Automatic speech recognition

Terminological note

This task is traditionally called *automatic speech recognition* (ASR) or simply *speech recognition*.

The term *speech-to-text* is commonly used among non-specialists but is arguably a bit of a shibboleth for specialists.

Downstream applications

- Voice user interfaces (virtual assistants, voice dialing, voice search, etc.)
- Speaker identification (your voice as a password)
- Telephony for the deaf, hearing- or motor-impaired
- Fluency evaluation, accent reduction, etc.
- Keyword spotting (e.g., at the NSA)
- Forced alignment for phonetic analysis
- Diarization (who's talking and when)
- Captioning (though this is mostly done manually on TV)

General issues

- Small vs. large vs. open vocabulary
- Constrained grammars
- Single- vs. multi-speaker
- High-quality vs. low-latency
- Clean environment vs. noisy environment
- Monolingual vs. multilingual vs. code-switched
- Adaptation to non-standard dialects, children, non-native speakers, etc.
- Adaptation to particular domains (medical scribes, speech pathology, assistive technologies, etc.)

Outline

- Evaluation
- A brief history
- Digital representations of speech
- Weighted-finite state automata-based approaches
- Neural network-based approaches
- Software

Evaluation

Word error rate (WER)

Given hypothesis transcription *h* and gold transcription *g*, let edits(*g*, *h*) be the minimum edit distance (or Levenshtein distance) between *g* and *h*. Then

$$\text{WER} = 100 \times \text{edits}(g, h) / |g|$$

NB:

- This is not a probability expressed as a percentage: if *h* is much longer than *g* it will be greater than 100 (though this is rare in practice).
- This is **not** the same WER we're using in earlier homeworks; there, the equivalent of WER would be called *phone error rate* (PER).

Computing WER

- Can be computed in $O(|g||h|)$ time using the Wagner-Fischer (1974) algorithm.
- Alternatively, can be computed by an edit transducer (e.g., see Gorman & Sproat 2021:§7.1) with arbitrary weights for insertion, deletion, substitution.

Wagner-Fischer edit distance algorithm: grid

		S	u	n	d	a	y
S							
a							
t							
u							
r							
d							
a							
y							

Wagner-Fischer edit distance algorithm: initialization

		S	**u**	**n**	**d**	**a**	**y**
	0	1	2	3	4	5	6
S	1	0					
a	2						
t	3						
u	4						
r	5						
d	6						
a	7						
y	8						

Wagner-Fischer edit distance algorithm: match

		S	**u**	**n**	**d**	**a**	**y**
	0	1	2	3	4	5	6
S	1	**0**					
a	2						
t	3						
u	4						
r	5						
d	6						
a	7						
y	8						

Wagner-Fischer edit distance algorithm: deletion

		S	**u**	**n**	**d**	**a**	**y**
	0	1	2	3	4	5	6
S	1	0					
a	2	**1**					
t	3						
u	4						
r	5						
d	6						
a	7						
y	8						

Wagner-Fischer edit distance algorithm: final state

		S	**u**	**n**	**d**	**a**	**y**
	0	1	2	3	4	5	6
S	1	0	1	2	3	4	5
a	2	1	1	2	3	3	4
t	3	2	2	2	3	4	4
u	4	3	2	3	3	4	5
r	5	4	3	3	4	4	5
d	6	5	4	4	3	4	5
a	7	6	5	5	4	3	4
y	8	7	6	6	5	4	**3**

Alternative metrics

- *Perplexity*: some work on language modeling just reports perplexity, a measure of corpus likelihood without bothering to plug in a full ASR system.
    - However, it has long been noted that improvements in perplexity need not produce reductions in WER (and may even make things worse), so I consider this practice flawed.
- *Side-by-side system comparison*: given triples of the form h_1, h_2, g where h_1 is the hypothesis transcription from system 1, h_2 the hypothesis transcription from system 2, and g the gold transcription, is h_1 a better, worse, or as-good a transcription of *gi* than h_2?
- Downstream metrics: evaluate the ASR system based on how good it is for whatever you're actually using the transcriptions for.

A brief history of ASR

History of ASR (after Mohri n.d.)

1939: voder and vocoder mechanical synthesizers
1952: single speaker, digit recognition in isolation
1960s: linear predictive coding
1970s: start of (D)ARPA funding of speech understanding research
1980s: *n*-gram based language modeling; mel-frequency cepstral coefficients; hidden Markov models; many new resources (e.g., ATIS); small-vocabulary recognizers.
1990s: discriminative training; vocal tract normalization; large-vocabulary recognizers; WFST algorithms and end-to-end WFST recognition.
2000s: broadcast news and conversational speaker large-vocabulary recognizers; open-source WFST software.
2010s: CTC neural network recognizers

Digital representations of speech

Digital speech

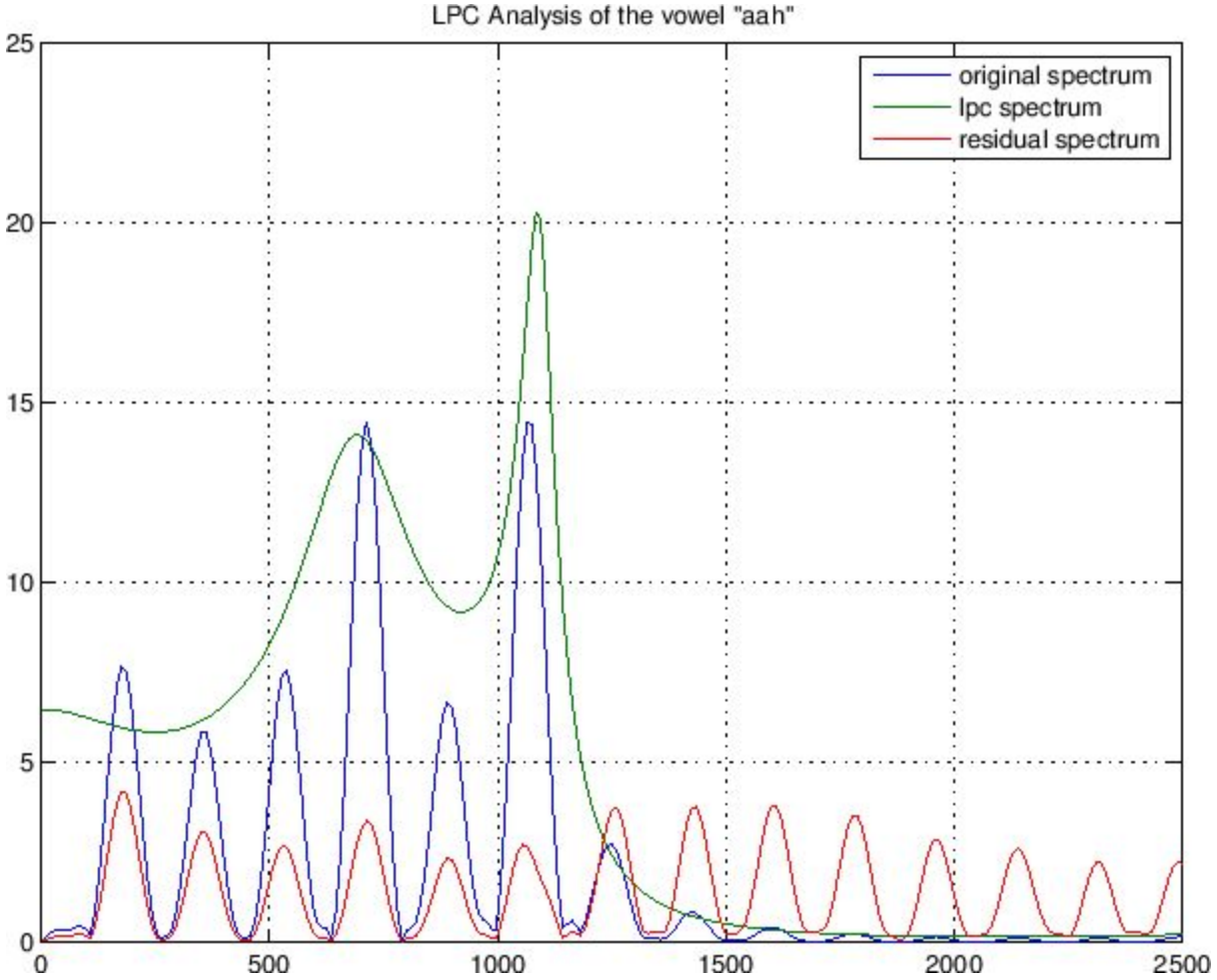
Audio files like `.wav` contain streams of fixed-precision integers sampled at a regular rate.

