EN. 601.467/667 Introduction to Human Language Technology End-to-End Speech Recognition

Shinji Watanabe

Johns Hopkins Whiting School of Engineering

Today’s agenda

- Introduction to end-to-end speech recognition
- Connectionist Temporal Classification (CTC)
- Attention

CENTER FOR LANGUAGE AND SPEECH PROCESSING

Frederick Jelinek (1932 –2010)

Statistical speech recognition and machine translation

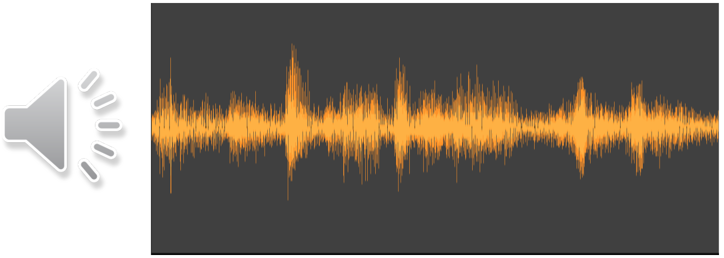
1972 - 1993: IBM

1993 - 2010: JHU

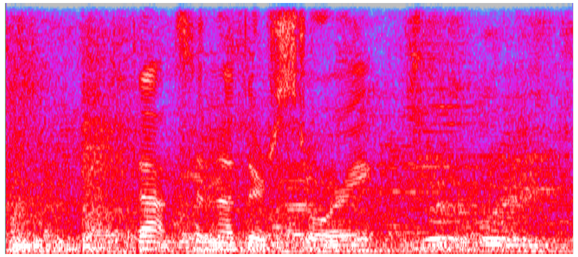
Jelinek methodology (1970s-)

Jelinek methodology (1970s-)

- Automatic Speech Recognition: Mapping ***physical signal sequence*** to ***linguistic symbol sequence***



$$X = \{x_l \in \mathbb{Z} | l = 1, \dots, L\}$$
$$L = 43263$$



$$X = \{\mathbf{x}_t \in \mathbb{C}^D | t = 1, \dots, T\}$$
$$T = 268$$

"That's another story"

$$W = \{w_n \in \mathcal{V} | n = 1, \dots, N\}$$
$$N = 3$$

Jelinek methodology (1970s-)

$$\arg\max_{\mathrm{W}} p(W|O)$$

O: Speech sequence
W: Text sequence

Jelinek methodology (1970s-)

L: Phoneme sequence

$$\arg\max_{W} p(W|O) = \arg\max_{W} p(O|W)p(W)$$
$$\approx \arg\max_{W,L} p(O|L)p(L|W)p(W)$$

$$\sum_{L} P(O|L,\cancel{W})P(L|W)P(W)$$

- **Speech recognition**
  - $p(O|L)$: Acoustic model (Hidden Markov model)
  - $p(L|W)$: Lexicon
  - $p(W)$: Language model (n-gram)

Jelinek methodology (1970s-)

$$\arg\max_{W} p(W|O) = \arg\max_{W} p(O|W)p(W)$$
$$\approx \arg\max_{W,L} p(O|L)p(L|W)p(W)$$

- **Speech recognition**
  - $p(O|L)$: Acoustic model (Hidden Markov model)
  - $p(L|W)$: Lexicon
  - $p(W)$: Language model (n-gram)

- Factorization
- **Conditional independence (Markov) assumptions, CIA**

Jelinek methodology (1970s-)

$$\arg\max_{\mathrm{W}} p(W|O) = \arg\max_{\mathrm{W}} p(O|W)p(W)$$

- **Machine translation**
  - $p(O|W)$: Translation model
  - $p(W)$: Language model

Jelinek methodology (1970s-)

$$\begin{aligned}\arg\max_{\mathrm{W}} p(W|O) &= \arg\max_{\mathrm{W}} p(O|W)p(W) \\ &\approx \arg\max_{\mathrm{W,L}} p(O|L)p(L|W)p(W)\end{aligned}$$

- **Speech recognition**
  - $p(O|L)$: Acoustic model (Hidden Markov model)
  - $p(L|W)$: Lexicon
  - $p(W)$: Language model (n-gram)
- Continued 40 years

Jelinek methodology (1970s-)

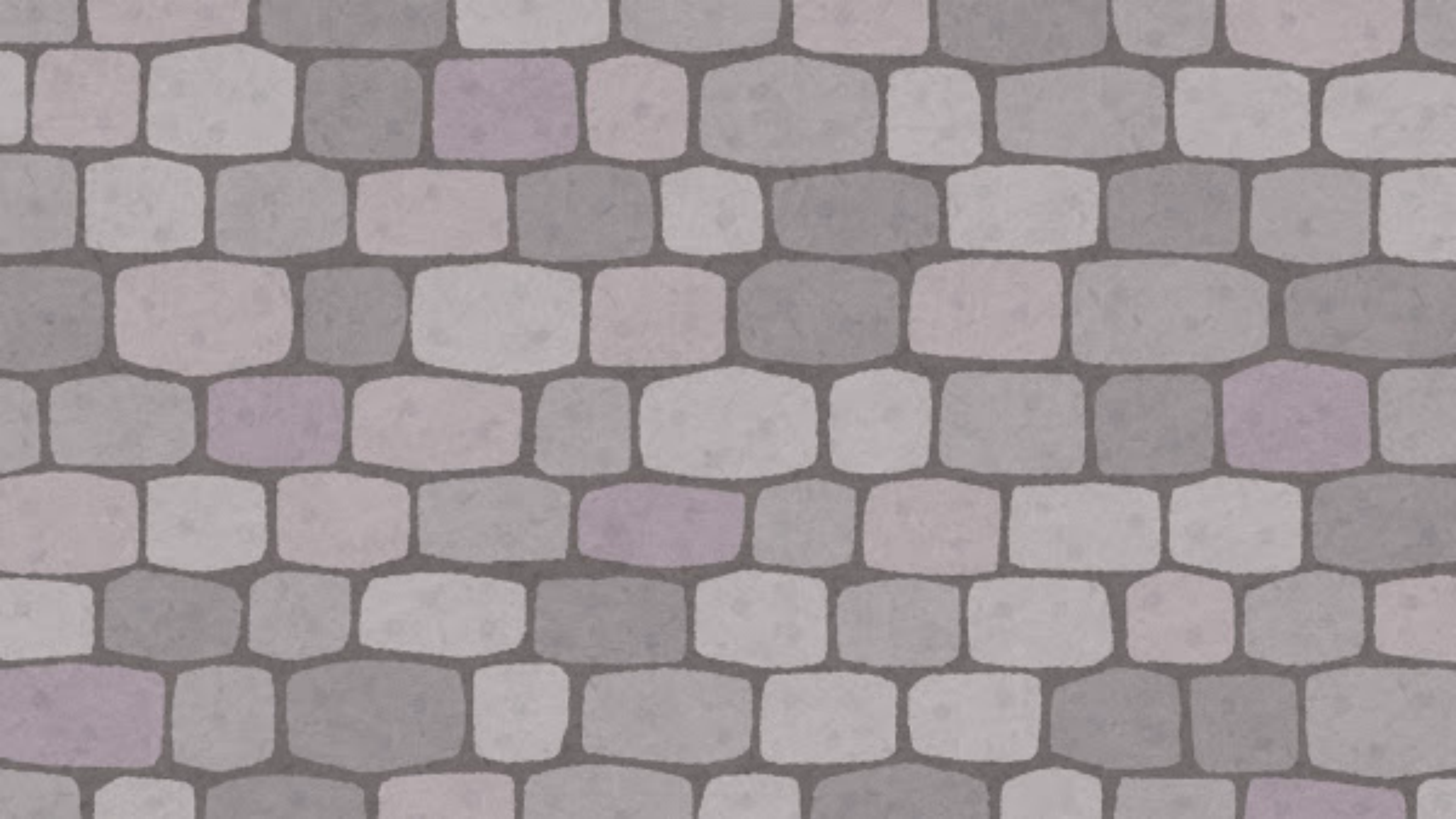
$$\arg\max_{\mathrm{W}} p(W|O) = \arg\max_{\mathrm{W}} p(O|W)p(W)$$
$$\approx \arg\max_{\mathrm{W,L}} p(O|L)p(L|W)p(W)$$

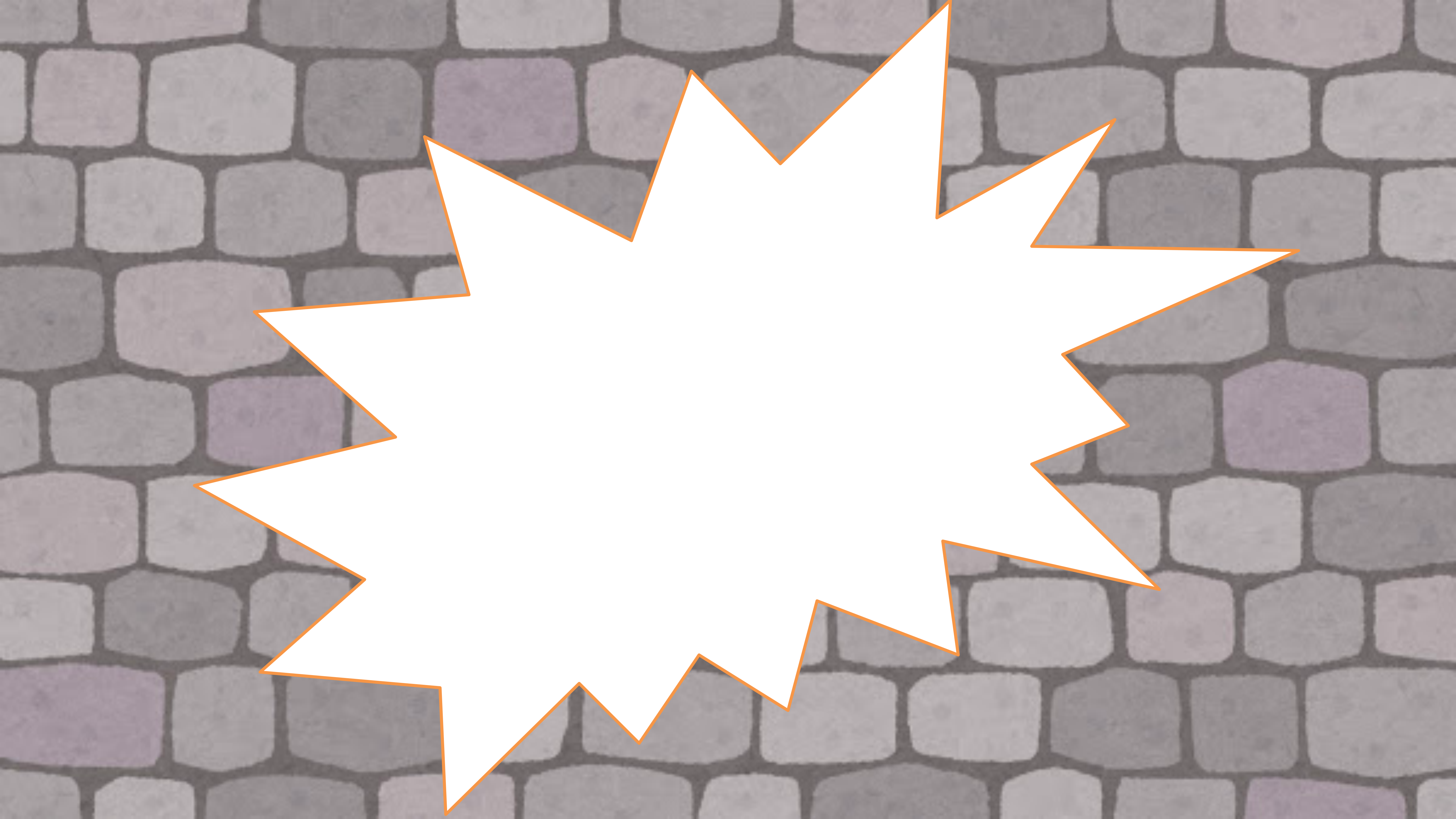
- **Speech recognition**
  - $p(O|L)$: Acoustic model
  - $p(L|W)$: Lexicon
  - $p(W)$: Language model
- Continued 40 years

