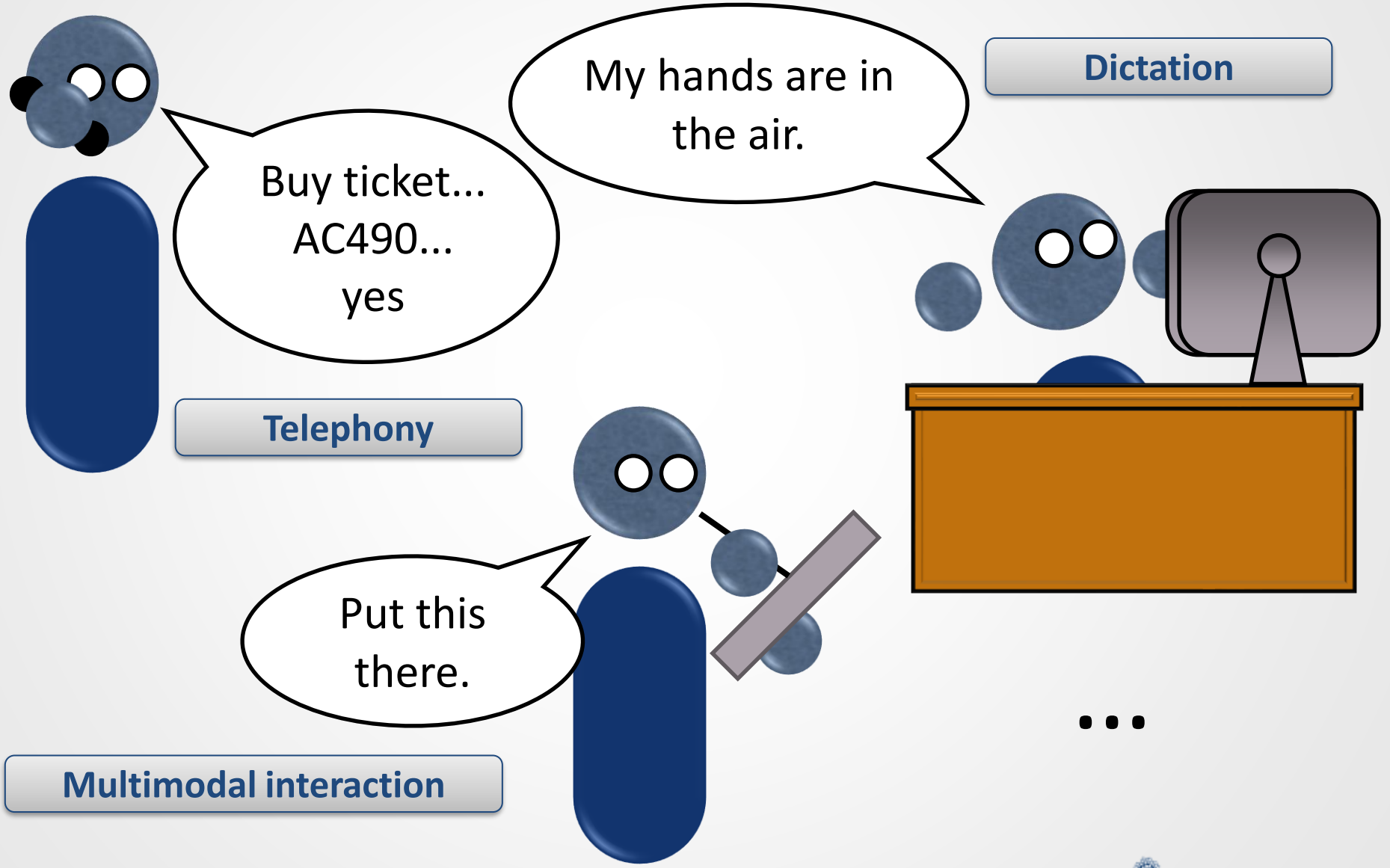
ASR and Dynamic Programming

Consider what we want speech to do



Automatic speech recognition

- Given an utterance, recorded as a waveform…

- Automatic Speech Recognition (ASR), *a.k.a.* speech-to-text (STT), transcribes it as a sequence of tokens, usually words

```
a dog
```

- Though increasingly as sub-words, like
  - Phones: `AH0 D AO1 G`
  - Character-by-character: `a _ d o g`
  - Spans of characters: `a_ do g`

Aspects of ASR systems in the world

- **Speaking mode**: **Isolated** word (e.g., "*yes*") vs. **continuous** (e.g., *"Hey Siri, ask Cortana for the weather")*
- **Speaking style**: **Read** speech vs. **spontaneous** speech; the latter contains many **dysfluencies** (e.g., stuttering, *uh*, *like*, …)
- **Enrolment**: **Speaker-dependent** (all training data from one speaker) vs. **speaker-independent** (training data from many speakers).
- **Vocabulary**: **Small** (<20 words) or **large** (>50,000 words).
- **Transducer**: Cell phone? Noise-cancelling microphone? Teleconference microphone?

Speech features

- Waveform inputs are very, very long
  - Usually: 1 second = 16,000 samples
- Dilated convolutional neural networks (CNNs) can learn & process the waveform directly
- Speech embeddings/representations can be learned with unsupervised objectives
- The log-Mel spectrogram remains a convenient, fast, and widely used pre-processing step
- The result is always a sequence of speech feature vectors spaced 10-30ms apart in time

A neural approach

- We are given a sequence speech features

$$x = x_1, x_2, \ldots, x_T, \qquad x_t \in \mathbb{R}^D$$

- We want a sequence of tokens (a transcription)

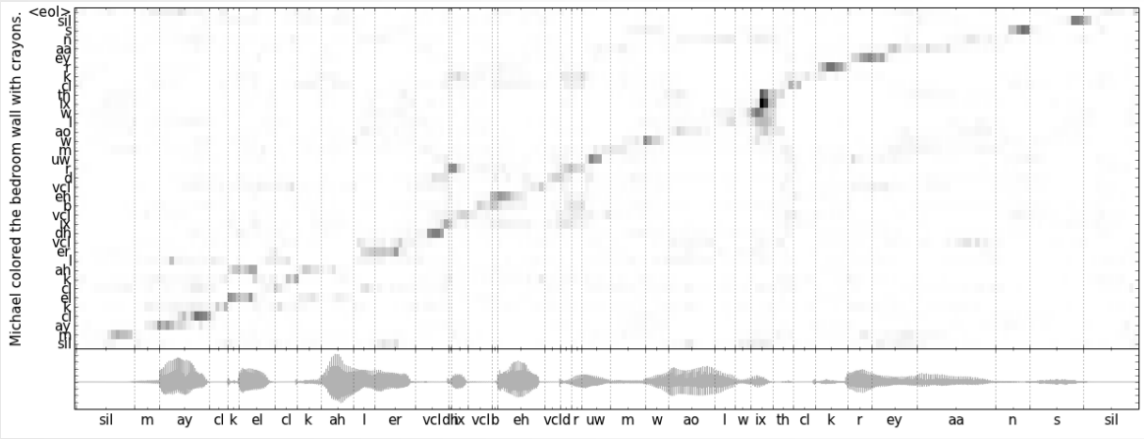
$$y = y_1, y_2, \ldots, y_U, \qquad y_u \in \{1,2, \ldots, V\}$$

  - y_u could be a character, word, phone, *etc.*
- $T \neq U$
- **Sound familiar?**
- We can do encoder/decoder NMT!
  - Source sequence (F) embeddings are now features (x)
  - Target sequences (E) are now transcriptions (y)