- A music CD contains 74 or 80 minutes of audio with 16-bit samples at 44,100 Hz; this is roughly 700 MB before compression.
- Formats like MP3 are "lossy" in that they do not perfectly approximate "CD quality" (they contain less detail at frequencies where the human auditory cortex is less likely to notice).
- In contrast, formats like FLAC are "lossless"; they perfectly reconstruct the original audio stream.


vevo

Linear predictive coding

Speech coding refers to techniques for compressing digital audio streams containing speech; e.g., *linear predictive coding* (LPC), the technology that gives us formant analysis, provides the frequencies and bandwidths of *n* poles and antipoles across the frequency spectrum.



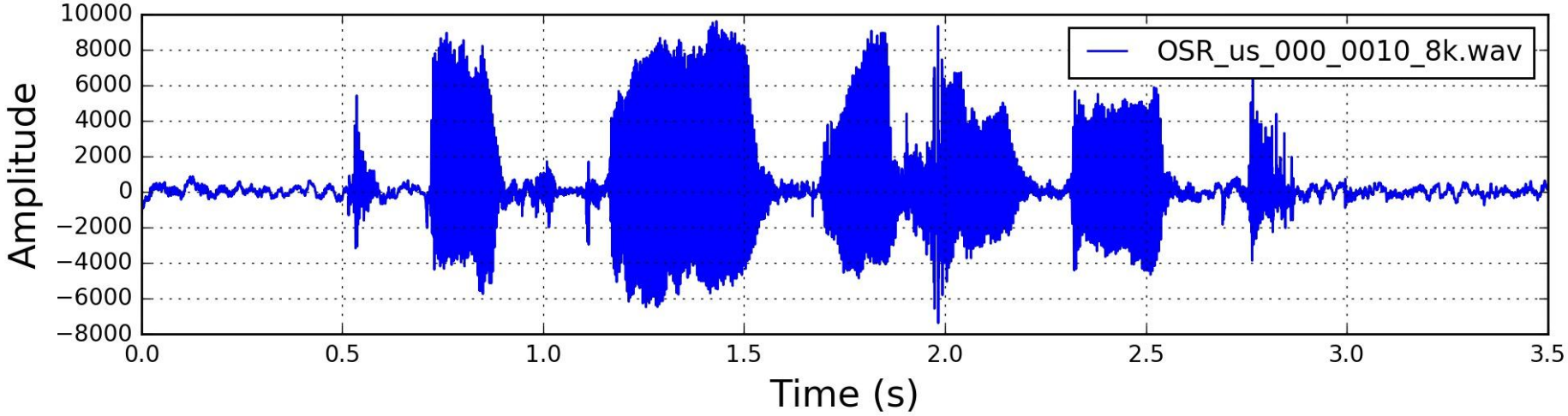
(Image credit: http://musicweb.ucsd.edu/~trsmyth/analysis/LPC_Analysis_in.html)

Mel-frequency cepstral coefficients

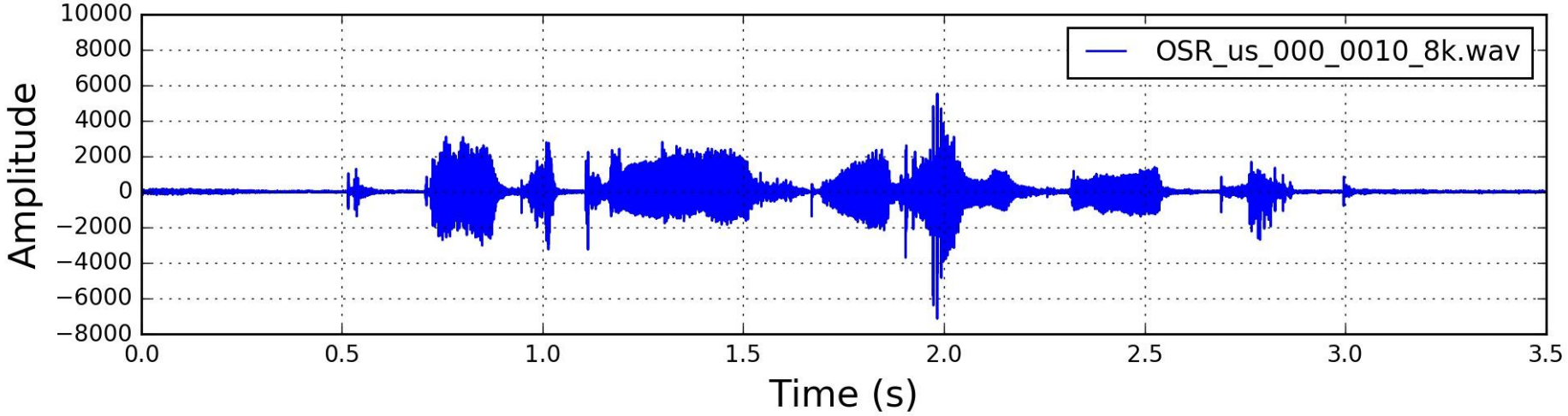
The most popular featural representation of speech are *mel-frequency cepstral coefficients* (MFCCs), representing a *mel-frequency cepstrum*.