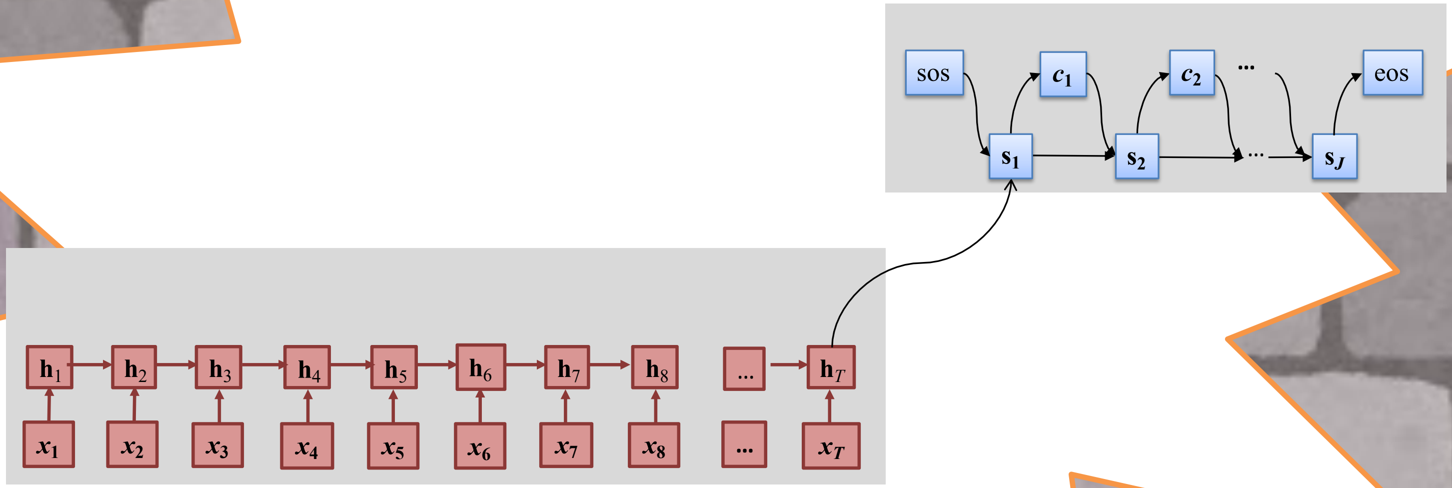
Big barrier:

noisy channel model
HMM
n-gram
etc.

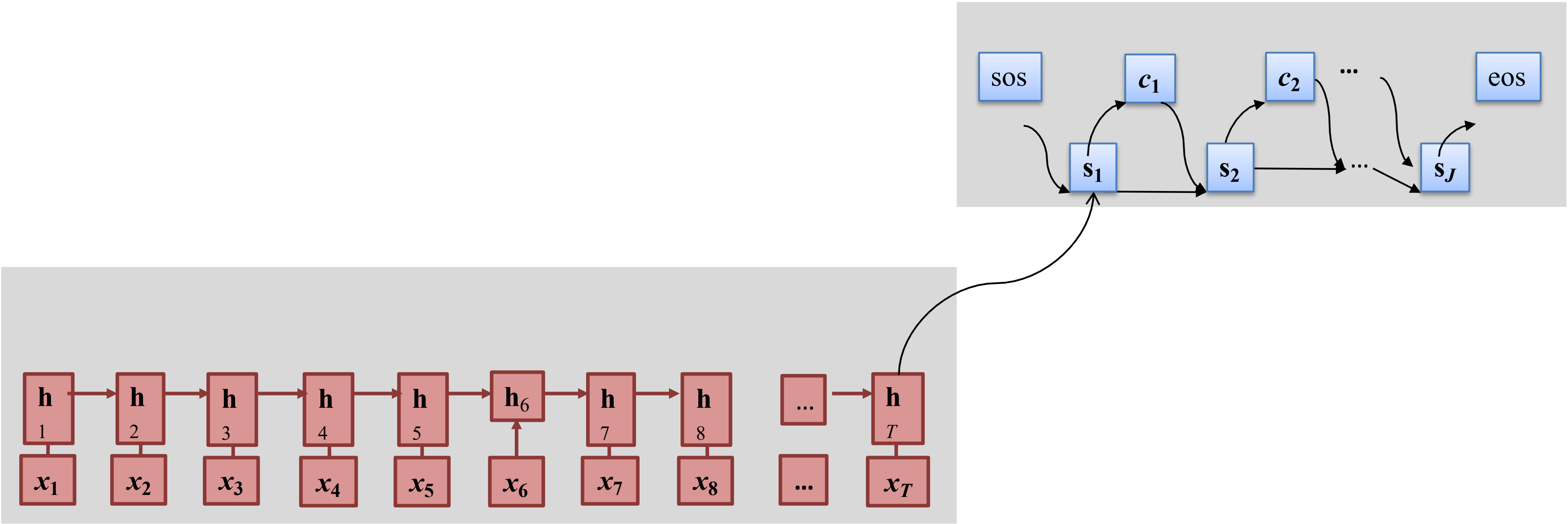
However,





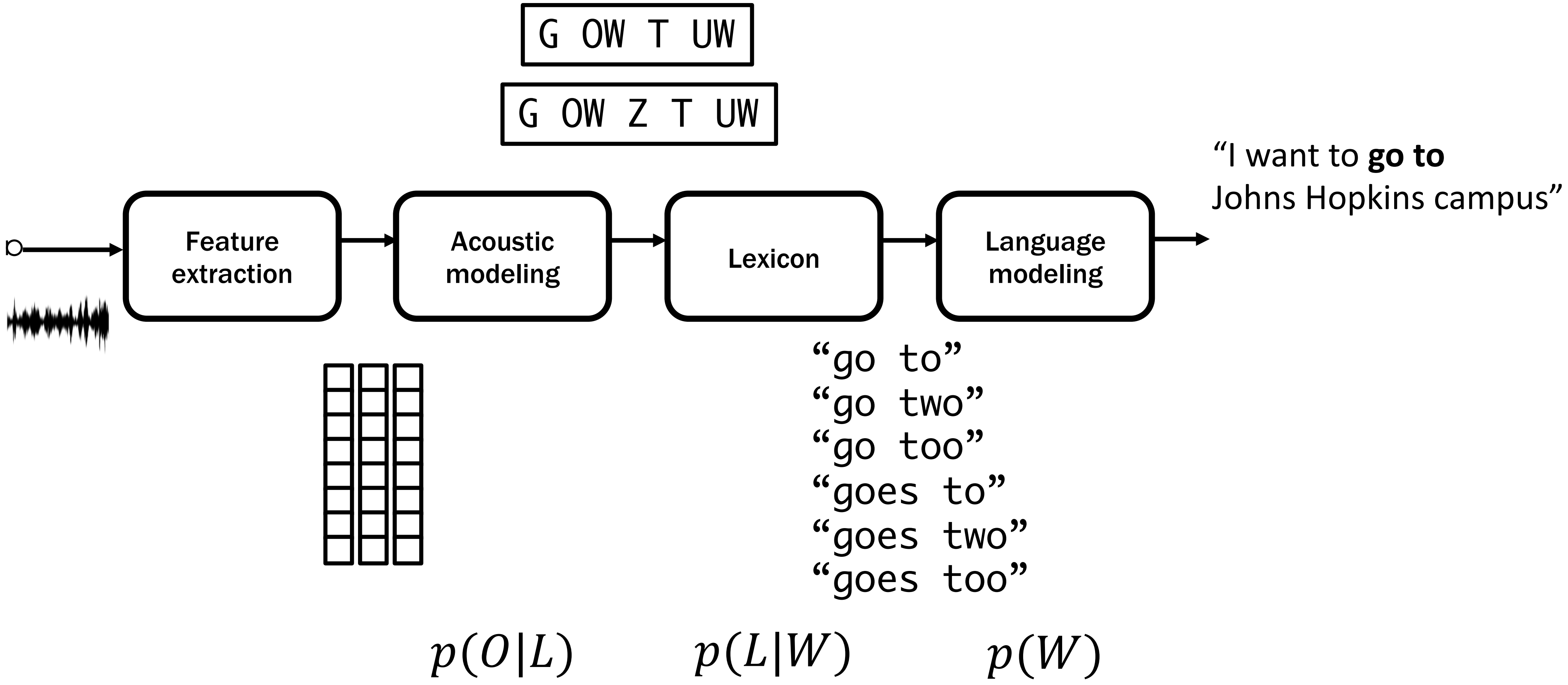

sos
c_1
c_2
...
eos
$\mathbf{s}_1$
$\mathbf{s}_2$
...
$\mathbf{s}_J$
$\mathbf{h}_1$ $\mathbf{h}_2$ $\mathbf{h}_3$ $\mathbf{h}_4$ $\mathbf{h}_5$ $\mathbf{h}_6$ $\mathbf{h}_7$ $\mathbf{h}_8$ ... $\mathbf{h}_T$
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 ... x_T