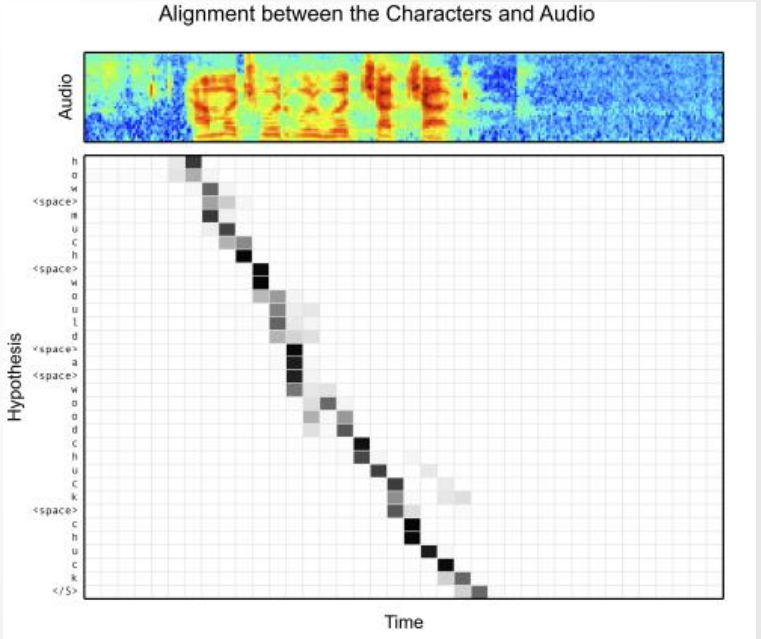
Encoder/decoder ASR

- Networks such as the Attention-based Encoder Decoder or Listen, Attend, and Spell are RNN-based encoder-decoders trained with teacher forcing (ML)

$$\mathcal{L} = -\sum_{u=1}^{U} \log P_\theta(y_u | y_{<u}, x)$$



From Chorowski *et al*. (2014) "End-to-end continuous speech recognition using attention-based recurrent NN: First results"



From Chan *et al*. (2016) "Listen, attend, and spell"

Decoding

- Like in NMT, we approximate the hypothesis transcription

$$y^* = \text{argmax}_y P_\theta(y|x)$$

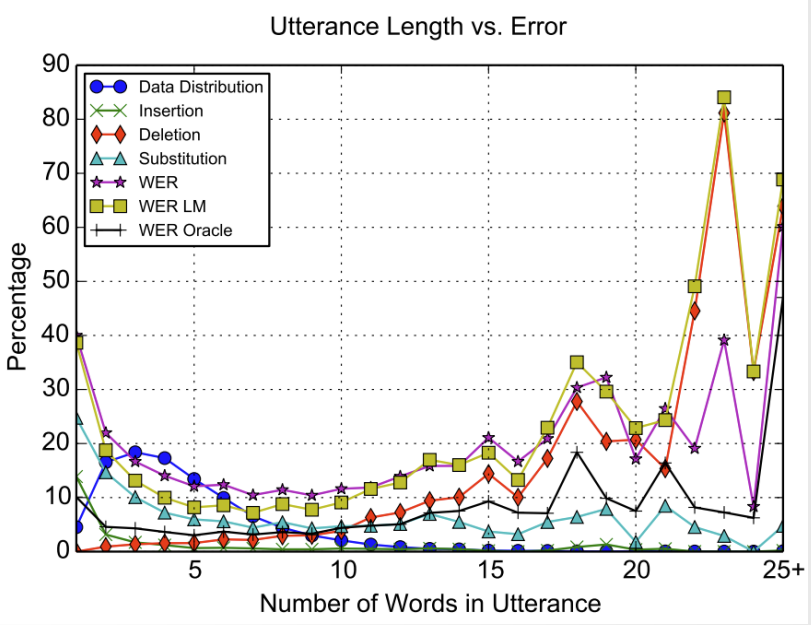
  with the beam search algorithm
- For best performance, an external, auto-regressive language model may be incorporated into each time step via shallow fusion:

$$\underbrace{\log P'_\theta(y_u|y_{<u}, x)}_{\text{Total score}} \approx \underbrace{\log P_\theta(y_u|y_{<u}, x)}_{\text{Encoder-decoder score}} + \underbrace{\lambda \log P_\xi(y_u|y_{<u})}_{\text{External LM score}}$$

- The impact of the external LM grows with λ

Pros and cons

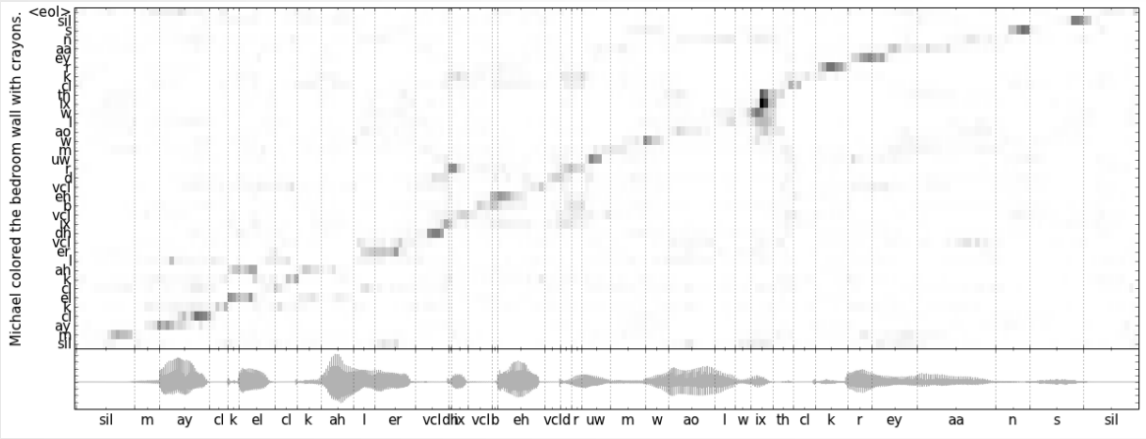
- Encoder/decoder ASR with Transformers is near state-of-the-art on ASR benchmarks
- They do have some drawbacks
  - They are unsuited to streaming (real-time transcription)
  - Performance suffers on long utterances



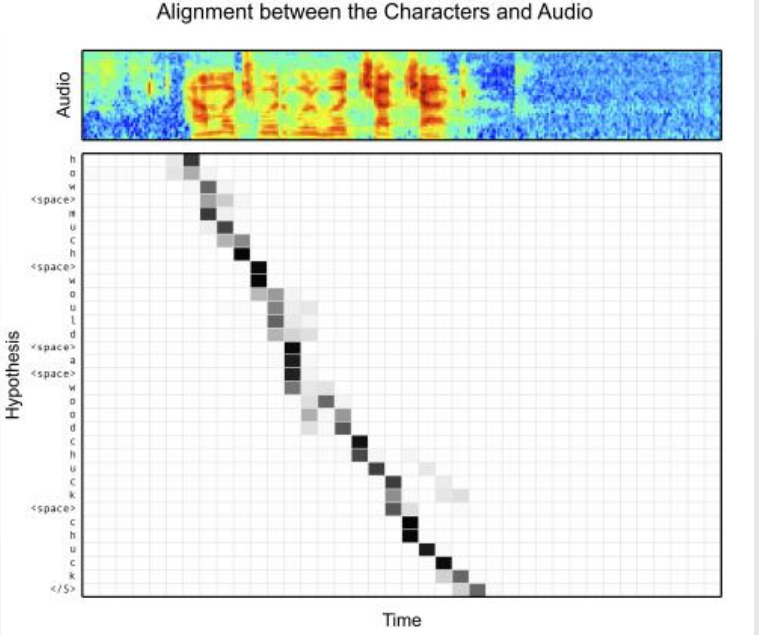
From Chan *et al*. (preprint) "Listen, attend, and spell"

An alternative

- A decoder can attend to any hidden state $h_1, \ldots, h_T$
- This is useful for NMT: tokens or phrases can be re-ordered, added, or removed
- But it is excessive in ASR: tokens are transcribed in the **same order** that they are uttered
- **Can we exploit this monotonicity?**



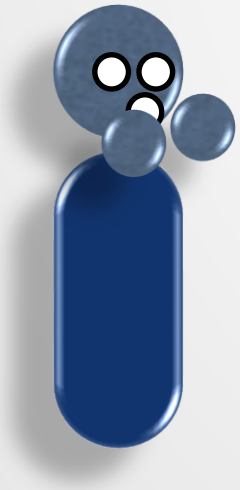
From Chorowski *et al*. (2014) "End-to-end continuous speech recognition using attention-based recurrent NN: First results"



From Chan *et al*. (2016) "Listen, attend, and spell"

Recognizing ~~speakers~~ phones

- A first idea: since GMM can be used to recognize speakers, it can be used to recognize phonemes.
  - For each frame (15~25ms), GMM (or DNN) classifies a phone.
  - Then we can look up pronunciations to convert them into words!



