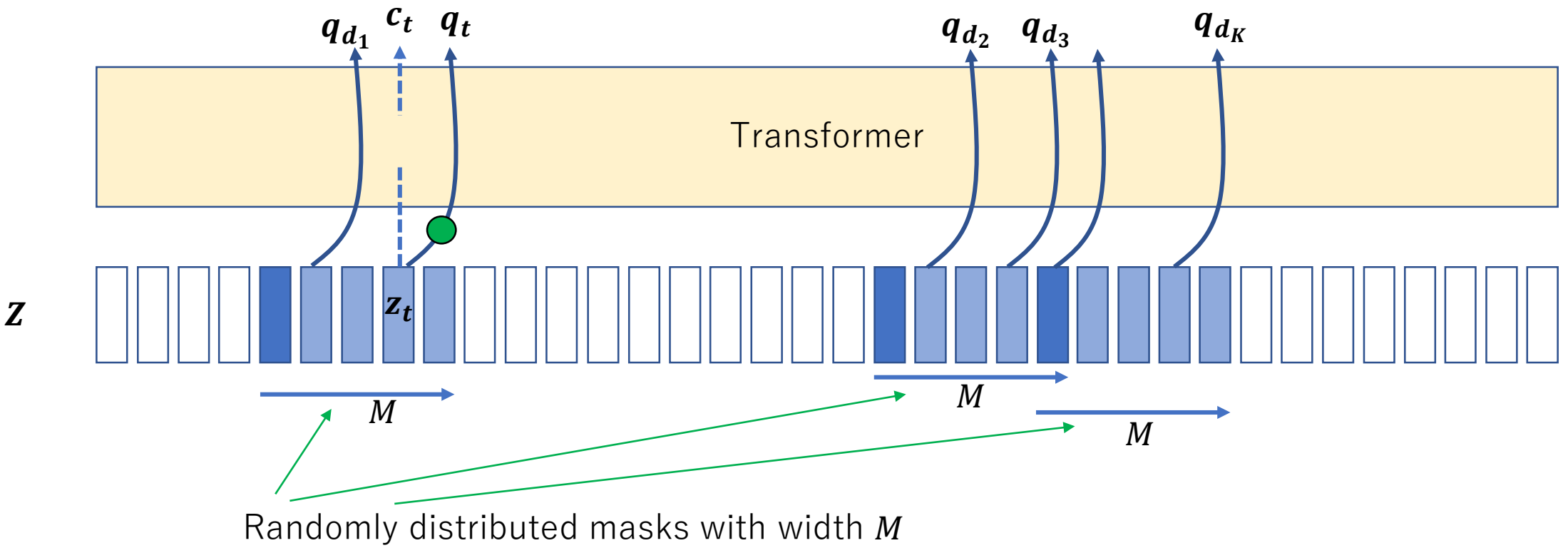
Contrastive Loss

$\boldsymbol{c_t}$: context representation at time frame t

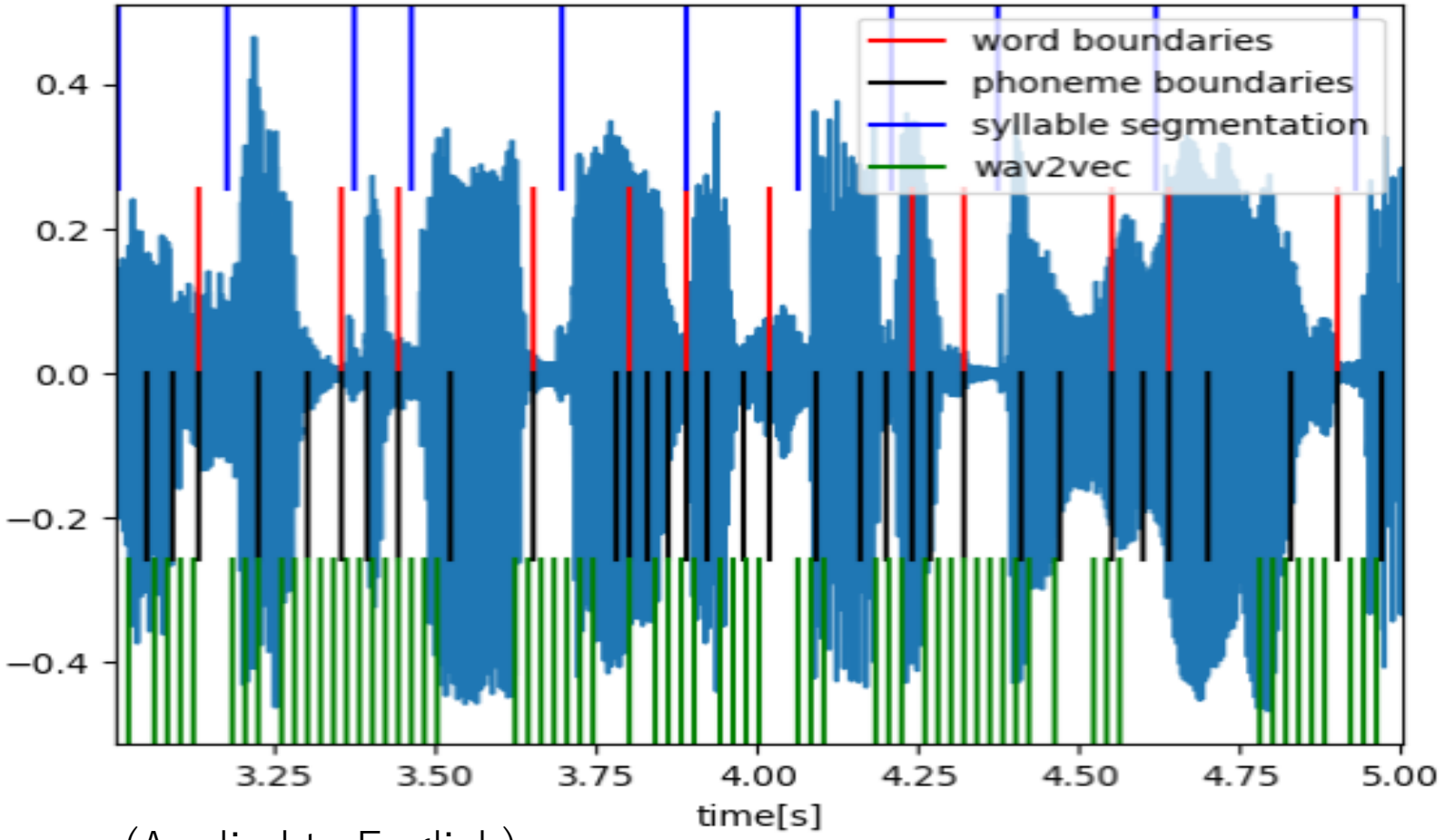
$\boldsymbol{q_t}$: quantized representation at time frame t

$\boldsymbol{q_{d_1}, q_{d_2}, \cdots, q_{d_K}}$: K distractors of quantized representation

$$\boldsymbol{L_m} = -log \frac{exp\left(\frac{sim(\boldsymbol{c_t}, \boldsymbol{q_t})}{\kappa}\right)}{\sum_{\hat{q} \in \{\boldsymbol{q_t}, \boldsymbol{q_{d_1}}, \boldsymbol{q_{d_2}}, \cdots, \boldsymbol{q_{d_K}}\}} exp\left(\frac{sim(\boldsymbol{c_t}, \hat{\boldsymbol{q}})}{\kappa}\right)}$$



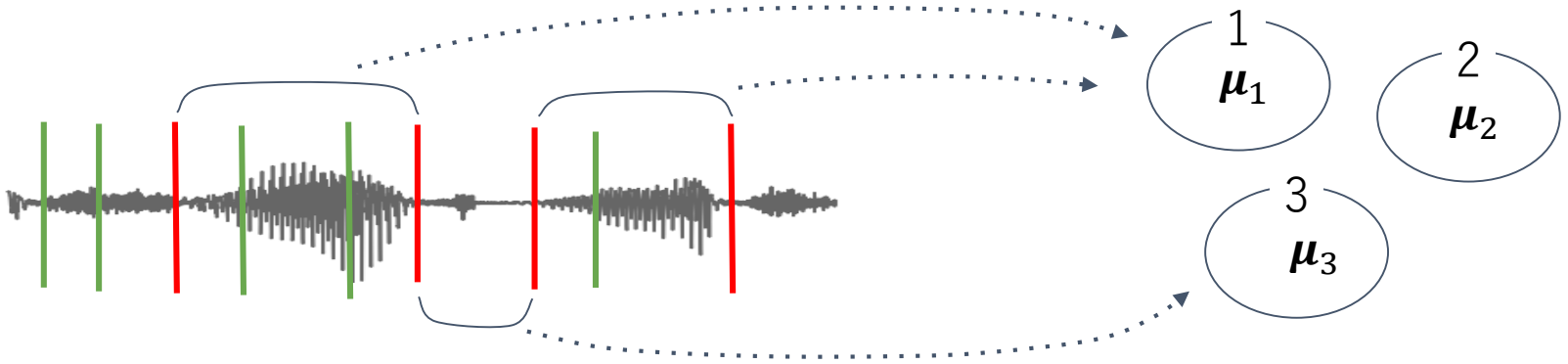
Segmentation Examples



(Applied to English)

Word Segmentation by ES-KMeans

- Input: Candidate word boundaries (e.g. (pseudo) syllable segmentation)
- Output: (Pseudo) word units and (pseudo) word segmentation
- Algorithm components:
  - acoustic embedding function
  - K-means clustering
  - Dynamic programming based search



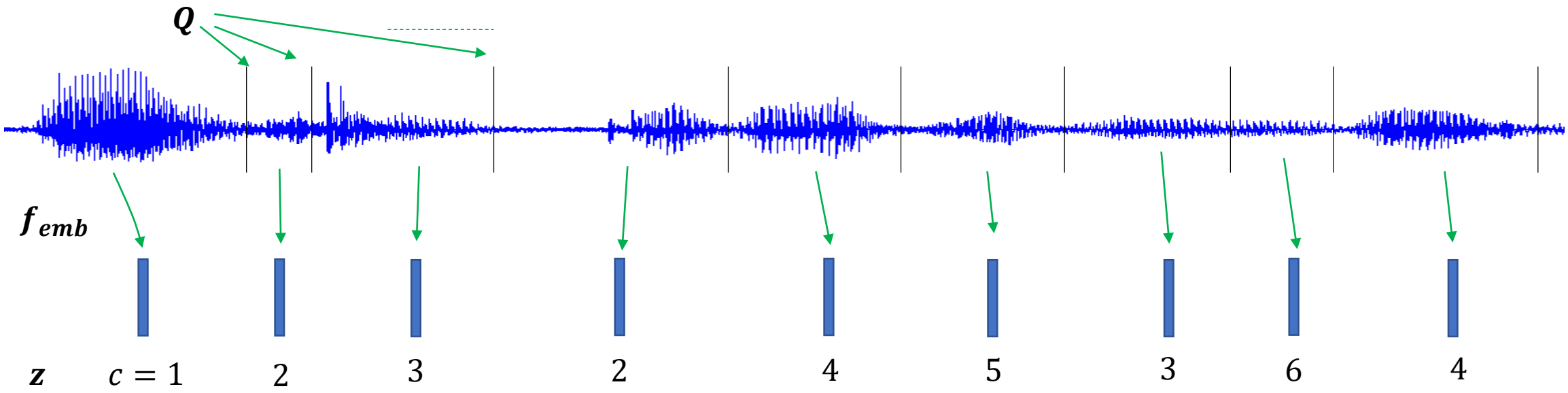
H. Kamper+, “An embedded segmental k-means model for unsupervised segmentation and clustering of speech," Proc. ASRU 2017.

Algorithm of ES-KMeans

1. Randomly initialize word segmentation $\boldsymbol{Q}$ and cluster assignment $\boldsymbol{z}$ for K clusters.
2. Apply the embedding function $\boldsymbol{f_{emb}}$ to the segments with $\boldsymbol{Q}$, and obtain a set of embedding vectors $f_{emb}\{\boldsymbol{Q}\}$. Following $\boldsymbol{z}$, obtain a set of cluster means $\boldsymbol{\mu_c}$
3. Given $\boldsymbol{z}$ and $\boldsymbol{\mu_c}$, obtain new $\boldsymbol{Q}$ that minimizes segmentation score L_{seg} using Dynamic Programing

$$L_{seg}(\boldsymbol{Q}) = \sum_{c=1}^{K} \sum_{x \in f_{emb}\{\boldsymbol{Q}\} \cap \boldsymbol{z}(x)=c} len(x) \, \|\boldsymbol{x} - \boldsymbol{\mu_c}\|^2$$

4. Given new $\boldsymbol{Q}$, obtain new $\boldsymbol{z}$ and $\boldsymbol{\mu_c}$ by K-means clustering. Go to step 3 or terminate when converged.



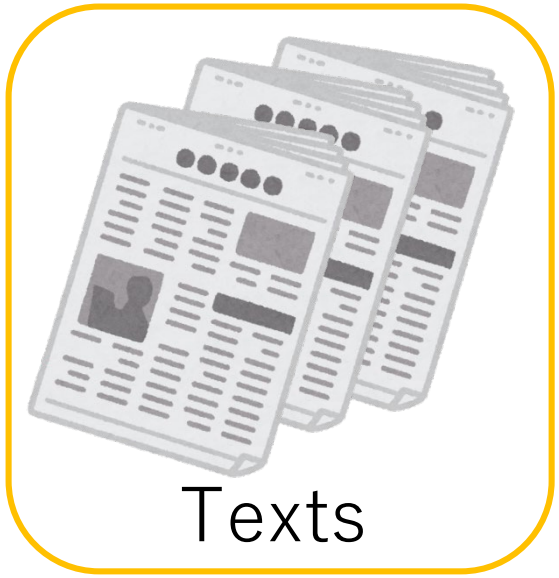
Language Model by Stochastic Segmentation

- Stochastic word segmentation [Mori+ ICSLP2004]
  1. P_i : Word boundary probability at position i
     - Estimator built by a segmented corpus
  2. Word 1-gram frequency of w given as the summation of expected frequency at all the occurrences $O_1 = \{i \mid x_{i+1}^{i+k} = w\}$

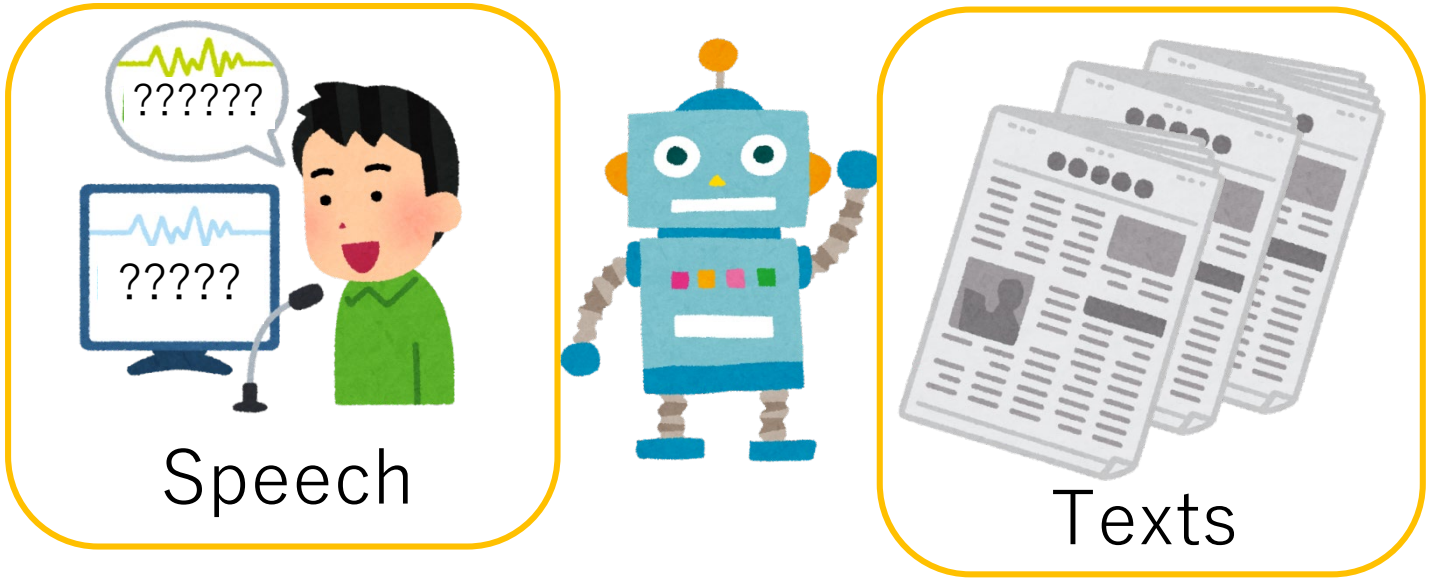
$$f_r(w) = \sum_{i \in O_1} P_i \left[\prod_{j=1}^{k-1} (1 - P_{i+j}) \right] P_{i+k}.$$

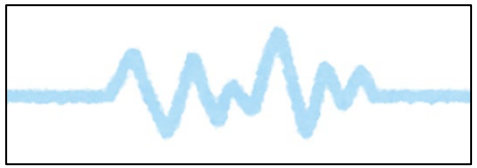
  3. Word 1-gram probability calculated by MLE
  4. Word *n*-gram probability given in a similar way

- Stochastic pronunciation estimation
  - The same idea, that is tag (pronunciation) distribution instead of word boundary probability



Acquiring Word-Pron. Pairs from Speech





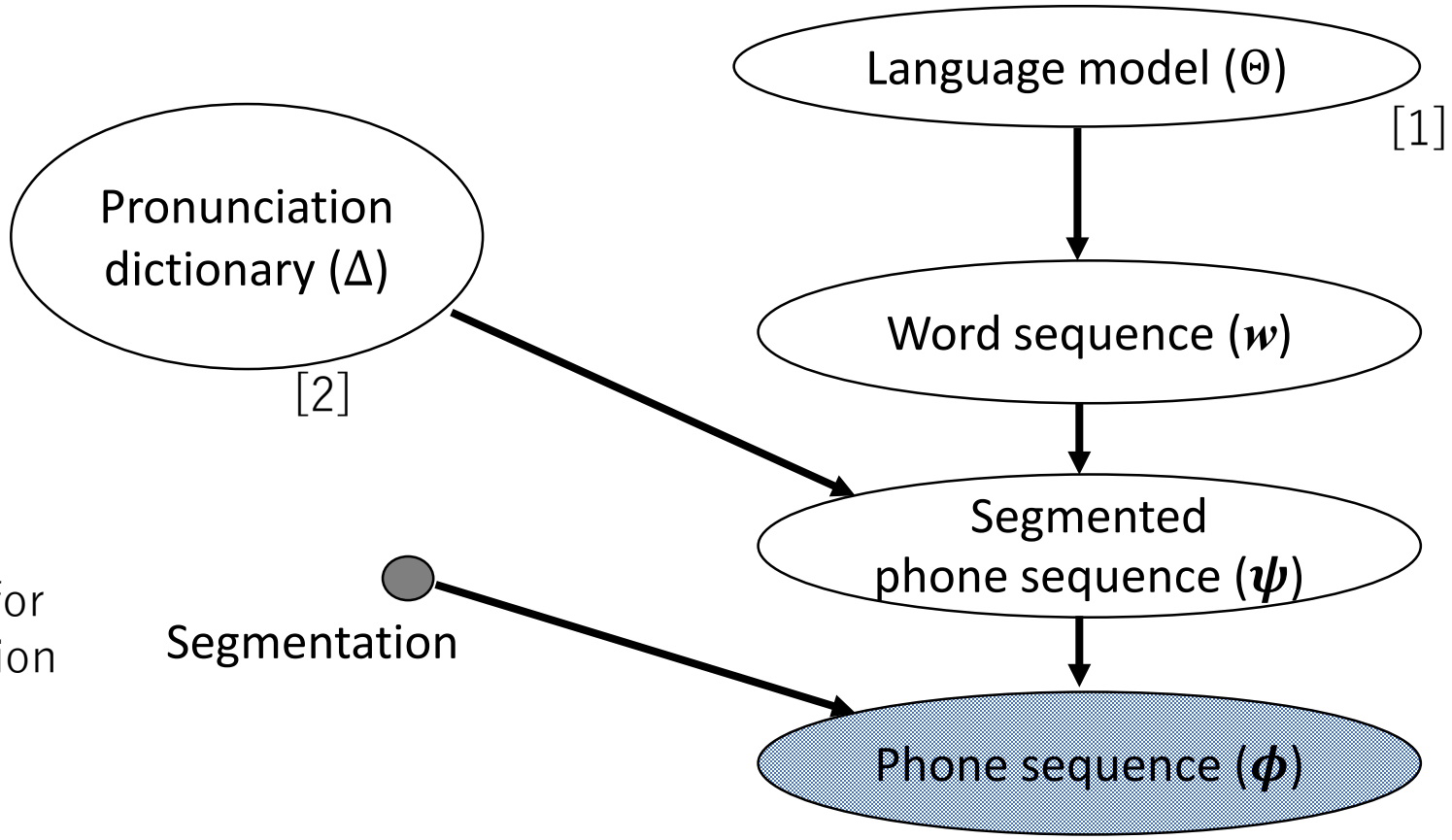
/kouvid…/

⟷ “COVID-19”

[Kurata+ ICASSP07]

1. Build an LM from texts with *all the possible* segmentations and pronunciations [Mori+ ICSLP2004]
2. Run ASR on speech data similar to the texts

Unified Bayes Model Integrating Dictionary and LM



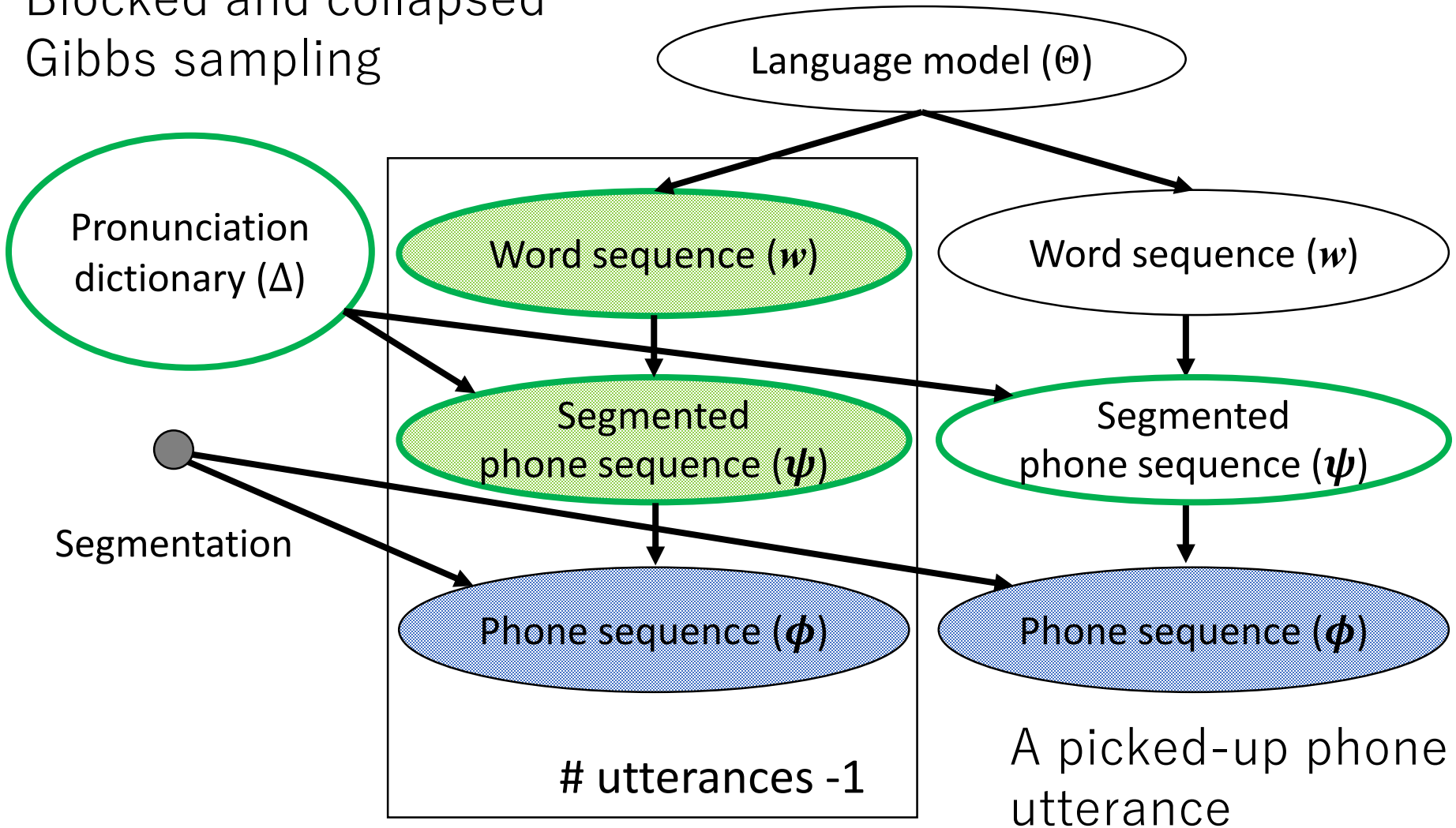
Weighted finite state transducer (WFST) is used for the implementation

[1] G. Neubig+, "Learning a language model from continuous speech." Proc. Interspeech, 2010

[2] T. Shinozaki+, "Semi-Supervised Learning of a Pronunciation Dictionary from Disjoint Phonemic Transcripts and Text," Proc. Interspeech, 2017

Gibbs Sampling Based Inference

Blocked and collapsed Gibbs sampling



Zero Resource Speech Challenge

A series of more and more challenging tasks

- Task 1: subword units discovery / representations learning
- Task 1b: discrete resynthesis
- Task 2: spoken term discovery / audio word segmentation
- Task 3: spoken language modeling

ZR15 (InterSpeech)	Task 1, 2
ZR17 (ASRU)	Task 1, 2
ZR19 (InterSpeech)	Task 1b
ZR20 (InterSpeech)	Task 1b, 2
ZR21 (new!)	*Task 3*

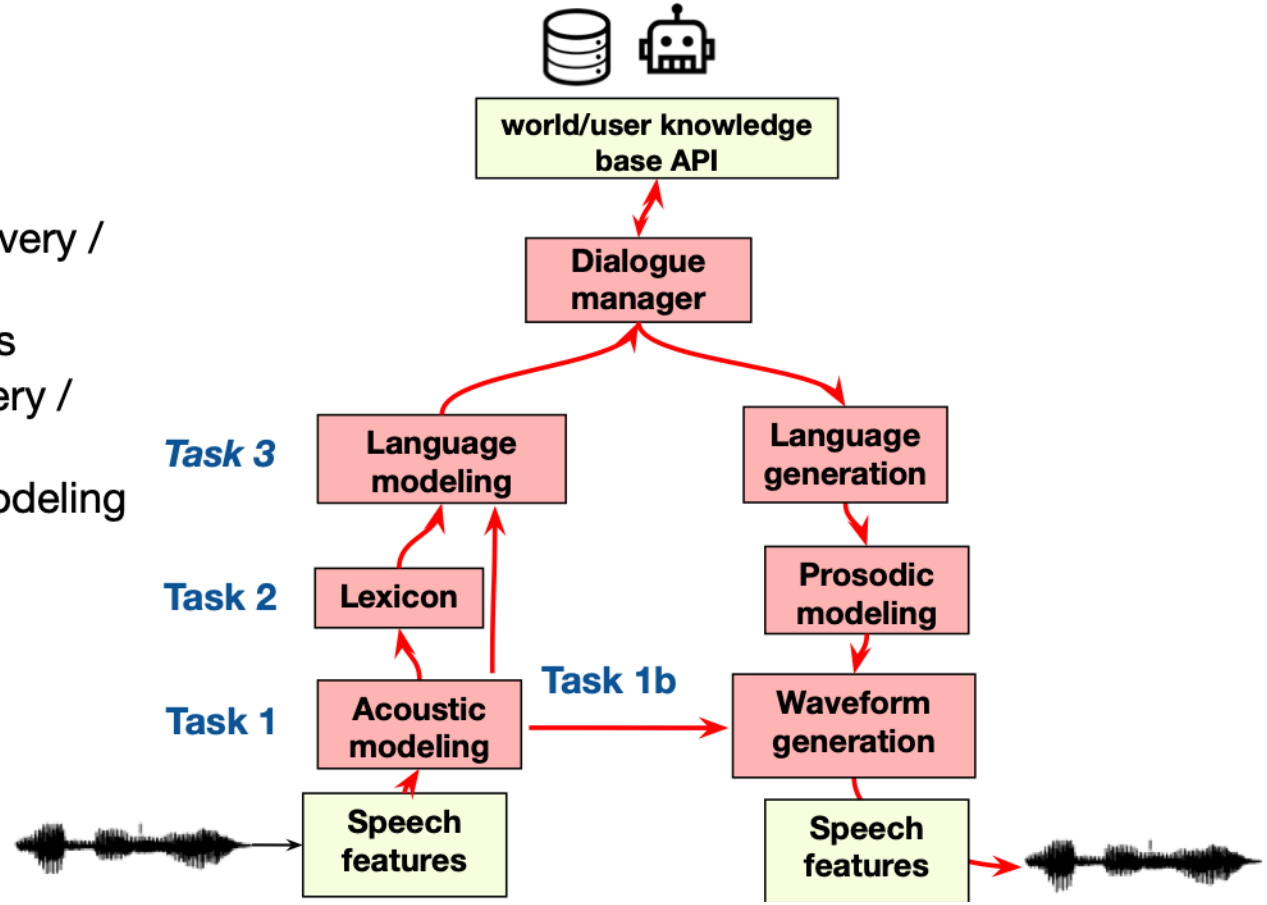


Figure is cited from [1]