"End-to-End" Processing Using Sequence to Sequence

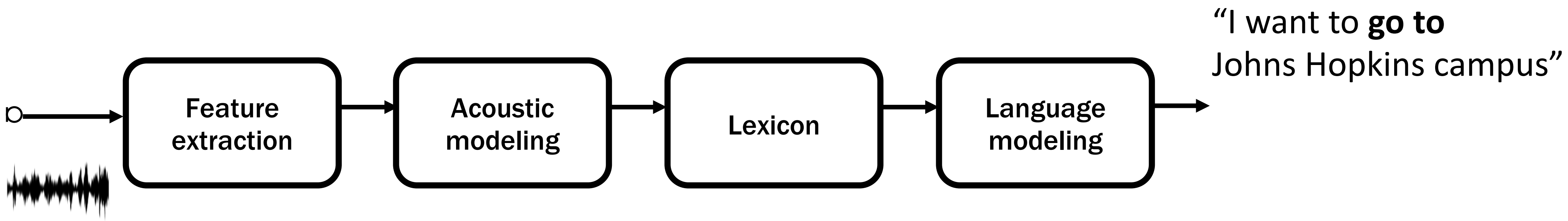


- Directly model $p(W|O)$ with a **single neural network**
- Great success in neural machine translation

Speech recognition pipeline

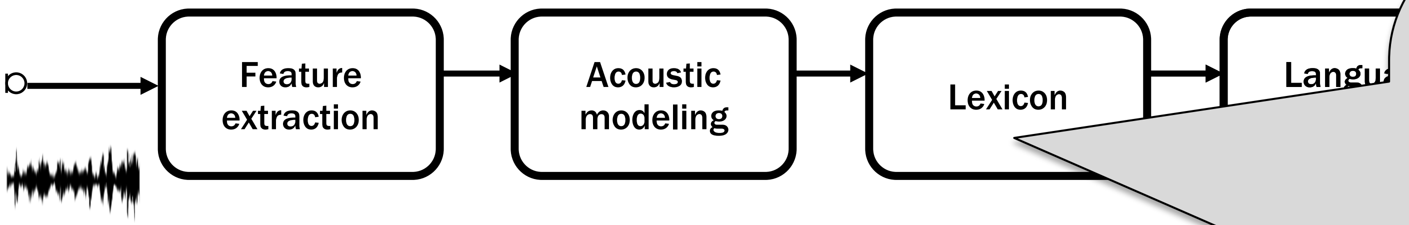


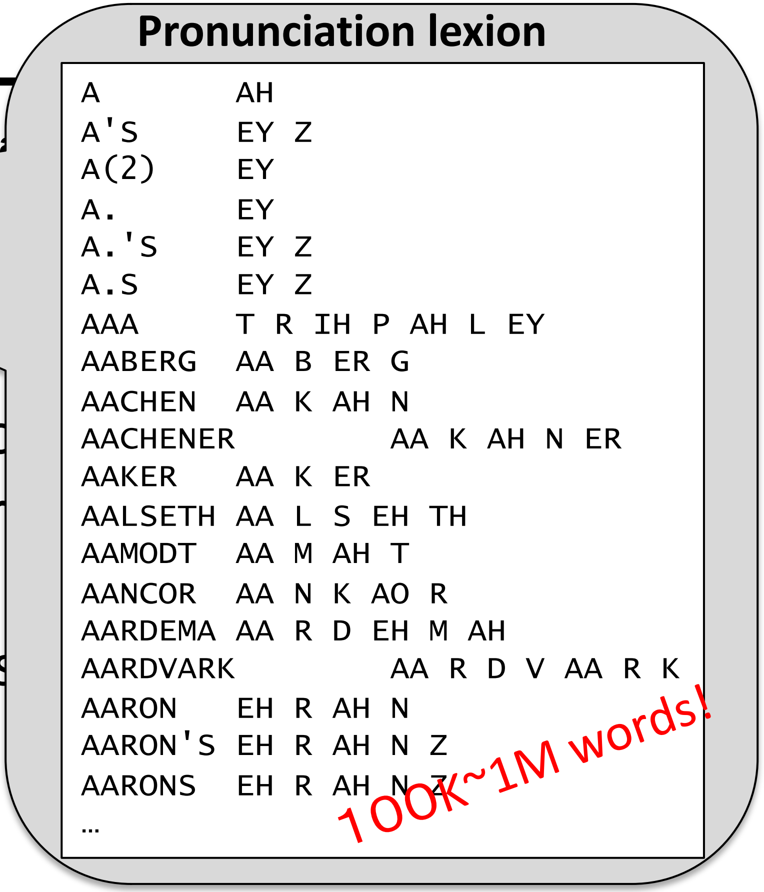
Speech recognition pipeline



- Require **a lot of development** for an acoustic model, a pronunciation lexicon, a language model, and finite-state-transducer decoding
- Require linguistic resources
- Difficult to build ASR systems for non-experts

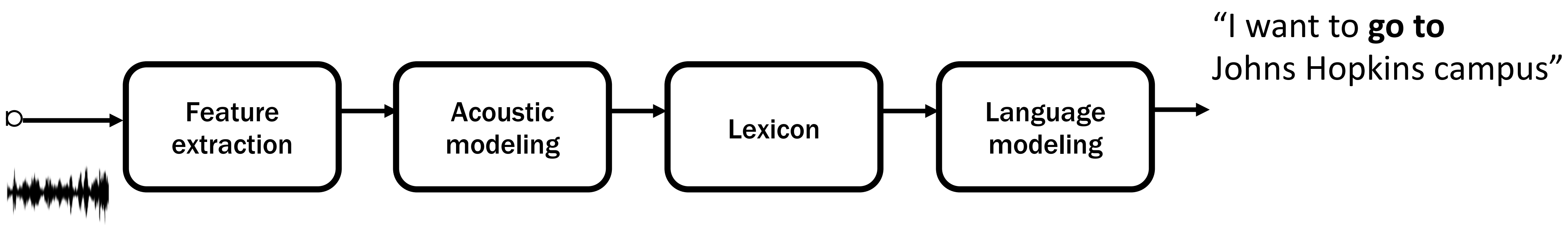
Speech recognition pipeline





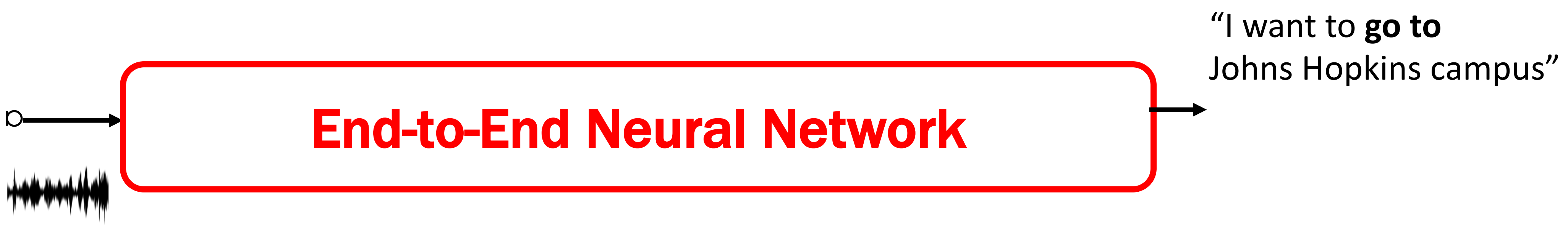
- Require **a lot of development** for an acoustic lexicon, a language model, and finite-state-tr
- Require linguistic resources
- Difficult to build ASR systems for non-experts

Speech recognition pipeline



- Require **a lot of development** for an acoustic model, a pronunciation lexicon, a language model, and finite-state-transducer decoding
- Require linguistic resources
- Difficult to build ASR systems for **non-experts**

From pipeline to integrated architecture

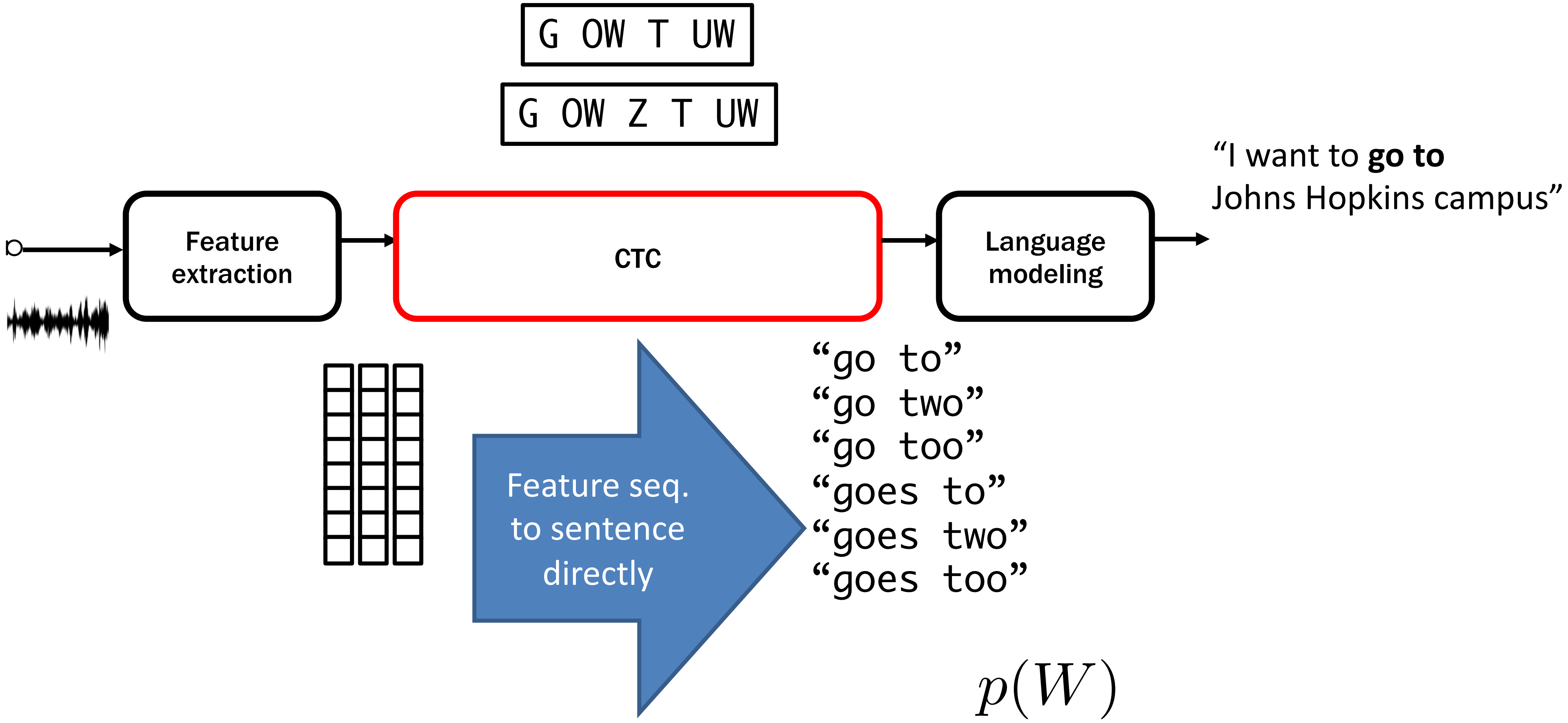


- Train a deep network that directly maps speech signal to the target letter/word sequence
- Greatly simplify the complicated model-building/decoding process
- Easy to build ASR systems for new tasks **without expert knowledge**
- Potential to outperform conventional ASR by **optimizing the entire network** with a single objective function

Today's agenda

- Introduction to end-to-end speech recognition
- Connectionist Temporal Classification (CTC)
- Attention

Speech recognition pipeline



Character seq. vs. word seq.