1. *Pre-emphasize* the speech signal.
2. *Window* the signal to focus in a single reasonably-narrow time-slice (e.g., 50ms).
3. Compute the *Fourier transform* of each windowed slice, decomposing it into a set of powers at particular frequency bins.
4. Map these powers onto the mel scale (a log-scale measurement).
5. Compute the *cepstrum*, the discrete cosine transform of the mel log frequencies, to detect periodic structure, returning the vector of amplitudes.

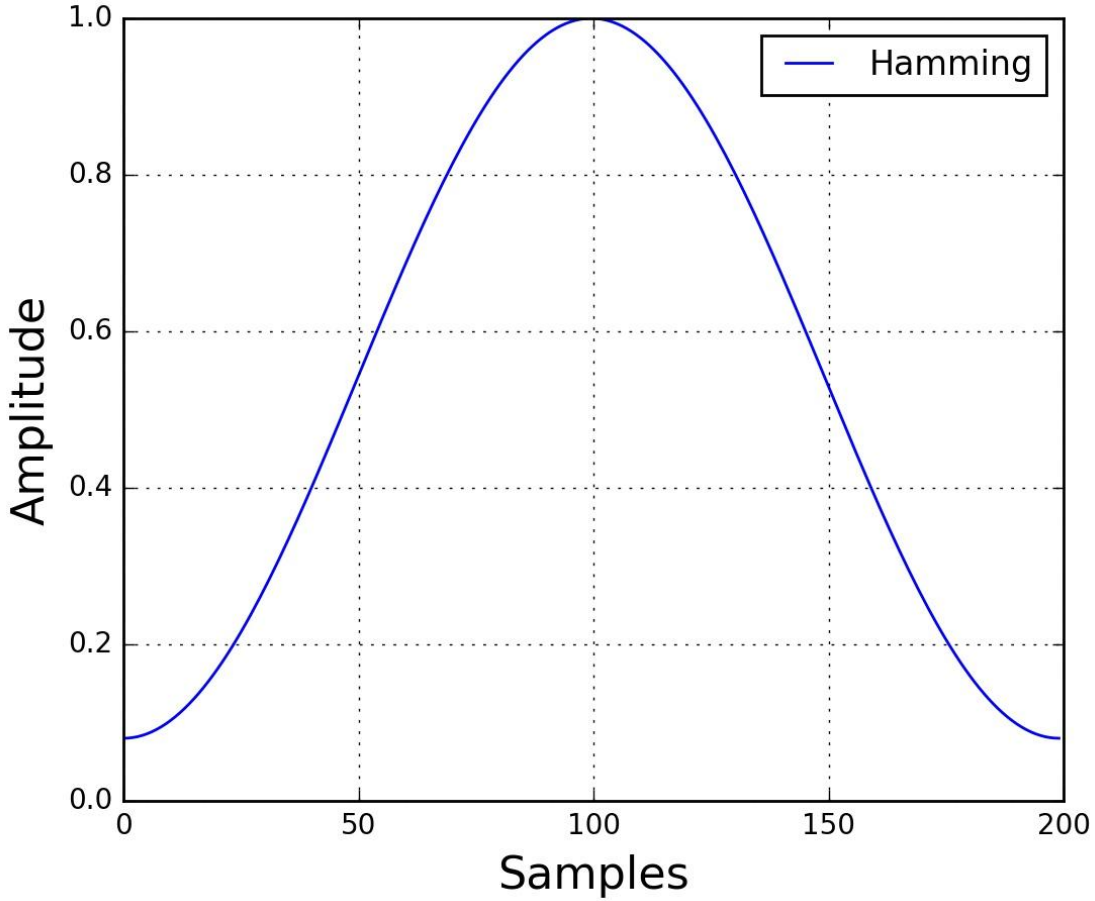
One may also add "delta" or "delta-delta" features to capture changes across slices.



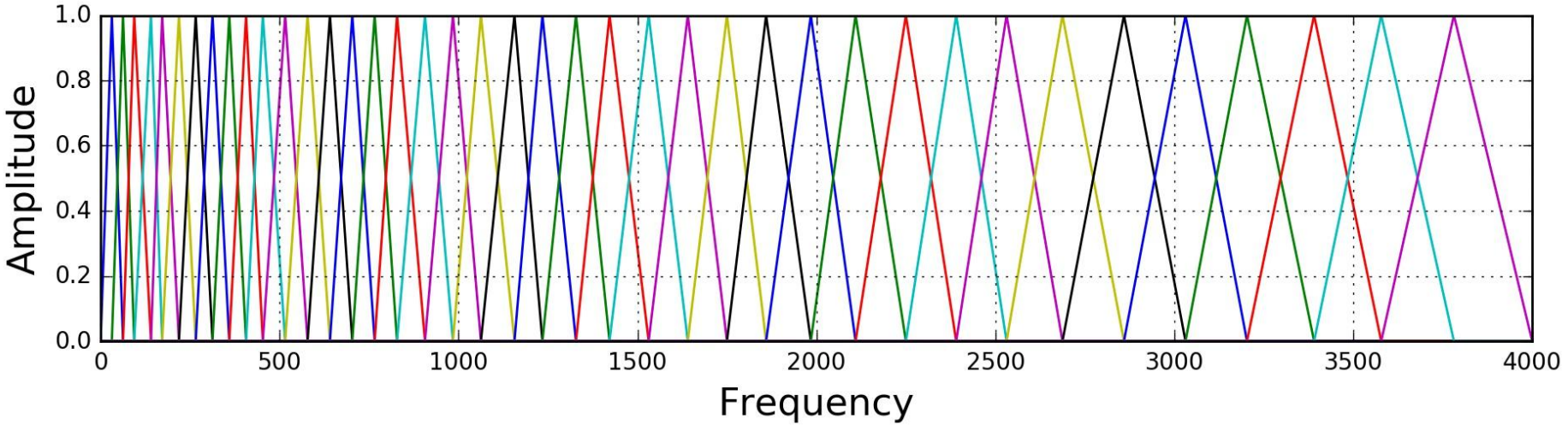
(Image credit: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html)