[1] https://www.zerospeech.com/

Grounding

Sound-Image Similarity Learning

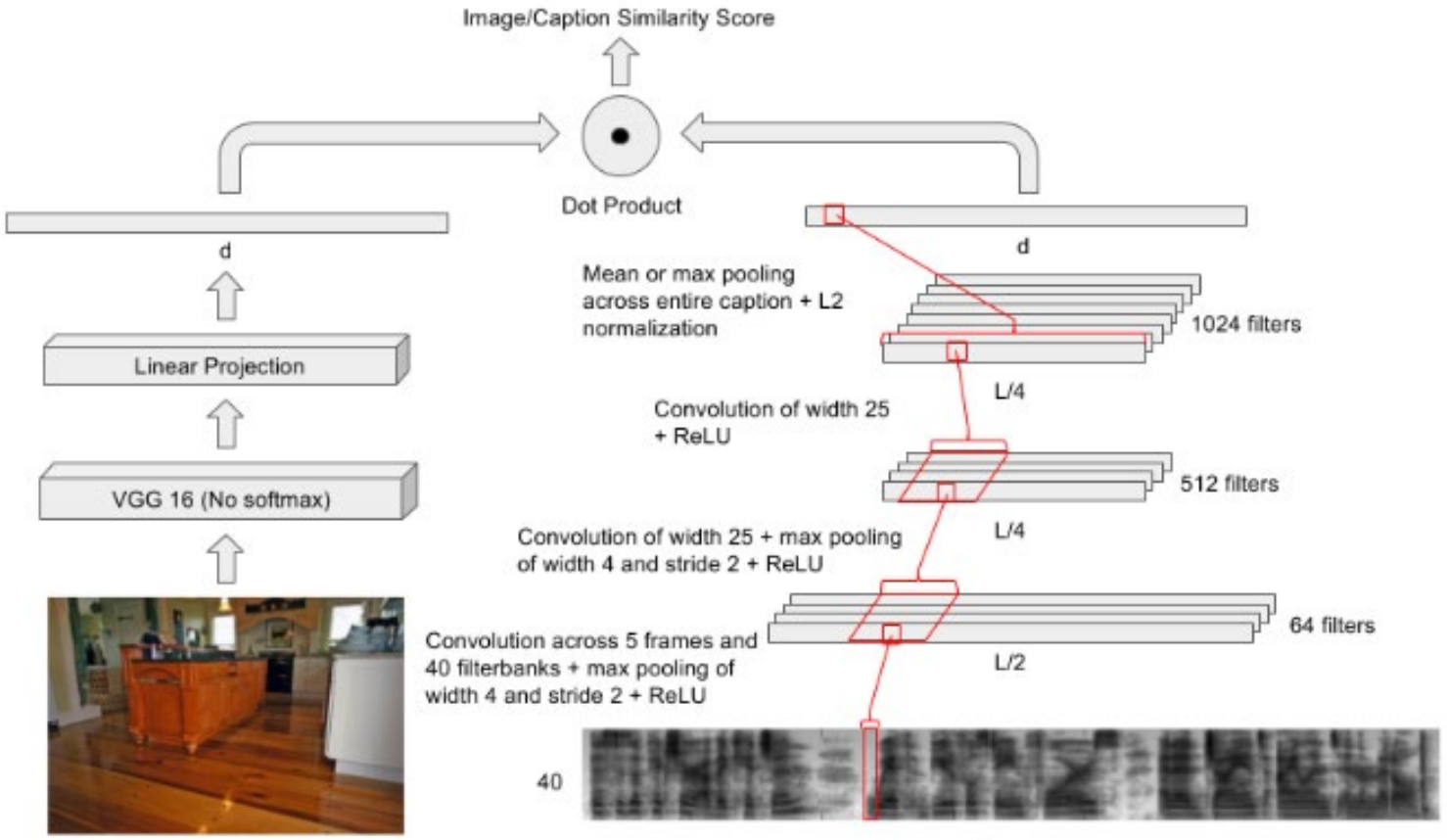


Image is cited from [1]

[1] D. Harwath+, “Unsupervised learning of spoken language with visual context," NIPS, 2016

NN for Game States and Language Expressions

1. Build a named entity recognizer (NER) designed for the game domain (ex. chess)
   - St: strategy names (ex. queen's gambit)
   - Mn: move names (ex. check mate)
   - …

2. Collect game states and commentaries on them
   - Run NER on the commentary texts

3. Train a neural network connecting
   - Encoding vector from a game state
   - Vector for the game NE vocabulary
     - 1 if the NE appears in the commentaries corresponding to the state,
     - 0 if not

- Tested on Japanese Chess [Ushiku+ SIGIR17]

https://simplifychess.com/

1
0
.
.
.

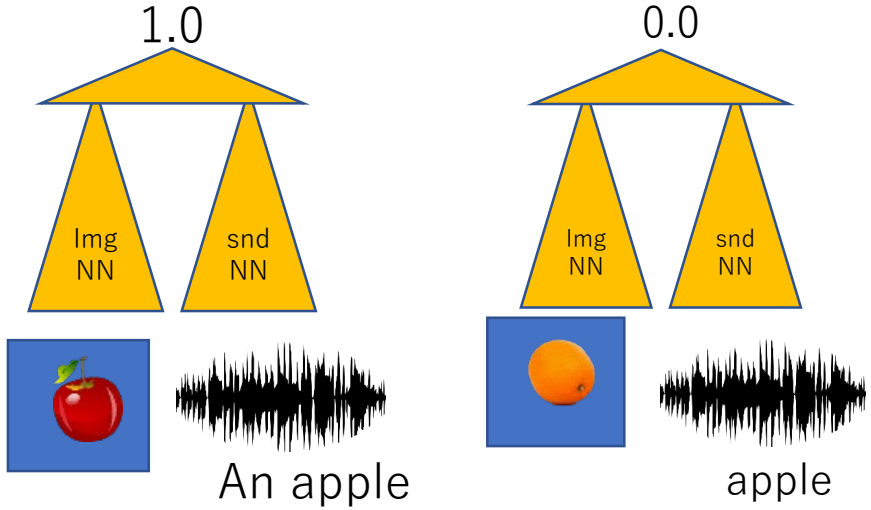
neural network

1 St: queen's gambit
0 St: king's gambit
.
.
.
0 Mn: check mate
1 Mn: en passant
.
.
.

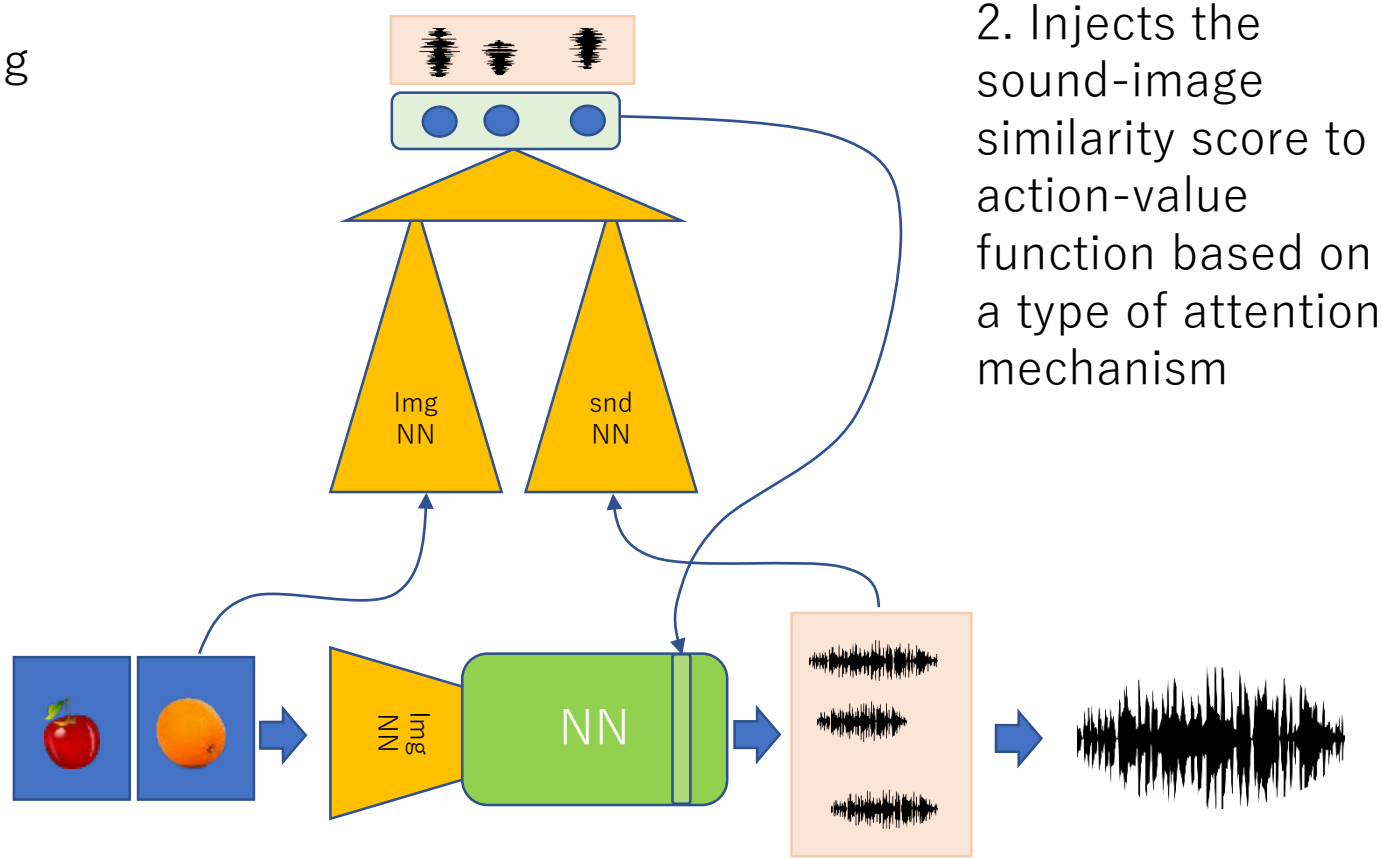
Vision Based Vocabulary Focusing Mechanism

Applies sound-image grounding to improve RL learning efficiency. Guide the agent's focus on those words that are related to current agent's eyesight

1. Triplet loss based unsupervised pre-training



2. Injects the sound-image similarity score to action-value function based on a type of attention mechanism



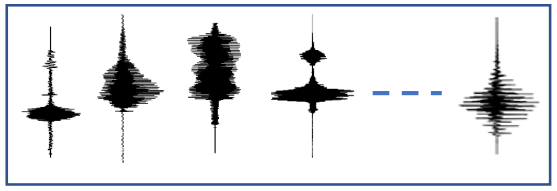
Zhang+, "Sound-Image Grounding Based Focusing Mechanism for Efficient Automatic Spoken Language Acquisition," Proc. Interspeech, 2020

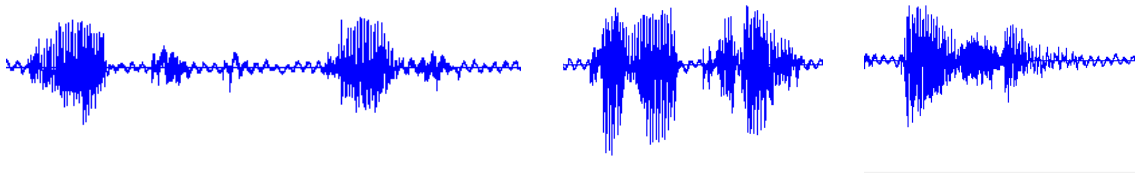
Speech Synthesis

Record and Playback

- Makes sound dictionary, and simply replay it
  - The dictionary contains waveform segments of word or phrase
  - Sentences can be made by concatenating the segments
- Limitation
  - No control with the pronunciation other than selection
  - Intonation becomes discontinuous at word/phrase boundaries

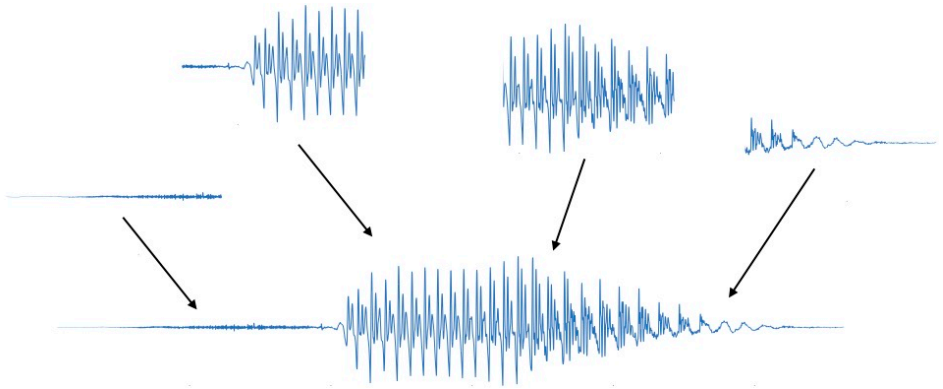
Sound dictionary





Unit Selection Methods

- Concatenating short speech waveform units
  - Units: phone, di-phone, etc.
  - Prepare many variations of the same unit, and select the best one when synthesizing an utterance to make the seams smooth
- Problems
  - Large collection of unit segments is needed
  - Difficult to control

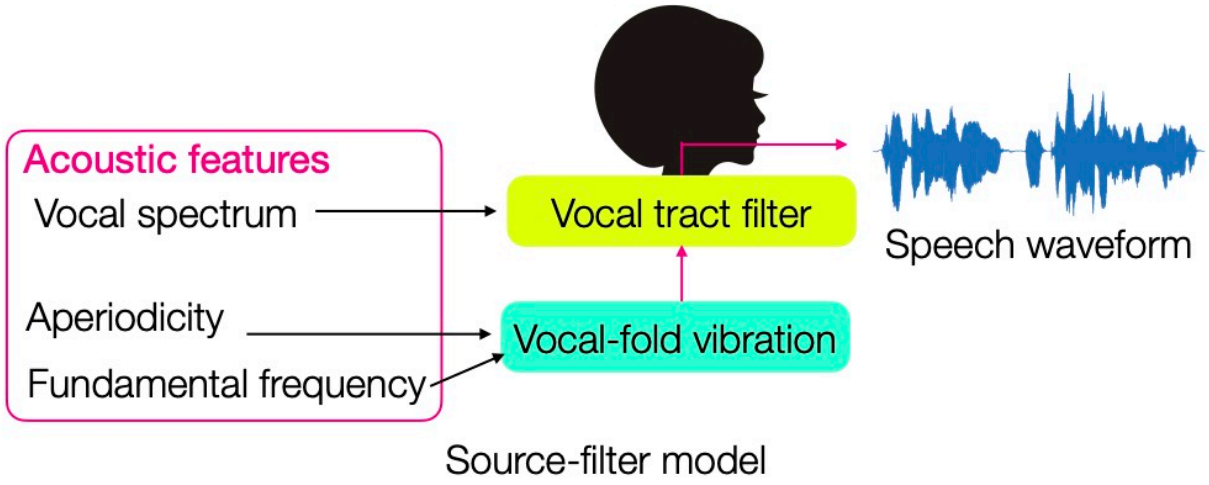


Source-filter vocoders

- Generating speech waveform from acoustic features
- Controllable
- Problems
  - lower synthesis quality than unit selection
  - Minimum phase approximation
  - Stationary assumption in an analysis frame
  - Analysis error of acoustic features

(e.g. STRAIGHT, WORLD)

Signal processing-based approach

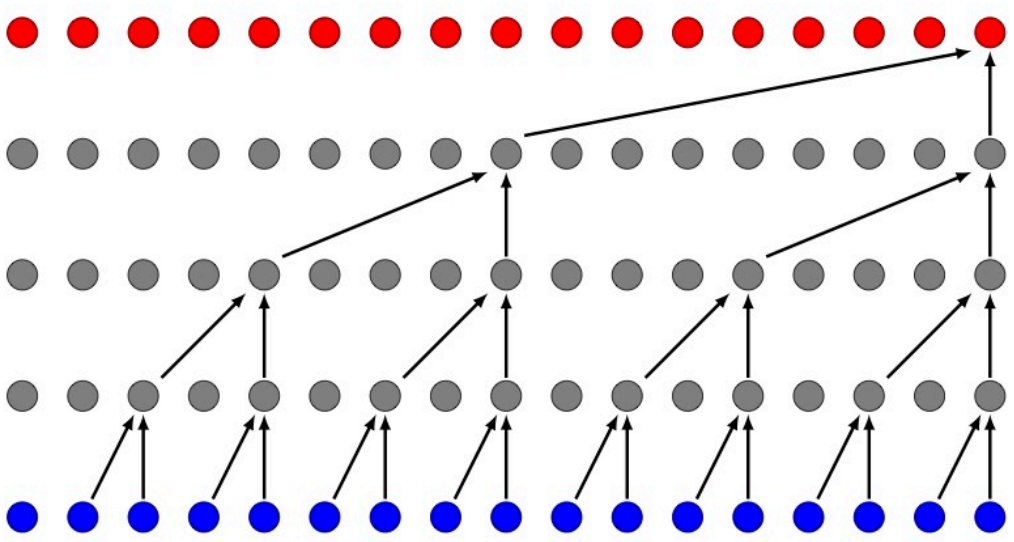


Source-filter model

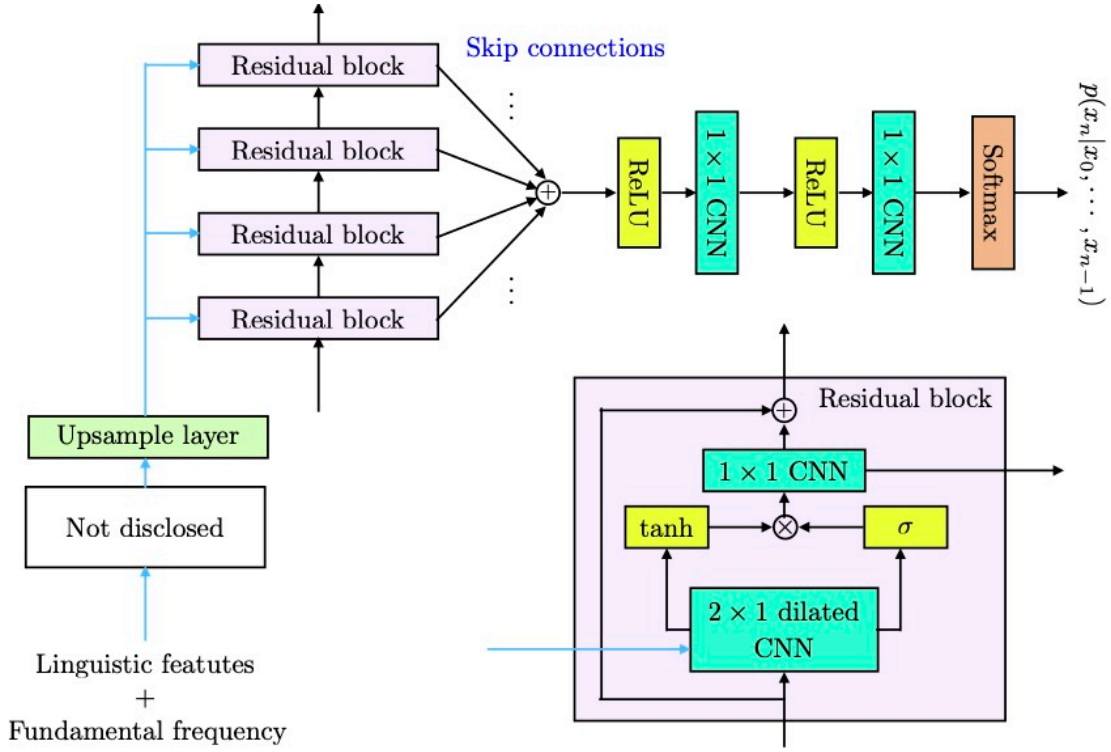
WaveNet

- Neural network-based speech waveform generative model: WaveNet
  - Autoregressive neural network with dilated causal convolution layers
    - Predicting joint probability of next sample from previous waveform samples

$$p(\boldsymbol{x} \mid \boldsymbol{h}) = \prod_{t=1}^{T} p(x(t) \mid x(1), \cdots, x(t-1), \boldsymbol{h})$$



Dilated causal convolution

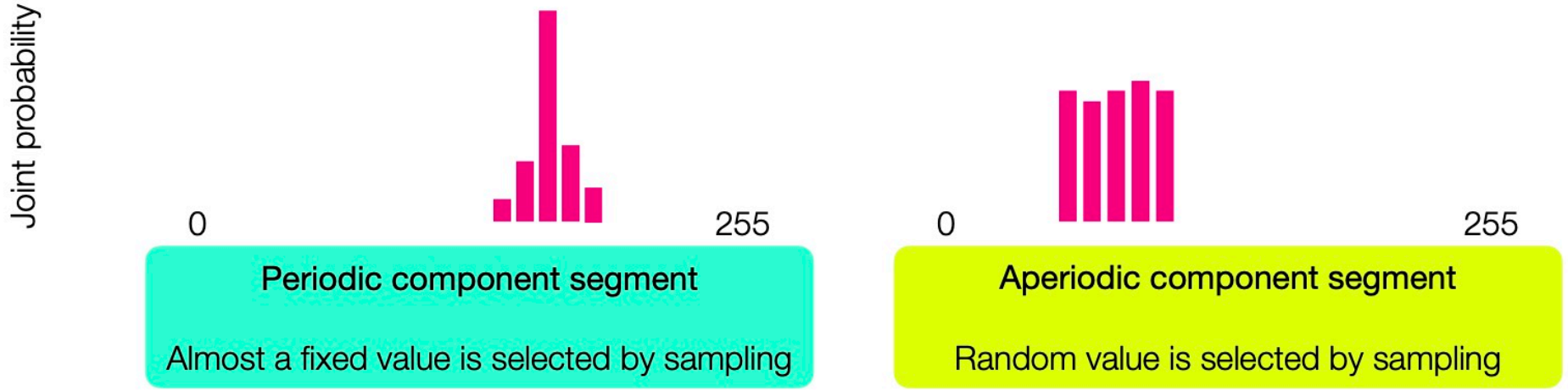


Network structure of WaveNet

How Realize High-Fidelity Synthesis in WaveNet?