- Example: "I see"
  - $W = (w_i \in \{\text{"i"}, \text{"see"}, \dots .\} | i = 1, 2)$
  - $C = (c_j \in \mathbb{U} | j = 1, \dots, 5)$, where $\mathbb{U} = \{\text{"a"}, \text{"b"}, \text{"c"}, \text{"d"}, \text{"e"}, \dots\}$ (Latin alphabet)
- Low/zero count problem
  - Word "bitcoin" is not appeared in old WSJ sentences, but character seq. can cover it
- Semantic context, lexicon constraint
  - Word unit can handle them, but not in the character unit
- No word unit in some languages
  - Some languages do not have word boundaries (no explicit word units)

- Remark: Subword/token (e.g., "_i", "_s", "ee"), data-driven ways to tokenize a character sequence to consider the benefits of both character and word units

Connectionist temporal classification

- Formulation
  - Let character seq. be $C = (c_t \in \mathbb{U} | j = 1, \ldots, J)$ and feature seq. be $O = (\mathbf{o}_t \in \mathbb{R}^D | t = 1, \ldots, T)$
  - Focus on the posterior distribution $p(C|O)$, and we can start from the Bayes decision theory:

$$\hat{C} = \underset{C}{\operatorname{argmax}}\, p(C|O)$$

  - This is the same start as the HMM except that we use the character seq. instead of the word seq.

Connectionist temporal classification

- Formulation
  - Focus on the posterior distribution $p(C|O)$, and rewrite it as

$$p(C|O) = \sum_Z p(C|Z, \cancel{O})p(Z|O)$$

$$\approx \sum_Z \underbrace{p(C|Z)}_{\text{CTC LM}} \underbrace{p(Z|O)}_{\text{CTC AM}}$$

  - No Bayes theorem, but use conditional independence assumption
  - Introduce latent variable seq. $Z = (z_t \in \{\mathbb{U}, <\mathbf{b}>\} | t = 1, \ldots, T)$ that has **the same length** as input feature seq.
    - We can use a conventional RNN to model this $p(Z|O)$
  - Similarly to HMM, we'll consider the summation of all possible Z

Introduction of blank symbol <b>

- First, we insert <b> to the character seq.

”see”

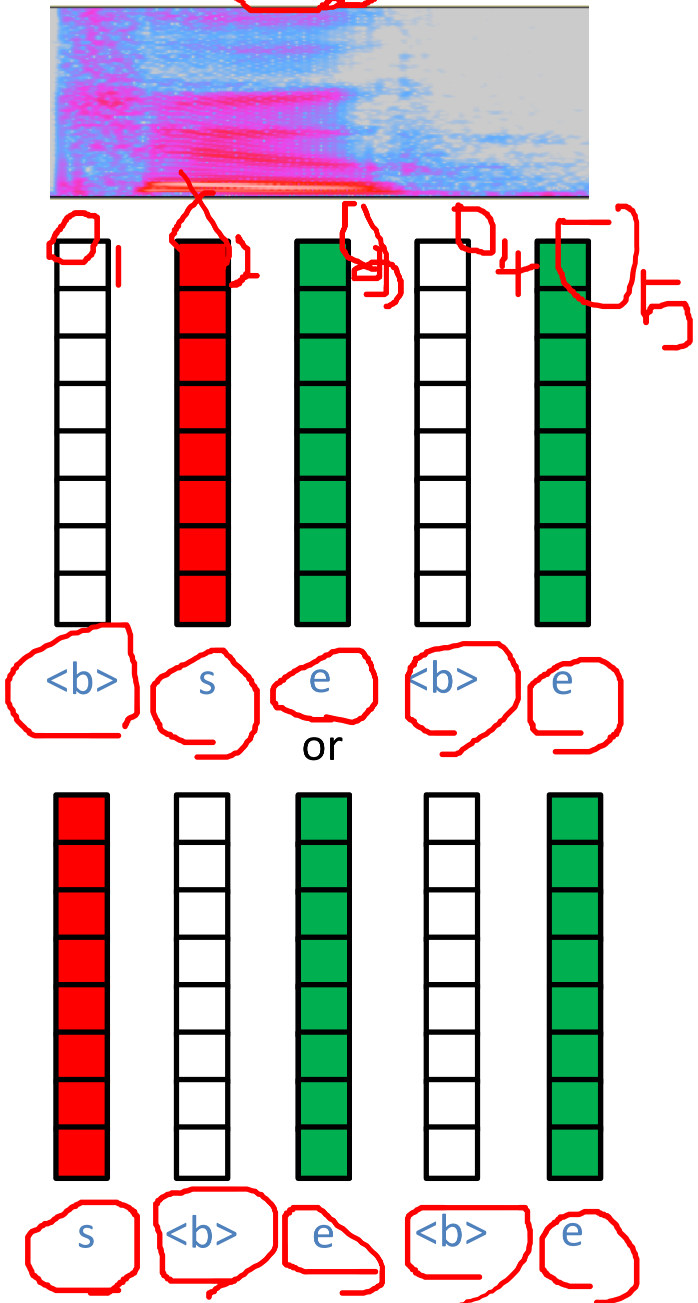
→ C = (“s”, “e”, “e”), where $|C| = J$

→ C' = (“<b>”, “s”, “<b>”, “e”, “<b>”, “e”, “<b>”), where $|C'| = 2J + 1$

- Then, expand C' to the frame length T to form Z
  - All characters can be repeated
  - <b> can be skipped except when it is inserted between repeated character
    - “s”, ”<b>”, ”e”: we can skip <b>
    - “e”, ”<b>”, ”e”: we **cannot** skip <b>
    - See A. Graves et al. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks.” in ICML

Example of Z

“see” (only 5 frames)



- C = (“s”, “e”, “e”)
- C' = (“<b>”, “s”, “<b>”, “e”, “<b>”, “e”, “<b>”)
- $T = 5$
- Z = (“<b>”, “s”, “e”, “<b>”, “e”), (“s”, “<b>”, “e”, “<b>”, “e”), (“s”, “s”, “e”, “<b>”, “e”),….

- This is an alignment problem
- $C \rightarrow Z$: one to many mapping

CTC Formulation

- CTC acoustic model

$$p(Z|O) = \prod_{t=1}^{T} p(z_t|\cancel{z_1, \cdots, z_{t-1}}, O)$$

$$\approx \prod_{t=1}^{T} p(z_t|O).$$

  - Using conditional independence assumption to factorize the posterior $p(Z|O)$ but this is not bad assumption compared with HMM
  - This can be realized by Bidirectional LSTM or self attention

$$p(z_t = j|O) = [\mathrm{softmax}(\mathbf{W}\mathbf{h}_t + \mathbf{b})]_j \,,$$

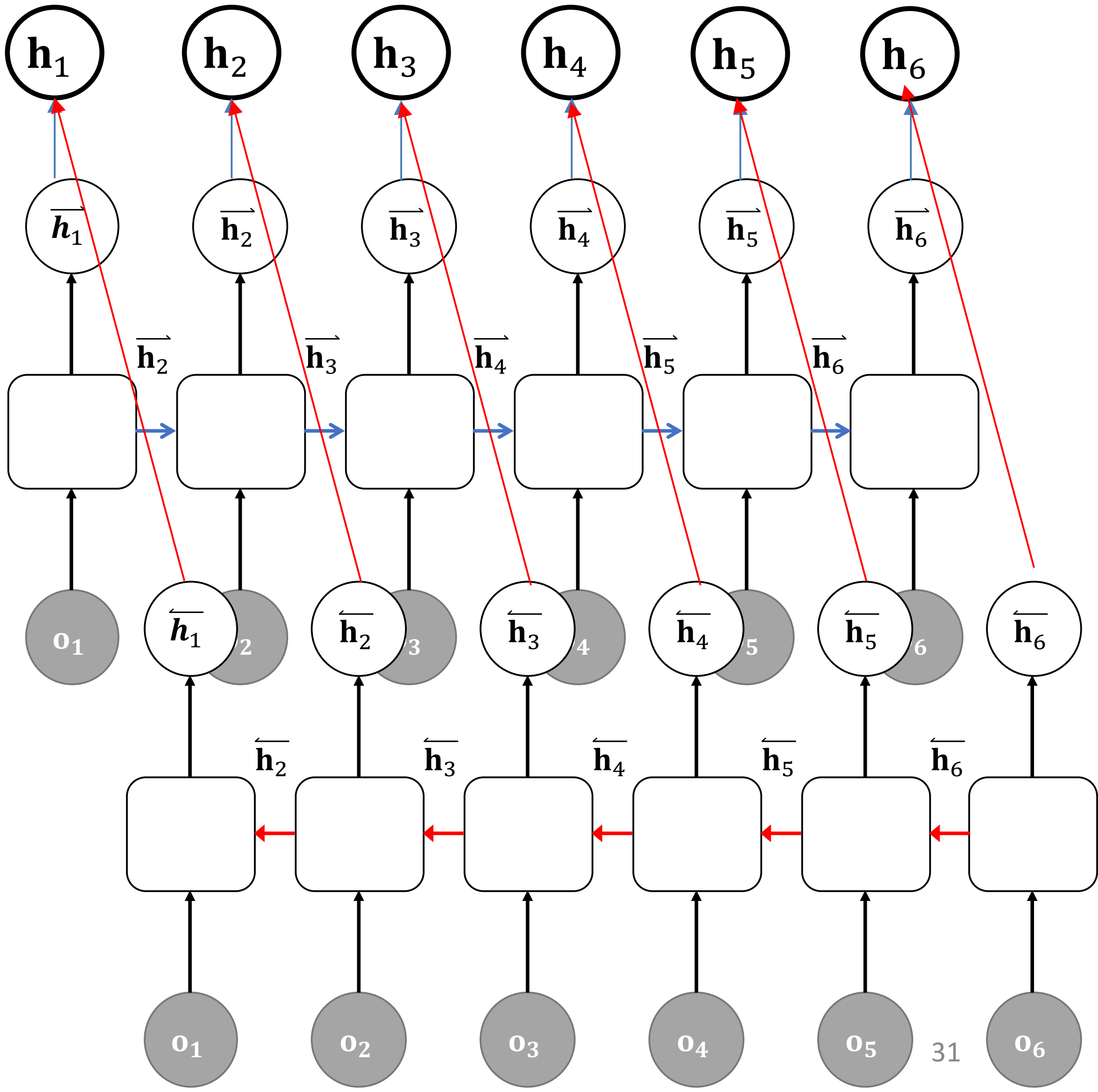
$$\mathbf{h}_t = \mathrm{BLSTM}(O) \text{ for } t = 1, \ldots, T.$$