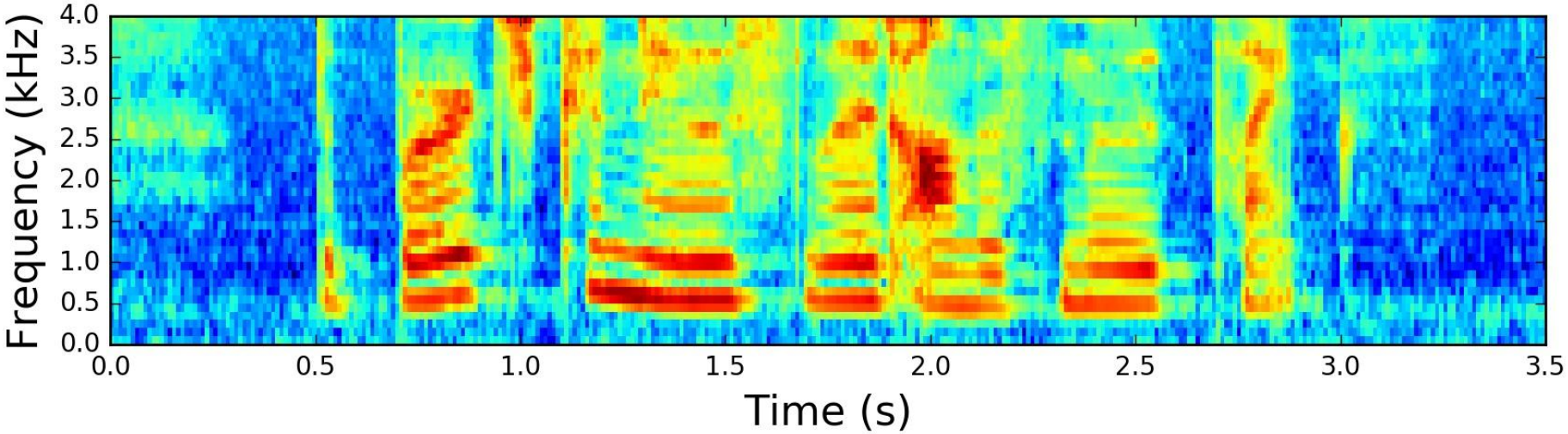
(Image credit: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html)



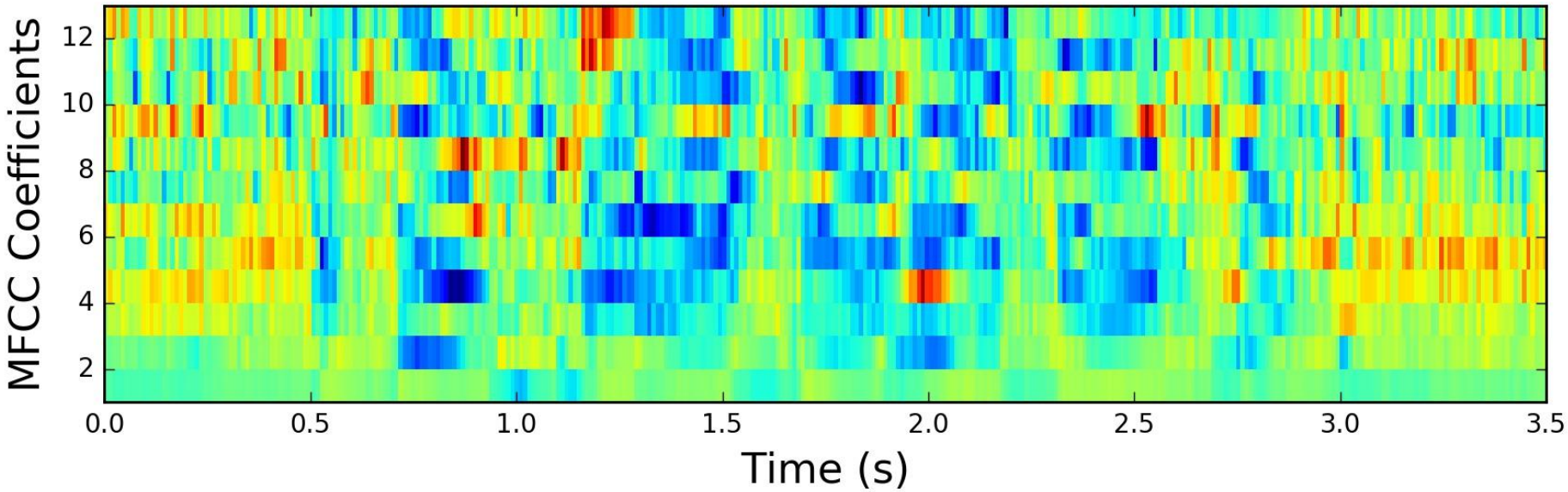
(Image credit: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html)



(Image credit: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html)



(Image credit: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html)



(Image credit: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html)

WFST-based speech recognition

The fundamental theorem of speech recognition

Let $\boldsymbol{A}$ be a sequence of acoustic frames (e.g., a matrix of MFCC coefficients), and $\boldsymbol{W}$ be a sequence of words.

If $P(\boldsymbol{W} \mid \boldsymbol{A})$ is the probability that words $\boldsymbol{W}$ were spoken given acoustic observations $\boldsymbol{A}$, then the recognizer should output a transcription such that

$$\begin{aligned} \hat{\boldsymbol{W}} &= \arg\max_{\boldsymbol{W}} P(\boldsymbol{W} \mid \boldsymbol{A}) \\ &= \arg\max_{\boldsymbol{W}} P(\boldsymbol{W})\, P(\boldsymbol{A} \mid \boldsymbol{W}) \,/\, P(\boldsymbol{A}) \end{aligned}$$

Since $P(\boldsymbol{A})$ is constant, we can simplify this to $\hat{\boldsymbol{W}} = \arg\max_{\boldsymbol{W}} P(\boldsymbol{W})\, P(\boldsymbol{A} \mid \boldsymbol{W})$.

We refer to $P(\boldsymbol{W})$ as the *language model*, and $P(\boldsymbol{A} \mid \boldsymbol{W})$ as the *acoustic model*.