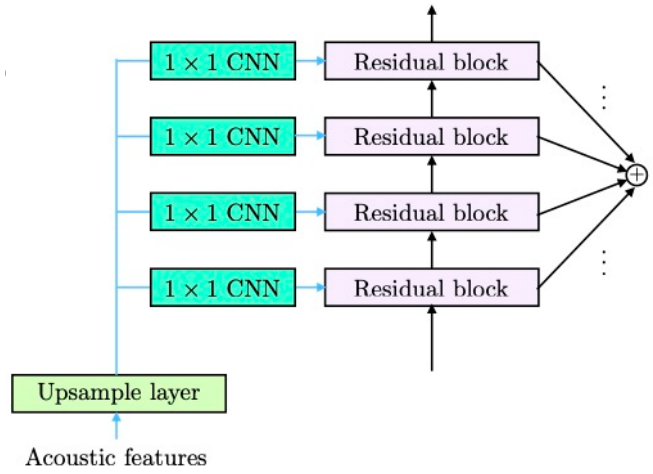
- Categorical problem
  - Applying 8 bit μ-law quantization and modeling continuous speech waveforms as 256 discretized values
    - Any shape of distribution can be realized compared with Gaussian distribution in MSE loss

- Sampling-based inference for speech waveform generation
  - Sampling next speech waveform value based on joint probability
    - Efficiently modeling periodic and aperiodic components
    - Different speech waveforms are synthesized in each inference



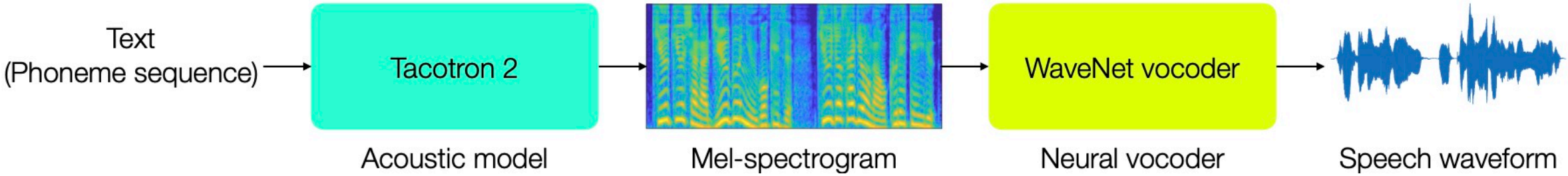
Neural Vocoder

- WaveNet vocoder
  - Conditioned on acoustic features (instead of linguistic featur
  - Higher synthesis quality than STRAIGHT and WORLD
  - Begging of "Neural vocoder" synthesizing speech waveform from acoustic features



Conditioned on acoustic features

- End-to-end neural text-to-speech: Tacotron 2
  - Comparable synthesis quality to human speech
  - Introducing mel-spectrogram as acoustic features instead of source-filter vocoder features
    - Without analysis error and widely used in neural vocoder



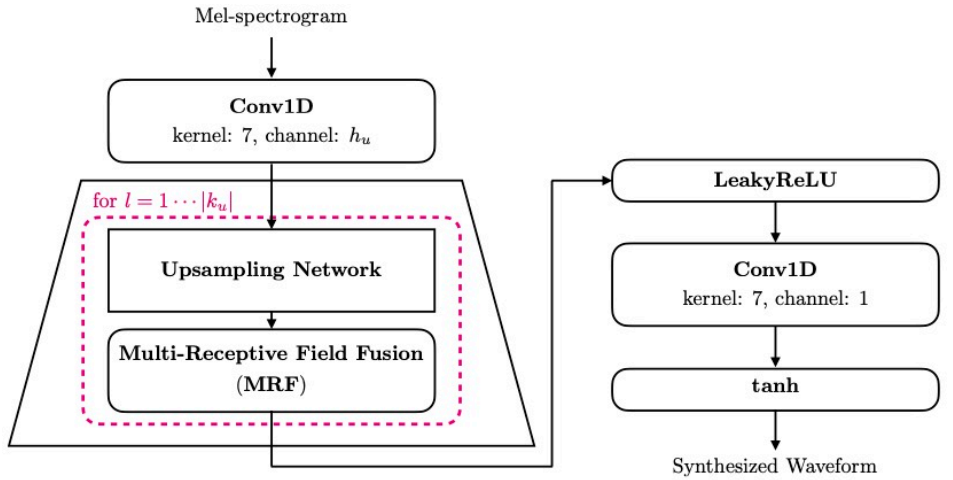
- Problem of WaveNet vocoder
  - Very slow inference due to auto-regressive structure and huge size of network

Real-Time Neural Vocoders

- Key point in neural vocoder
  - How accurately synthesize fine structure and aperiodicity components of speech waveforms?
- Fast autoregressive models
  - Still autoregressive as WaveNet but lightweight network structure for real-time inference (e.g. WaveRNN, LPCNet)
    - Based on sampling
- Non-autoregressive models
  - Flow-based methods (e.g. Parallel WaveNet, WaveGlow)
    - Based on autoregressive flow or generative flow with Gaussian white noise input
  - Generative adversarial network (GAN)-based methods (e.g. MelGAN, Parallel WaveGAN, HiFi-GAN)
    - Based on adversarial training with generators and discriminators
  - Diffusion probabilistic models (e.g.WaveGrad, DiffWave)
    - Based on diffusion with additive Gaussian white noise in training and iterative denoising in inference

HiFi-GAN

- Recent widely used model
  - GAN-based real-time neural vocoder on CPUs
  - Upsample-based high-speed generator
  - Multi-period discriminator
    - Modeling periodic patterns
  - Multi-scale discriminator
    - Capturing consecutive patterns and long-term dependencies



HiFi-GAN generator network

- Property
  - Tradeoff between synthesis quality and inference speed on CPUs
    - HiFi-GAN V1 (large model): High-quality but slow inference speed
    - HiFi-GAN V1 (small model): Fast inference speed but lower quality

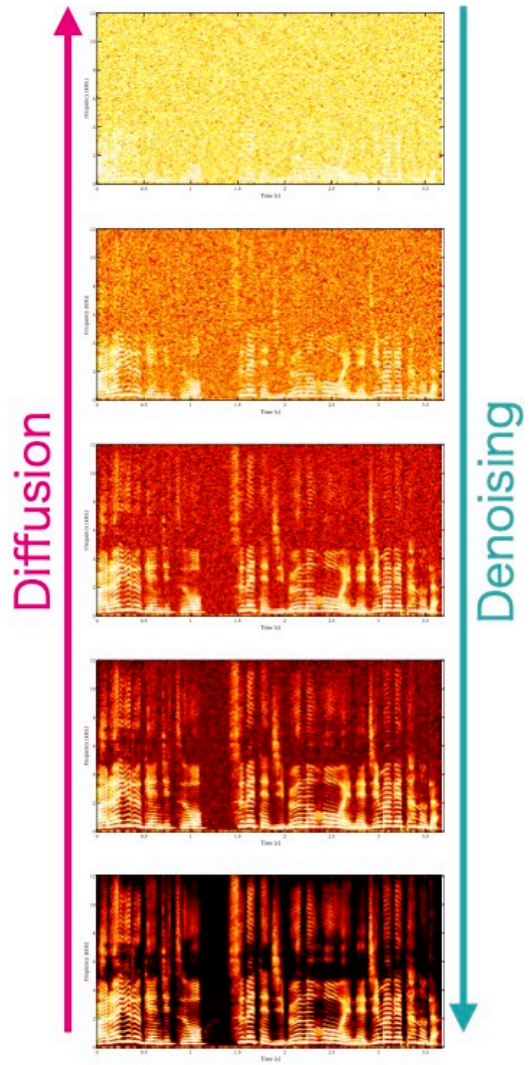
- Official implementation on GitHub
  - https://github.com/jik876/hifi-gan

Diffusion Probabilistic Neural Vocoders

- WaveGrad and DiffWave
  - Training: Predicting noise component from mixture of speech waveform and white noise
    - Loss function: $\mathbb{E}_{\epsilon,c}\left[\left\|\epsilon-\epsilon_\theta\left(\sqrt{\bar{\alpha}}x_0+\sqrt{1-\bar{\alpha}}\epsilon,h,c\right)\right\|_2^2\right]$ c : WaveGrad: discretized & continuous DiffWave: only discretized
    - Example of noise schedule (N=1000):

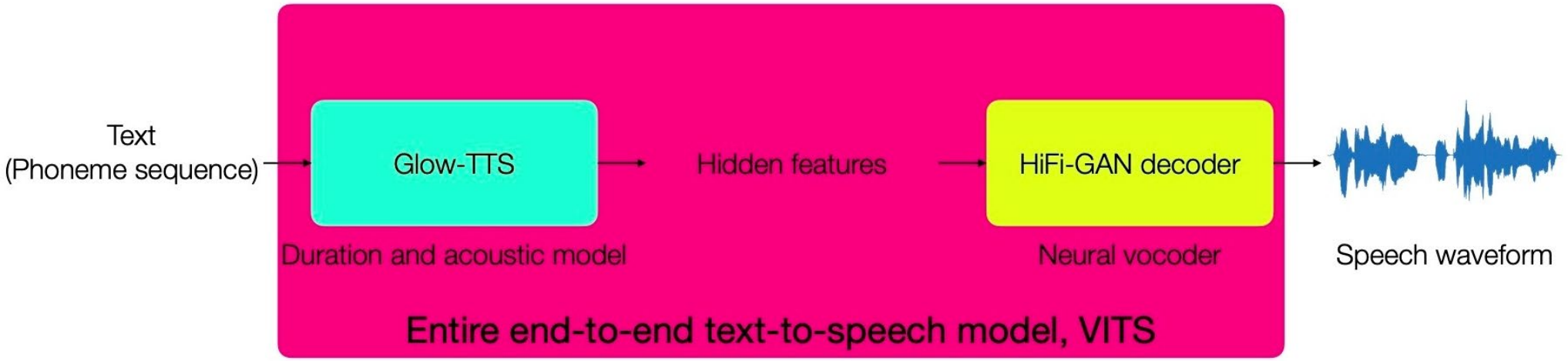
$$\beta_{1,\cdots,n,\cdots,1000}=\text{Linear}(1\times10^{-4},0.005,1000)\quad \alpha_n=1-\beta_n\quad \bar{\alpha_n}=\prod_{s=1}^{n}\alpha_s$$

  - Inference: Input noise is gradually converted to a speech waveform based on Langevin dynamics
    - Update function: $x_{n-1}=\frac{1}{\sqrt{\alpha_n}}\left(x_n-\frac{1-\alpha_n}{\sqrt{1-\bar{\alpha}_n}}\epsilon_\theta\left(x_n,h,c\right)\right)+\sigma_n z$
  - Difference of model structures
    - WaveGrad: Multiple upsampling and downsampling layers
    - DiffWave: Multiple non-causal dilated convolution layers as Parallel WaveGAN



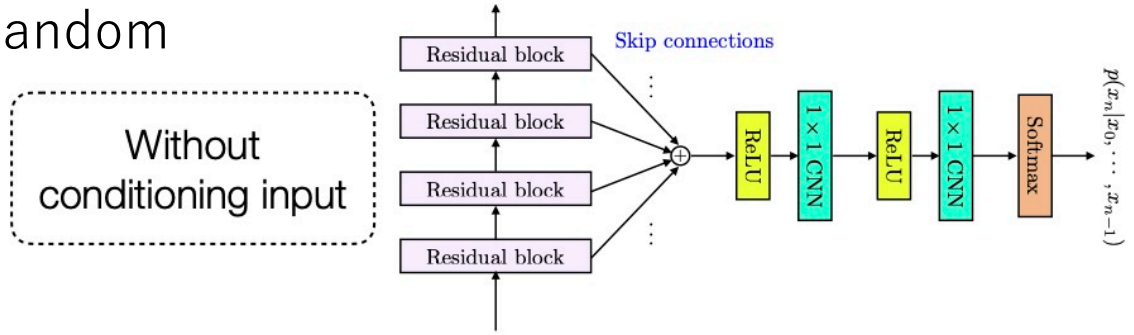
Entire End-to-End Text-to-Speech Models

- Single network directly synthesizing speech waveforms from texts
  - Duration model, acoustic model and neural vocoder are jointly trained as single network without acoustic features
    - EATS: including GAN-TTS
    - FastSpeech 2+: including Parallel WaveGAN
    - Wave-Tacotron: including WaveGlow
    - VITS: including HiFi-GAN (Official implementation: https://github.com/jaywalnut310/vits)
    - Reinforced-Aligner: including HiFi-GAN



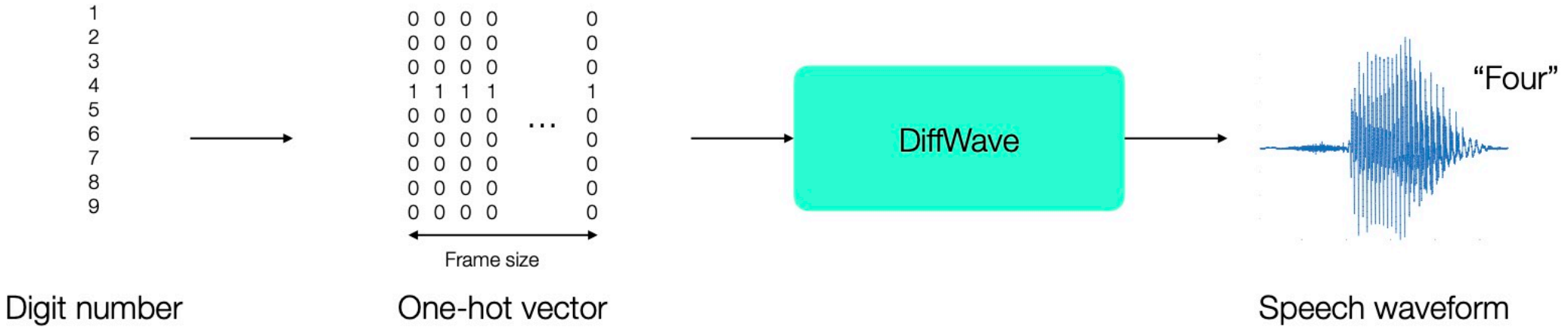
Unconditional and Semi-Conditional Models

- Unconditional models (initially investigated in WaveNet)
  - Synthesizing speech waveforms only with random sampling as image generative models
    - Clean human speech but no specific words



Network structure of unconditional WaveNet

- Semi-conditional models
  - Synthesizing speech waveforms conditioned on time-invariant information (e.g. digit number in DiffWave)



Open-source Implementations on GitHub

- Official implementations (PyTorch except for LPCNet)
  - LPCNet: https://github.com/mozilla/LPCNet (TensorFlow)
  - WaveGlow: https://github.com/NVIDIA/waveglow
  - HiFi-GAN: https://github.com/jik876/hifi-gan
  - VITS: https://github.com/jaywalnut310/vits
  - Quasi-Periodic Parallel WaveGAN: https://github.com/bigpon/QPPWG
  - Unified source-filter GAN: https://github.com/chomeyama/UnifiedSourceFilterGAN
  - MDDLP [38]: https://github.com/patrickltobing/cyclevae-vc-neuralvoco

- Unofficial implementations (PyTorch)
  - WaveNet with noise shaping: https://github.com/kan-bayashi/PytorchWaveNetVocoder
  - WaveNet with mixture of logistics (MoL): https://github.com/r9y9/wavenet_vocoder
  - WaveRNN with 9 bit μ-low: https://github.com/fatchord/WaveRNN
  - Parallel WaveGAN, MelGAN, Multi-band MelGAN, HiFi-GAN: https://github.com/kan-bayashi/ParallelWaveGAN
  - WaveGrad and DiffWave: https://github.com/lmnt-com
  - VITS in ESPnet2-TTS: https://github.com/espnet/espnet

Related Work Conducted in NICT

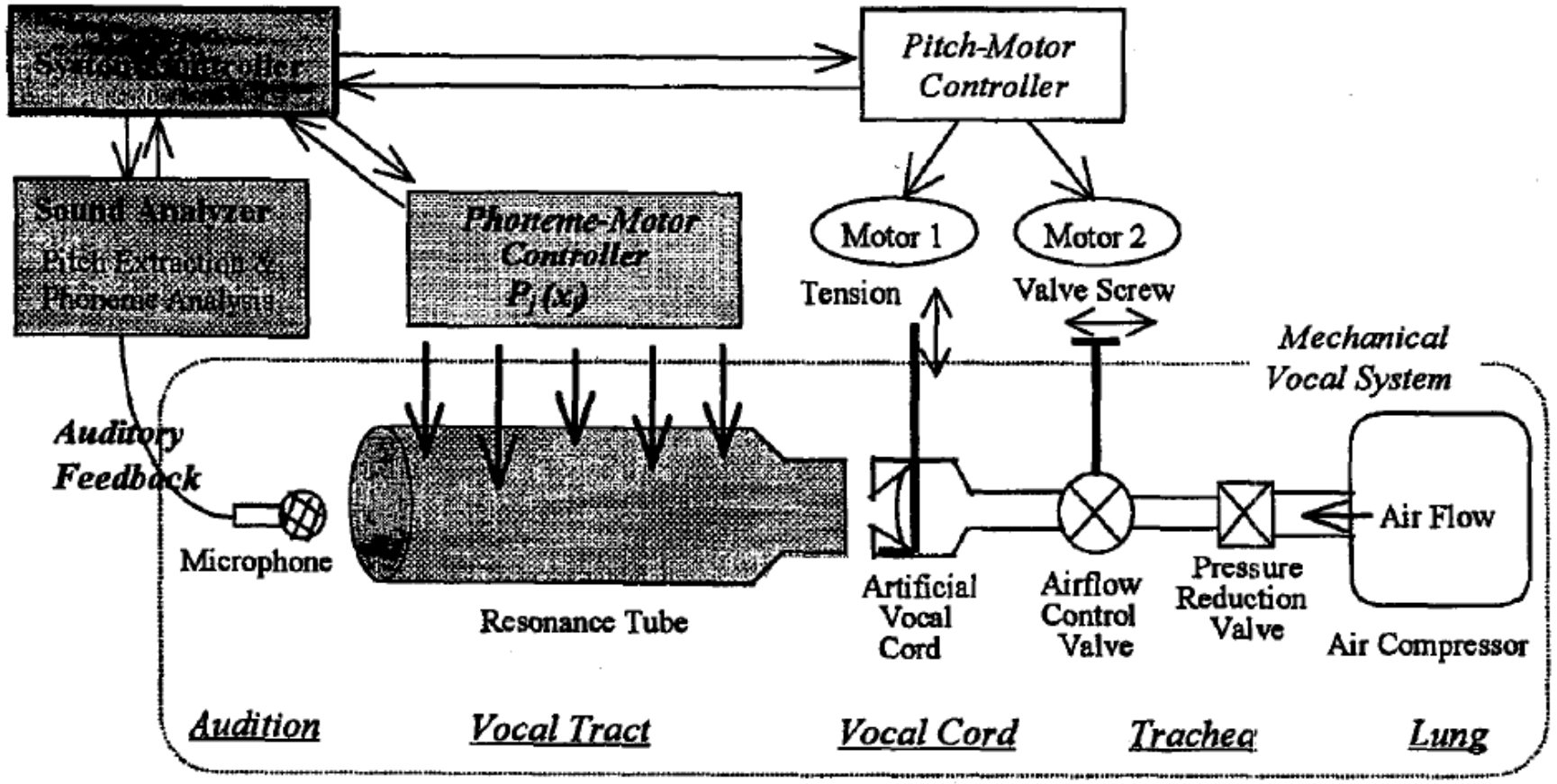
- Improving autoregressive models
  - Subband WaveNet (ASRU 2017, ICASSP 2018) and FFTNet (SLT 2018)
  - Single Gaussian FFTNet with noise shaping (ICASSP 2019)
  - Single Gaussian WaveRNN with noise shaping (Interspeech 2010)
  - Training data size for LPCNet (Acoust. Sci. Tech. 2021)
  - LPCNet-based text-to-speech and voice conversion (ICASSP 2021)
  - Full-band LPCNet (IEEE Access 2021)

- Real-time neural text-to-speech with WaveGlow
  - Stable acoustic models: BLSTM+Taco2Dec (ASRU 2019)
  - Transformer with weighted forced attention (ICASSP 2020)

- Improving diffusion probabilistic models
  - Noise level limited sub-modeling: DiffWaveGrad (ICASSP 2021)

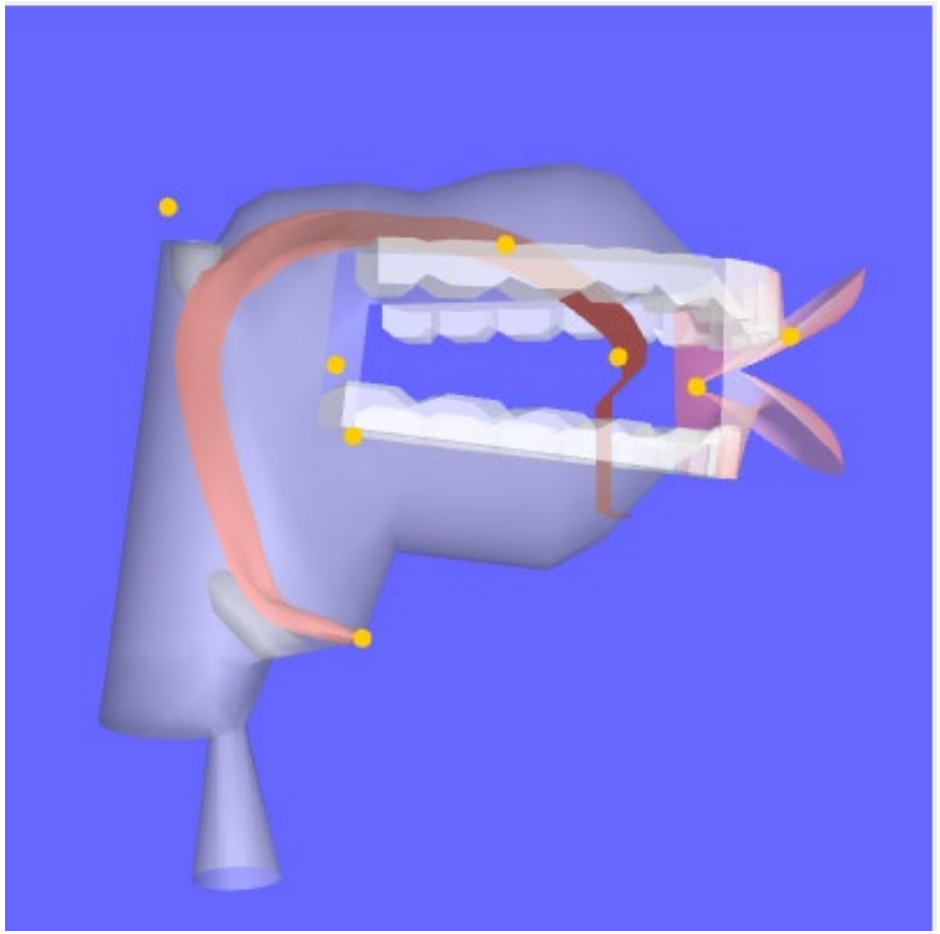
- Improving HiFi-GAN vocoder
  - Multi-stream HiFi-GAN (ASRU 2021)
  - HiFi-GAN with LPCNet features (Acoust. Sci. Tech. [accepted, in press])

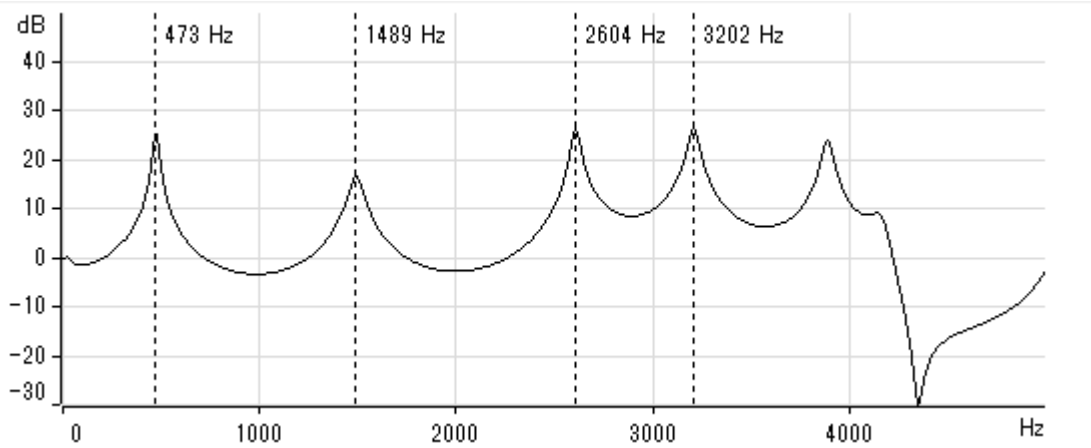
Physical Speech Synthesis



T. Higashimoto+, "Speech production by a mechanical model: Construction of a vocal tract and its control by neural network," Proc. EEE International Conference on Robotics and Automation, 2002

Vocal Tract Lab

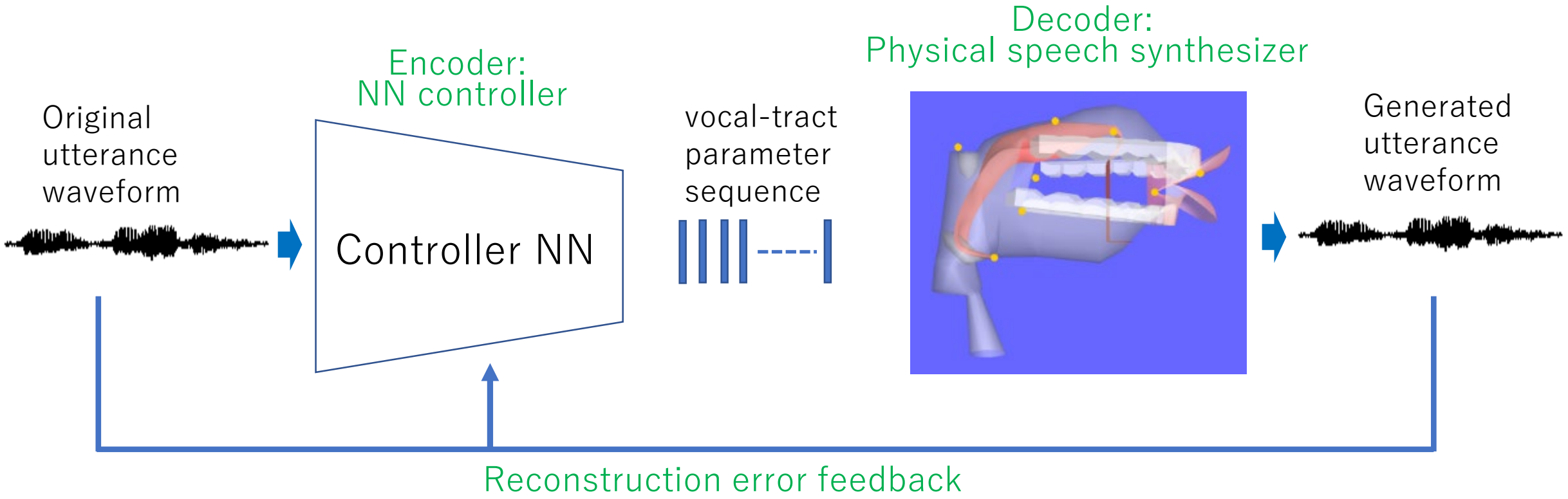




https://www.vocaltractlab.de/

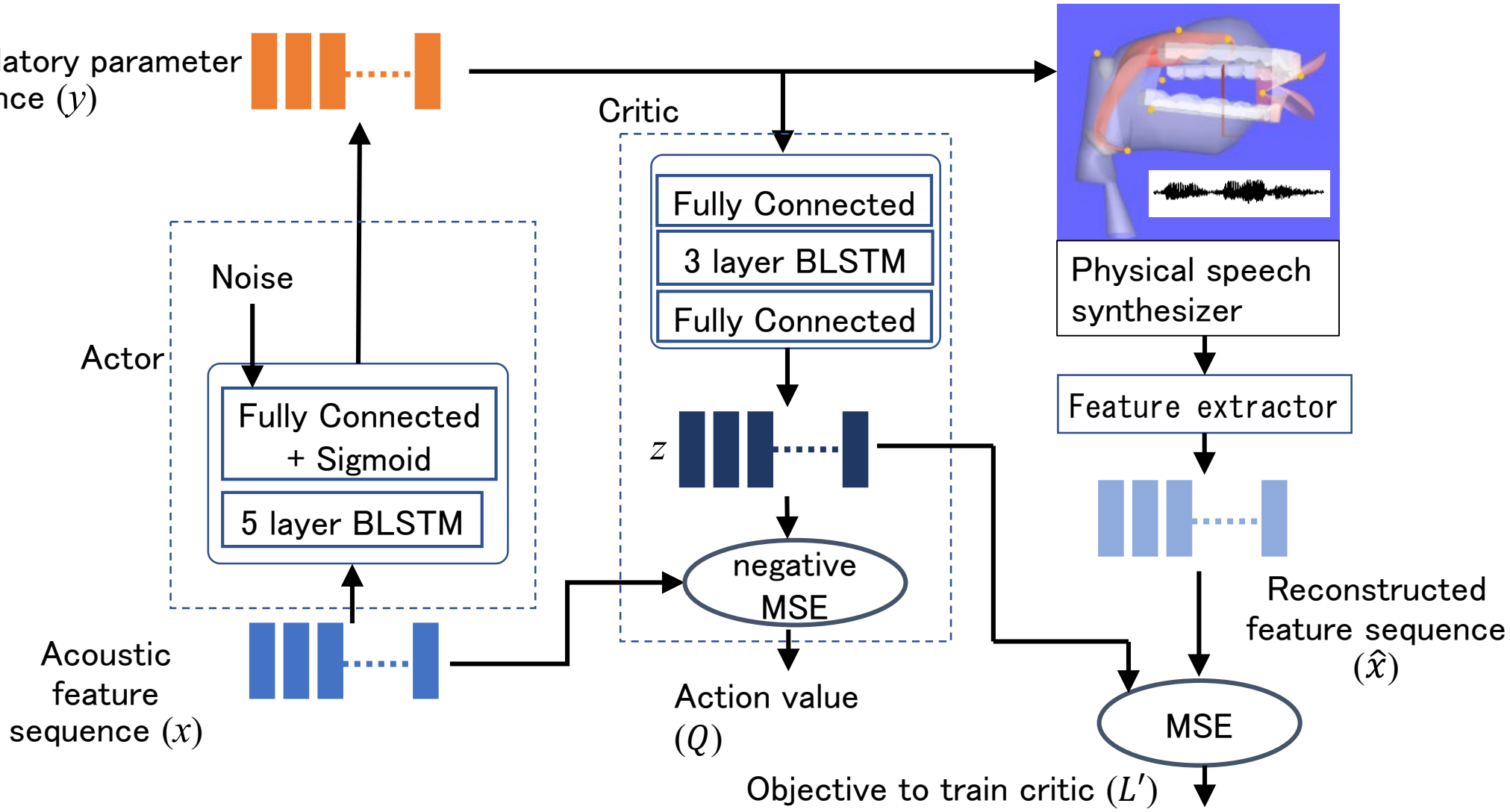
P. Birkholz P, "Modeling consonant-vowel coarticulation for articulatory speech synthesis," PLoS ONE, 2013

Hybrid Auto-Encoder Based Control Learning



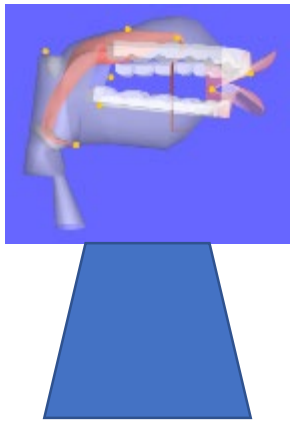
H. Shibata+, "Unsupervised Acoustic-To-Articulatory Inversion Neural Network Learning Based on Deterministic Policy Gradient," IEEE Spoken Language Technology, 2021