Bidirectional RNN

- $\mathbf{h}_t = \begin{bmatrix} \overrightarrow{\mathbf{h}_t} \\ \overleftarrow{\mathbf{h}_t} \end{bmatrix} = f(O = (\mathbf{o}_1, \ldots, \mathbf{o}_T))$

Then,

- $p(z_t|O)$
  $\approx p(z_t|\mathbf{h_t})$



CTC Formulation

- CTC Language model (generative model view)

$$
\begin{aligned}
p(C|Z) &= \frac{p(Z|C)p(C)}{p(Z)} \\
&= \prod_{t=1}^{T} p(z_t|z_1, \cdots, z_{t-1}, C)\frac{p(C)}{p(Z)} \\
&\approx \prod_{t=1}^{T} p(z_t|z_{t-1}, C)\frac{p(C)}{p(Z)},
\end{aligned}
$$

  - Using conditional independence assumption (1st order Markov) to factorize the posterior, same as the HMM
  - $p(C)$: Letter language model (we can also combine the word language model)
  - $p(Z)$: Prior probability for the state sequence

Summary of CTC formulation

- $p(C|O)$ is rewritten as follows

$$p(C|O) \approx \sum_{Z} \prod_{t=1}^{T} p(z_t|z_{t-1}, C) p(z_t|O) \frac{p(C)}{p(Z)}$$

- In general, prior probabilities $p(C)$ and $p(Z)$ are separately obtained (not fully end-to-end)
- We can further eliminate the prior probabilities by assuming the uniform distributions as follows ($\mathcal{Z}(C)$ denotes all possible CTC paths given C):

$$p(C|O) \approx \underbrace{\sum_{Z \in \mathcal{Z}(C)} \prod_{t=1}^{T} p(z_t|O)}_{\triangleq p_{\mathrm{ctc}}(C|O)}$$

  – Basically, we can use **a forward-backward algorithm** to estimate the parameter

HMM/DNN vs. CTC

- Conditional independence assumptions
- Language models
- Use of pronunciation lexicon information
- Implementation

Let’s discuss the difference

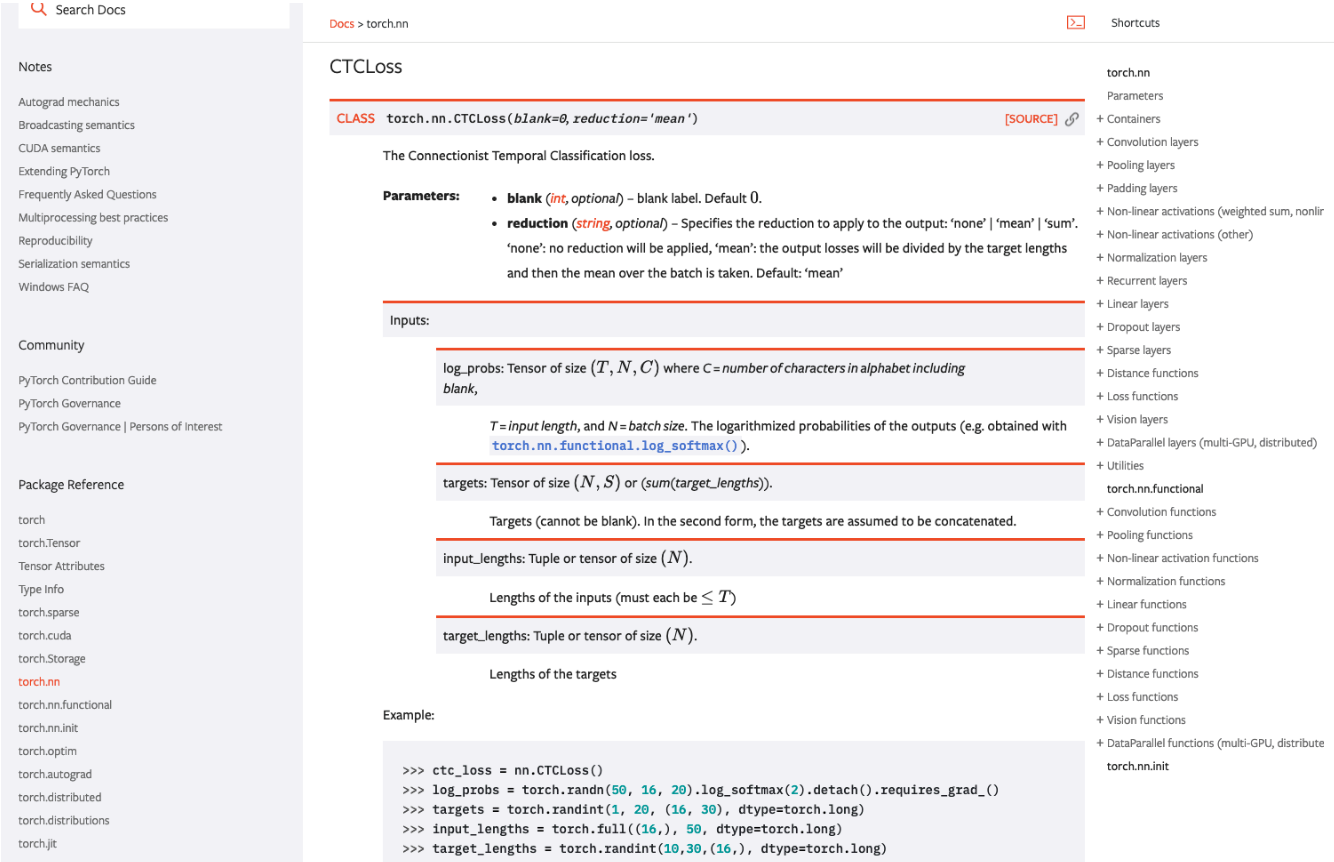
CTC vs. HMM

HMM	CTC
$p(W\|O)$	$p(C\|O)$
• With Bayes rule and CIA (separate acoustic, lexicon, and language models)	• **No Bayes rule**, but CIA (separate acoustic and language model)
$\sum_{Z,L} p(O,Z\|L)p(L\|W)p(W)$	$\sum_{Z} p(C\|Z)p(Z\|O)$
• 1st order Markov and frame-level decomposition	• 1st order Markov and frame-level decomposition
$p(O\|Z)p(Z\|L) \rightarrow \prod_t p(o_t\|z_t)p(z_t\|z_{t-1},L)$	$p(Z\|O)p(C\|Z) \rightarrow \prod_t p(z_t\|O)\, p(z_t\|z_{t-1},C)p(C)$
• Replace the likelihood function $p(o_t\|z_t)$ with a DNN based on the pseudo likelihood trick	• Replace the frame-level posterior distribution $p(z_t\|O)$ with a DNN

Basic assumptions are very similar (CIA, 1st order Markov assumptions)

Implementation of CTC

- During training
  - Major toolkit supports CTC
    - Tensorflow, Pytorch, Chainer, etc.
    - Nvidia cuDNN also supports CTC
- During recognition
  - You have to implement the following search:
    $\text{argmax}_C \, p(C|Z)p(Z|O)$
  - This can be efficiently performed by using a finite state transducer



Baidu CTC
[Amodei+(2015)]

- Optimization of computational cost of CTC dynamic programming
- Multiple GPUs
- Architecture optimization (BLSTM -> GRU, use of CNN)
- Use 12,000 hours of data for training
- Data augmentation (noise)

Read Speech			
Test set	DS1	DS2	Human
WSJ eval'92	4.94	3.60	5.03
WSJ eval'93	6.94	4.98	8.08
LibriSpeech test-clean	7.89	5.33	5.83
LibriSpeech test-other	21.74	13.25	12.69

Google CTC
[Soltau+(2016)]

- Word-level CTC, conventional BLSTM
- No language model
- 125,000 hours of training data (!) from Youtube

Model	Layers	Outputs	Params	Vocab	OOV(%)	Spoken WER(%)	
						w/ LM	w/o LM
~~CTC~~ HMM CD phone	7x1000	6400	43m	500000	0.24	12.3	—
CTC spoken words	7x1000	82473	116m	82473	0.63	11.6	12.0

- Word-level CTC obtains comparable performance (even without LM)