Factoring recognizers as WFST cascades

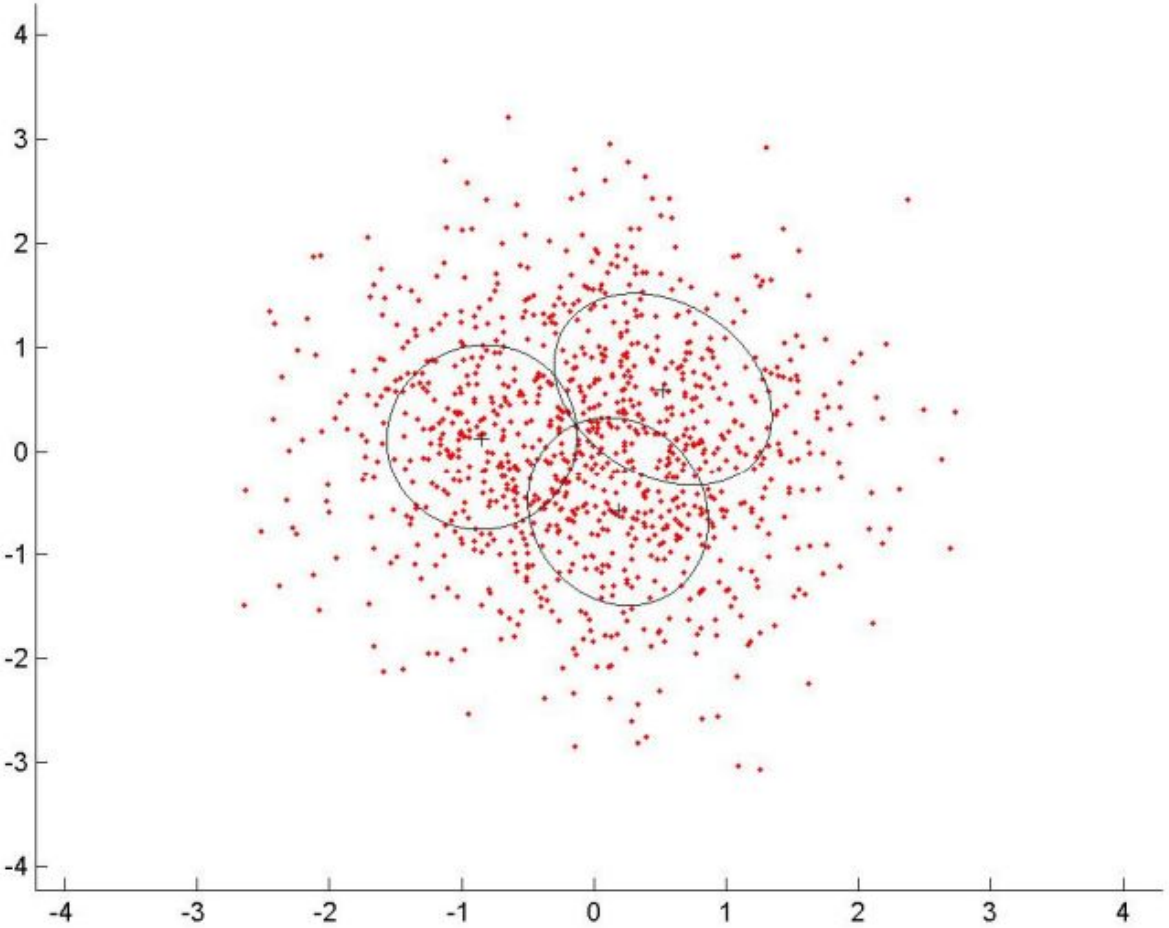
This approach recognizes that both the acoustic model and language model can be expressed as (the composition of) WFSTs, fused together via WFST composition, and exhaustively and efficiently searched using WFST decoding algorithms.

Multivariate Gaussian mixture model (GMMs)

Let $\boldsymbol{X}$ be a single MFCC vector. Then the simplest model is a single multivariate Gaussian distribution over the MFCC vector $N(\boldsymbol{X}; \mu, \sigma)$ with a simple covariance matrix σ (e.g., diagonal).

For more complex models, we use a weighted mixture of Gaussians.

The parameters of these models are estimated using the expectation maximization algorithm, essentially the same problem as forced alignment.



(Image credit: https://cs.nyu.edu/~mohri/asr12/lecture_9.pdf)

Working backwards...

1. A language model favors likely transcriptions over unlikely ones:

   P(dog bites man) > *P*(man bites dog)

2. Word tokens are mapped onto their pronunciations (*monophone* sequences):

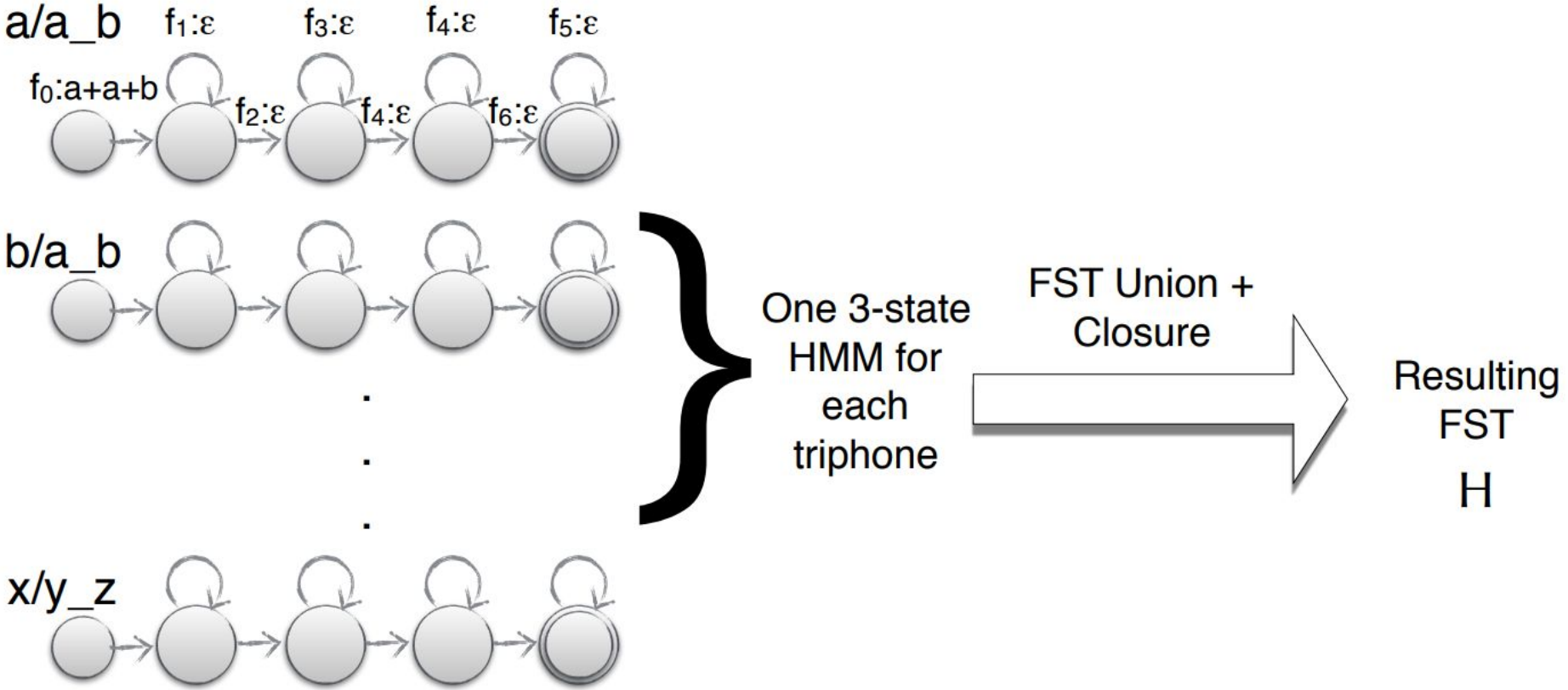
   In practice we store frequent words in a dictionary and generate the rest via G2P.