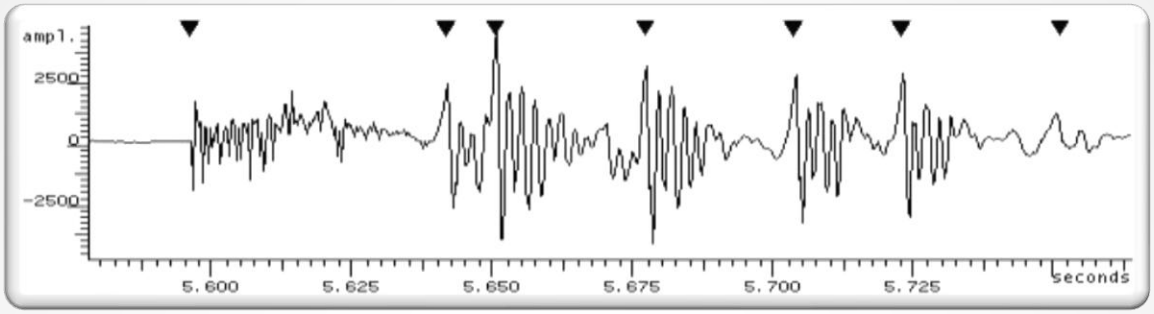
/ow ow ow ow ow p p p p ah ah ah ah ah n/

/ow p ah n /

Open

It's even easier for characters!

Some issues



- Speech **changes** over time
  - Per-frame decisions do not encode label order
  - This is valuable predictive context!
- During training, we don't know every frame's phone label
  - We have "Open", not /ow ow ow ow ow p p p .../
  - How do we maximize the likelihood of "Open"?

Learning alignments

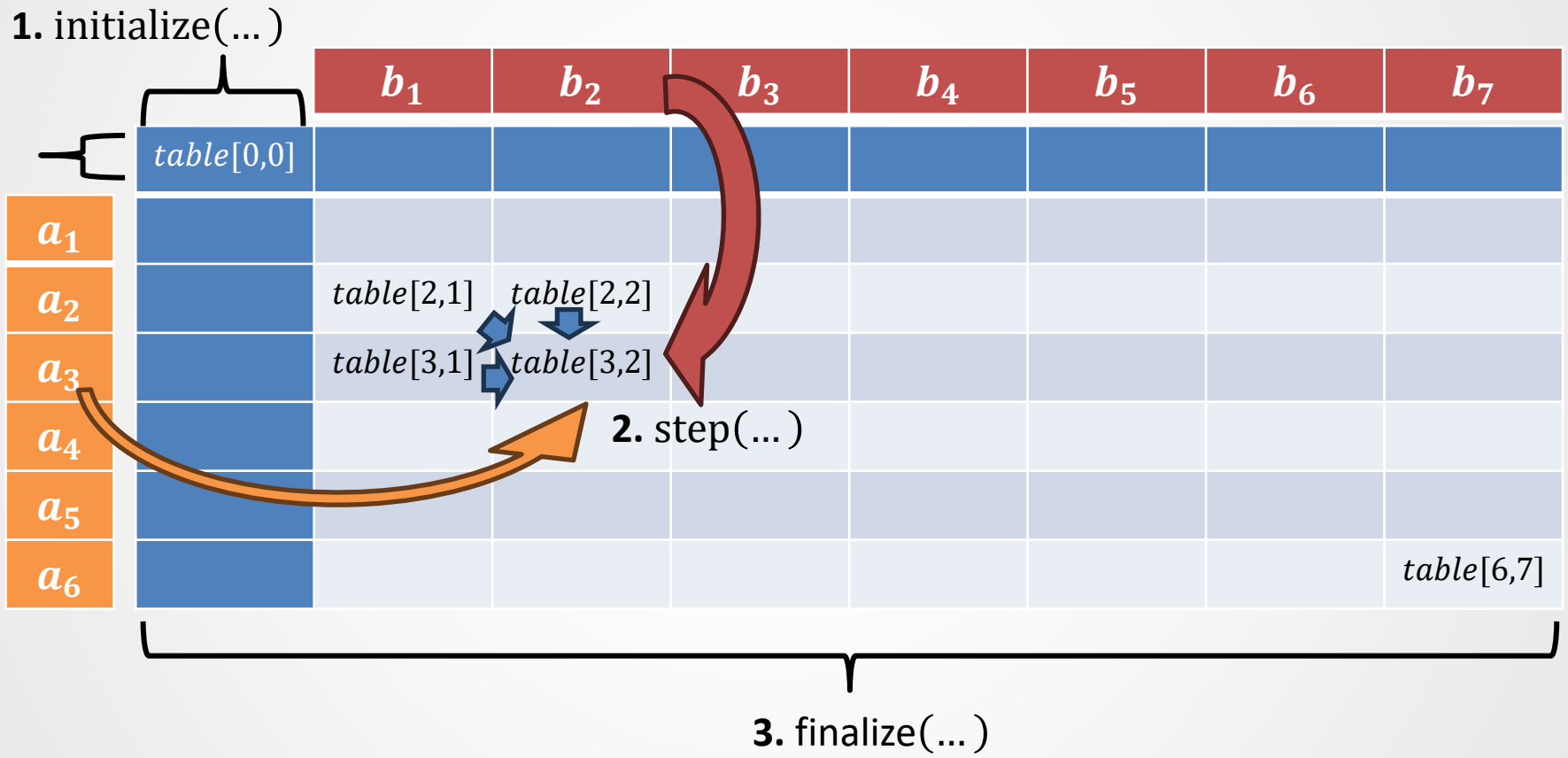
- We can solve both problems by learning monotonic alignments between sequences of frames and tokens
- To do so, both pre-neural and neural ASR rely on Dynamic Programming
  - Pre-neural: Dynamic Time Warping (DTW), HMMs
  - Neural: Connectionist Temporal Classification (CTC), RNN-Transducer (RNN-T)
- DP can be used to align arbitrary sequences a and b
  - Error rates, bitext alignment, phrase-based NMT…
- The forward algorithm applies to all of these