Deterministic Policy Gradient Based Learning



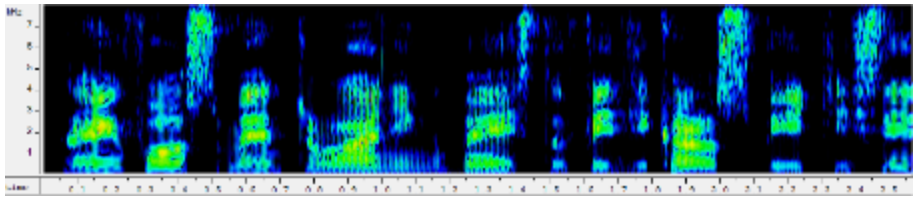
Resynthesis Result

Train the system using
Google Speech Commands
(English data)

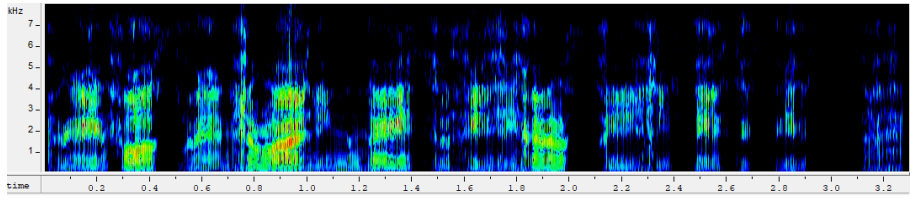


Input an English utterance

Reinforcement learning based physical speech synthesis



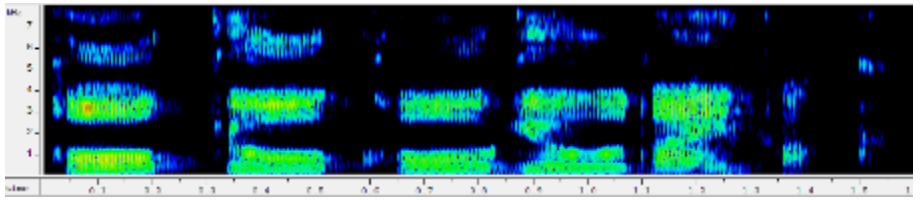
Original 



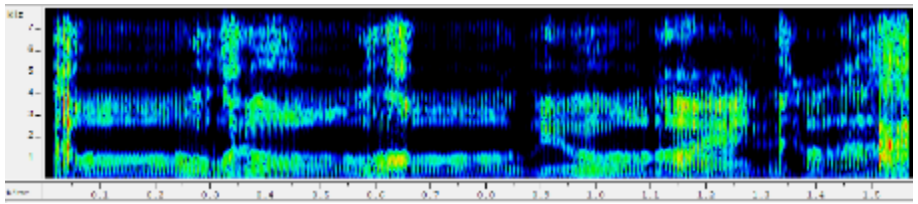
Re-generated 

Input a Japanese utterance

Tokyo kogyo daigaku



Original 



Re-generated 

Task Oriented Dialogue Systems

Traditional Dialogue System Structure

Task oriented dialogue systems typically consists of three modules: Natural Language Understanding (NLU), Dialogue Manager (DM), and Natural Language Generation (NLG)

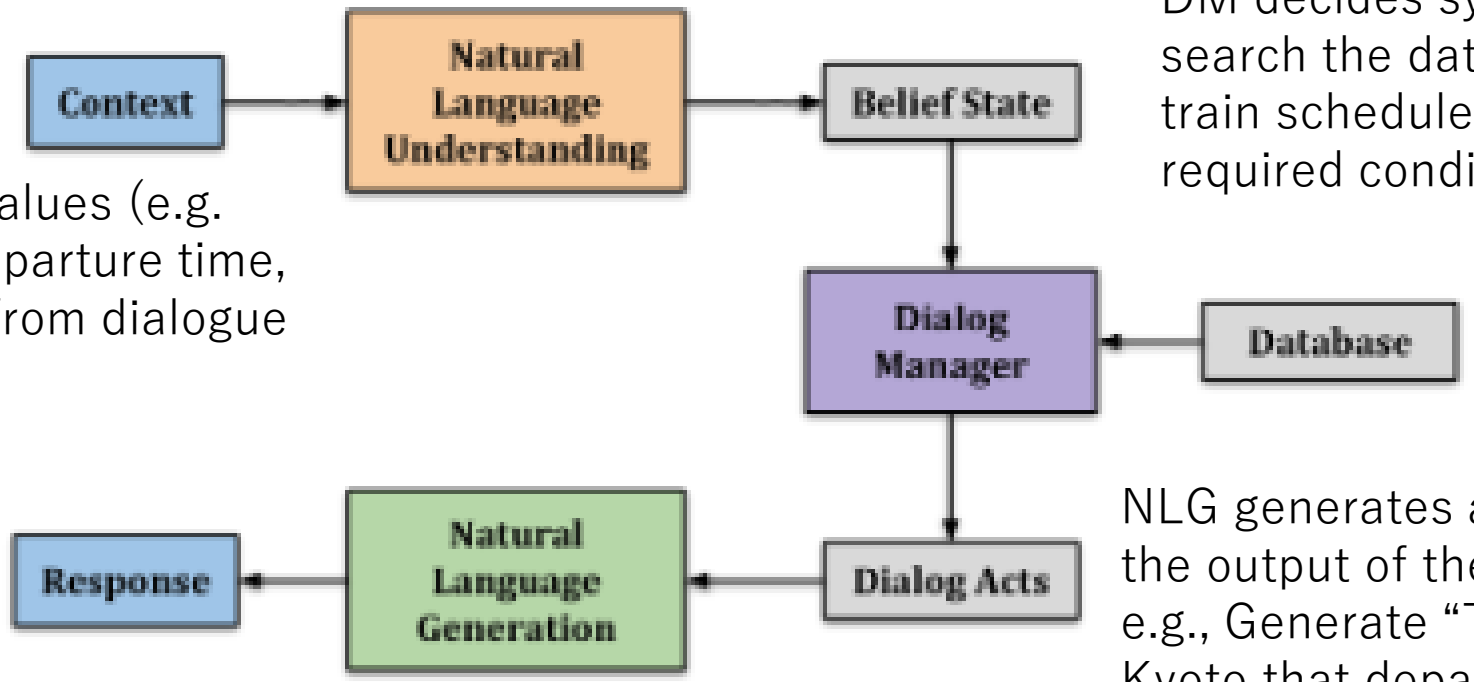


Figure is cited from [1]

DM decides system action (e.g. search the database and find train schedule that meet the required condition)

NLU extracts slot values (e.g. current position, departure time, destination place) from dialogue history

NLG generates a sentence from the output of the dialogue manager. e.g., Generate “There is a train to Kyoto that departs Tokyo at 13:30” from <Tokyo, Kyoto, 13:30>

[1] S. Mehri+, "Structured Fusion Networks for Dialog," arXiv 2019

Training of Task Oriented Dialogue Systems

- Typically, the three modules are separately optimized by supervised training
- When applying reinforcement learning to DM, usually a fixed NLU is assumed that has been trained. In that condition, NLU plays a role in providing observation state to DM that satisfies the Markov property, gathering required information from the dialogue history
- If the observation state does not have complete information and DM directly needs to consider history, POMDP is needed
- When the state transition probability (i.e. model) is available, belief state is obtained by Bayes inference, and the problem is converted to MDP
- Researches on those topics such as model-free POMDP and sequence generating reinforcement learning from scratch combined with unsupervised pre-training are challenging

End-to-End Task Oriented Dialogue System with RL

A research working on
RL + pre-training

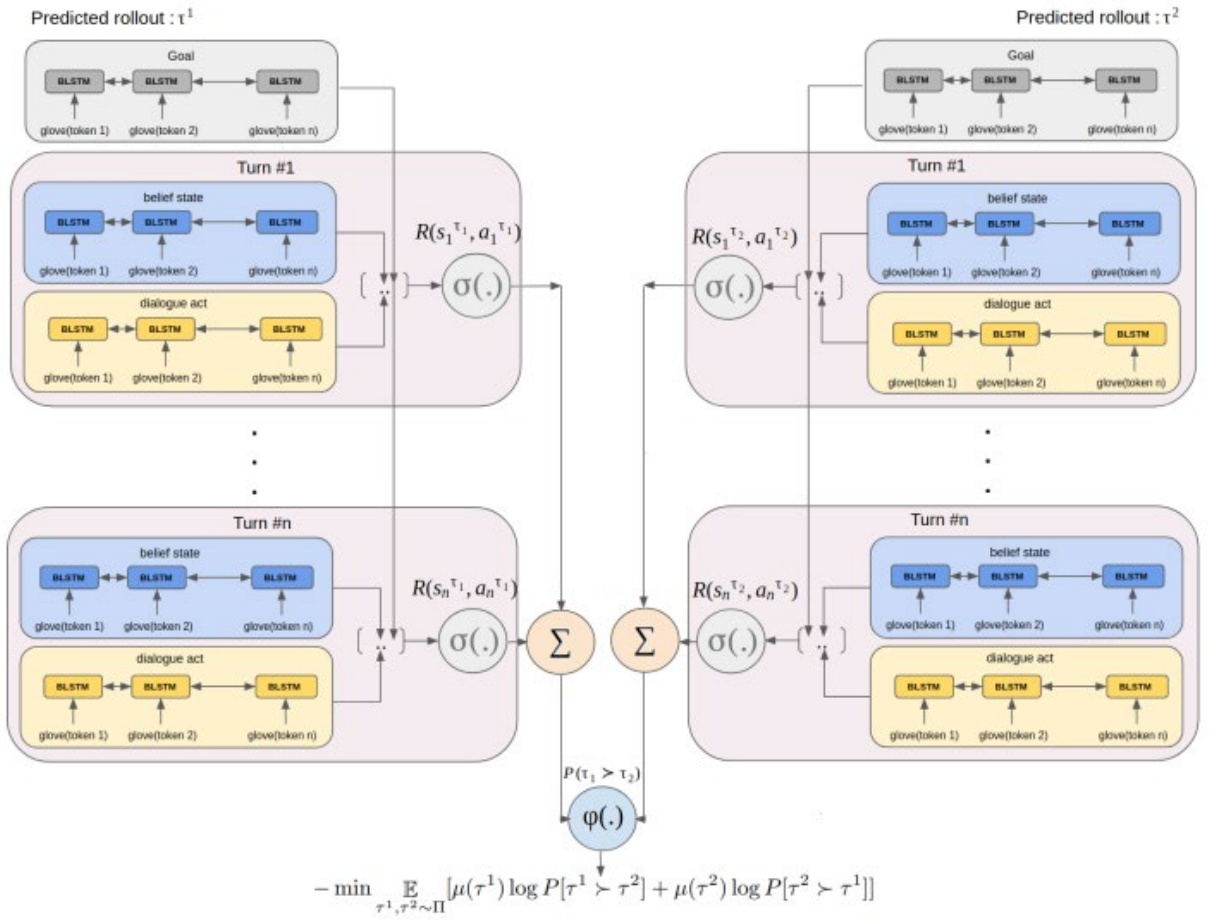


Figure is cited from [1]

[1] G. Ramachandran+, "Causal-aware Safe Policy Improvement for Task-oriented dialogue," arxiv 2021

PART 4

Agent System (1)

Uses Unsupervised Sound Dictionary Based Action Space

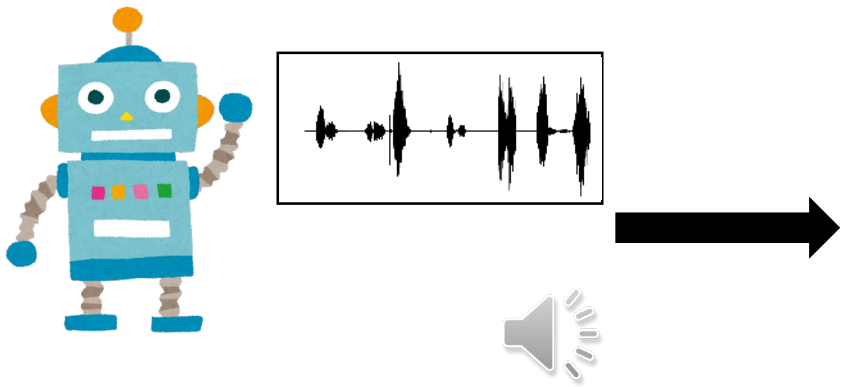
① Word discovery
② Grounding
③ Action learning
④ Pronunciation learning

Gao+, "Spoken Language Acquisition Based on Reinforcement Learning and Word Unit Segmentation," Proc. ICASSP2020, 2020

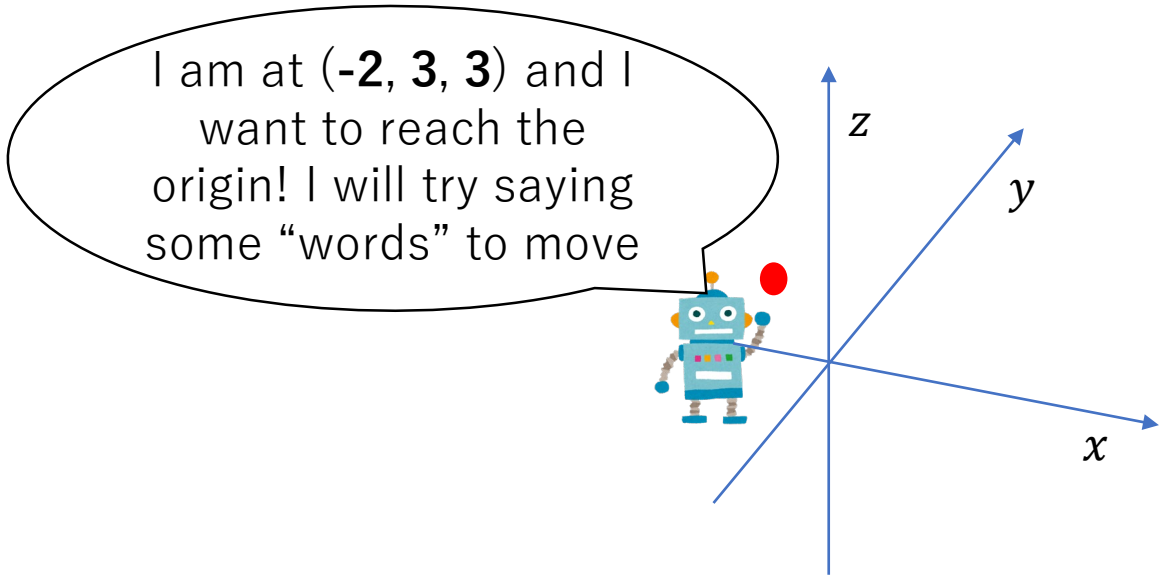
Task Setting

1. Observation phase
Agent observes unsegmented spoken sound examples

2. Dialogue phase
By pronouncing a command (up, down, left, right, forward, backward), agent can move one step to that direction at a time



Unsegmented sound sample



Initially, agent has no language knowledge

Agent is randomly positioned at a 3-D point (x, y, z) and wants to reach the origin

Unsupervised Sound Dictionary Learning

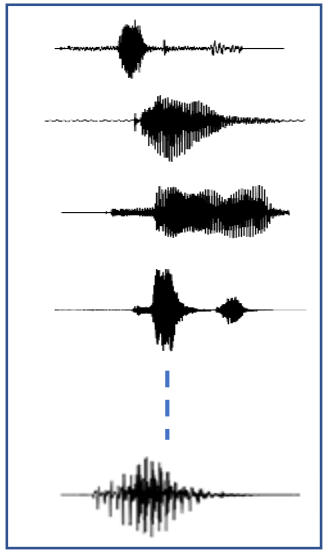
- Makes sound dictionary using ES-KMeans using the sound data obtained in the observation phase
- Use the dictionary as the action space for RL

Continuous sound

ES-KMeans

Unsupervised word segmentation

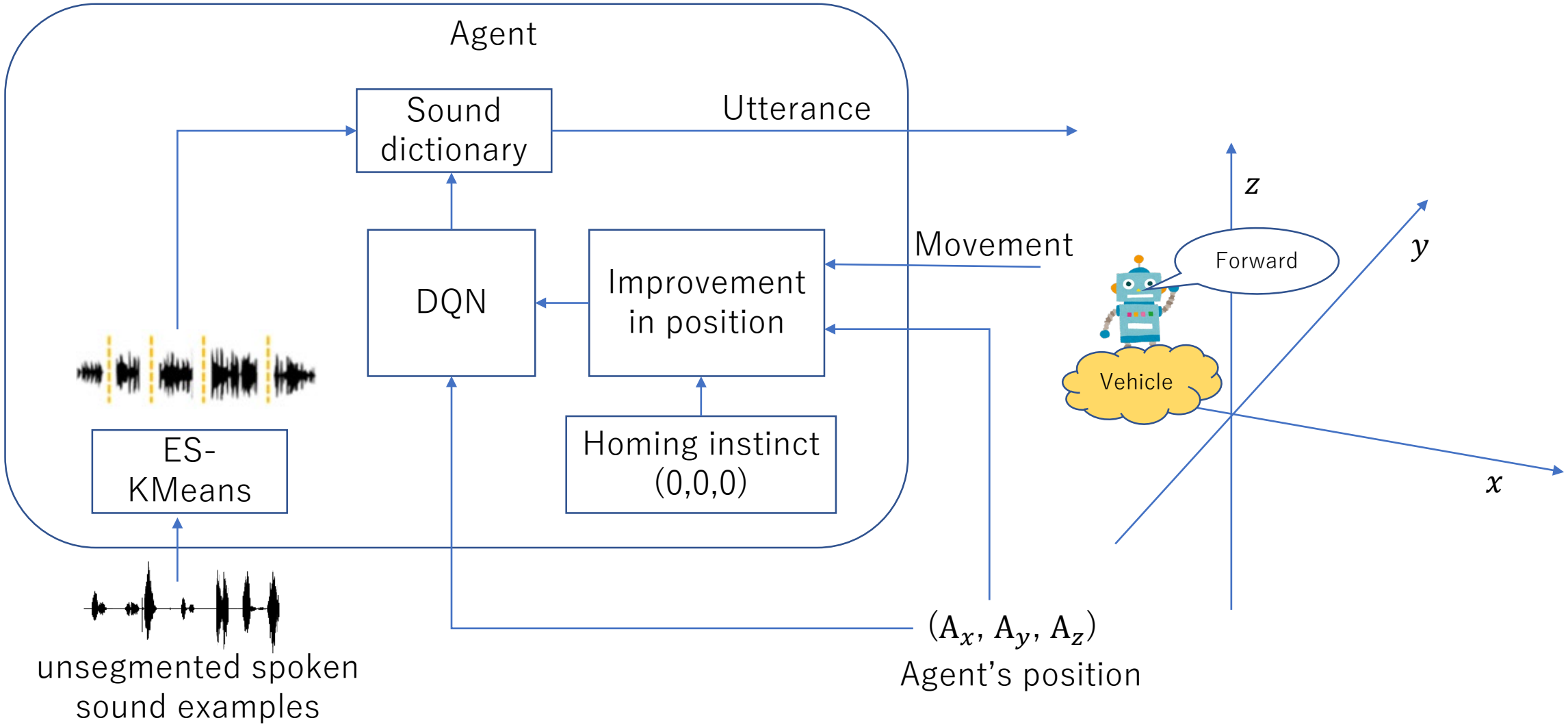
(Prototype)



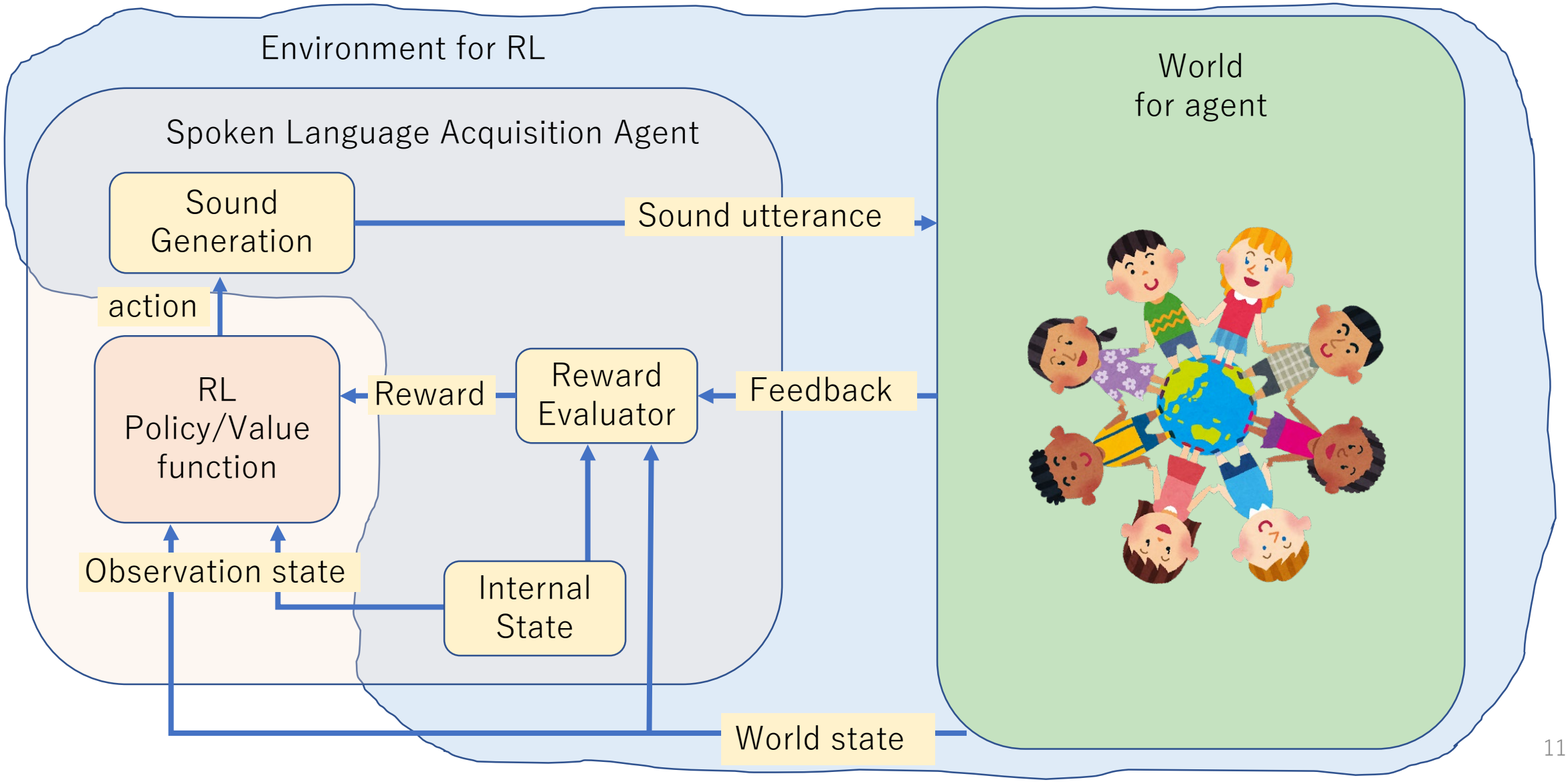
Sound dictionary

- ➢ The dictionary (i.e., action space) is open and learned entirely from scratch
- ➢ While the dictionary contains many broken segments, RL gradually finds useful segments for conversation

Utterance Learning by RL



World for Agent and Environment for RL



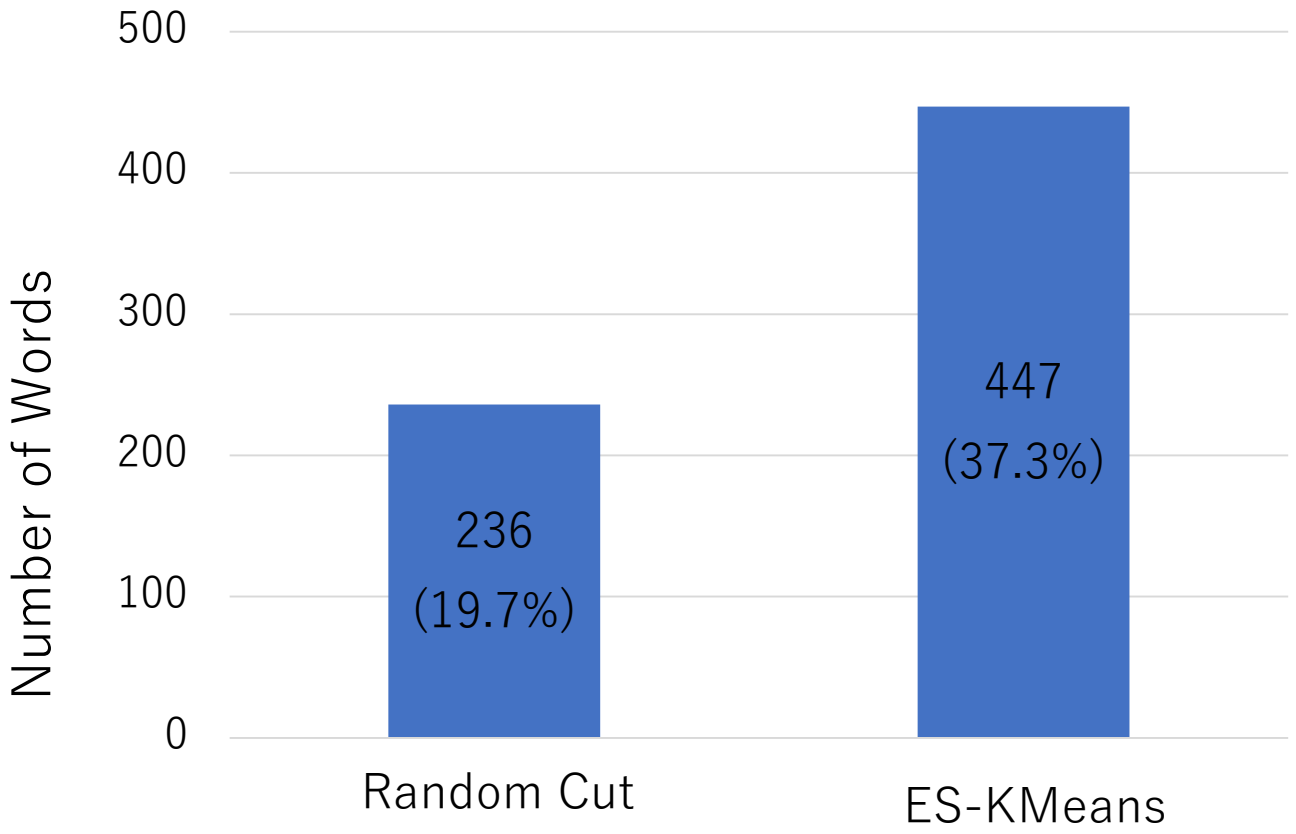
Experimental Setup

- **Dataset**: Google Speech Command
- **Preprocessing**: 200 samples are randomly picked from 6 types of command words plus a noise word “Marvin” and concatenated into a continuous speech
- **Environment module**: Google Speech-to-Text API [1], a general-purpose ASR system

Total number of one-second utterances	65,000
Vocabulary (of the corpus)	35
Number of command word types	6 (up, down, left, right, forward, backward)