3. To model local coarticulation, we convert monophones to triphones, phones conditioned on the previous and following segment:

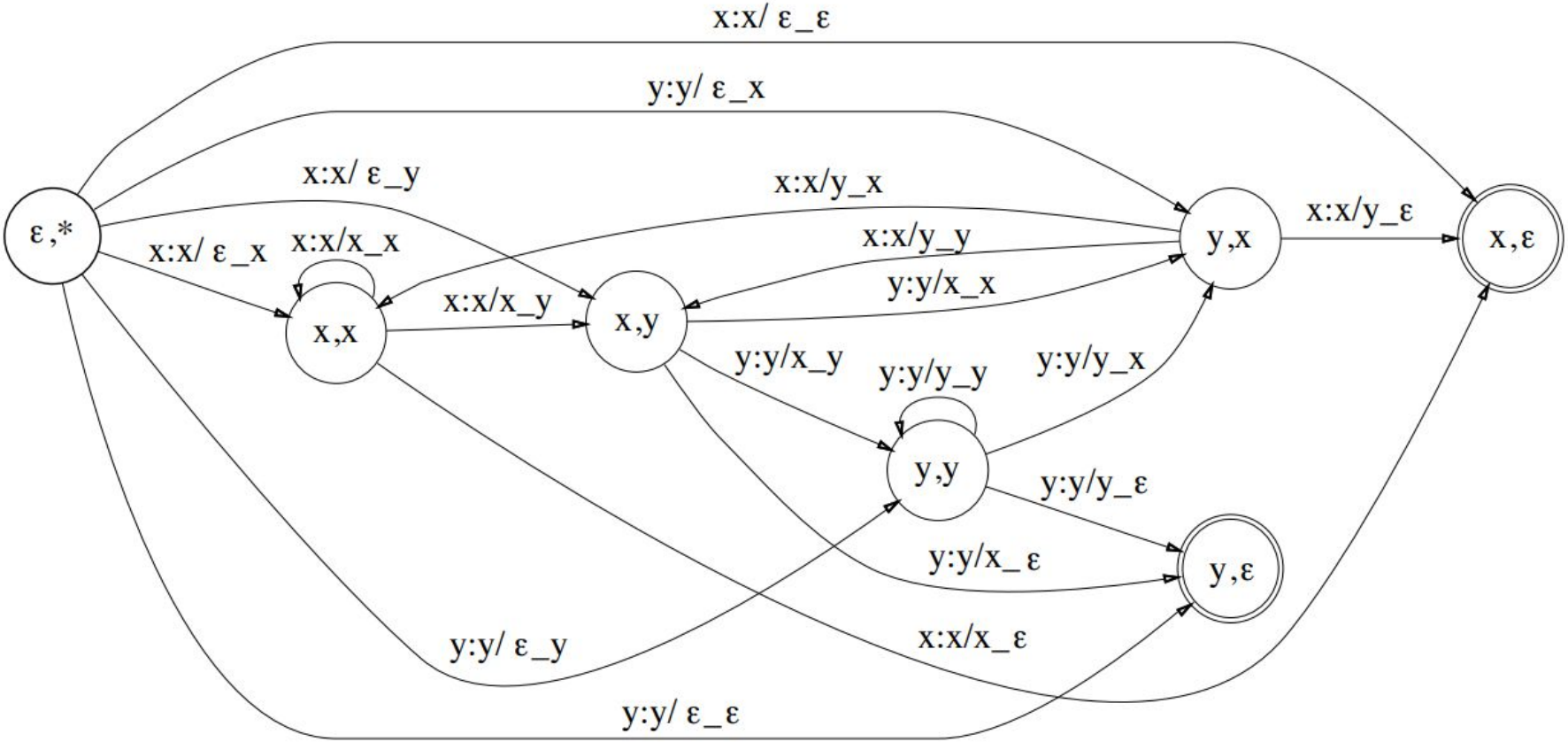
   For example, the vowel in *cat* might be represented as `ae/k_t`.

WFST construction (after Mohri et al. 1997, 2002)

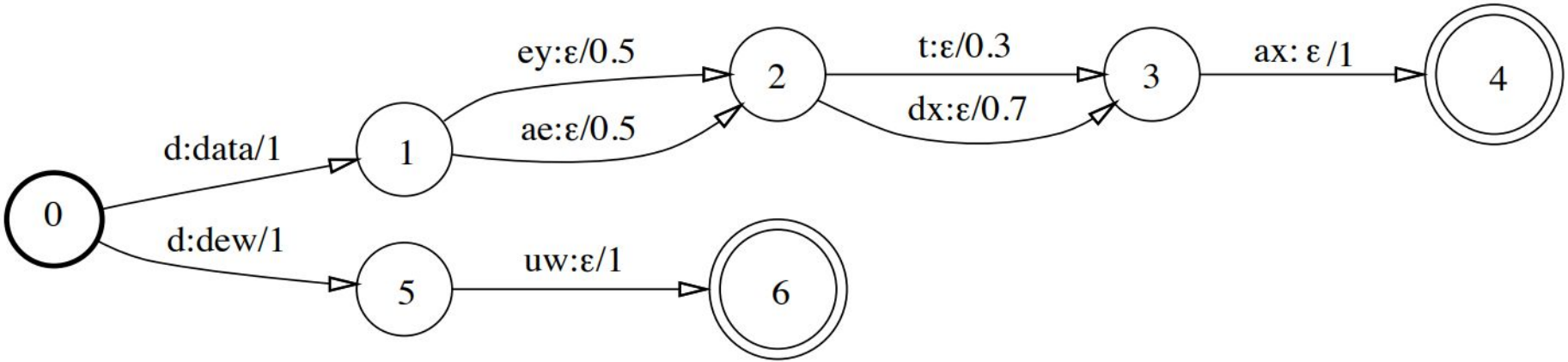
1. The acoustic model constructs a weighted lattice (a WFSA) of possible triphones for each frame, *H*; each state corresponds to a frame, and the arcs leaving that state define a probability distribution over triphones for that frame.
2. The context transducer, a unweighted FST, *C* maps from triphones to monophones.
3. The pronunciation model *L*, a FST (with or without weights), maps from monophones to words.
4. The language model *G*, a WFSA, favors likely transcriptions.