A monotonic forward algorithm

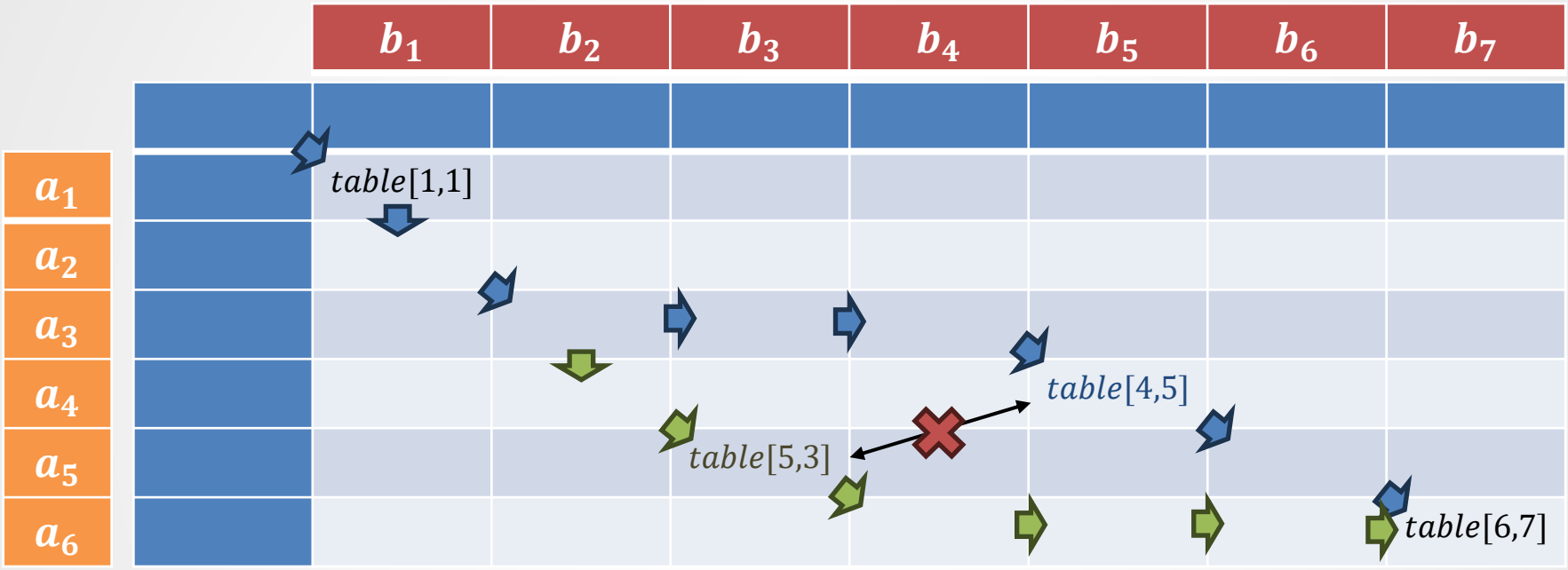
Function monotonic_forward
Inputs $a = a_1, a_2, \ldots, a_U$ **and** $b = b_1, b_2, \ldots, b_T$
1: **Define** $table[0 \ldots U, 0 \ldots T]$
2: initialize$(table[0 \ldots U, 0], table[0,1 \ldots T])$
3: **For each** u **in** $1 \ldots U$:
4: **For each** t **in** $1 \ldots T$:
5: $table[u,t] = \text{step}(a_u, b_t, table[u-1, t-1], table[u-1, t], table[u, t-1])$
6: **Return** finalize$(table[0 \ldots U, 0 \ldots T])$

1: Define $(U+1) \times (T+1)$ table $table$ to store partial results
2: Initialize first row and column of $table$
3+4: Iterate over columns and rows with t and u
4: Use a_u, b_t, and left, up, and diagonal cells to compute $table[u,t]$
5: Use $table$ to compute final results

A tabular representation



Order of operations

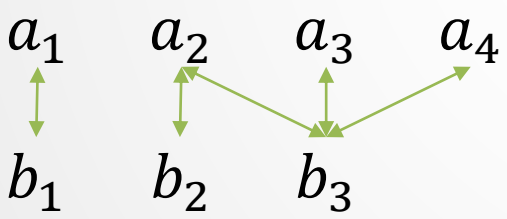


- Cells are populated in **non-decreasing** order of indices
  - $table[u, t]$ depends on all $table[u', t']$ where $u' \leq u$ and $t' \leq t$
  - **Not** when either $u' > u$ and/or $t' > t$

Valid alignment

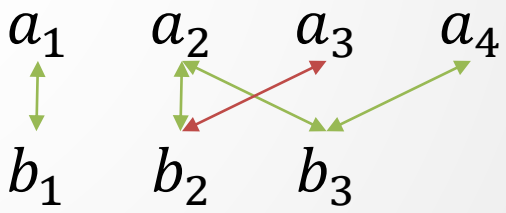
- We use monotonic_forward when solutions involve aligning each element of a to one from b and vice versa
- **All** elements of a and b must be **aligned without crossing**
- The solution may involve one or more alignments

A valid alignment



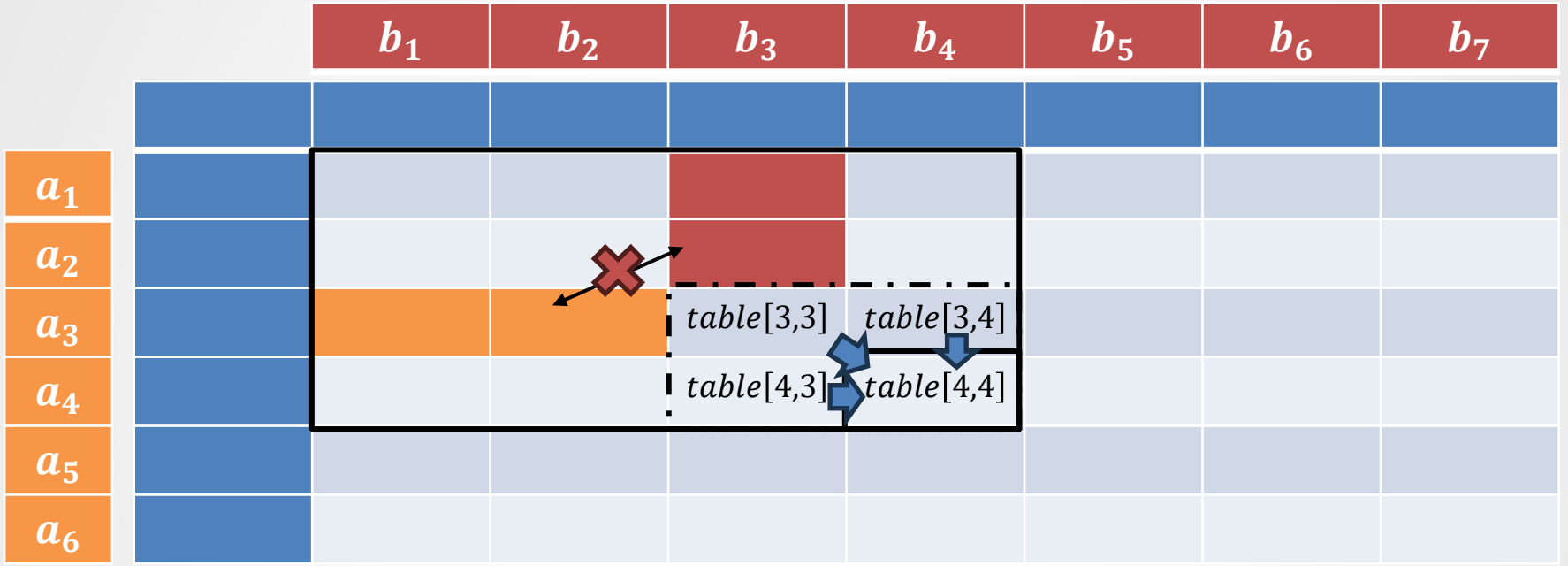
$\{a_1, b_1\} \leq \{a_2, b_2\} \leq \{a_2, b_3\} \ldots$
$\ldots \leq \{a_3, b_3\} \leq \{a_4, b_3\}$