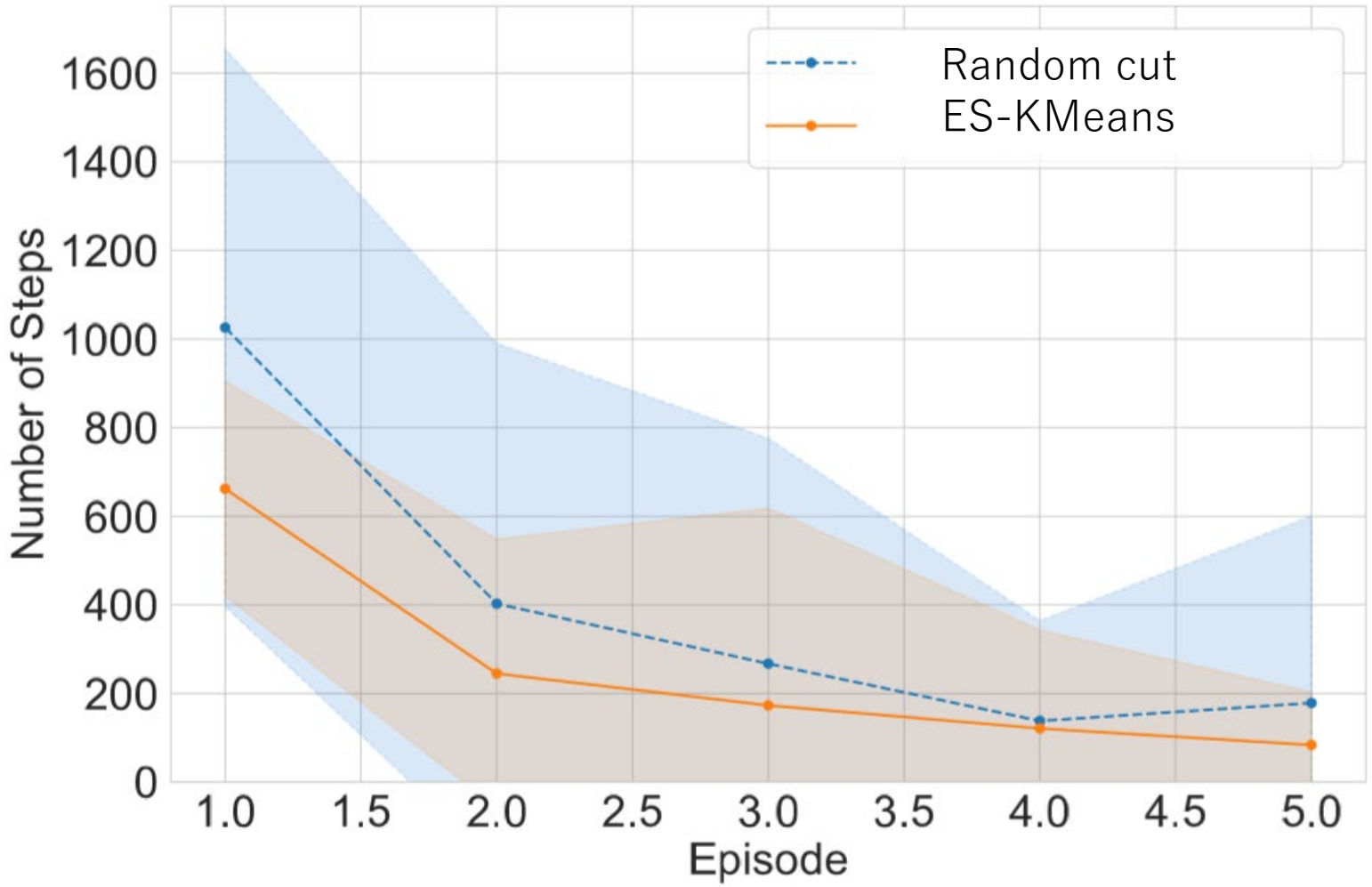
[1] https://cloud.google.com/speech-to-text/docs/reference/rest/

Useful Segment Ratio in Sound Dictionary



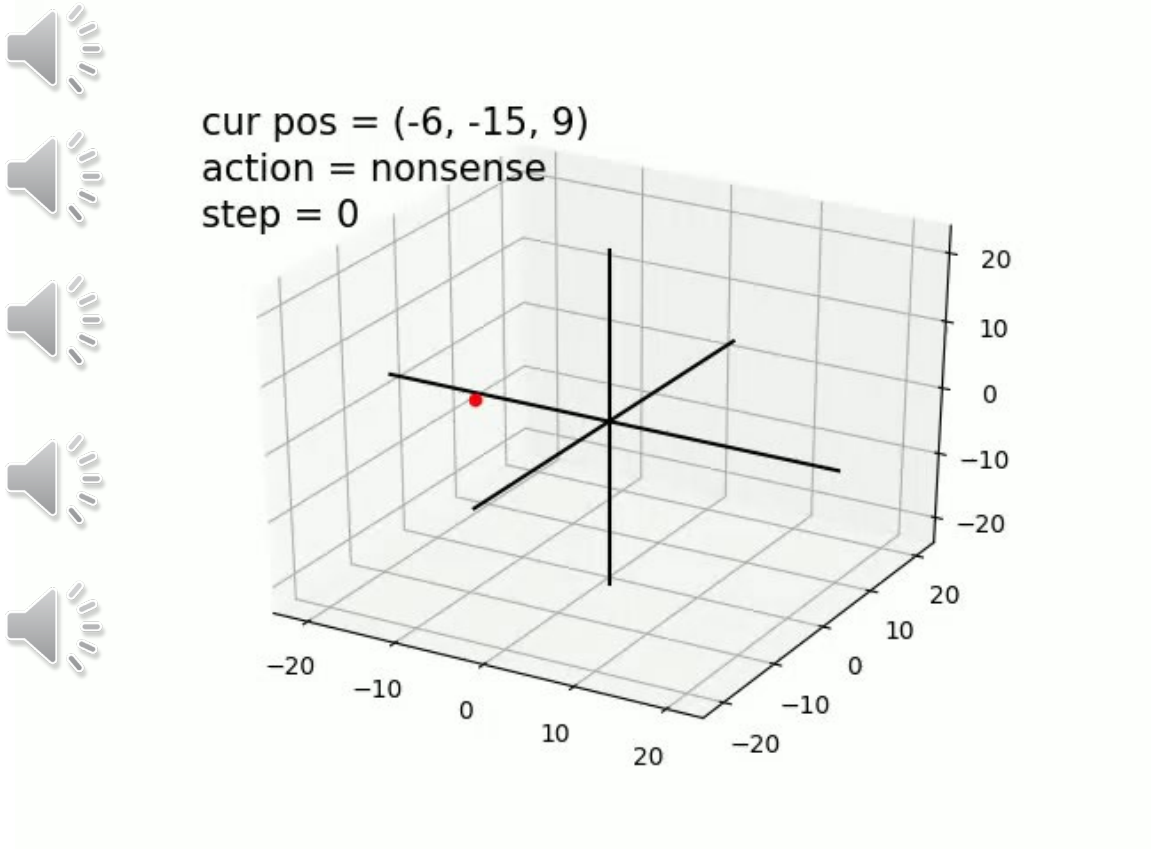
Proportion of segmented words recognized by the environment among 1200 command words

Result


Random cut
ES-KMeans
Number of Steps
0
200
400
600
800
1000
1200
1400
1600
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Episode

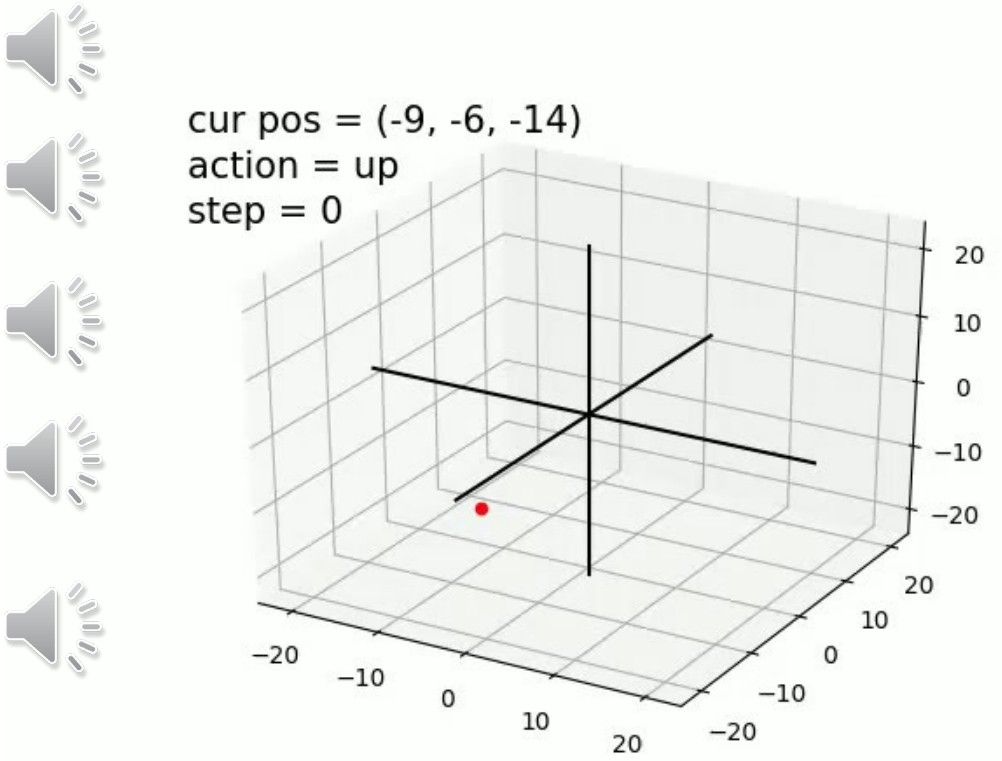
Demo

First five actions at episode 0



Moving process at episode 0

First five actions at episode 5



Moving process at episode 5

Agent System (2)

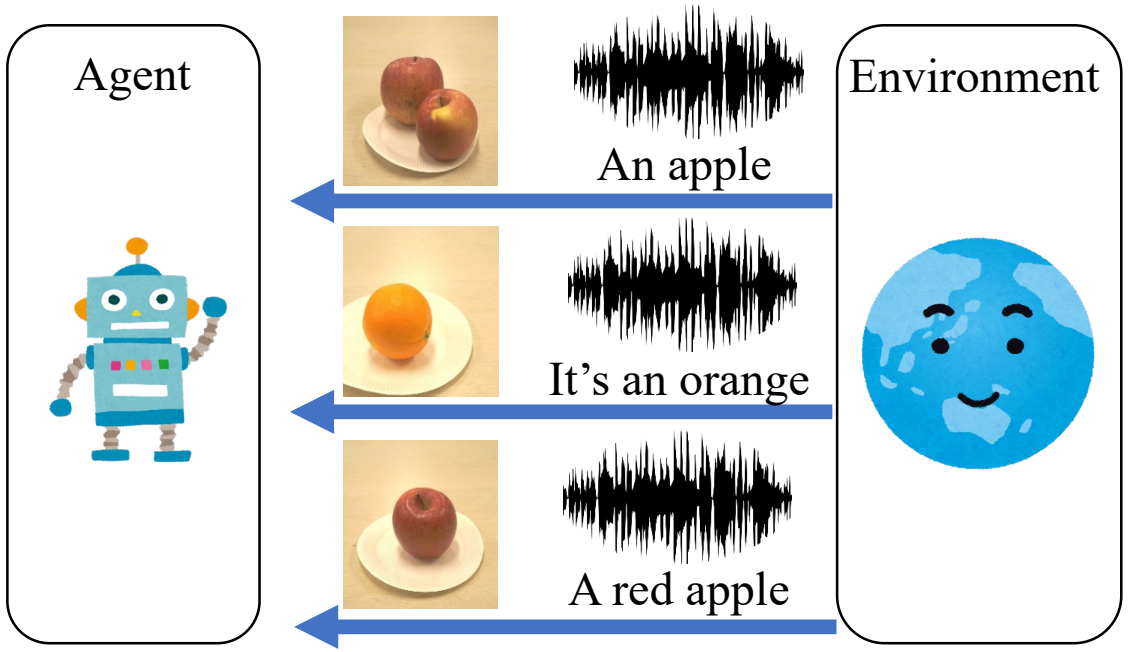
Introduces sound-image grounding based focusing mechanism to improve the learning efficiency

① Word discovery
② Grounding
③ Action learning
④ Pronunciation learning

Zhang+, "Sound-Image Grounding Based Focusing Mechanism for Efficient Automatic Spoken Language Acquisition," Proc. Interspeech, 2020
Toyoda+, "Self-Supervised Spoken Question Understanding and Speaking With Automatic Vocabulary Learning,“ Proc. O-COCOSDA 2021

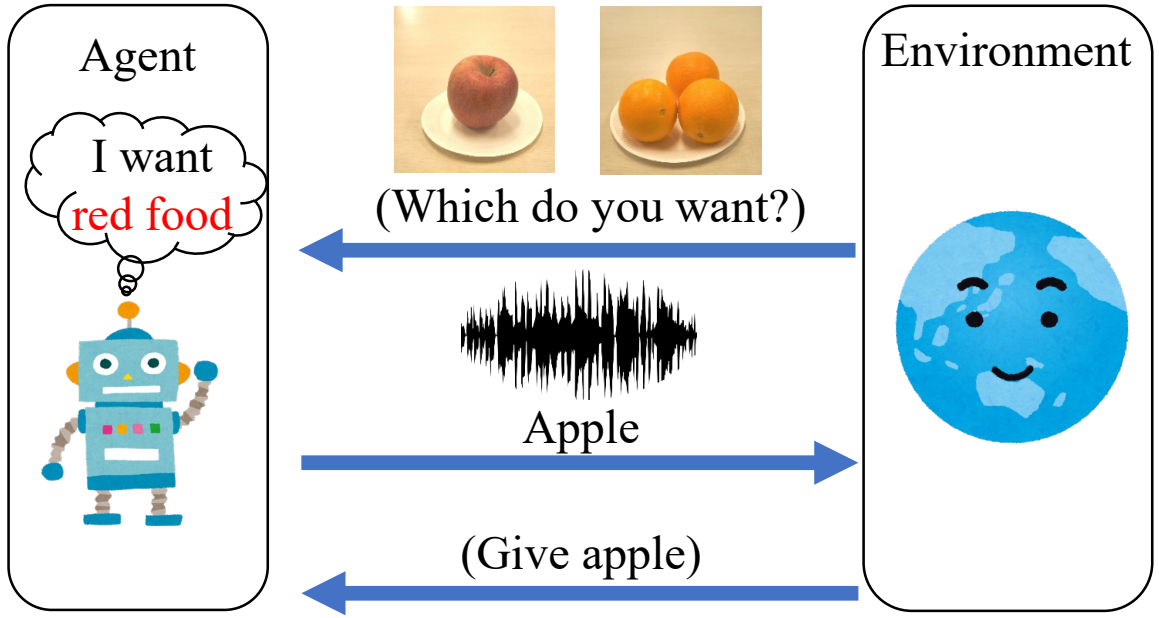
Task Setting

1. Observation phase
Agent observes 1) continuous sound, and 2) a food photo and its sound description



Initially, agent has no language knowledge

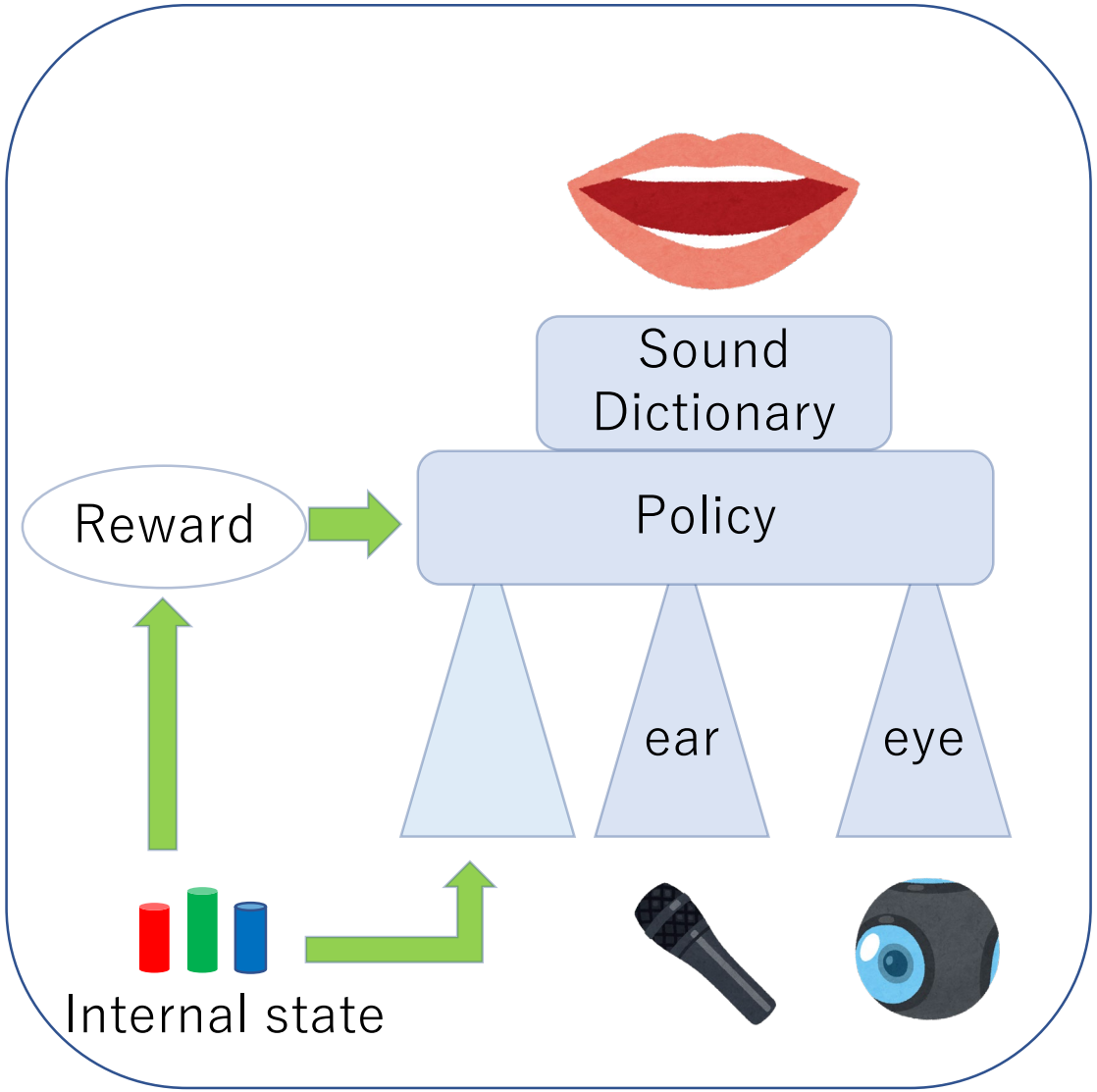
2. Dialogue phase
By pronouncing one of two indicated food names, agent can get it



Agent has random preference of food color and becomes happy if it gets the desired food.
Environment give the food if it recognizes the name

Basic Agent Design

- Basic structure is the same as Gao+'s system adding sound and image inputs
- The internal state is color preference instead of the homing instinct



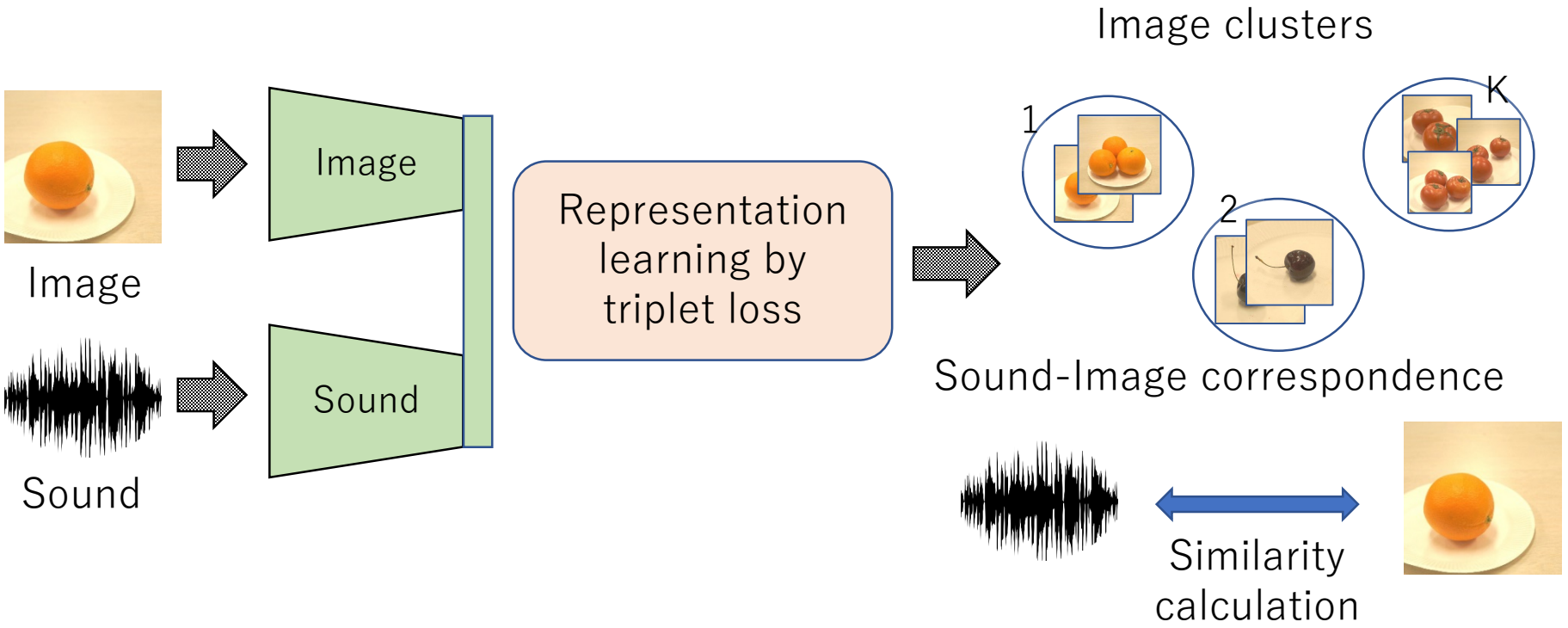
Problem:

As the agent's vocabulary increases, RL learning becomes difficult, especially at the beginning since the random action rarely succeed

Solution:

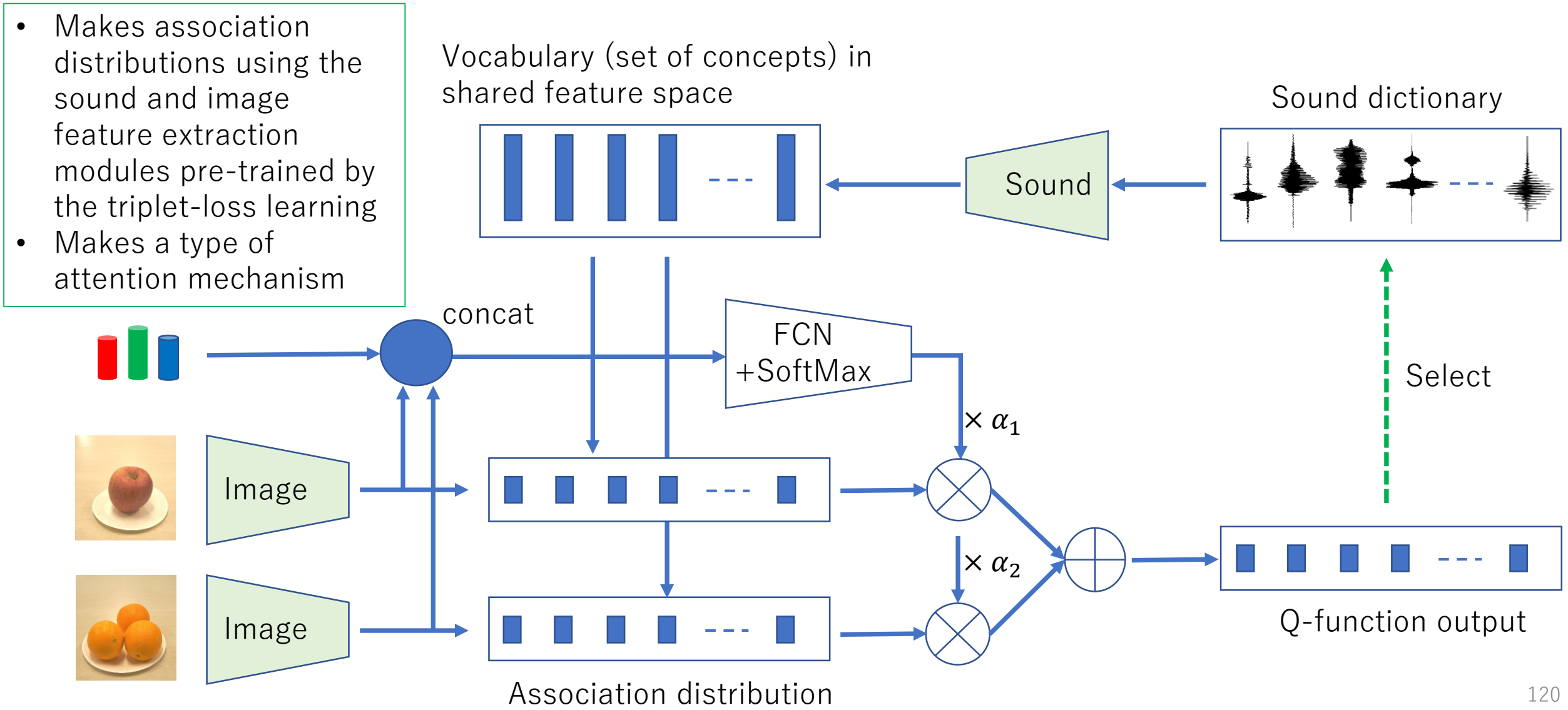
Improve the learning efficiency by guide agent's attention to what is in sight with the help of image-sound grounding

Sound-Image Grounding and Concept Clustering



Introduce a Focusing Mechanism to Q-function

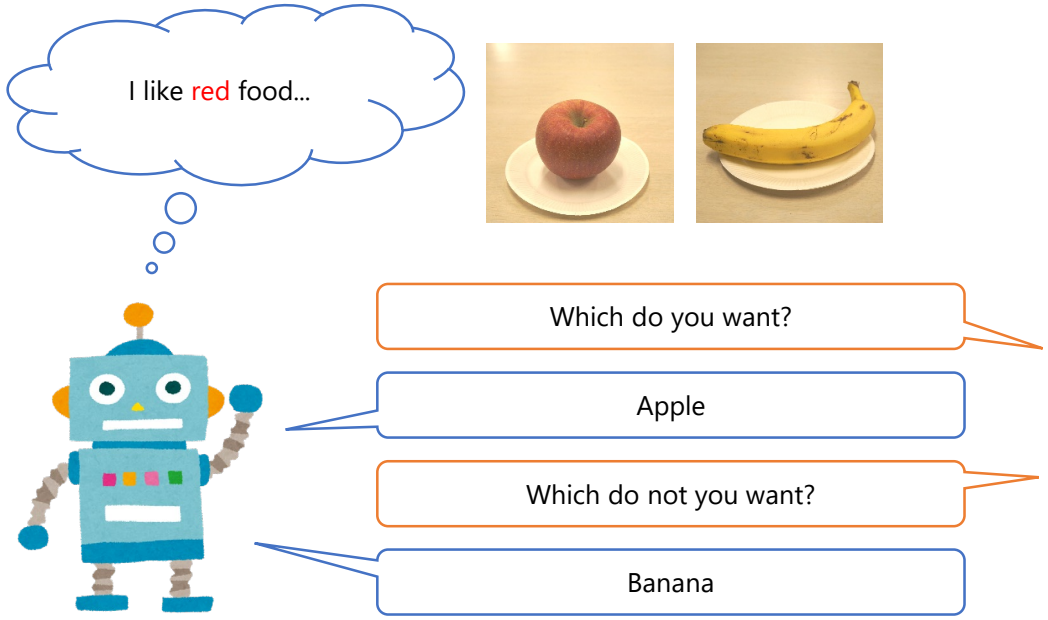
- Makes association distributions using the sound and image feature extraction modules pre-trained by the triplet-loss learning
- Makes a type of attention mechanism



Dialogue Scenario

Two Questions
- “Which do you want?”
- “Which do not you want?”