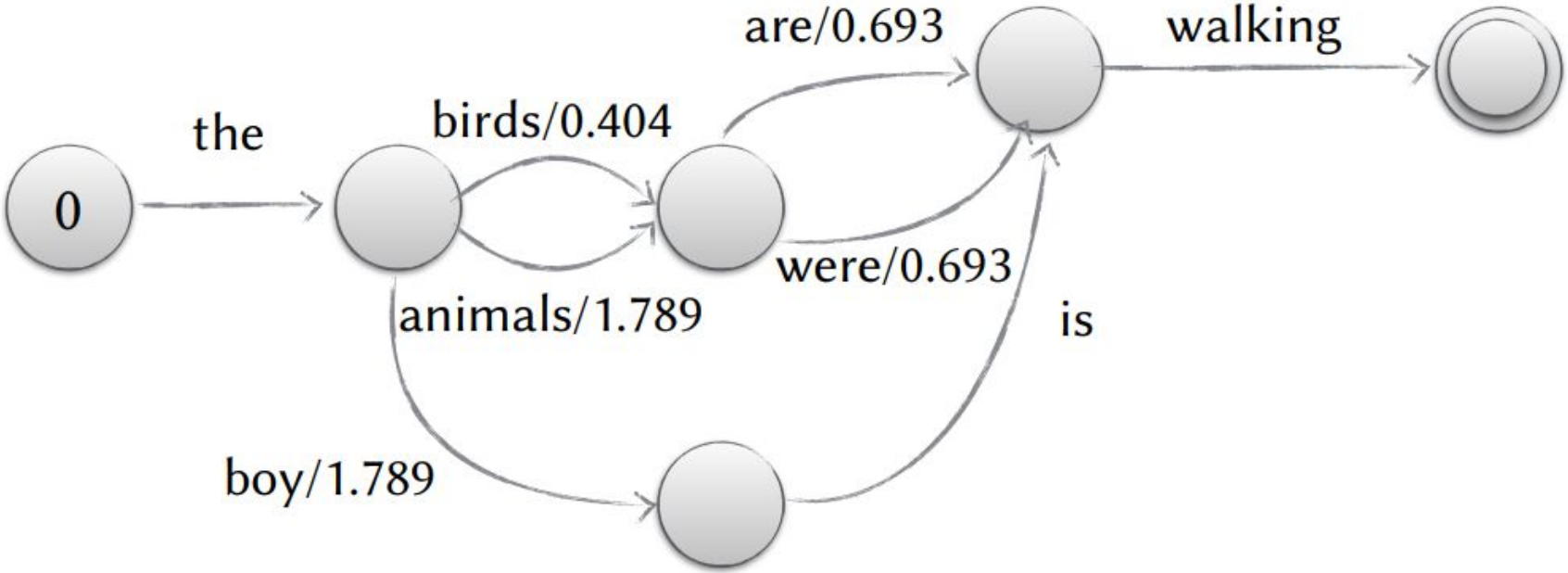
(Image credit: https://www.cse.iitb.ac.in/~pjyothi/cs753_spr16/slides/lecture4.pdf)



(Image credit: https://www.cse.iitb.ac.in/~pjyothi/cs753_spr16/slides/lecture4.pdf)



(Image credit: https://www.cse.iitb.ac.in/~pjyothi/cs753_spr16/slides/lecture4.pdf)



(Image credit: https://www.cse.iitb.ac.in/~pjyothi/cs753_spr16/slides/lecture4.pdf)

Optimization

In place of $H \circ C \circ L \circ G$, we may use the equivalent, but much smaller

$$\texttt{push}(\texttt{min}(\texttt{det}(H \circ \texttt{det}(C \circ \texttt{det}(L \circ G)))))$$

where

- `push` refers to *weight pushed* (an equivalent FST where weights are moved as early as possible along paths),
- `min` refers to the *minimization* (an equivalent deterministic FST with the minimal number of states), and
- `det` refers to *determinization* (an equivalent automaton where no two transitions from any state share the same input labels).

Decoding

- *Viterbi algorithm* (just like HMM tagging)
- *Time-synchronous beam search*: at each time *t* keep only *n* states, or states within a certain threshold of the best state.
- *On-the-fly composition*: build HCLG states only as needed by the decoder.
- *Second-pass reranking*: extract the *k*-best paths using the Viterbi algorithm, and rescore them according to some global, task-specific discriminative model.

NN-based speech recognition

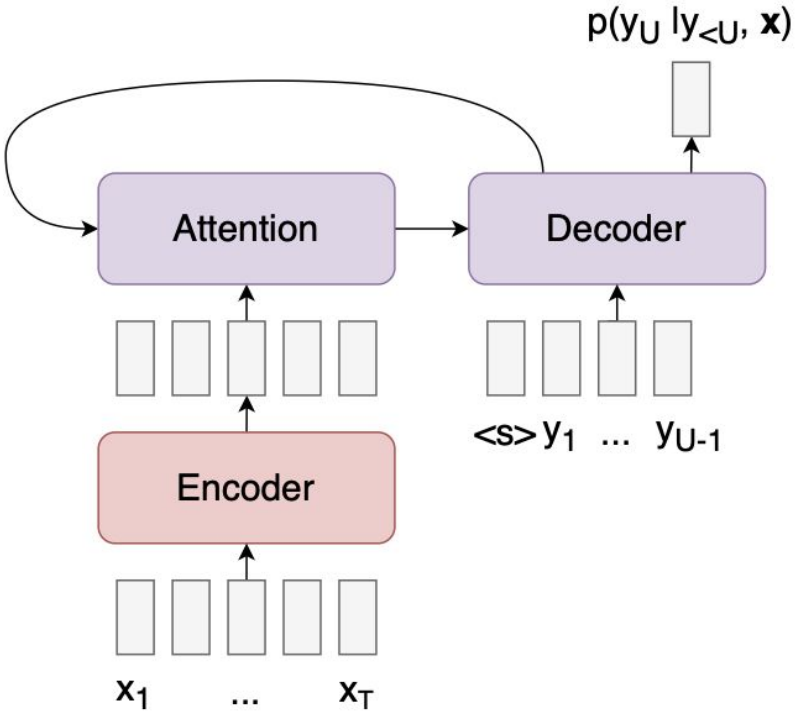
Neural speech recognition

...is still in its infancy. We can achieve very good results on standard "research-sized" data sets, but low-latency, online ("streaming"), or embedded recognition is still dominated by WFST approaches. Some common approaches include:

- Connectionist temporal classification (CTC; Graves et al. 2006)
- Sequence-to-sequence models with attention (e.g., "Listen, Attend, and Spell"; Chan et al. 2015)