An invalid alignment



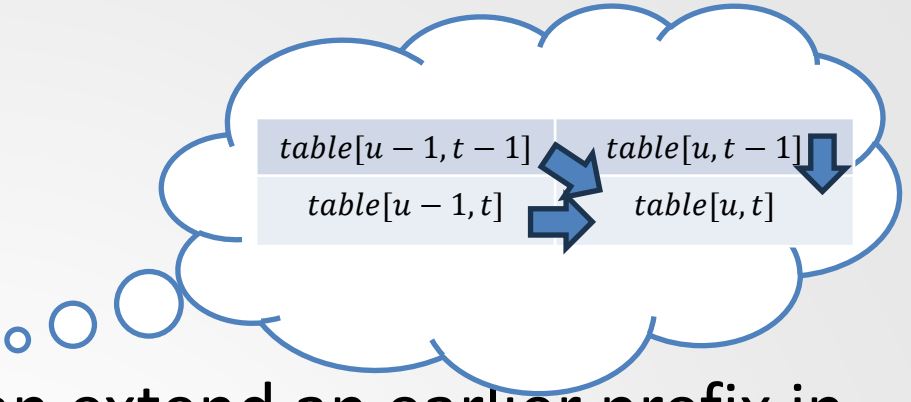
$\{a_1, b_1\} \leq \{a_2, b_2\} \leq \{a_2, b_3\} \ldots$
$\ldots ??? \{a_3, b_2\} \leq \{a_4, b_3\}$

Completeness

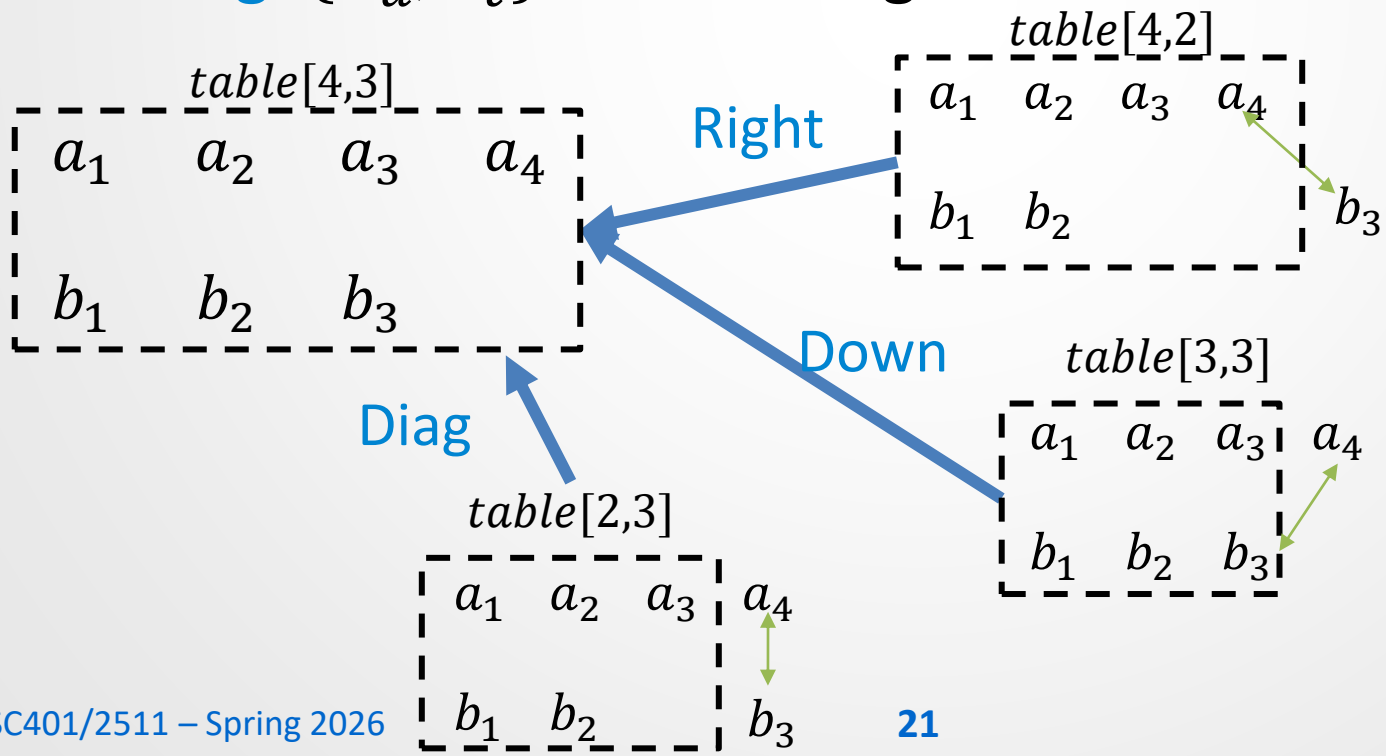


- $table[4,4]$ aligns a_3 to one of $\{b_1, b_2, b_3\}$, b_3 to $\{a_1, a_2, a_3\}$
- It **must** contain one of $\{a_4, b_3\}$, $\{a_3, b_3\}$ or $\{a_3, b_4\}$
  - We extend one of $table[4,3]$, $table[3,3]$, or $table[3,4]$

Extensions

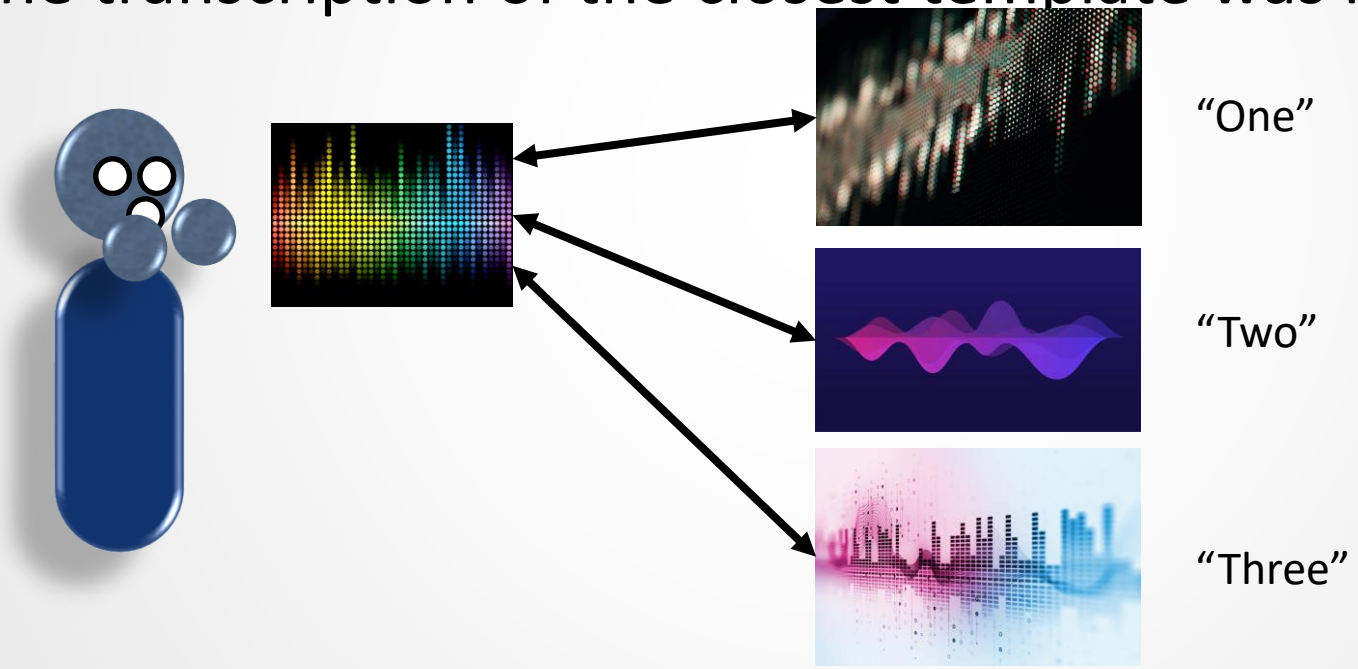


- The alignments in $table[u,t]$ can extend an earlier prefix in one of three ways:
  - Right: b_t extends alignments in $table[u,t-1]$
  - Down: a_u extends alignments in $table[u-1,t]$
  - Diag: $\{a_u, b_t\}$ extends alignments in $table[u-1,t-1]$



Template matching in ASR

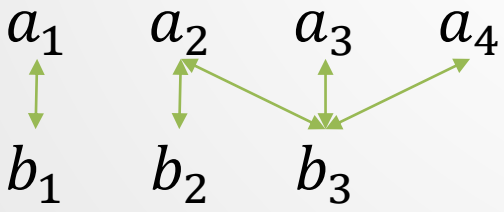
- Early ASR systems were based on template matching
- Input utterances were compared against stored templates
- The transcription of the closest template was returned



- How to deal with stretching and shrinking in time?