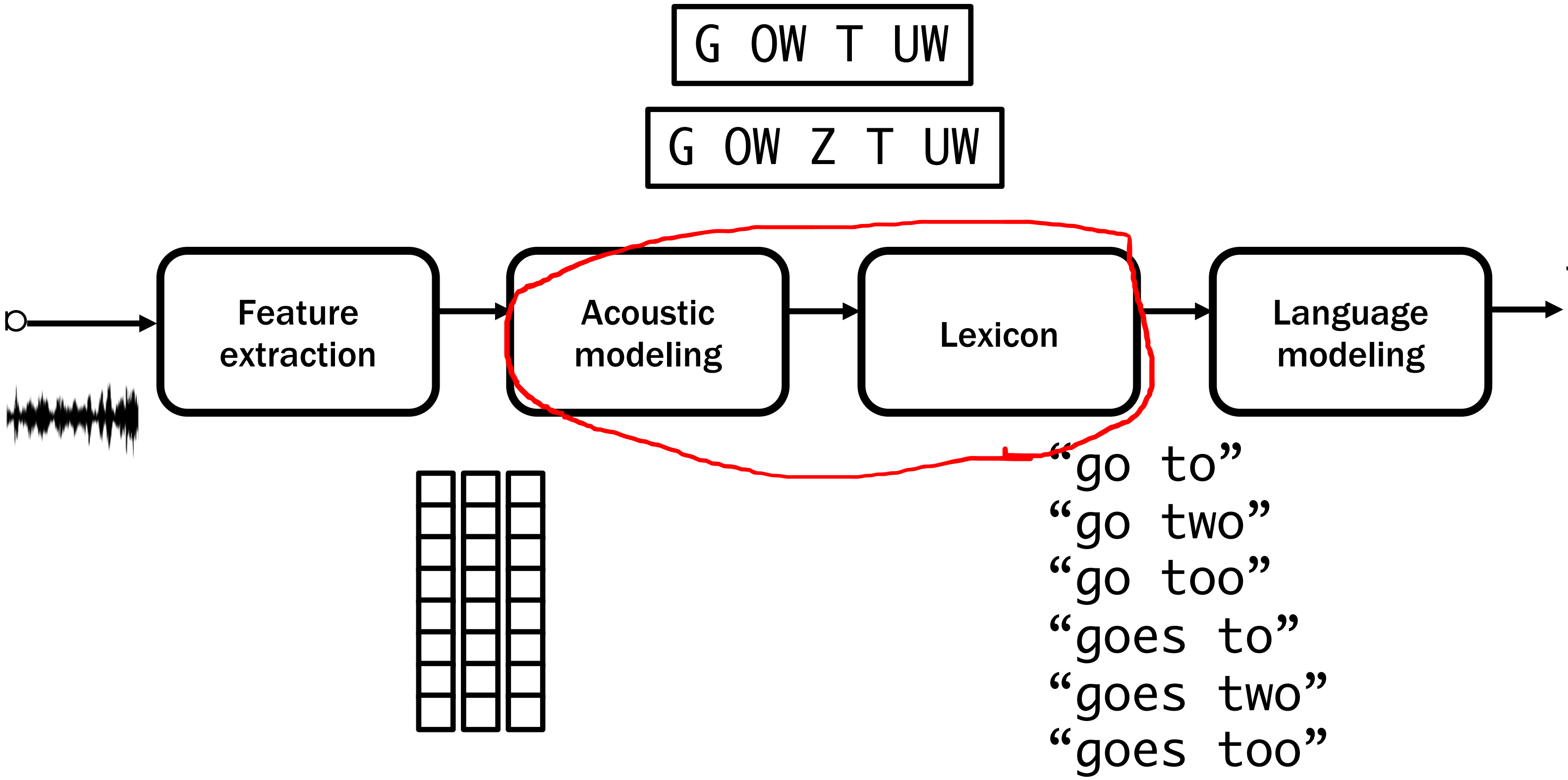
Summary

- CTC
  - One promising direction of end-to-end
  - No language model (but it can be combined with an LM)
  - Still based on conditional independence assumptions and Markov model
  - CTC is really end-to-end?
- Can we use it to any of sequence to sequence task?
  - The alignment should be monotonic (HMM like task, it cannot be applied to machine translation)
  - The input length must be longer than the output length (it cannot be applied to speech synthesis)
- Attention
  - Another end-to-end

Today’s agenda

- Introduction to end-to-end speech recognition
- Connectionist Temporal Classification (CTC)
- Attention

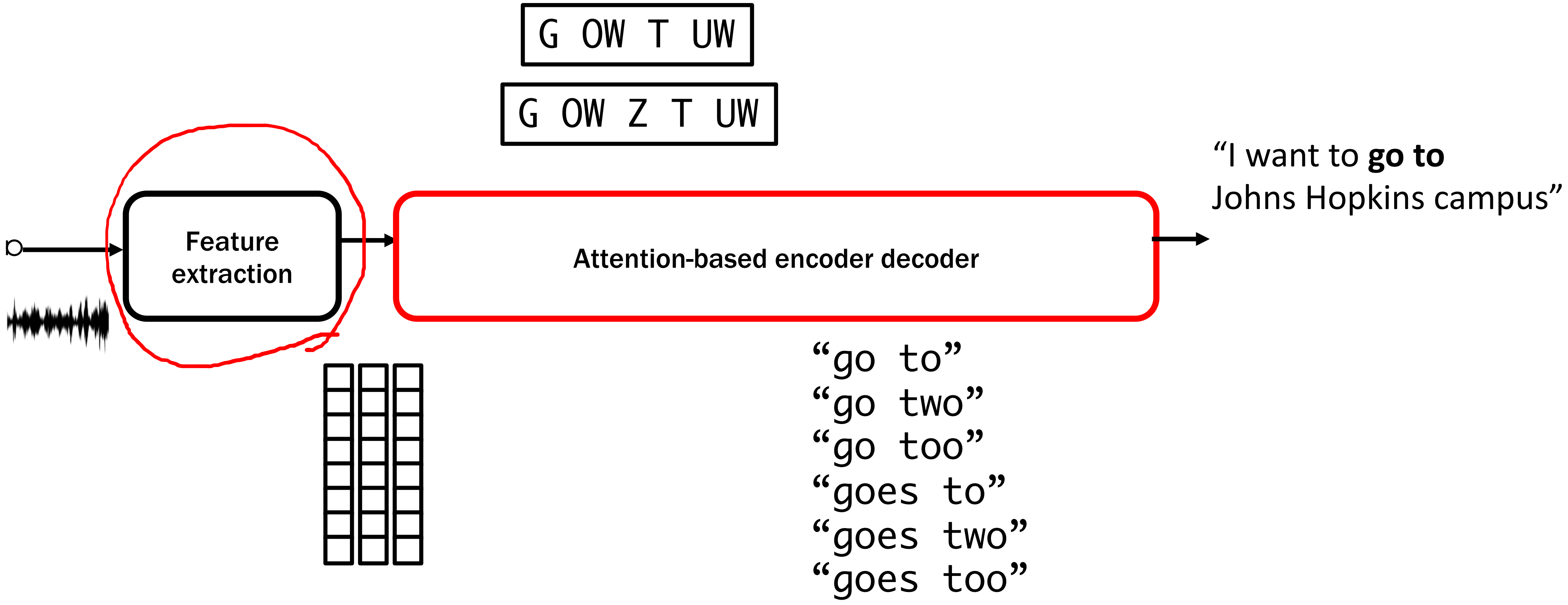
Speech recognition pipeline



“I want to **go to** Johns Hopkins campus”

$$p(O|L) \quad p(L|W) \quad p(W)$$

Speech recognition pipeline



$$p(O|L) \quad p(L|W) \quad p(W)$$

Attention based encoder-decoder

- Let $C = (c_j \in \mathbb{U} | j = 1, \dots, J)$, be a character sequence
  - $\mathbb{U}$: set of characters
- Let $O = (\mathbf{o}_t \in \mathbb{R}^D | t = 1, \dots, T)$, be a sequence of D dimensional feature vectors

$$\hat{C} = \operatorname{argmax}_C p(C|O)$$

- Problem: T and J are different, and we cannot use normal neural networks
- Sequence to sequence is a solution to deal with it

Problem of original encoder-decoder architecture

$$p(C|O) = \prod_j p(c_j | c_{1:j-1}, \mathbf{h'}_T)$$

- We cannot explicitly connect the relationship between input and output (an alignment property)
  - No explicit connection with between frame-level activations $\mathbf{h'}_t$ with output labels y_j

Instead, we consider the following extension

$$p(C|O) = \prod_j p(c_j | c_{1:j-1}, \mathbf{v}_j)$$

$v_j(O)$

- $\mathbf{v}_j$ has an explicit dependency for character c_j

Attention mechanism

$$p(C|O) = \prod_j p(c_j | c_{1:j-1}, \mathbf{v}_j)$$

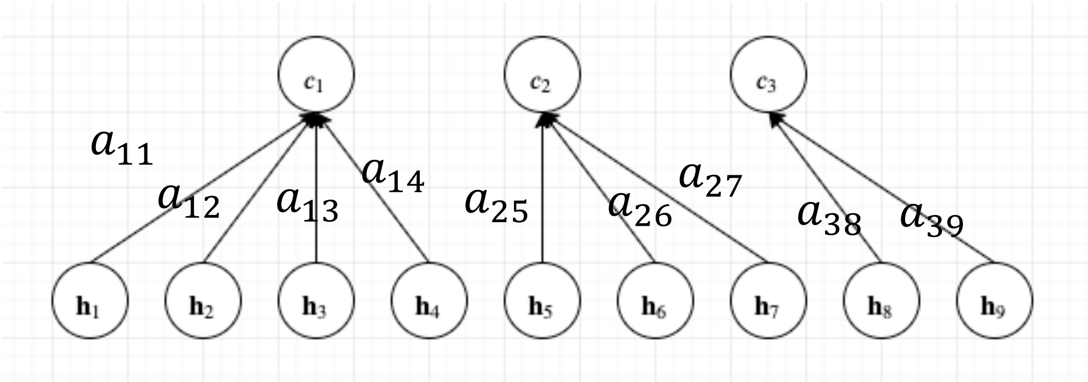
– Obtain the context vector

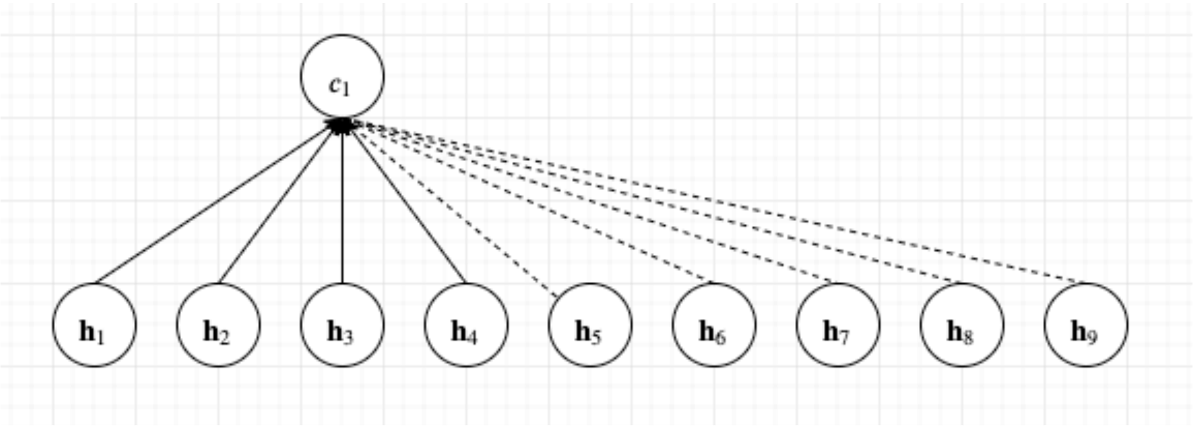
$$\mathbf{v}_j = \sum_{t=1}^{T} a_{jt} \mathbf{h}'_t$$