- If the question is the former, the agent gets the favorite food when it answers the name
- If the question is the latter, the agent receives the favorite food when it answers the name of the opposite one



Dataset

- Food images

  cherry, green pepper, lemon, orange, potato,
  strawberry, sweet potato, tomato

- Speech description –We generated synthetic sounds using Google text to speech
  - “<food name>”
  - “a/an <food name>”
  - “a <food color><food name>”
  - “it’s a/an <food name>”

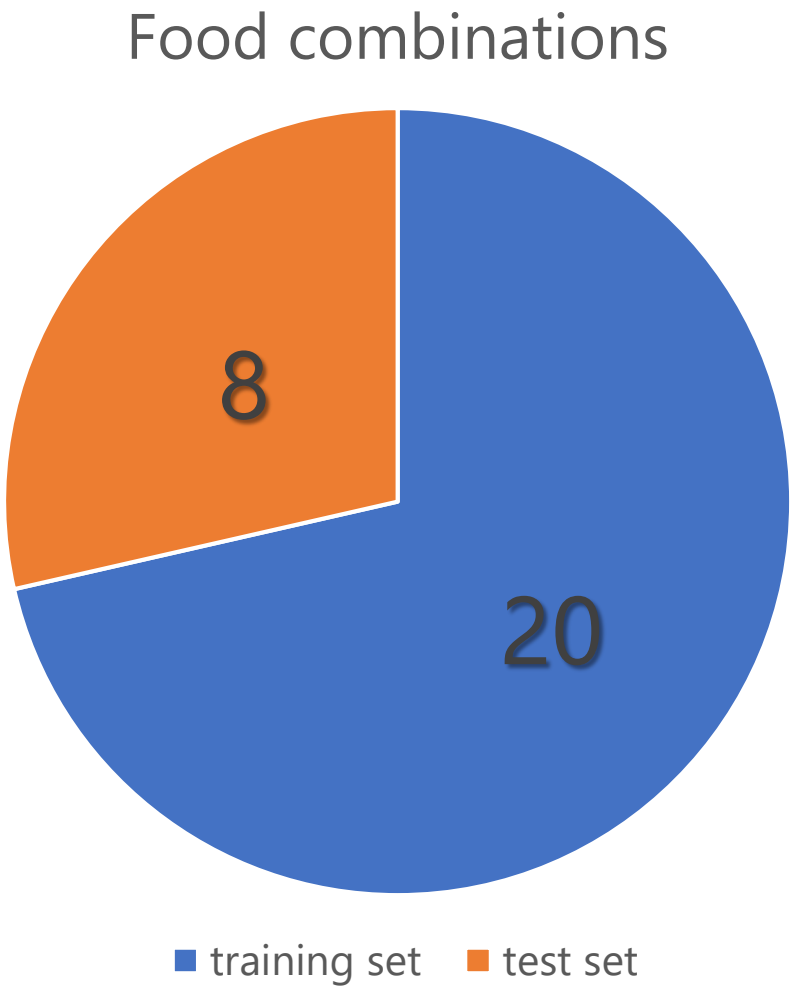
Question Types

Exp.	Input Question
1	Which do you want? / Which do not you want? *1
2	Which do you want?
3	Which do not you want?
4	No question utterance input (Which do you want) *2
5	No question utterance input (Which do not you want) *2

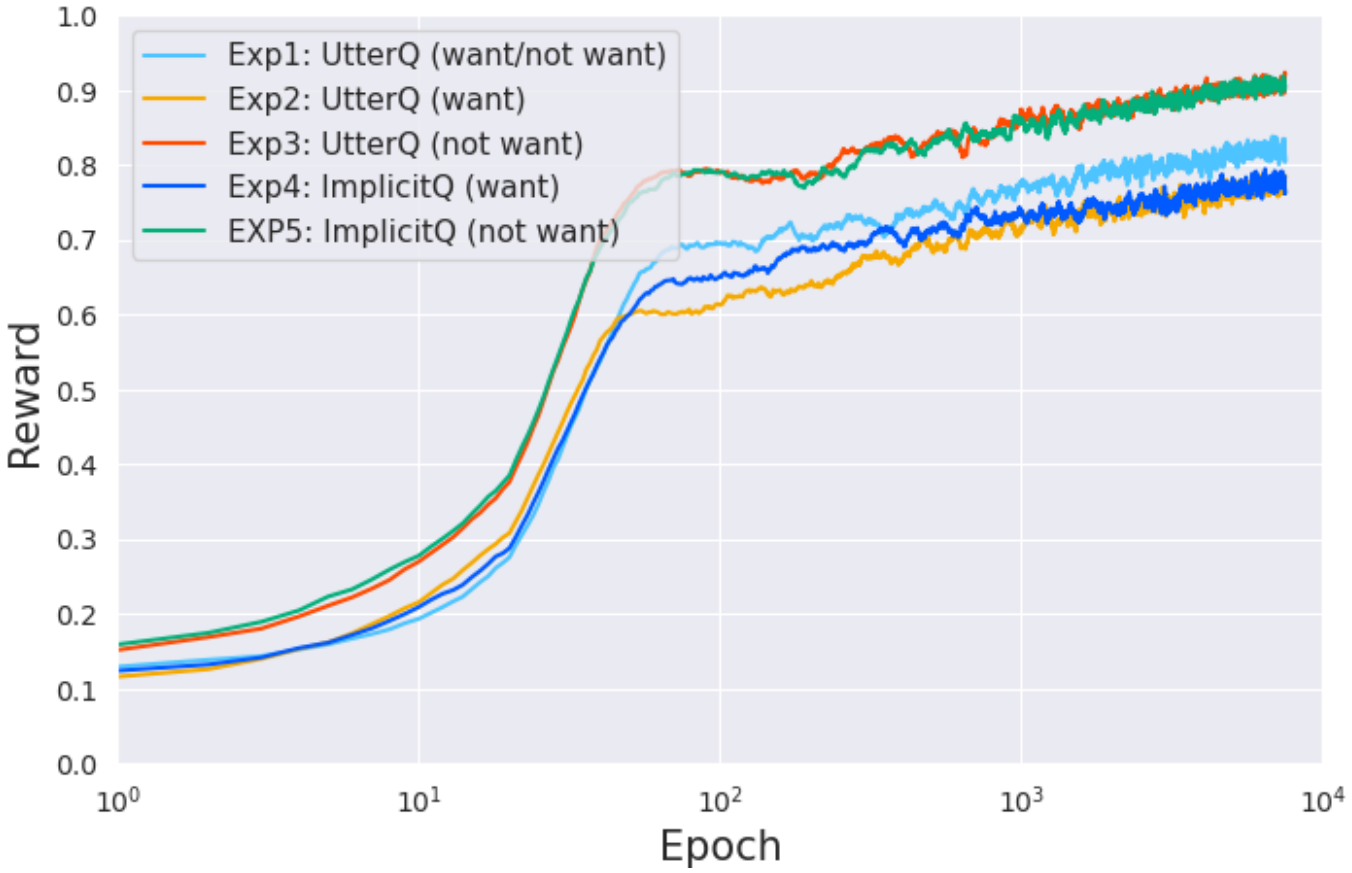
*1) randomly input one of two questions as a waveform utterance
*2) no input question utterance. Implicitly assumes a single question

Train & Testing Partitioning

- There are ${}_8C_2 = 28$ food combinations
  - Among them, we use 20 for the training and 8 for the testing
  - In the test phase, the agent need to infer the answer based on the understanding of logical “not” for the unseen input image combinations



Result



- When asking ”which do you want” explicitly (exp2) or implicitly (exp4), the rewards were mostly the same
- When asking ”which do not you want” explicitly (exp3) or implicitly (exp5), the rewards were mostly the same
- When asking two questions, the rewards were somewhere between that of exp3,5 and exp2.4. The difference of the positive/negative questions is due to the biased distribution of food colors

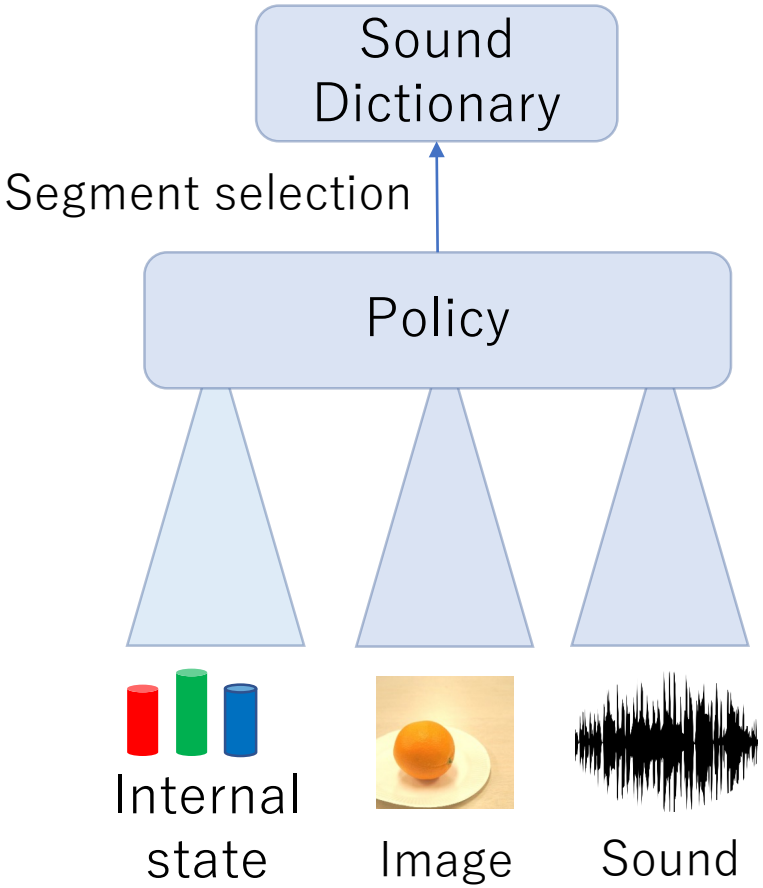
Agent is successfully discriminating the two questions

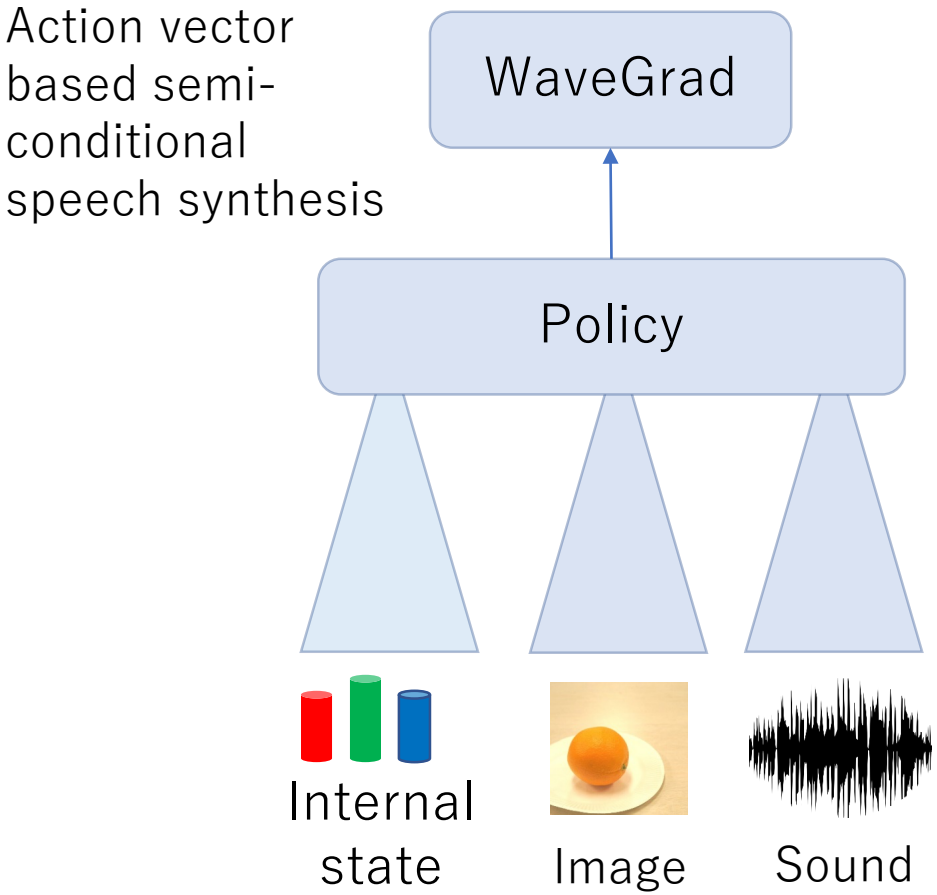
Agent System (3)

Replaces sound dictionary with neural vocoder for flexible utterance pronunciation

① Word discovery
② Grounding
③ Action learning
④ Pronunciation learning

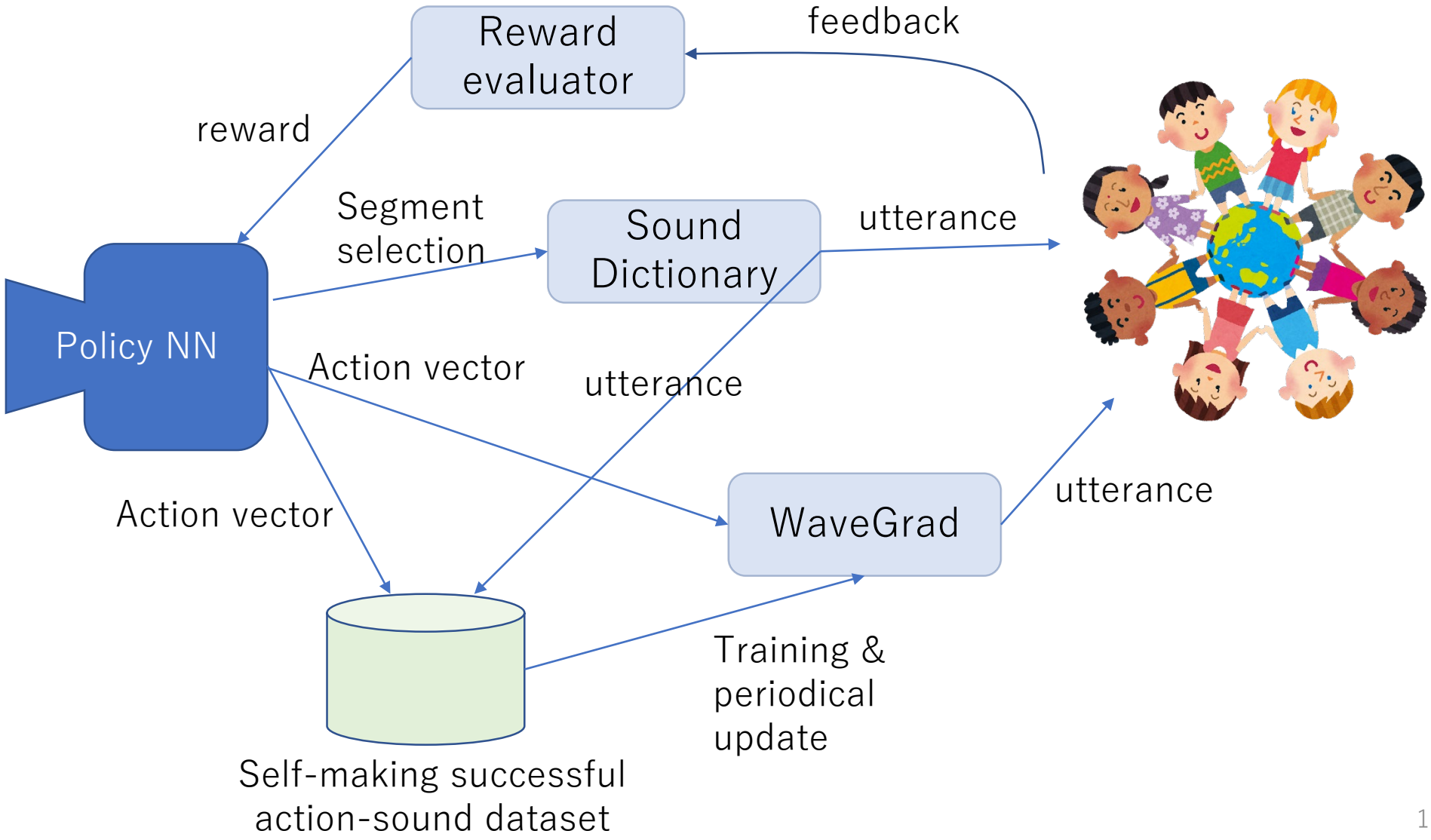
Uses WaveGrad as Speech Organ



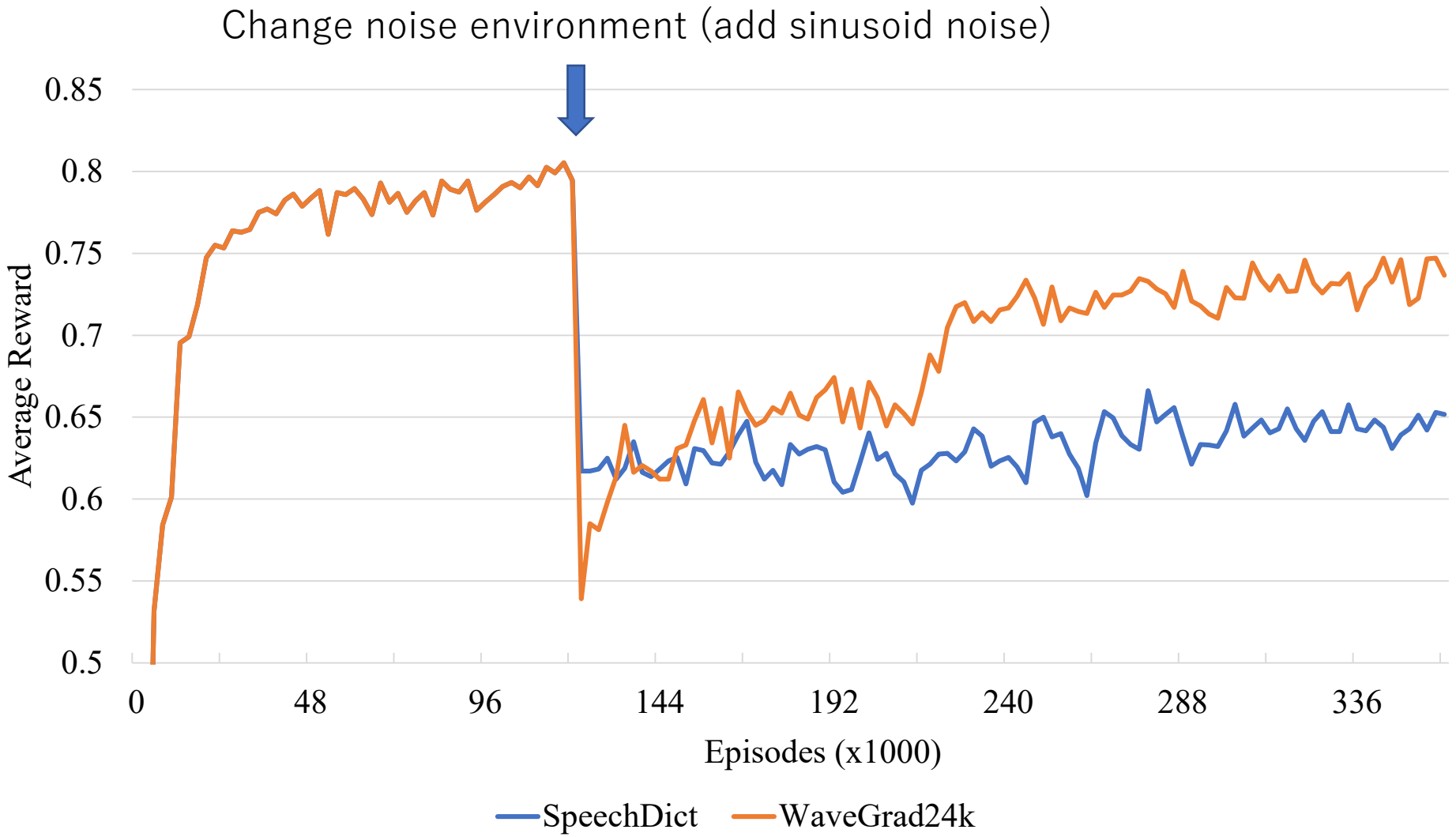


Self-supervised WaveGrad Training

1. Agent initially uses the sound dictionary approach and accumulate action-sound dataset through dialogue
2. After accumulating enough data, train WaveGrad and switches to it



Result

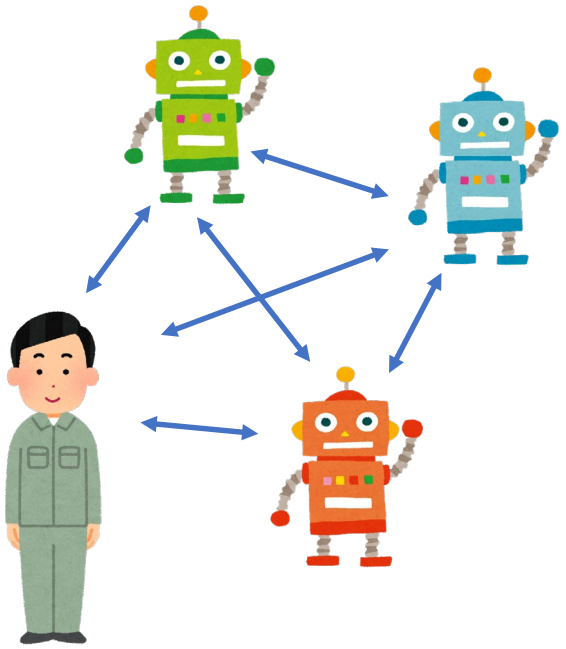

Change noise environment (add sinusoid noise)
Average Reward
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
48
96
144
192
240
288
336
Episodes (x1000)
SpeechDict
WaveGrad24k

Future Directions

Open Tasks

- Expansion of the learnable vocabulary size, and support for multi-word utterances
- Support for video stream
- Curriculum learning
- Implementation of emotion
- Online parallel learning where an agent communicates with a large number of people
- Multi-agent system where agents communicate each other for faster language learning
- Agent with physicality

Full of opportunities for challenges!



PART 5

Toolkit Introduction

RL Libraries

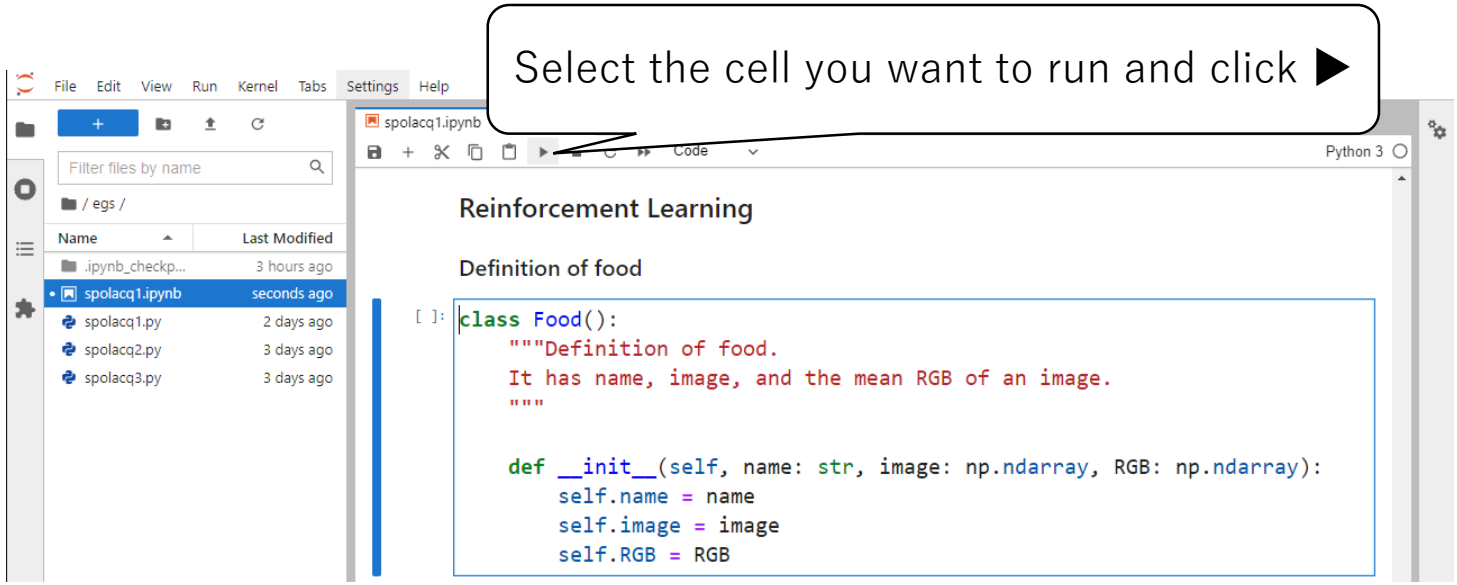
- RL environment: **OpenAI Gym** [1]
  - Easy to use
  - Provide many environments

- RL algorithm: **Stable Baselines3** [2]
  - Custom Policy Network using PyTorch
  - Training RL model **in a line**
  - Easy to save and load the entire model
  - Tensorboard support

[1] G. Brockman+, "OpenAI Gym," arXiv, 2016
[2] A. Raffin+, "Stable-Baselines3: Reliable Reinforcement Learning Implementations," JMLR, 2021

Setup environment

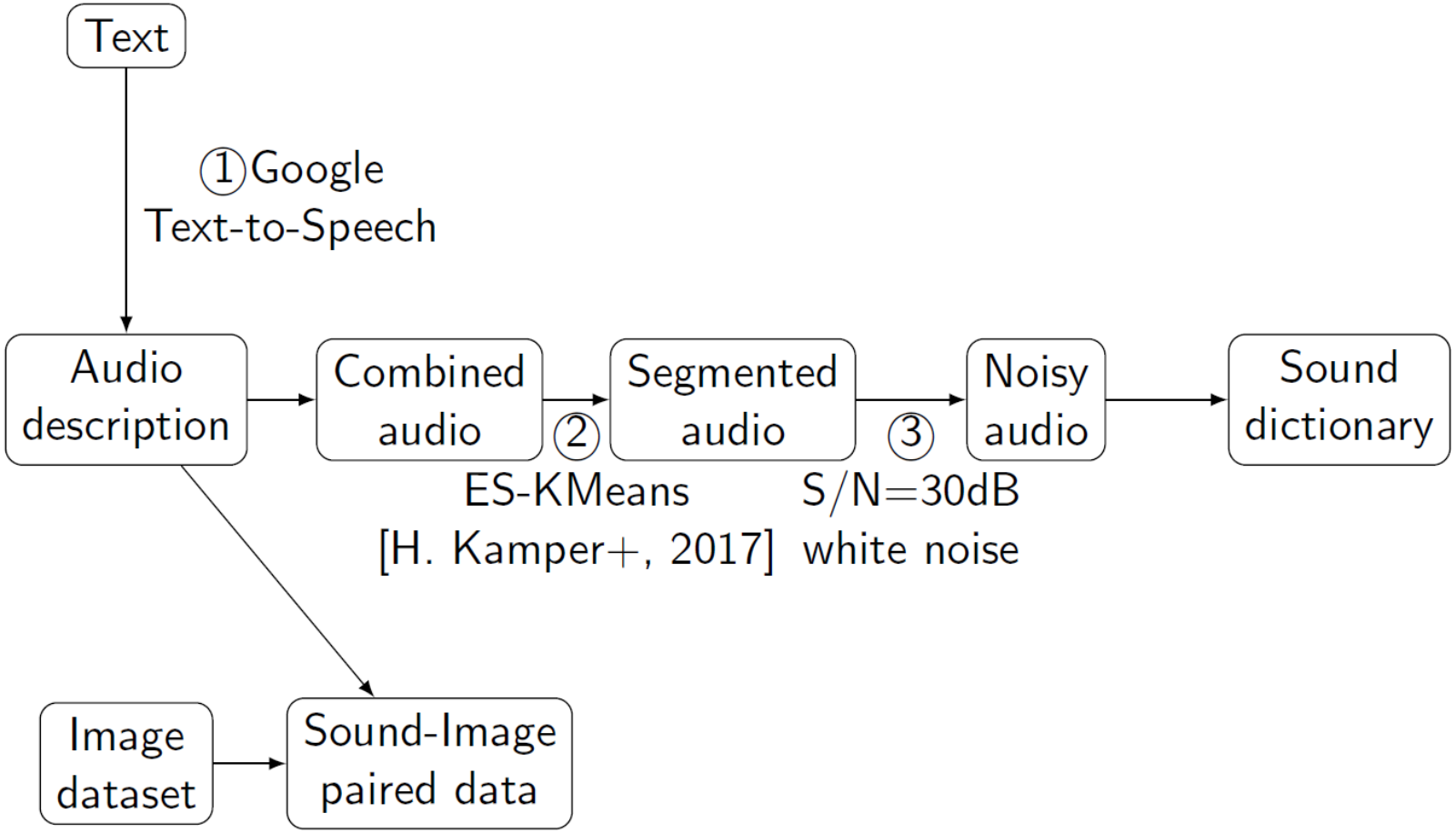
- $ git clone https://github.com/tttslab/spolacq.git
- $ cd spolacq
- $ conda env create -f=spolacq.yml
- $ conda activate spolacq
- $ jupyter lab --no-browser --port=1XXXX