Aligning features

- Let $a \in \mathbb{R}^{U \times D}$ and $b \in \mathbb{R}^{T \times D}$ be seqs. of feature frames
  - MFCCs, LPCs, filter bank coefficients, *etc*.
- Choose some frame-wise (vector) distance function
  - E.g. $d(a_u, b_t) = \sum_{d=1}^{D} \left| a_{u,d} - b_{t,d} \right|$
- We can sum those frame-wise distances in a monotonic alignment to get a "distance" between utterances!



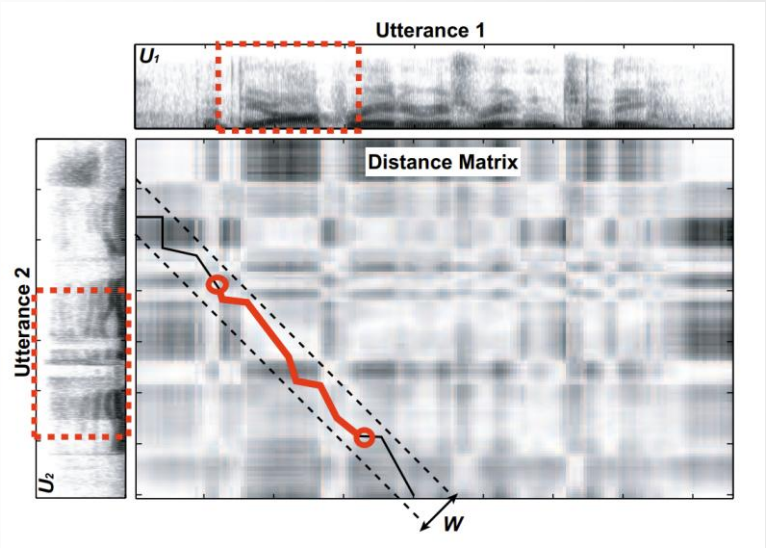
$$d(a_1, b_1) + d(a_2, b_2) + d(a_2, b_3) + d(a_3, b_3) + d(a_4, b_3)$$

Dynamic time warping

- There are many possible ways to align a and b (call them $\mathcal{A}$)
- In Dynamic Time Warping, *the* "distance" between a and b is that with the minimal frame-wise sum

$$\mathcal{D}_d(a,b) = \min_{\alpha \in \mathcal{A}} \sum_{\{u,t\} \in \alpha} d(a_u, b_t)$$

- $\mathcal{D}_d(a,b)$ can be computed with monotonic_forward



From Park and Glass (2006) "Towards unsupervised pattern discovery in speech"

Specifying DTW

1. What are a_u and b_t? → Feature vectors of utts a and b
2. What is a solution to a prefix?
   $\rightarrow table[u,t] = \mathcal{D}_d(a_{1,\ldots,u}, b_{1,\ldots,t})$
3. How do we build the initial prefix(es)?
   $\rightarrow table[1,1] = d(a_1, b_1)$
4. How do we extend a prefix correctly?

$$\rightarrow table[u,t] = d(a_u, b_t) + \min \begin{cases} table[u-1,t] \\ table[u-1,t-1] \\ table[u,t-1] \end{cases}$$

5. How is the result computed from $table$?
   $\rightarrow \mathcal{D}_d(a,b) = table[U,T]$

Adapting monotonic_forward

Function monotonic_forward
Inputs $a = a_1, a_2, \ldots, a_U$ **and** $b = b_1, b_2, \ldots, b_T$
1: **Define** $table[0 \ldots U, 0 \ldots T]$
2: initialize($table[0 \ldots U, 0]$, $table[0,1 \ldots T]$)
3: **For each** u **in** $1 \ldots U$**:**
4: **For each** t **in** $1 \ldots T$**:**
5: $table[u,t] = \text{step}(a_u, b_t, table[u-1, t-1], table[u-1, t], table[u, t-1])$
6: **Return** finalize($table[0 \ldots U, 0 \ldots T]$)

initialize: set $table[0,0] = 0$, $table[1 \ldots U, 0] = table[0,1 \ldots T] = \infty$

$$\text{step: } table[u,t] = d(a_u, b_t) + \min \begin{cases} table[u-1, t] \\ table[u-1, t-1] \\ table[u, t-1] \end{cases}$$

finalize: $table[U, T]$

A DTW example

		b_1	b_2	b_3
a_1		??+0	??+3	??+1
a_2		??+1	??+2	??+5
a_3		??+1	??+2	??+4
a_4		??+1	??+0	??+1

Since $d(a_u, b_t)$ is a fixed cost in $table[u, t]$, we denote it as "+ x"

A DTW example

		b_1	b_2	b_3
	0	∞	∞	∞
a_1	∞	??+0	??+3	??+1
a_2	∞	??+1	??+2	??+5
a_3	∞	??+1	??+2	??+4
a_4	∞	??+1	??+0	??+1

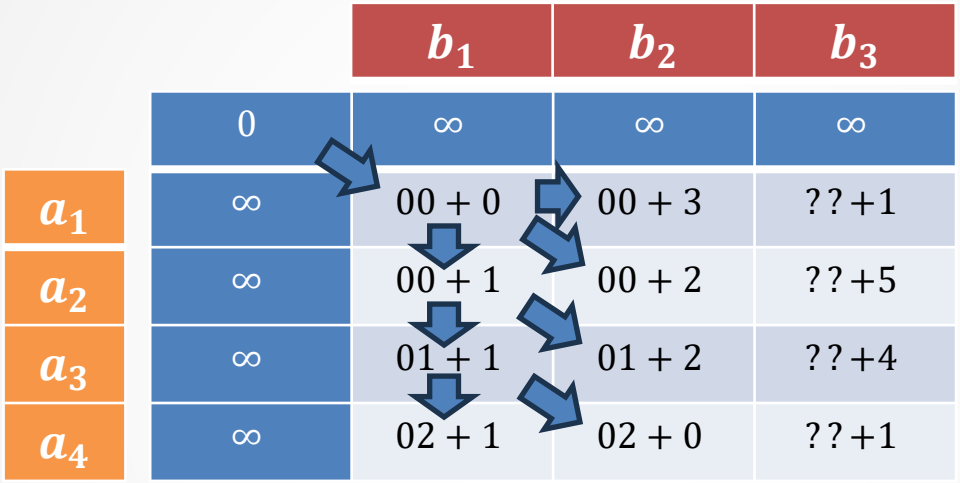
Initialize row 0 and column 0

A DTW example

		b_1	b_2	b_3
	0	∞	∞	∞
a_1	∞	$00+0$	$??+3$	$??+1$
a_2	∞	$00+1$	$??+2$	$??+5$
a_3	∞	$01+1$	$??+2$	$??+4$
a_4	∞	$02+1$	$??+0$	$??+1$