– Compute the assignment probability for each output j from a neural network

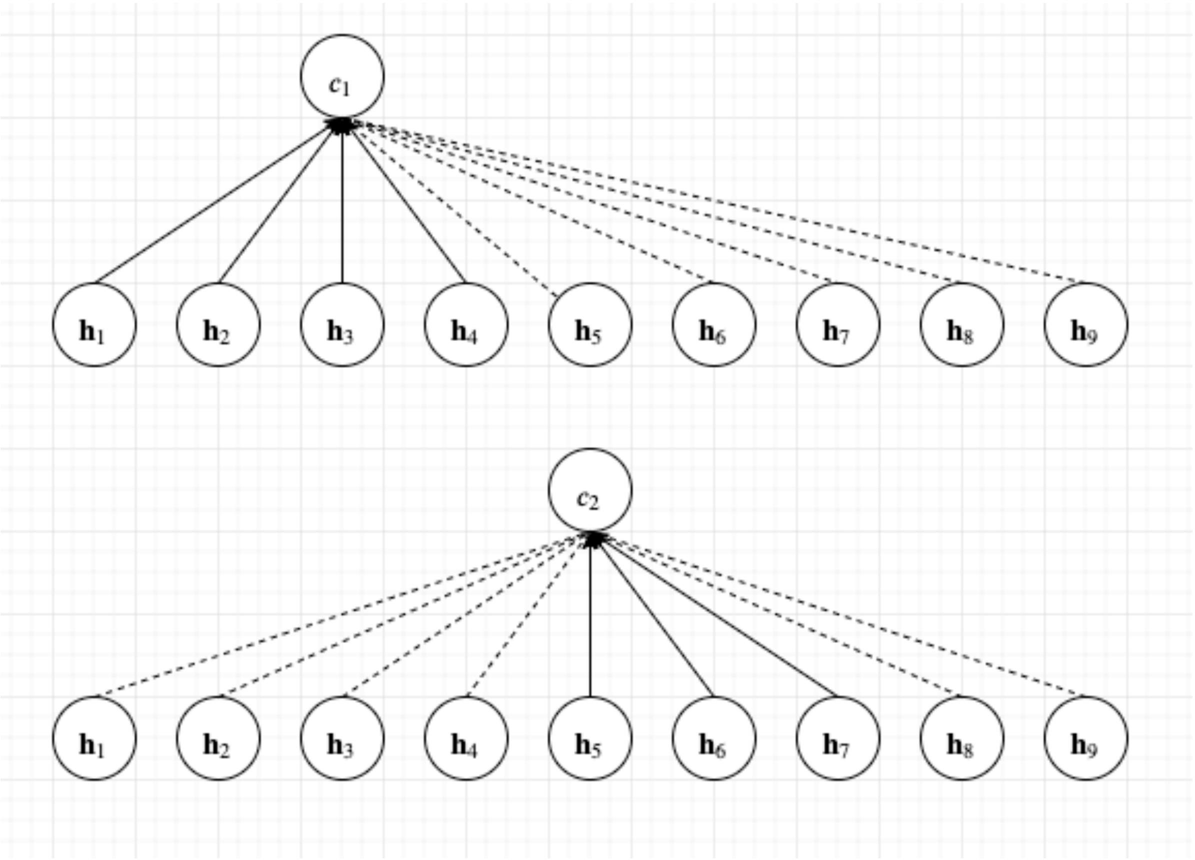
– $\mathbf{a}_j = \{a_{jt} | t = 1, \dots, T\} \in \mathbb{R}^T, 0 < a_{jt} < 1, \sum_{t=1}^{T} a_{jt} = 1$

– a_{jt} is obtained by using a neural network

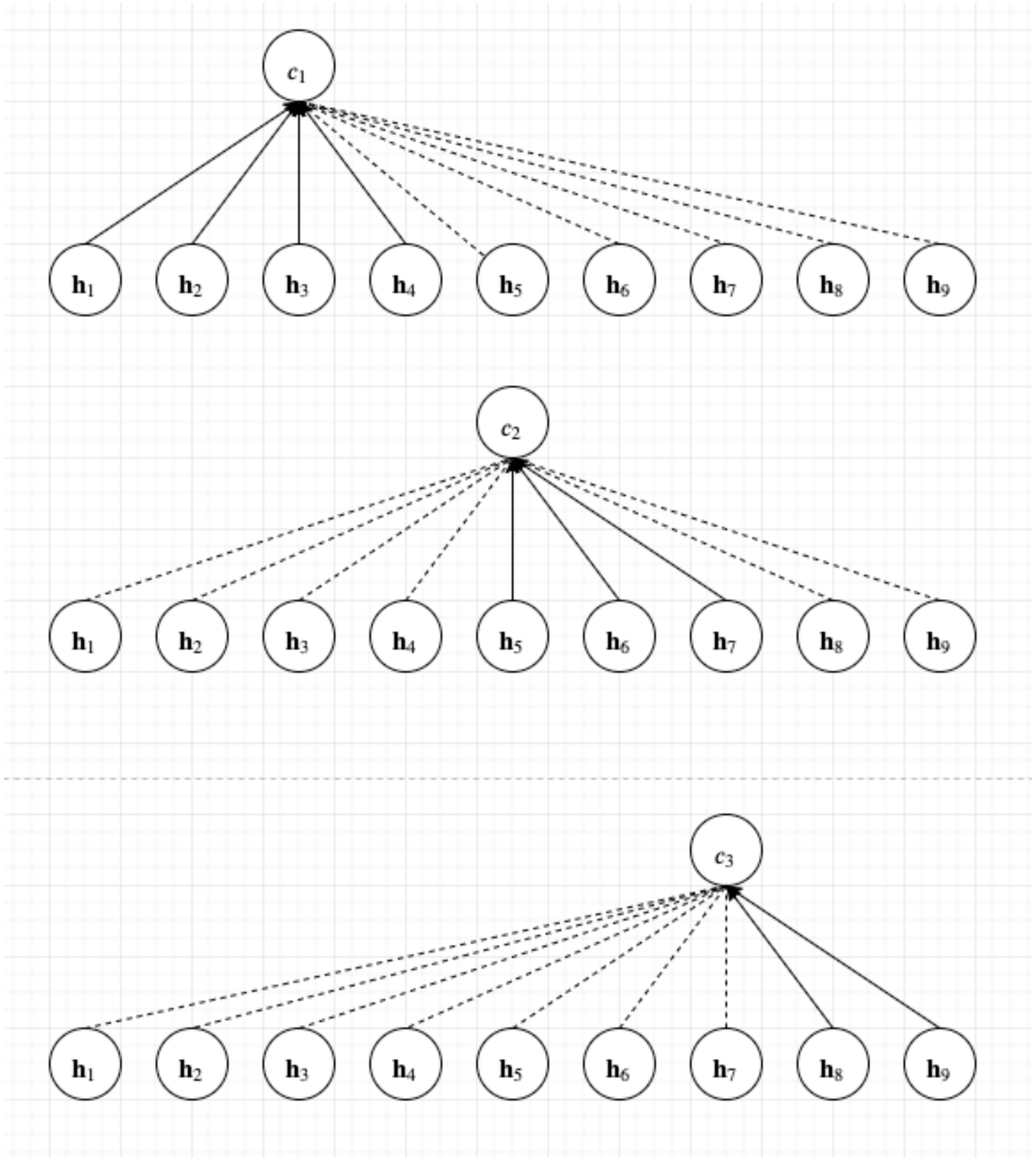




Normal arrow: high probability
Dashed arrow: low probability

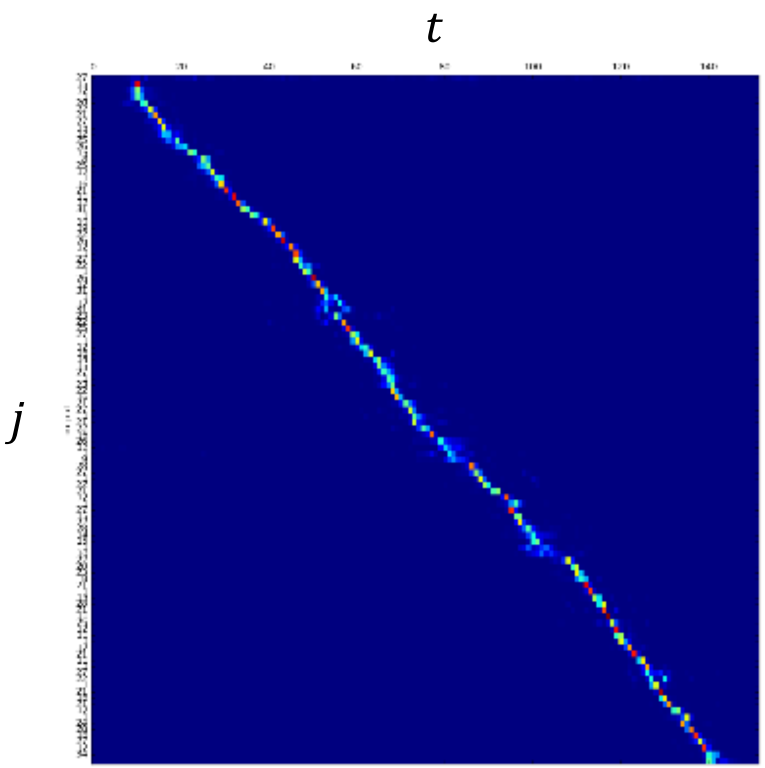


Normal arrow: high probability
Dashed arrow: low probability



Normal arrow: high probability
Dashed arrow: low probability

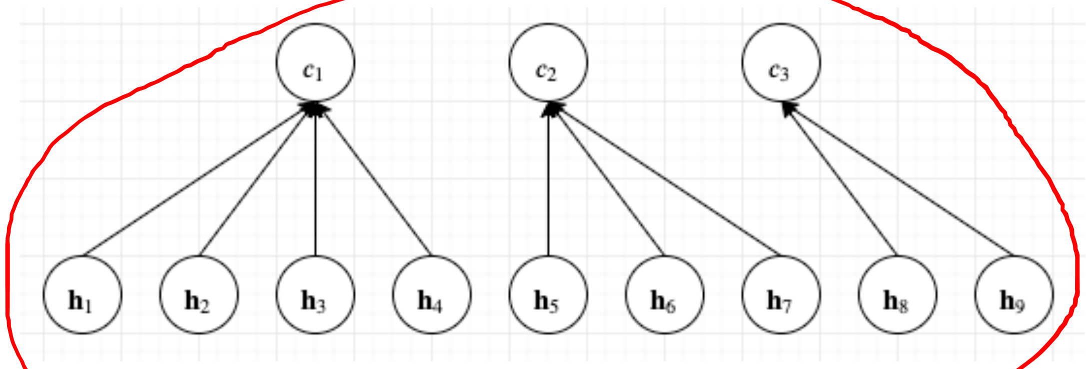
The attention mechanism performs a soft alignment



