CMPE 297, INSTRUCTOR: JORJETA JETCHEVA

AUTOMATIC SPEECH RECOGNITION

OVERVIEW

- Transcribing spoken language pre-dates the computer
- Radio Rex (1920) included a celluloid dog on a spring inside a little house, where the spring would get released at 500Hz acoustic energy (emitted when someone said the word Rex)
- The task of automatic speech recognition (ASR) is to covert a waveform to a string of words

"It's time for lunch!"

- Speech synthesis or text-to-speech is the reverse process of ASR (we will not discuss it in class)

APPLICATIONS OF ASR

- Communication in the context of devices, e.g., smart home appliances
- Interactions with digital devices when we can’t conveniently use our hands (e.g., car, cell phone)
- Call routing applications when calling customer service
- Captioning (e.g., for talks, meetings, court proceedings)
- Augmentative applications which enable people with disabilities to use technology (e.g., vision problems, hearing loss, motor control challenges, etc.)

ARS TASK OVERVIEW

Vocabulary dimension

- Tasks with limited vocabulary size can be solved with very high accuracy, e.g., vocabulary of just “yes” and “no”, digit recognition (recognizing sequences of digits)
- Open-ended tasks where videos or human conversations are transcribed are very hard (vocabulary sizes are in the range of thousands of words)

Speech participant dimension

- Human speaking to humans (conversational speech), e.g., transcribing a discussion or meeting, is very hard
- Human speaking to machine or reading out loud (e.g., audio books) are relatively easy to recognize as humans seem to speak more slowly and clearly, and also use simpler words/constructs

Channel & noise dimension

- Quiet space, nearby microphone vs. loud environment, far away microphone

Speaker characteristics

- Accent, regional or ethnic dialects, child vs. adult
- ASR not trained on the types of speakers that use the system will not perform well

EXAMPLE SPEECH-TO-TEXT DATASETS

- LibriSpeech - open source read speech from the LibriVox project, with over 1000 hours of audio books with transcripts aligned at the sentence level
  - Easier to understand speech was rates as “clean”, while the rest was annoyed as “other” (based on experiments with ASR)
  - LibriVox is a group of worldwide volunteers who read and record public domain texts, creating free public domain audiobooks
- Switchboard corpus - 2430 prompted conversations between strangers averaging 6 minutes each (total of 240 hours and 3 million words)
- CALLHOME corpus - 120 unscripted 30-minute telephone conversations between native speakers of English (usually friends or family)
- CORAAL - 150 sociolinguistic interviews with African American speakers
- CHiME Challenge — challenging tasks to evaluate ASR robustness
  - E.g., CHiME 5 includes conversational speech from 20 dinner parties in real homes, each with 4 participants, in 3 locations (kitchen, dining area, living room) recorded with both room microphones and body-worn microphones

FEATURE EXTRACTION FOR ASR

FEATURE EXTRACTION OVERVIEW

- We need to transform the input waveform into a sequence of acoustic feature vectors (it's like tokenization but of the waveform)
- We will cover
  - Sampling and Quantization
  - Windowing
  - Discrete Fourrier Transform
  - Mel filter bank and log

REVIEW OF WAVES

- Waves are approximated by sine functions $y = A \times sin(2\pi ft)$
- Amplitude (A) is the max values on the Y axis ($A = 1$ in the figure)
- Frequency (f): # of times/sec a wave repeats itself, i.e., number of cycles
  - Frequency is typically measured in cycles/sec
  - Cycles per second are usually called hertz (Hz)
- Period (T): the time it takes for one cycle to complete

  $T = \frac{1}{f}$ ($T = 0.1s$ in the figure)

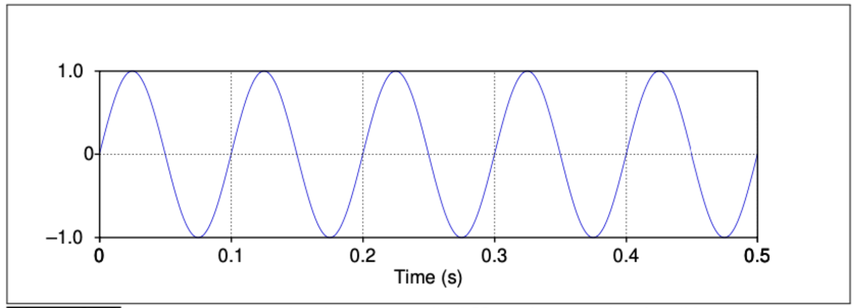

Figure 25.8 A sine wave with a frequency of 10 Hz and an amplitude of 1.