Directory Structure

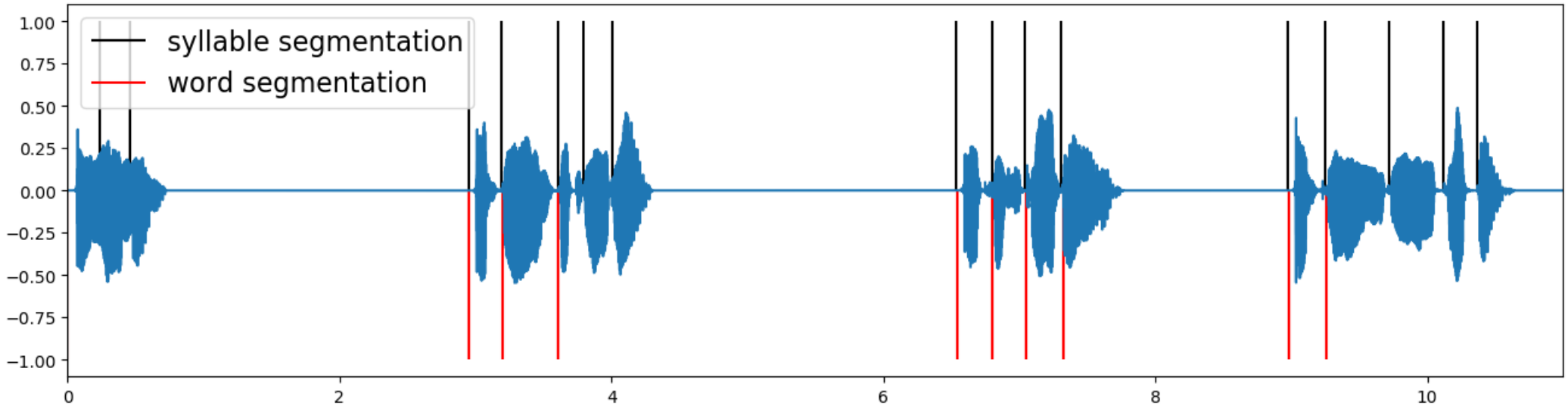
- conf: yaml file for hyperparameters
- data: Dataset
- egs: Main script
- steps: Shell script to run Python codes
- tools: Third party packages
- utils: Our own codes and APIs

Data Flow


Text
①Google
Text-to-Speech
Audio description
Combined audio
②
ES-KMeans
[H. Kamper+, 2017]
Segmented audio
③
S/N=30dB
white noise
Noisy audio
Sound dictionary
Image dataset
Sound-Image paired data

Unsupervised Word Segmentation

1. Combine audio description with 1-3 seconds of random intervals
2. Syllable segmentation: Candidate for word segmentation
3. Word segmentation: ES-KMeans [1]



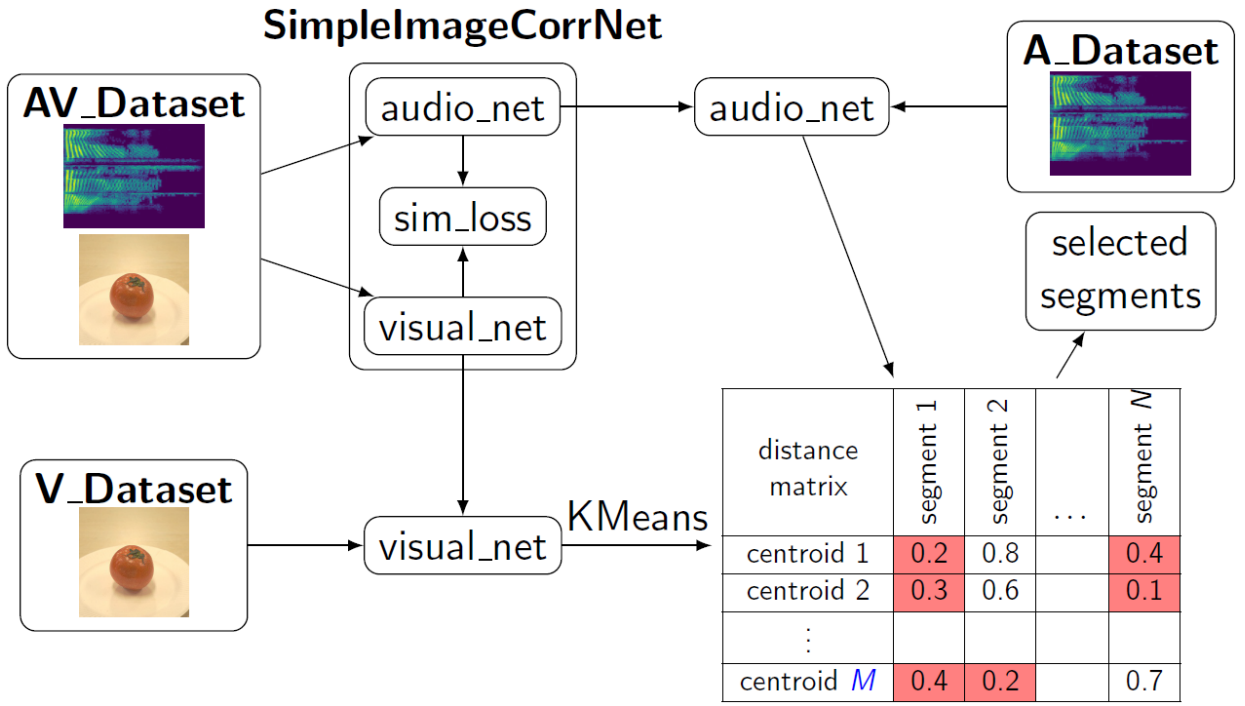
[1] H. Kamper+, "An embedded segmental K-means model for unsupervised segmentation and clustering of speech," ASRU, 2017

exercise1.ipynb (word segmentation)

1. Get landmarks
2. Get segmentation list
3. MFCC extraction
4. Embed MFCC
5. ES-KMeans
6. Wave file segmentation

Unsupervised Correspondence Learning

- audio_net, visual_net: ResNet50 [1]
- AV_Dataset: (Audio description, Food image)
- A_Dataset: Segmented audio including meaningless sound

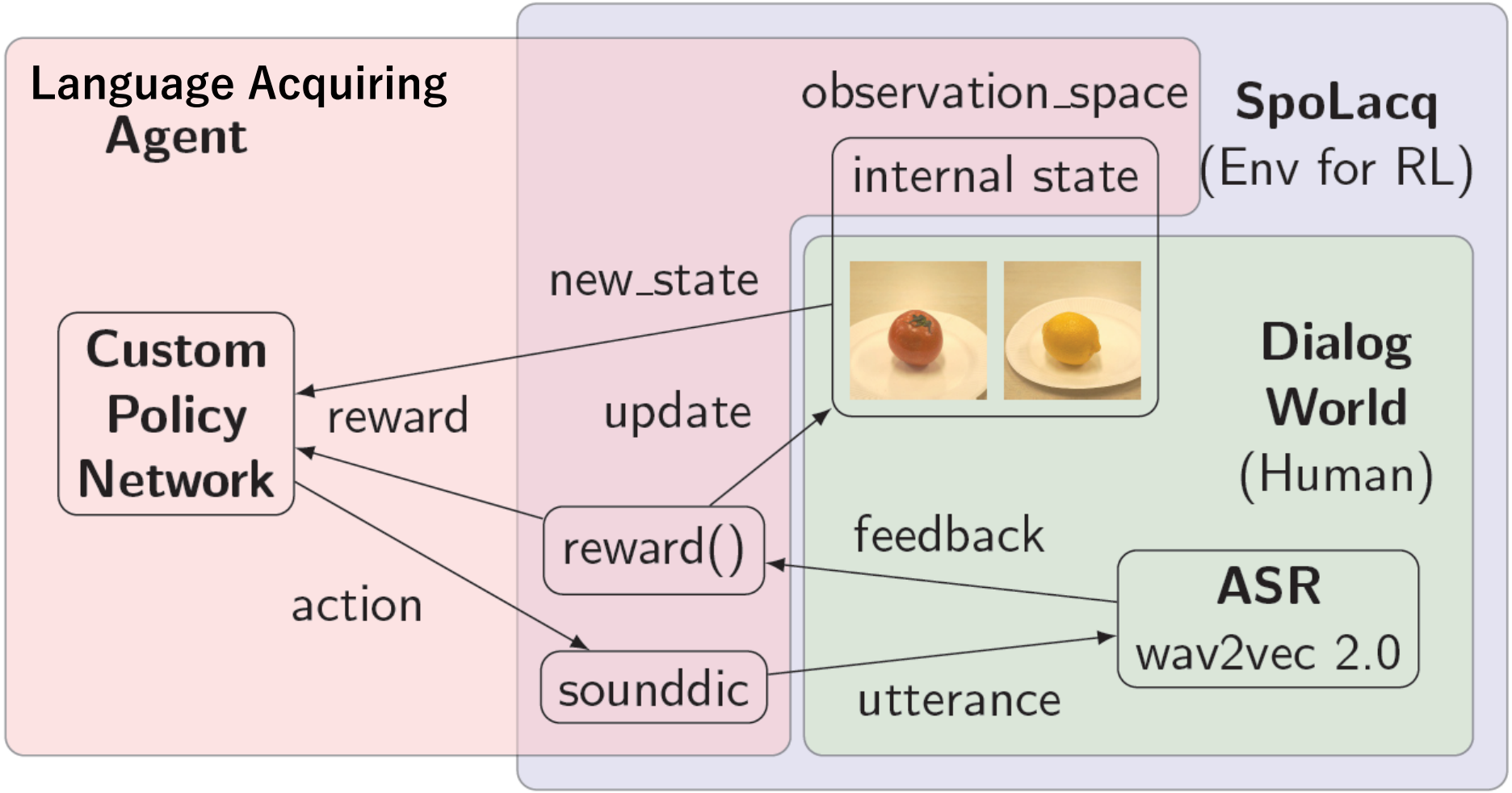


[1] K. He+, "Deep Residual Learning for Image Recognition," CVPR, 2016

exercise2.ipynb (correspondence learning)

1. Remove silence from segmented audio
2. Extract spectrogram from segmented audio
3. Add S/N=30dB white noise to segmented audio
4. Dataset
5. Loss function
6. Model
7. Train model
8. Extract features using pretrained model

RL Environment



[1] A. Baevski+, "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations," NeurIPS, 2020

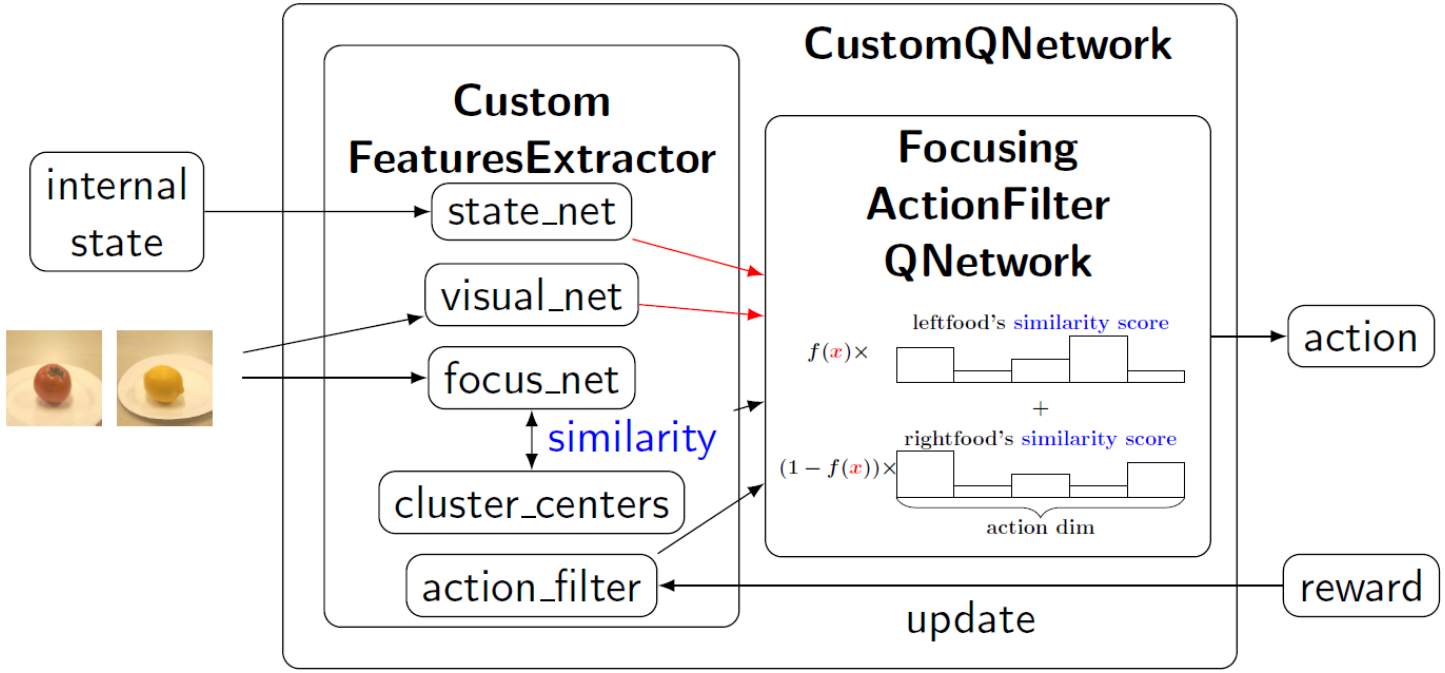
Detail of SpoLacq Environment

- observation_space: gym.spaces.Dict
  - internal state (preferred RGB): 3 dim space (gym.spaces.Box)
  - image: $224 \times 224 \times 3$ dim space (gym.spaces.Box)
- action_space: gym.spaces.Discrete (space size=12000)
  - sounddic: convert action (categorical ID) to wave utterance

- **step()**: one-step dialogue
- reset(): reset observation for a new episode
- render(): render observation

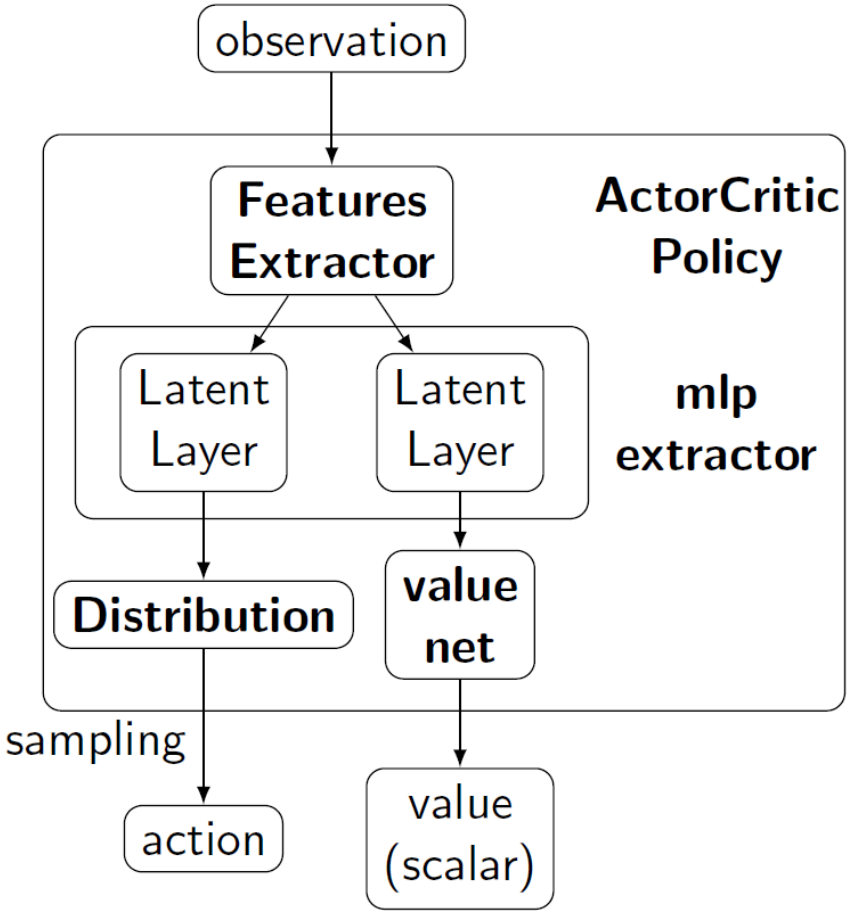
Structure of Custom Policy Network

- state_net: Linear layer
- visual_net: Pretrained ResNet50
- focus_net: Pretrained ResNet50 with fixed weights



Custom Policy Network for Actor Critic

- DQN [1]
  - Action space: Discrete
  - Multi processing: ×
- Actor Critic
  - A2C [2]
  - PPO [3]
  - Action space: Discrete or **Continuous**
  - Multi processing: ○



[1] V. Mnih+, "Human-level control through deep reinforcement learning," Nature, 2015
[2] V. Mnih+, "Asynchronous Methods for Deep Reinforcement Learning," ICML, 2016
[3] J. Schulman+, "Proximal Policy Optimization Algorithms," arXiv, 2017

Training RL Model

Simple example of training

RL environment and model creation

```
env = SpoLacq(FOODS, args.datadir, sounddic, asr)
model = CustomDQN(...)
```

Train RL model

```
model.learn(total_timesteps=360000, tb_log_name=args.tb_log_name)
```

Save RL model and replay buffer

```
model.save(args.workdir+'/dqn')
model.save_replay_buffer(args.workdir+'/replay_buffer')
```

Load RL model and replay buffer

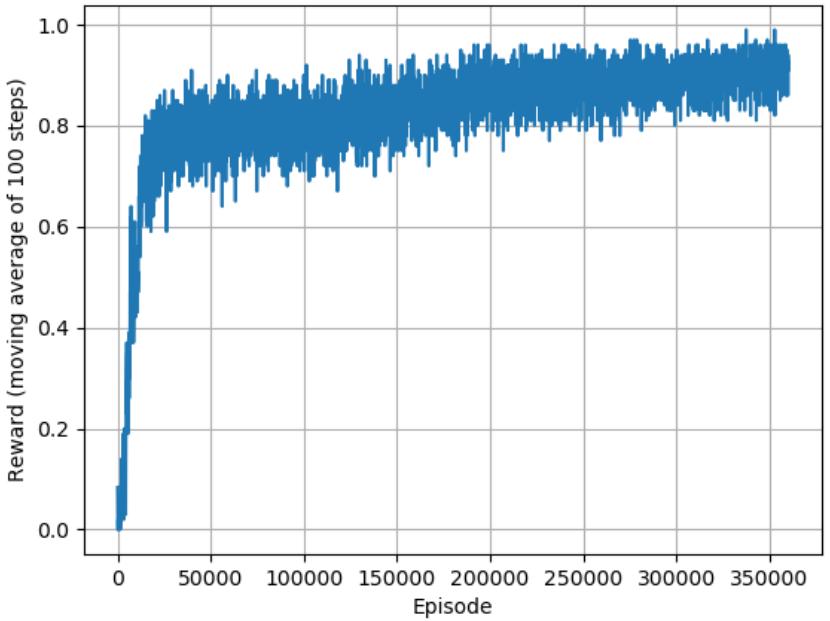
```
del model
del env
env = SpoLacq(FOODS, args.datadir, sounddic, asr)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = CustomDQN.load(args.workdir+'/dqn', env=env, device=device)
model.load_replay_buffer(args.workdir+'/replay_buffer')
```

Retrain RL model

If reset_num_timesteps=False, we can continue the previous tensorboard's log.

```
model.learn(total_timesteps=360000, tb_log_name=args.tb_log_name, reset_num_timesteps=False)
```

Average reward

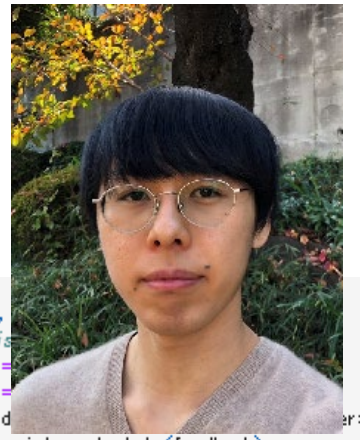


exercise3.ipynb (RL)

1. Dialogue partner
2. Spolacq environment
3. Prepare ASR model
4. Make sound dictionary
5. RL environment and model creation
6. Train RL model
7. Save and load RL model
8. Retrain RL model
9. Test the learnt agent

Toolkit Contributors

Ryota Komatsu*1



Yusuke Kimura*1

Mingxin Zhang*1

Kent Hino*1

Yu Iwamoto*1

Kosuke Mori*1

Keisuke Toyoda*1

Takahiro Shinozaki*2

*1 Master student at Tokyo Institute of Technology, Japan

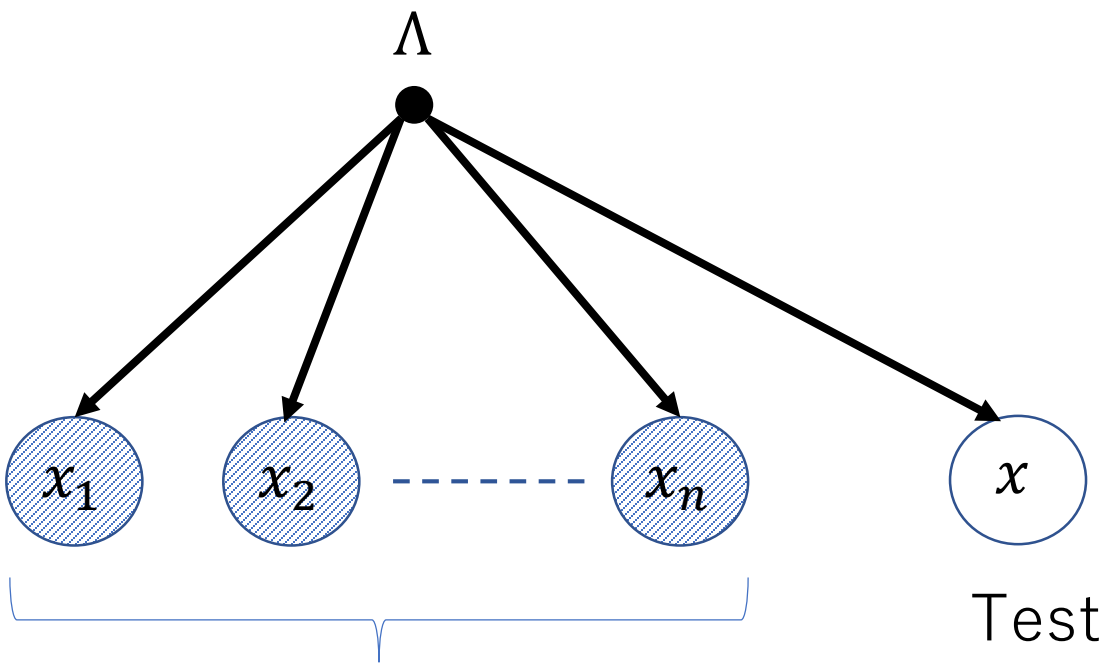
*2 Tokyo Institute of Technology, Japan http://www.ts.titech.ac.jp

We are open to research collaboration.
Please contact Takahiro Shinozaki if you are interested in.