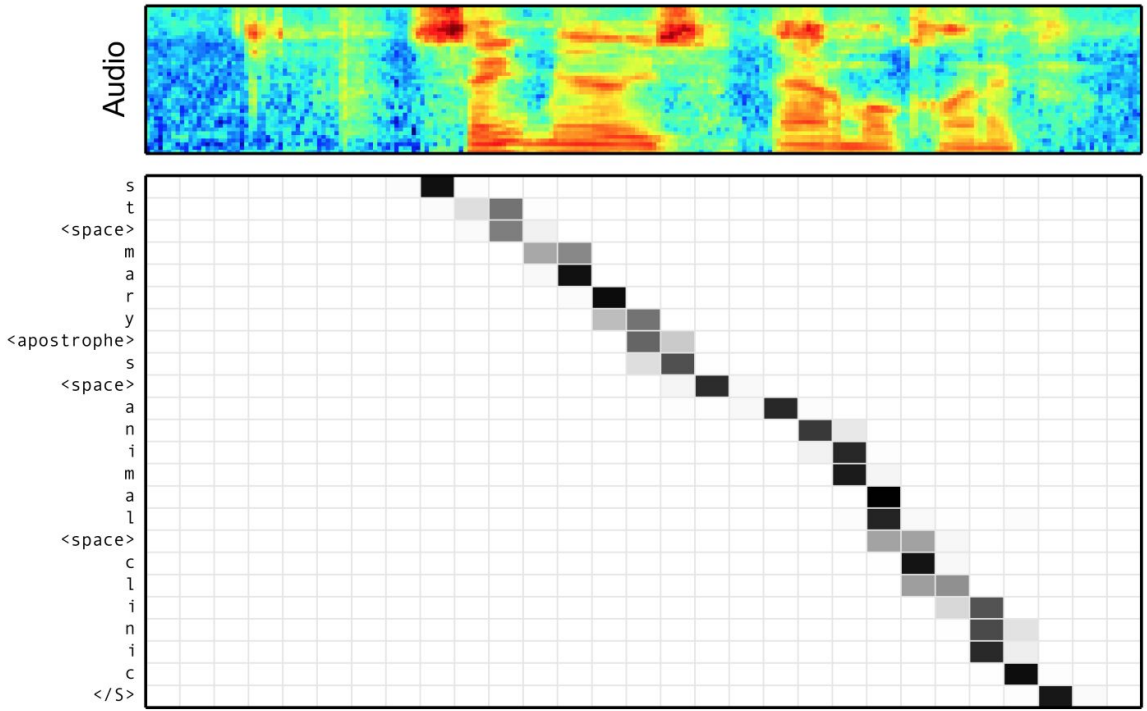
Though the former is older, I think it is superior, for reasons explained below...

Sequence-to-sequence attention-based ASR



(Image credit: https://lorenlugosch.github.io/posts/2020/11/transducer/)

Sequence-to-sequence attention-based ASR



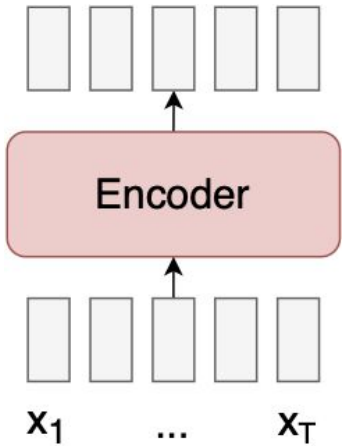
(Image credit: https://lorenlugosch.github.io/posts/2020/11/transducer/)

Deficiencies of the the LAS model

- Attention computations can become expensive with long sequences.
- Attention models cannot be directly run online because they cannot attend to elements of the sequence not yet presented (though one could imagine using Reinforcement Learning here to learn to wait just long enough…)
- Attention models don't take into account the fact that the alignment is largely monotonic, i.e., if frame t comes before $t + 1$ it cannot be the case that t aligns to a later word than $t + 1$.

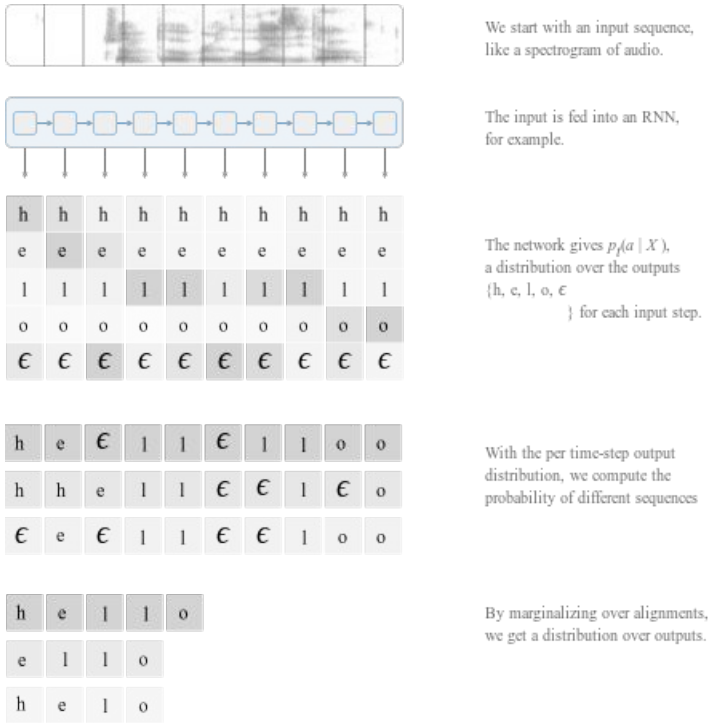
Connectionist temporal classification

CTC models assume that there is a monotonic alignment, eliminating the need for a decoder. They do this by treating transcription as a tagging problem and marginalizing over all monotonic alignments.



(Image credit: https://lorenlugosch.github.io/posts/2020/11/transducer/)

CTC alignment



(Image credit: https://distill.pub/2017/ctc/)

Complexities of CTC models

- We need to make a weighted combination of the various sequences of tags that produce equivalent transcriptions; in WFST-land this corresponds roughly to epsilon-removal, determinization, and weight pushing.
- We need a loss function sensitive to the final transcriptions, not the tags.
- We still need to fuse this model with a language model (perhaps a neural language model) to obtain good results.
- Beam search is likely needed for decoding.

Comparison to WFST GMM-HMM models

- Unlike HMMs, introduces a "blank" (which might be a good idea for HMMs too!).
- Unlike HMMs, the model is discriminative; it models $P(\boldsymbol{W} \mid \boldsymbol{A})$ directly.

Software & Data

Free (*gratis*, at least) software

- Kaldi is a free recognition toolkit for WFST-based models. It is primarily interacted with via the command-line. While it has some support for NN-based systems, it is being rewritten to support NN approaches.
- Mozilla Voice STT is an open-source NN-based ASR engine implemented in TensorFlow, based on Baidu's DeepSpeech, based on the CTC model.
- Google Cloud Speech-to-Text provides a free tier with high-quality recognizers for over 100 languages: one simply crafts a JSON query with the audio and it replies with the transcription.

Data

Unlike natural language processing, most speech data is proprietary and released by the Linguistic Data Consortium. Two popular LDC collections are CALLHOME (Egyptian Arabic, American English, German, Japanese, Mandarin, Spanish) and CALLFRIEND (Egyptian Arabic, American English, Farsi, Canadian French, German, Hindi, Japanese, Korean, Mandarin Chinese, Spanish, Tamil, Vietnamese).

The free LibriSpeech corpus contains 1,000 hours of English amateur audiobooks; the OpenSLR collection to which it belongs has many more nice resources.

The Mozilla Common Voice project is trying to crowdsource free multilingual data.

Project ideas

- Train your own recognizer (using an existing toolkit like Kaldi or Mozilla Voice STT), documenting your process so that it can be easily repeated with new data, and evaluate performance on held-out data.
- Look into ASR performance across a variety of accents (e.g., using TIMIT).