SAMPLING AND QUANTIZATION: ANALOG TO DIGITAL CONVERSION

Analog to Digital Conversion, **Step 1: Sampling**

- We measure the signal amplitude at different times based on the sampling rate we've chosen
  - Sampling rate is number of samples taken per second
- At a minimum, we need 2 samples per cycle: one to measure the positive part of the wave, and one to capture the negative part of the wave
- Conversely, the max frequency wave that be measured is one whose frequency is half the sampling rate (aka Nyquist frequency)
  - Most information in human speech is in frequencies below 10,000Hz and thus requires a 20,000Hz sampling rate (1 Hz is 1 cycle per) second
  - Telephone speech is filtered by the switching network, so frequencies are below 4,000Hz, so an 8,000Hz sampling rate is sufficient
  - Microphone speech requires 16,000Hz sampling rate
  - Note that we can't have different sampling rates for training and test data, so all data has to be downsampled to the lowest sampling rate across training and test

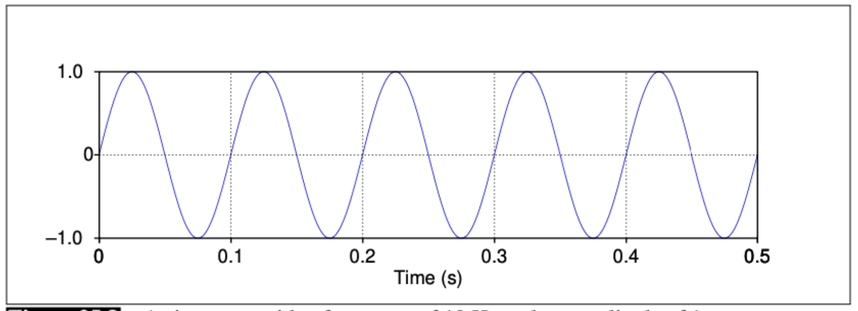

Figure 25.8 A sine wave with a frequency of 10 Hz and an amplitude of 1.

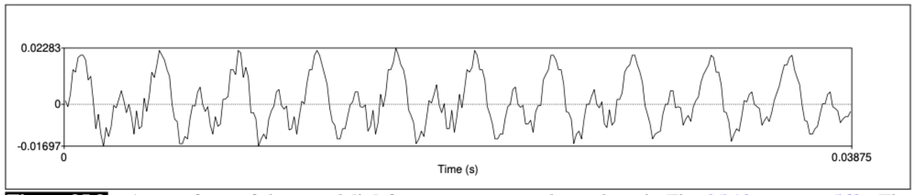

Figure 25.9 A waveform of the vowel [iy] from an utterance shown later in Fig. 25.13 on page 568. The *y*-axis shows the level of air pressure above and below normal atmospheric pressure. The *x*-axis shows time. Notice that the wave repeats regularly.

Analog to Digital Conversion, **Step 2: Quantization**

- We represent the real-valued sample measurements of the wave as 8 or 16 bit integers
- We will use $x[n]$ to refer to the quantized sample at time n

WINDOWING

- Phonemes are units of sound, e.g., corresponding to pronunciation of letters
- In order to try to extract phonemes or parts of phonemes from the digitized waveform, we use a sliding window approach to generate frames (each frame is a portion of the speech), parameterized as follows:
  - Window size (or frame size) - width in ms
  - Frame stride (or shift, or offset) between successive windows
  - Window shape, e.g., rectangular, Hamming

  To extract the signal at time n, we multiply the value of the signal at time n, by the value of the window at time n as follows: $y[n] = w[n]s[n]$

  Window shapes:

$$\textit{rectangular} \quad w[n] = \begin{cases} 1 & 0 \le n \le L-1 \\ 0 & \text{otherwise} \end{cases}$$

$$\textit{Hamming} \quad w[n] = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi n}{L}\right) & 0 \le n \le L-1 \\ 0 & \text{otherwise} \end{cases}$$

  The Hamming window shrinks the values of the signal toward 0 at the window boundaries to create a more continuous signal across frames, which is better for further analysis using Fourier Transforms, and is thus more commonly used