Appendix

Maximum Likelihood (ML) and Bayesian Approach

ML Estimation and Prediction



Training set $D = \langle x_1, x_2, \cdots, x_n \rangle$

Test sample

$$\Lambda^* = \underset{\Lambda}{\operatorname{argmax}}\, p_\Lambda(D) = \underset{\Lambda}{\operatorname{argmax}} \prod_{n=1}^{N} p_\Lambda(x_n)$$

$$p\left(x \mid \Lambda^*\right)$$

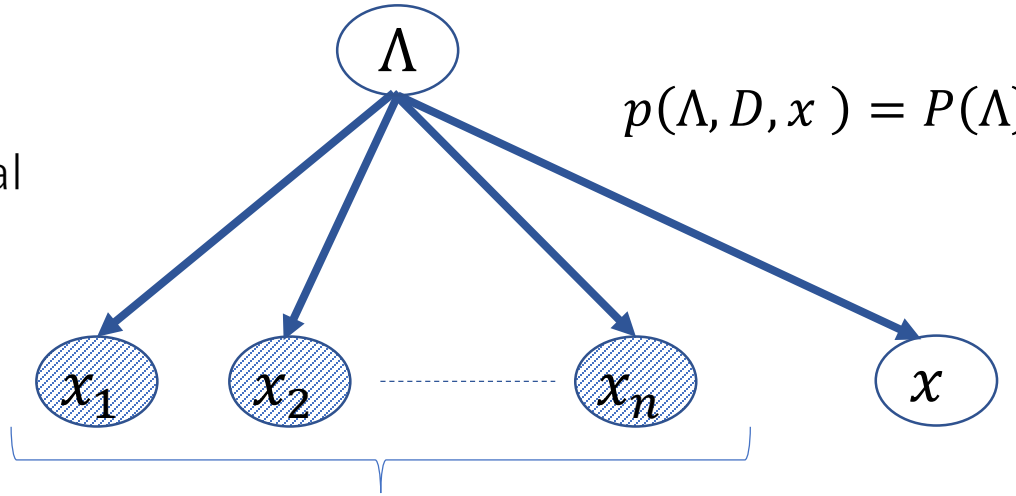
Maximum likelihood (ML) estimation

Prediction

Bayesian Approach

In Bayesian approach, we treat parameters as random variables

Consider a joint probability of all variables, and then compute desired conditional probability



$$p(\Lambda, D, x) = P(\Lambda)P(D|\Lambda)P(x|\Lambda)$$

$$D = \langle x_1, x_2, \cdots, x_n \rangle$$

Prediction of a new sample x is formulated as an evaluation of conditional probability of x given a training set D

$$p(x|D) = \frac{p(x,D)}{p(D)} = \frac{\int_\Lambda p(x,D,\Lambda)}{p(D)} = \int_\Lambda p(x|\Lambda)\frac{p(D|\Lambda)p(\Lambda)}{p(D)} = \int_\Lambda p(x|\Lambda)\,p(\Lambda|D)$$

Definitions of Terms

- Prior distribution of parameters

$$p(\Lambda)$$

- Probabilistic model

$$p(x|\Lambda)$$

- Posterior distribution of parameters

$$p(\Lambda|D) = \frac{p(D|\Lambda)p(\Lambda)}{p(D)}$$

- Predictive distribution

$$p(x|D) = \int_{\Lambda} p(x|\Lambda)\, p(\Lambda|D)$$

Evaluation of Posterior Distribution $p(\Lambda|D)$

- Analytical evaluation
  - Ideal, but only applicable for very simple models
  - For practical models, closed form solution is usually not obtained. Numerical integration is also not feasible when there are many variables
- Variational Bayes
  - Can be applied to large models if proper analytical approximation is introduced
- Sampling
  - Widely applicable, but requires large computational cost

Example: Bayesian GMM

Dirichlet distribution

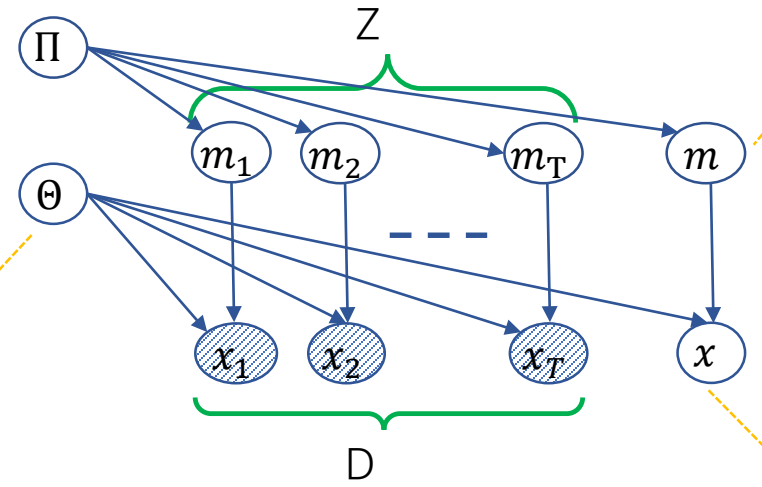
Prior distribution of mixture weights

$$P(\Pi; A) = \frac{1}{Z}\prod_{i=1}^{M} w_i$$

$$\Pi = \langle w_1, w_2, \cdots, w_M \rangle$$

Z:normalization constant

(M-mixture GMM with constant variances)



Mixture weight

$P(m_t = i) = w_i$

Prior distributions of means

$$\Theta = \langle \mu_1, \mu_2, \cdots, \mu_M \rangle$$

$$P(\mu_i) = N(\mu_i; 0,1)$$

$P(x_t|m_t, \Theta) = N(x; \mu_{m_t}, 1)$

$$P(\Pi, \Theta, Z, D, m, x) = P(\Pi)P(\Theta)P(Z|\Pi)P(D|\Theta, Z)P(m|\Pi)P(x|\Theta, m)$$

$$= P(\Pi)P(\Theta)\prod_t \{P(m_t|\Pi)P(x_t|\Theta, m_t)\}\, P(m|\Pi)P(x|\Theta, m)$$

Posterior distribution of parameters

$$P(\Pi, \Theta|D) = \frac{\sum_Z P(\Pi)P(\Theta)P(Z|\Pi)P(D|\Theta, Z)}{\sum_Z \int\int P(\Pi)P(\Theta)P(Z|\Pi)P(D|\Theta, Z)\, d\Pi d\Theta}$$

Conditional Independence and d-separation

Conditional Independence

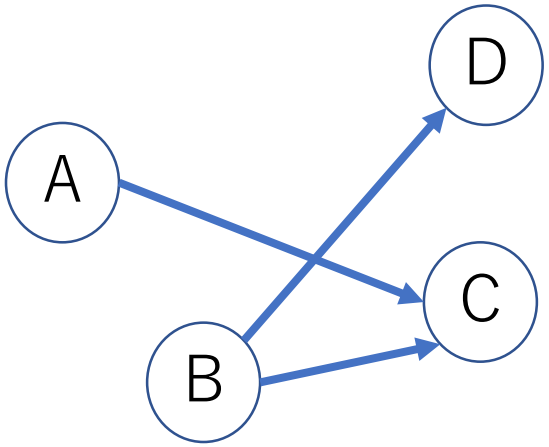
Let A, B, and C be disjoint sets of random variables. When the following equation holds, we say that A is independent of B given C, and denote it as $A \perp\!\!\!\perp B \mid C$

$$P(A \mid B, C) = P(A \mid C)$$

$$A \perp\!\!\!\perp B \mid C$$

Graph Structure and Conditional Independence

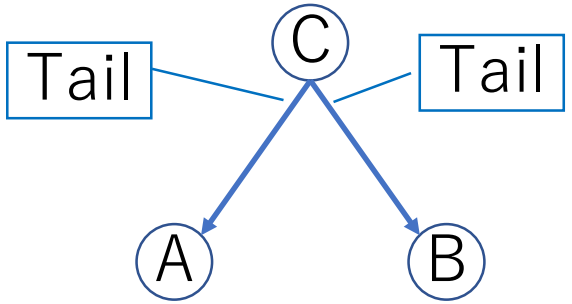
By investigating the graph structure, we can read relationships between random variables



$A \perp\!\!\!\perp B \mid C$?

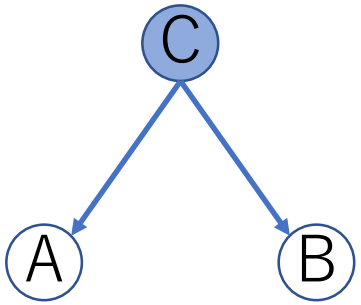
$A \perp\!\!\!\perp D \mid C, B$?

Tail-To-Tail



In general, P(A,B) is not expressed as P(A)P(B). Therefor, $A \perp\!\!\!\perp B \mid \Phi$ does not hold.
(Φ is an empty set)

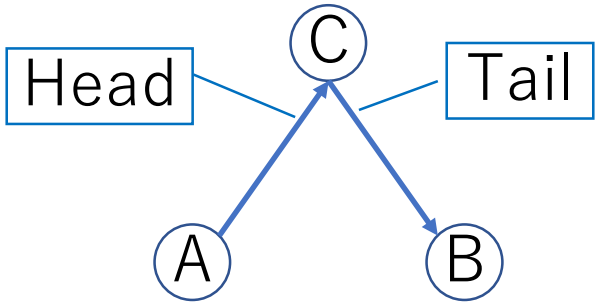
$$P(A,B) = \sum_C P(A,B,C) = \sum_C P(A \mid C)P(B \mid C)P(C)$$



P(A,B|C) is expressed as P(A|C)P(B|C). Therefore $A \perp\!\!\!\perp B \mid C$ holds.

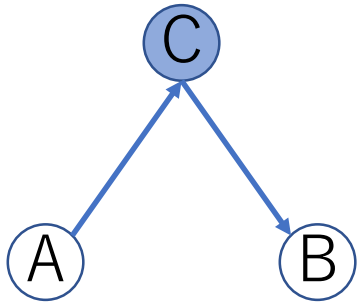
$$P(A,B \mid C) = \frac{P(A,B,C)}{P(C)} = \frac{P(A \mid C)P(B \mid C)P(C)}{P(C)}$$
$$= P(A \mid C)P(B \mid C)$$

Head-To-Tail



In general, P(A,B) is not expressed as P(A)P(B). Therefor, $A \perp\!\!\!\perp B \mid \Phi$ does not hold.

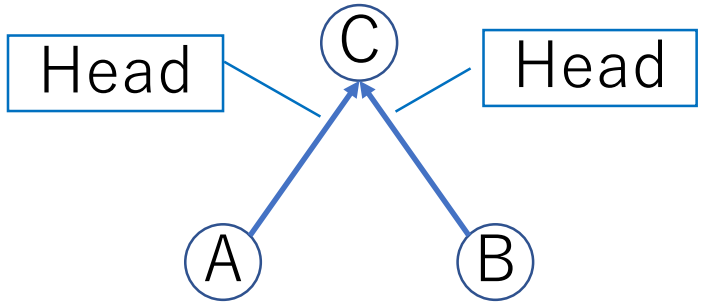
$$P(A,B) = \sum_C P(A,B,C) = \sum_C P(A)P(B \mid C)P(C \mid A)$$



P(A,B|C) is expressed as P(A|C)P(B|C). Therefore $A \perp\!\!\!\perp B \mid C$ holds.

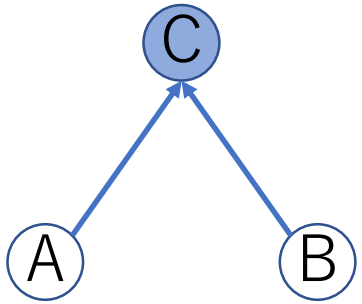
$$P(A,B \mid C) = \frac{P(A,B,C)}{P(C)} = \frac{(P(C \mid A)P(A))P(B \mid C)}{P(C)}$$
$$= P(A \mid C)P(B \mid C)$$

Head-To-Head



In general, P(A,B) is expressed as P(A)P(B). Therefor, $A \perp\!\!\!\perp B \mid \Phi$ holds.

$$P(A,B) = \sum_C P(A,B,C) = \sum_C P(A)P(B)P(C \mid A,B) = P(A)P(B)$$



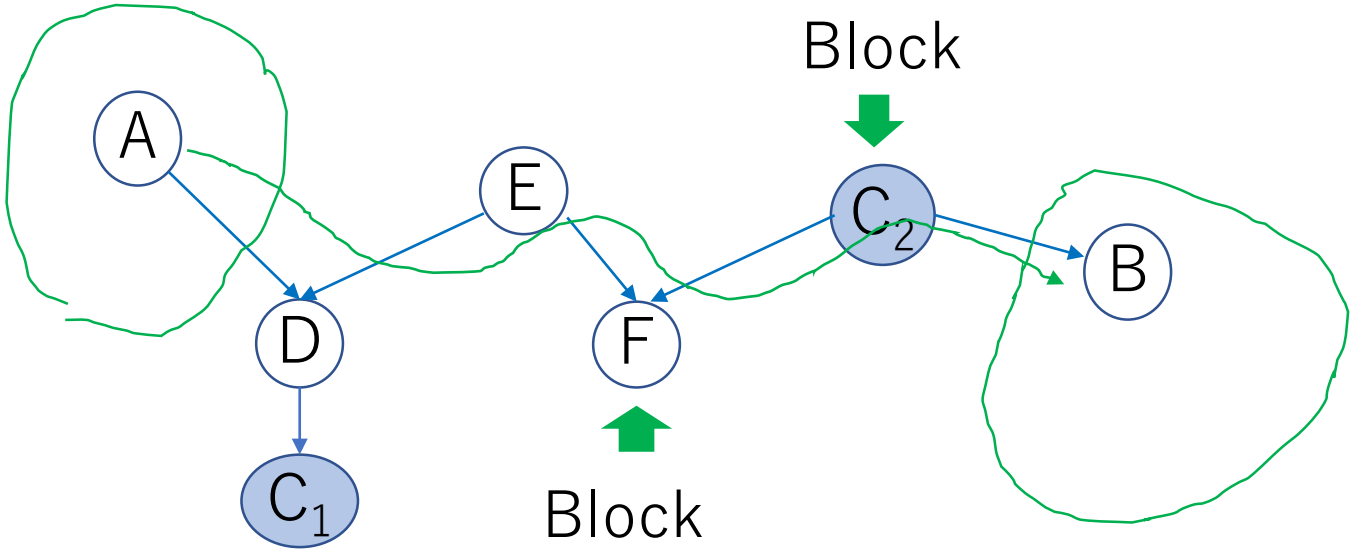
P(A,B|C) is not expressed as P(A|C)P(B|C). Therefore $A \perp\!\!\!\perp B \mid C$ does not hold.

$$P(A,B \mid C) = \frac{P(A,B,C)}{P(C)} = \frac{P(A)P(B)P(C \mid A,B)}{P(C)}$$

Blocking a Path

Let A and B be a node, and C be a set of nodes that does not include A and B. Path from A to B is blocked when either of the followings holds

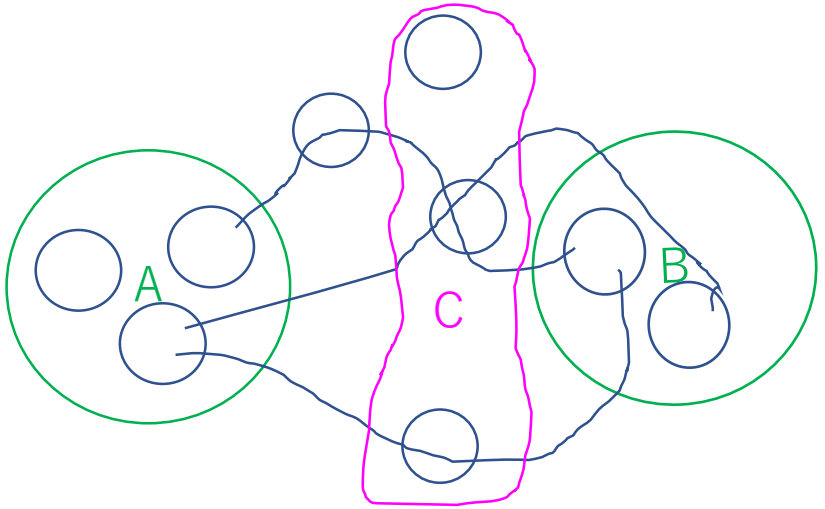
- On the path from A to B, there is a node in C and the connection of the arcs is tail-to-tail or head-to-tail
- At one of the nodes on the path from A to B, the connection of the arcs is head-to-head. In addition, the node and its all descendants are not included in C



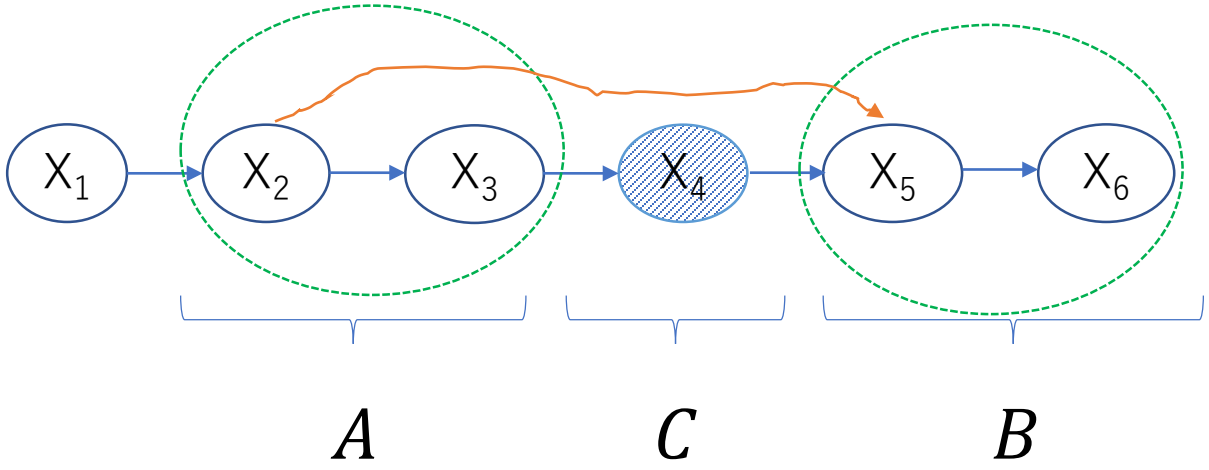
d-separation

Let A, B, and C be exclusive sets of nodes

- A is d-separated from B by C if all the paths starting from a node in A and ending at a node in B is blocked
- When A is d-separated from B by C, $A \perp\!\!\!\perp B | C$ holds for the joint probability defined by the Bayesian network (Pearl 1988)



Example



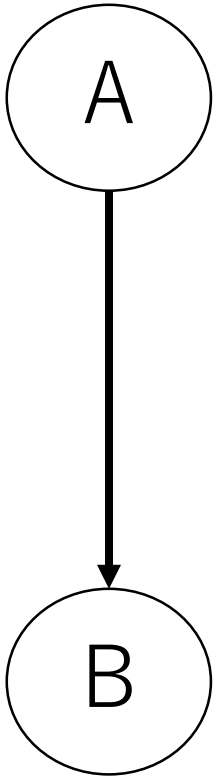
When X_4 is observed, all the paths from A to B is blocked at X_4

➡ $A \perp\!\!\!\perp B \mid C$

$$P(A,C,B) = P(A)P(C|A)P(B|A,C) = P(A)P(C|A)P(B|C)$$
$$= P(X_2,X_3)P(X_4|X_2,X_3)P(X_5,X_6|X_4)$$

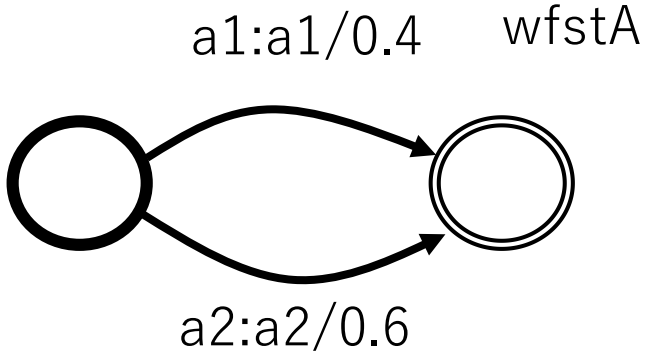
Weighted Finite State Transducer (WFST) and Probability Distribution Modeling

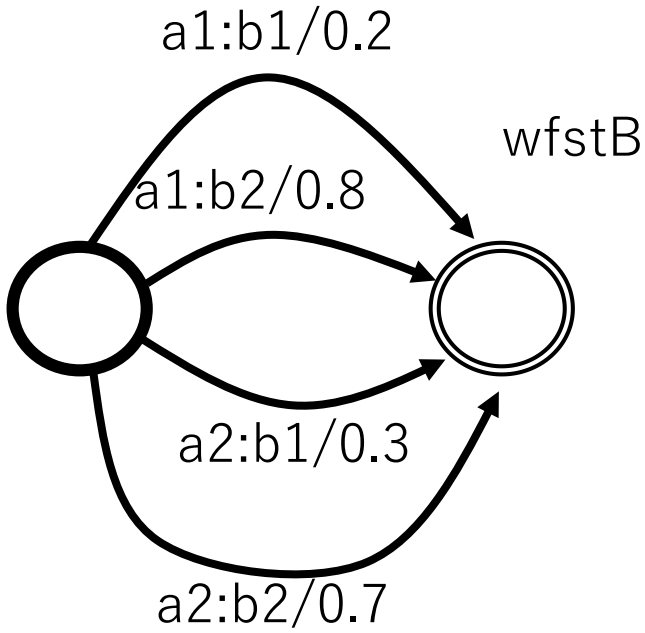
Bayesian Network and WFST



	A=a1	A=a2
P(A)	0.4	0.6

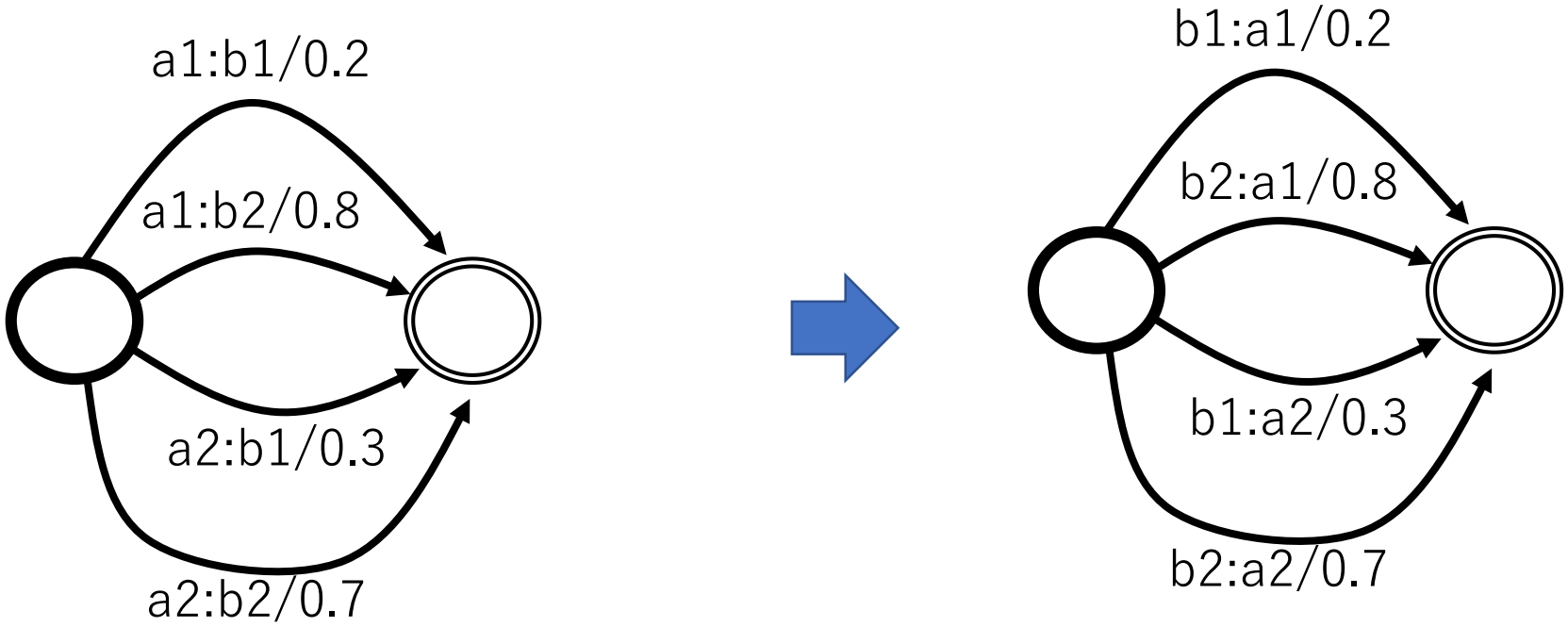
P(B|A)	B=b1	B=b2
A=a1	0.2	0.8
A=a2	0.3	0.7





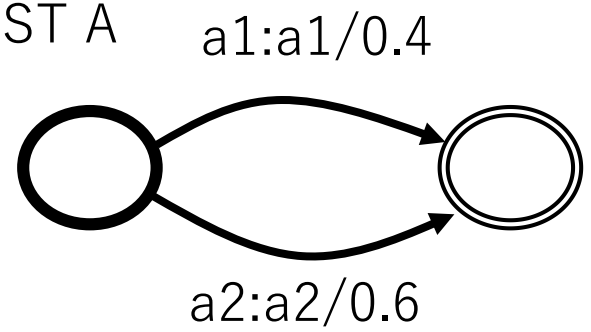
Invert of WFST

- Swap input symbol and output symbol

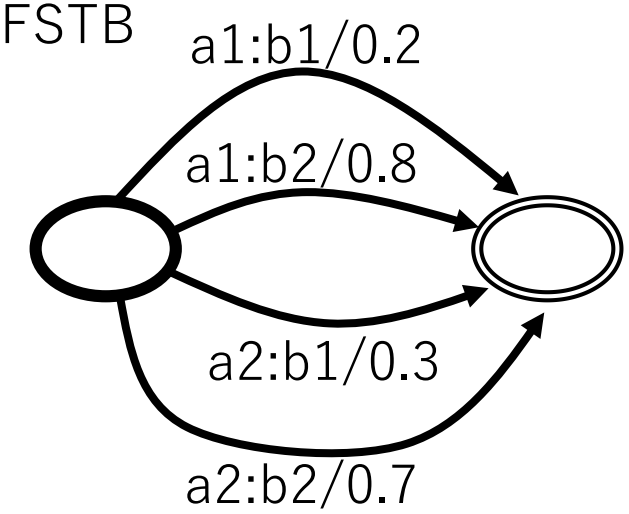


Composition and Joint Probability

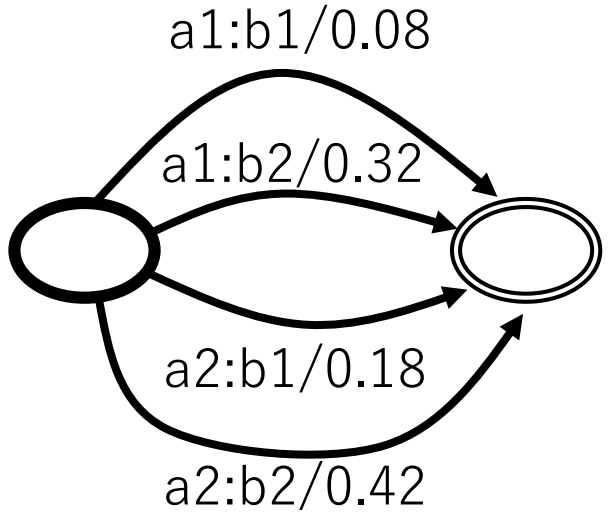
WFST A



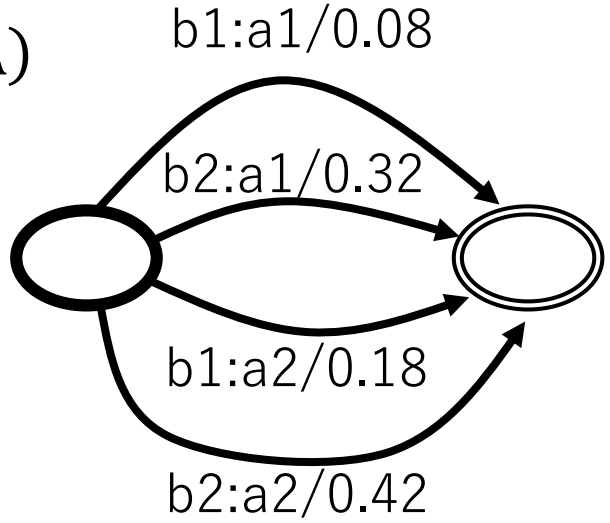
WFSTB



A∗B
P(A,B)
∝P(B|A=a)

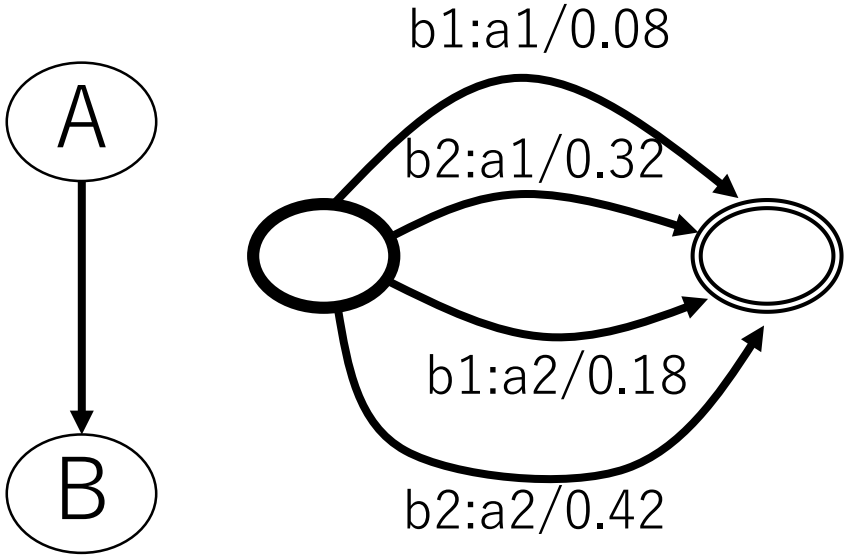


Inv(B)∗ **Inv(A)**
P(A,B)
∝P(A|B=b)

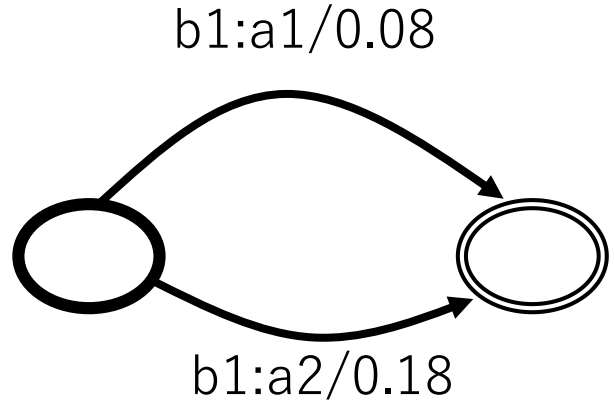


*real semi-ring

Composition and Inference



WFST: Inv(B)∗ **Inv(A)**
P(A,B)



Bb1∗(Inv(B)∗ **Inv(A)**)
∝P(A|B=b1)

WFST: Bb1
B=b1