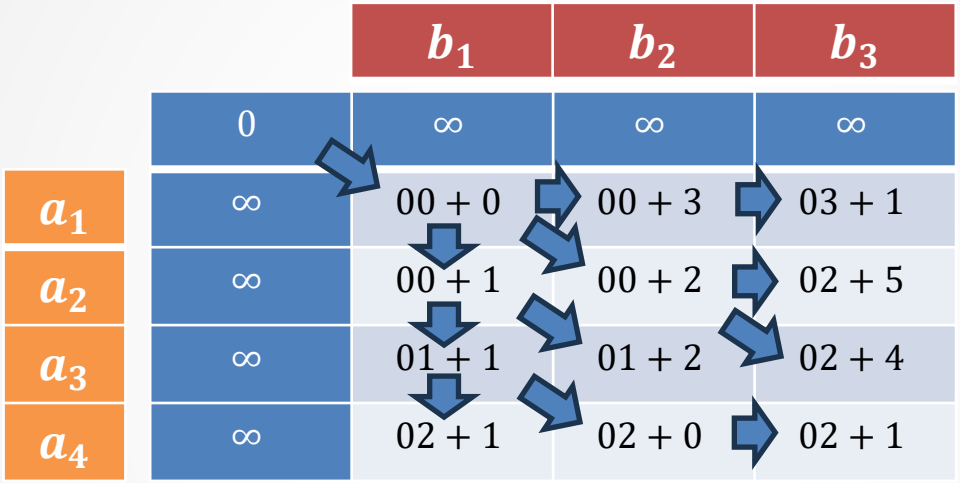
Fill column 1 by row

A DTW example



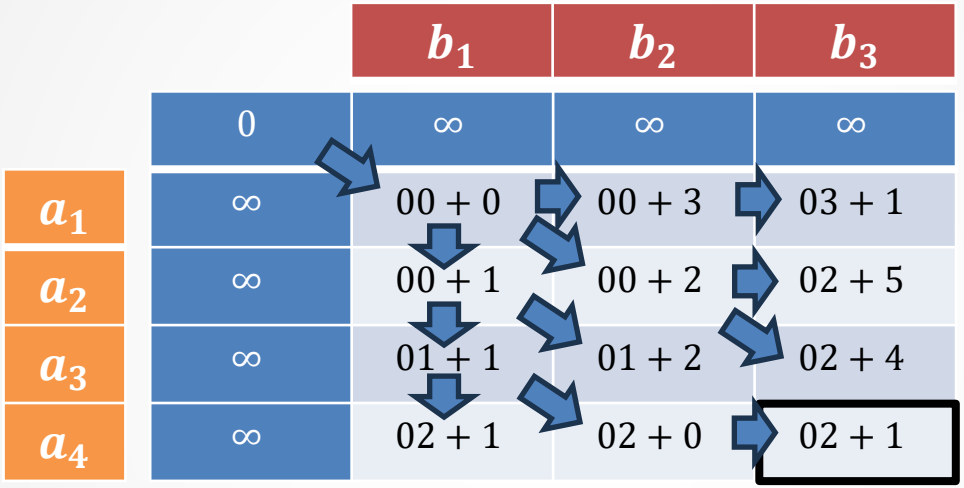
Then column 2

A DTW example



Then column 3

A DTW example



Return $table[4,3] = 3$

ASR evaluation

- ASR systems often generate hypothesis transcripts which don't match the gold-standard reference transcript
- But some are closer than others:
  - Reference: `errors are common here`
  - Hypothesis 1: `his errors are commas here`
  - Hypothesis 2: `here are are`
- We may describe the errors in terms of transformations:
  - Hypothesis 1 inserted `his` and substituted `commas` for `common`
  - Hypothesis 2 substituted `here` for `errors`, `are` for `common`, and deleted `here`
    - The `here`s cannot match without re-ordering

Word-error rate (WER)

- ASR enthusiasts are often concerned with **word-error rate (WER)**, which counts the number of 3 types of errors a hypothesis makes given a reference
  - **Substitution error**: One word being mistaken for another
    e.g., hyp: **`shift`**, ref: `ship`
  - **Deletion error**: An input word that is 'skipped'
    e.g., hyp: `I Torgo`, ref: `I` **`am`** `Torgo`
  - **Insertion error**: A 'hallucinated' word not in the input.
    e.g., hyp: **`steamed`** `hams`, ref: `hams`
- Given S substitutions, D deletions, I insertions, and N reference tokens:

$$WER = \frac{S + D + I}{N} \times 100\%$$

(this is not a valid percentage)

WER through Aligning strings

- We can count errors by building up prefixes of alignments between reference (a) and hypothesis (b)
  - Insertion: add b_t to hypothesis,
- Prepend ref + hyp with a special token <s> to allow for insertions/deletions at the beginning of the reference/hypothesis

A WER example

	<s>	his	errors	are	comma	here
<s>						
errors						
are						
common						
here						

Rows (a_u) are reference tokens, columns (b_t) are hypothesis tokens

A WER example

	<s>	his	errors	are	comma	here
<s>	0	1	2	3	4	5
errors	1					
are	2					
common	3					
here	4					

- Initialize row 0 and column 0
  - Row 0 inserts hypothesis tokens
  - Column 0 deletes references tokens

A WER example

	<s>	his	errors	are	comma	here
<s>	0	1	2	3	4	5
errors	1	1				
are	2	2				
common	3	3				
here	4	4				

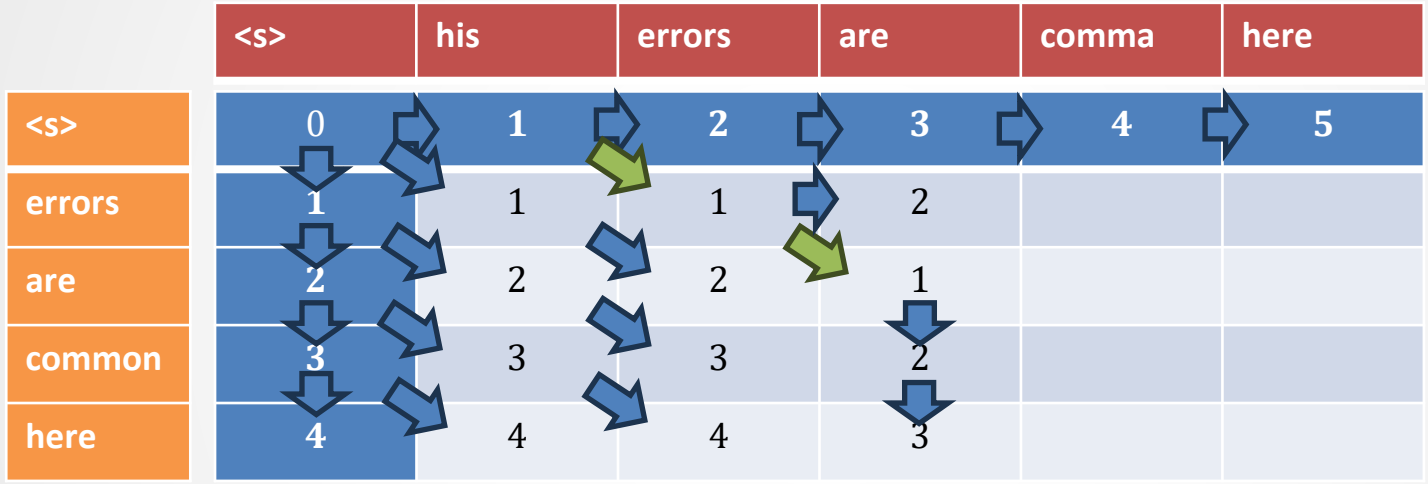
- Fill column 1

A WER example

	<s>	his	errors	are	comma	here
<s>	0	1	2	3	4	5
errors	1	1	1			
are	2	2	2			
common	3	3	3			
here	4	4	4			