- Then we compute a Discrete Fourier transform (DFT) over the input frames, which tells us the energy at each frequency band
  - Outputs the magnitude and phase of the frequency components in the signal across all N discrete frequency bands

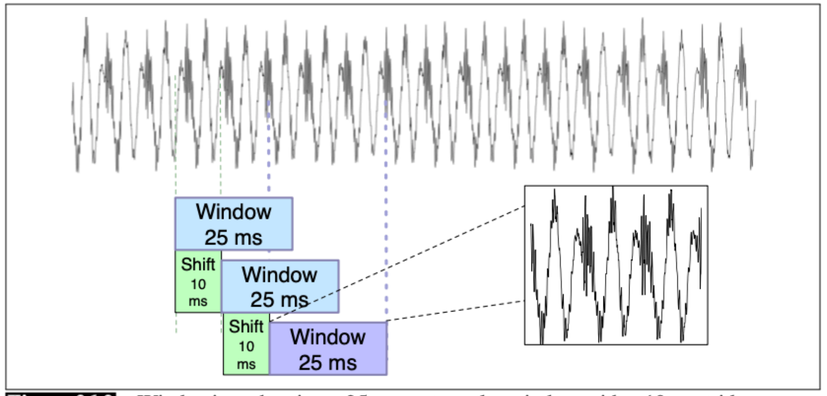

Figure 26.2 Windowing, showing a 25 ms rectangular window with a 10ms stride.

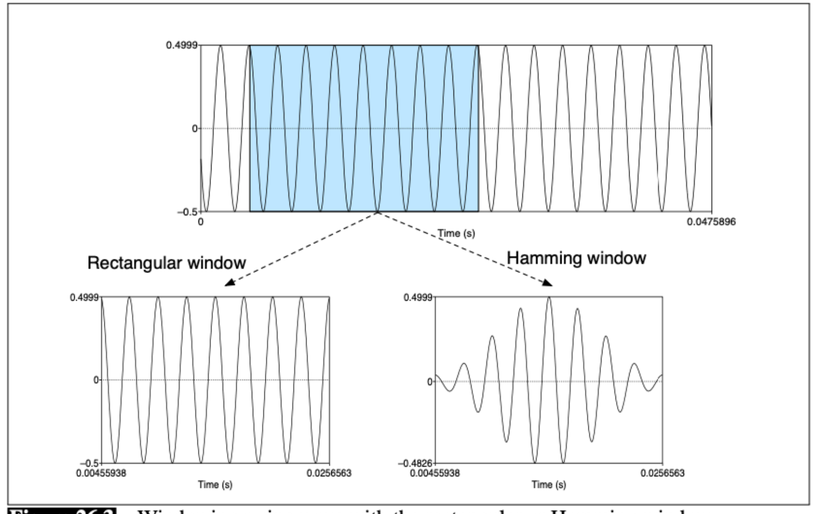

Figure 26.3 Windowing a sine wave with the rectangular or Hamming windows.

MEL FILTER BANK AND LOG

- Even though we can calculate the energy at each frequency band, human hearing is actually more sensitive to lower frequencies and ASR systems work better when they take advantage of this fact
- We create a bank of filters that collect energy from each frequency range
  - The filter sizes are logarithmic since we want to have finer resolution at lower frequencies (e.g., the triangular filters below get larger at higher frequencies)
- The filters are multiplied by the spectrum to get a mel spectrum (mel is a unit of pitch, and the mel scale is an auditory frequency scale)
- Then we take a log of the mel spectrum values
  - Human response to signal level is logarithmic (like human response to frequency) – humans are less sensitive at signal differences at high amplitude than they are at low amplitudes