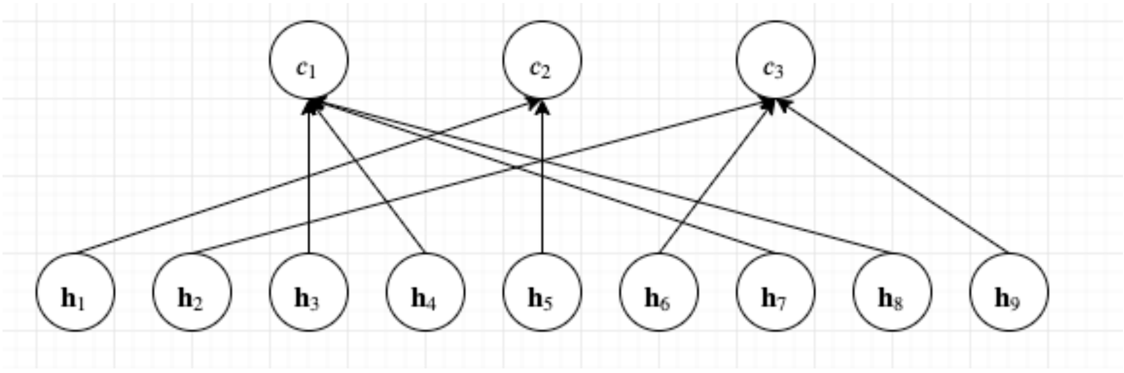
- $\mathbf{v}_j = \sum_{t=1}^{T} a_{jt}\, \mathbf{h}'_t$
- Attention weight a_{jt} determines whether encoder $\mathbf{h}'_t$ is assigned to a character c_j or not
  - $a_{jt} \approx 0$: no assignment
  - $a_{jt} \neq 0$: assigned

The attention mechanism performs a soft alignment

- There is no constraint for the alignment
- The order can be changed (good for machine translation, but it does not happen in speech recognition)

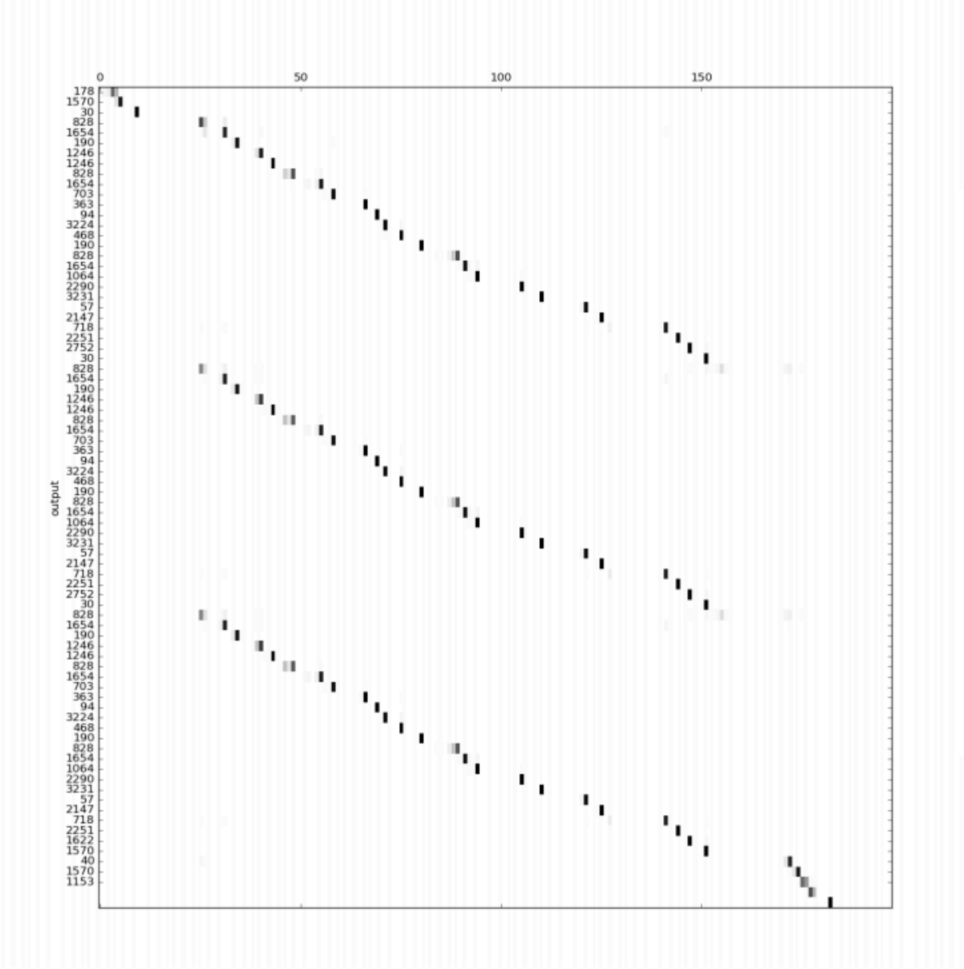


Monotonic



Non monotonic

Examples of wrong alignments



id: (20040717_152947_A010409_B010408-A-057045-057837)

Reference

但是如果你想想如果回到了过去你如果带着这个现在的记
忆是不是很痛苦啊

MTL

Scores: (#Correctness #Substitution #Deletion #Insertion) 28 2 3 45

但是如果你想想如果回到了过去你如果带着这个现在的节
如果你想想如果回到了过去你如果带着这个现在的节如果
你想想如果回到了过去你如果带着这个现在的机是不是很
. . .

Difference in training and recognition

- **During training**
  - $\hat{\theta} = \mathrm{argmax}_\theta \prod_j p(c_j | c_{1:j-1}, \mathbf{v}_j)$
  - We use the transcriptions for $c_{1:j-1}$
- **During recognition**
  - $\hat{C} = \mathrm{argmax}_c \prod_j p(c_j | c_{1:j-1}, \mathbf{v}_j)$
  - However, we don't know the correct transcription $c_{1:j-1}$ during recognition
  - We use estimated history $\hat{c}_{1:j-1}$ instead of the correct transcription

  → mismatch of training and recognition

  - Recognition result is stopped when we observe the "eos" symbol
  - argmax_C is impossible → Approximately only consider possible high-score hypotheses (beam search)

Summary of attention encoder-decoder

- No conditional independence assumption
  - No need for pronunciation lexicon
  - Attention & Encoder: acoustic model
  - Decoder: language model
  - Combine acoustic and language models with single network
- Attention model is too flexible for alignment issues
- Not easy to combine the language model trained with a bunch of text data

Google’s Experiments (12,500 hours)

- They use huge amount of training data (pair data)
  - 12,500 hours
- A lot of techniques in addition to a simple end-to-end ASR
- End-to-End ASR system finally achieved 5.8%
- Classical HMM system (Hybrid DNN/HMM system) 6.7%

Experiments (< 80 hours, Nov 2018)

- Word Error Rate [%] in **English** Wall Street Journal (WSJ) task

Models	dev93	eval92
ESPnet	7.0	**4.7**
Attention model + word 3-gram LM [Bahdanau 2016]	-	9.3
CTC + word 3-gram LM [Graves 2014]	-	8.2
CTC + word 3-gram LM [Miao 2015]	-	7.3
Attention model + word 3-gram LM [Chorowski 2016]	9.7	6.7
Hybrid CTC/attention, multi-level LM	-	**5.6**
Wav2Letter with gated convnet	-	**5.6**
HMM/DNN + sMBR + word 3-gram LM	6.4	3.6
HMM/DNN + sMBR + word RNN-LM	5.6	**2.6**

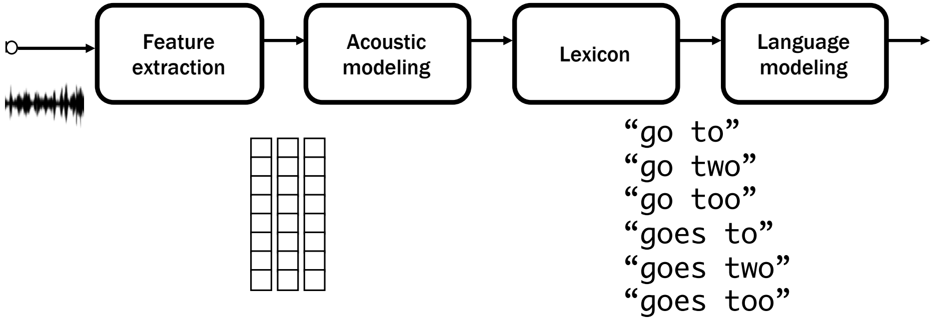
Our best end-to-end

End-to-end best

DNN/HMM (pipeline) best

Why HMM-based classical ASR is better than End-to-End ASR?

- Classical HMM



$$\hat{W} = \underset{W}{\operatorname{argmax}} \max_{L} p(O|L)p(L|W)p(W)$$

$p(O|L)$ $p(L|W)$ $p(W)$

- We can ***separately*** train acoustic, lexicon, and language model
- We can ***incorporate pronunciation dictionary information*** through $p(L|W)$
- We can ***train the language model $p(W)$ only with text*** (newspaper, web, etc.)

On the other hand end-to-end ASR ***always require the pair data*** to train $p(W|O)$

(We can incorporate a language model $p(W)$ by $p(W)^{\alpha}p(W|O)$ heuristically)

HMM/DNN vs. CTC vs. Attention

- Conditional independence assumptions
- Language models
- Use of pronunciation lexicon information
- Implementation

Let’s discuss the difference

CTC vs. HMM vs. Attention

HMM

$$p(W|O)$$

- With Bayes rule and CIA (separate acoustic, lexicon, and language models)

$$\sum_{Z,L} p(O,Z|L)p(L|W)p(W)$$

- 1st order Markov and frame-level decomposition

$$p(O|Z)p(Z|L) \rightarrow \prod_t p(o_t|z_t)p(z_t|z_{t-1},L)$$

- Replace the likelihood function $p(o_t|z_t)$ with a DNN based on the pseudo likelihood trick

CTC

$$p(C|O)$$

- **No Bayes rule**, but CIA (separate acoustic and language model)

$$\sum_{Z} p(C|Z)p(Z|O)$$

- 1st order Markov and frame-level decomposition

$$p(Z|O)p(C|Z) \rightarrow \prod_t p(z_t|O)\, p(z_t|z_{t-1},C)p(C)$$

- Replace the frame-level posterior distribution $p(z_t|O)$ with a DNN

Attention

$$p(C|O)$$

- **No CIA, no separate LM**

$$\prod_j p(c_j|c_{1:j-1},\mathbf{v}_j)\,,\mathbf{v}_j = \sum_j a_{jt}\mathbf{h}'_t$$