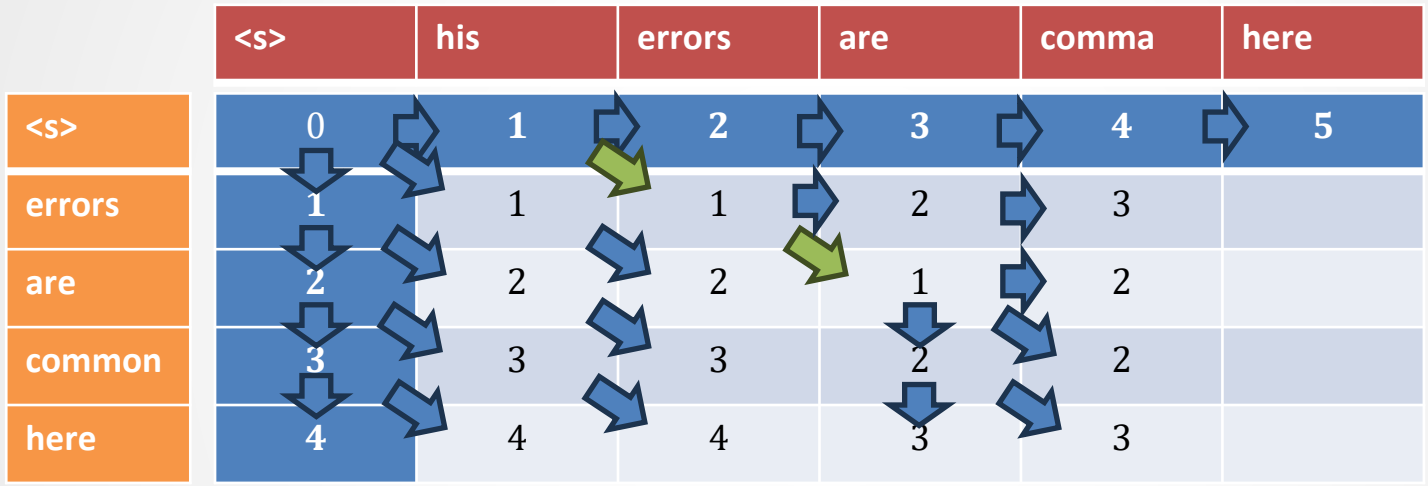
- Fill column 2
  - “errors” is a match coming from row 0

A WER example



- Fill column 2
  - “are” is a match coming from row 1

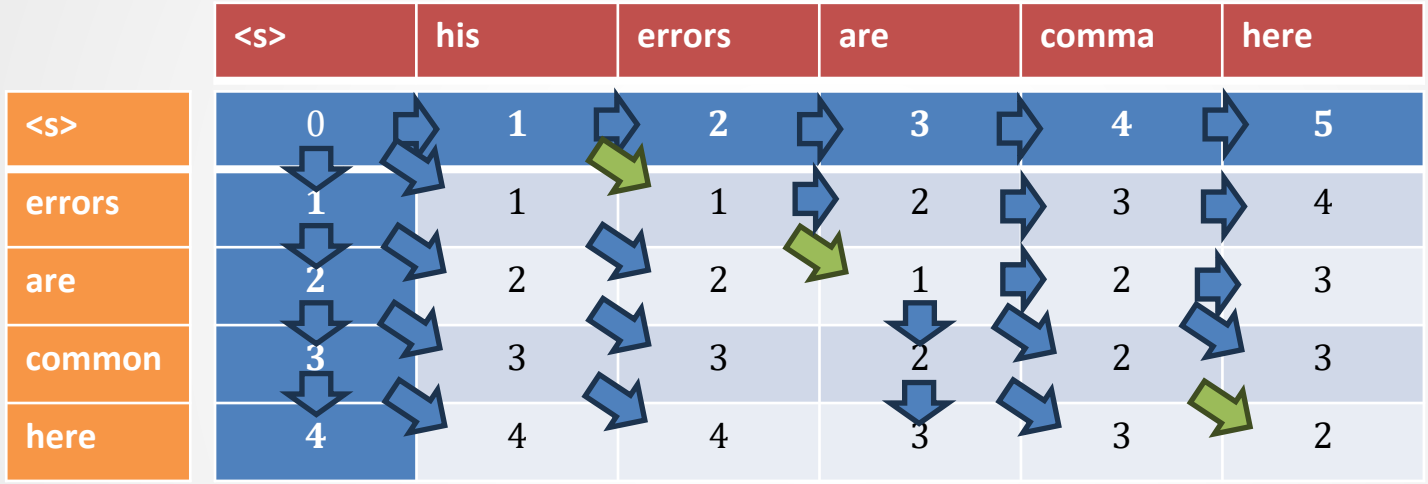
A WER example



	<s>	his	errors	are	comma	here
<s>	0	1	2	3	4	5
errors	1	1	1	2	3	
are	2	2	2	1	2	
common	3	3	3	2	2	
here	4	4	4	3	3	

- Fill column 3

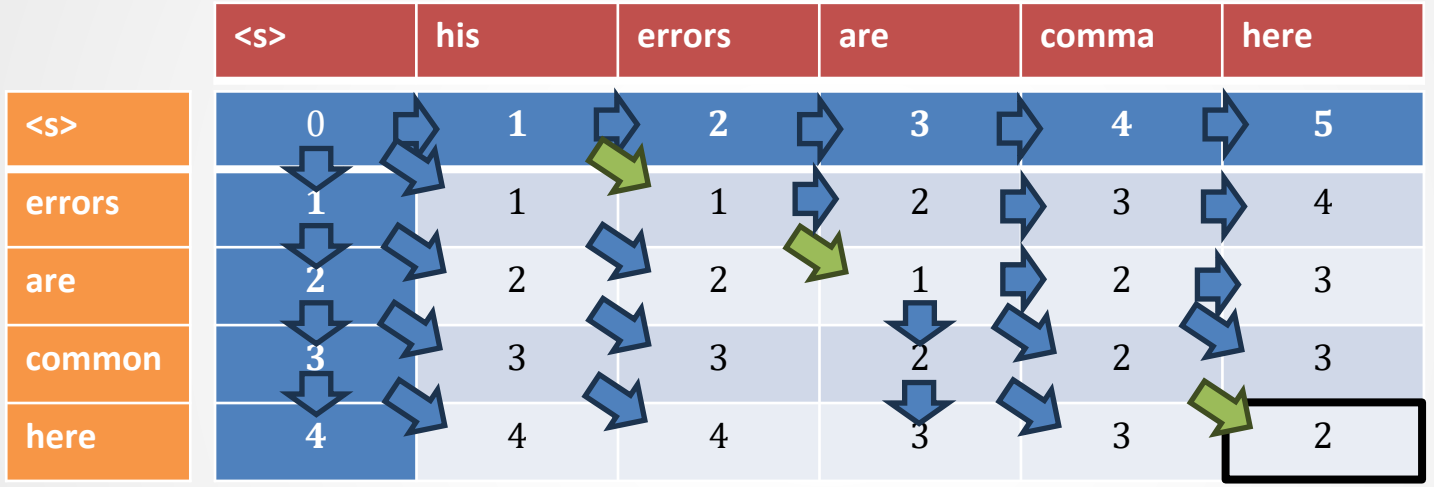
A WER example



	<s>	his	errors	are	comma	here
<s>	0	1	2	3	4	5
errors	1	1	1	2	3	4
are	2	2	2	1	2	3
common	3	3	3	2	2	3
here	4	4	4	3	3	2

- Fill column 4
  - “here” is a match coming from row 4

A WER example



	<s>	his	errors	are	comma	here
<s>	0	1	2	3	4	5
errors	1	1	1	2	3	4
are	2	2	2	1	2	3
common	3	3	3	2	2	3
here	4	4	4	3	3	2

- Return $\frac{table[4,5]}{5} \times 100\% = 40\%$