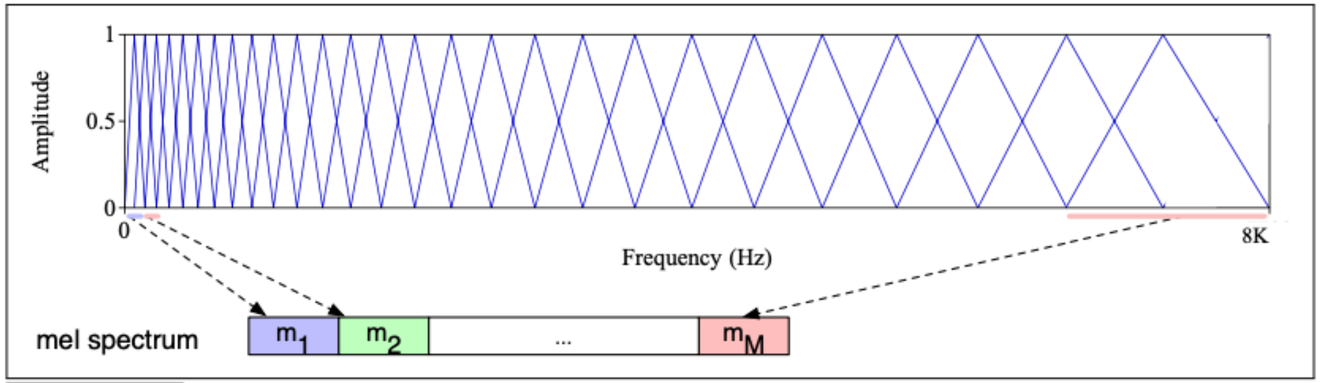

Figure 26.5 The mel filter bank (Davis and Mermelstein, 1980). Each triangular filter, spaced logarithmically along the mel scale, collects energy from a given frequency range.

SPEECH RECOGNITION ARCHITECTURE

ATTENTION-BASED ENCODER DECODER

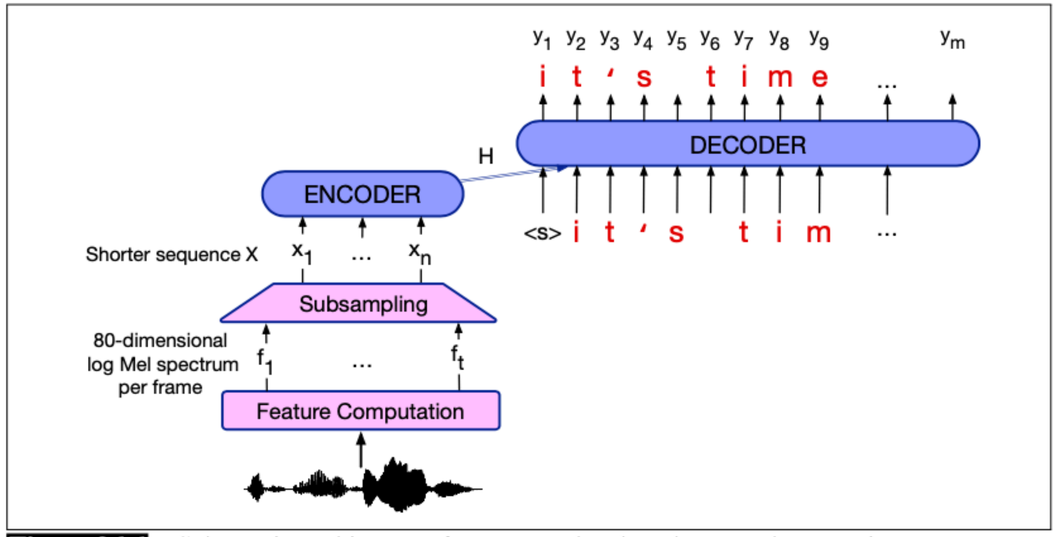

Figure 26.6 Schematic architecture for an encoder-decoder speech recognizer.

- Typically use encoder decoder architecture (similar to MT)
  - Input: the log mel spectral feature vectors:

    $F = f_1, f_2, \ldots, f_t$ (one vector per 10ms frame)
  - Output: letters (though word pieces also used in some approaches)

    $Y = (\langle SOS \rangle, y_1, \ldots, y_m \ \langle EOS \rangle)$, where $SOS =$ start of sequence, and $EOS =$ end of sequence

    and $y_i \in \{a, b, c \ldots, z, 0, \ldots, 9, \langle space \rangle, \langle comma \rangle, \langle period \rangle, \langle apostrophe \rangle, \langle unk \rangle\}$
- Differences in length between input and output are a great fit for encoder-decoder architectures
  - However, compression/subsampling of the input is needed for ASR since the difference between input and output are too extreme
    - A single word is short relative to the number of frames in the acoustic feature sequence (e.g., 5-letter word can last 2s and take up 200 acoustic frames of 10ms each)
    - A possible compression approach known as **low frame rate** is to concatenate each 3 vectors together thus reducing the input length by 3 (though each input feature is 3 times longer)

ENCODER-DECODER OPTIMIZATIONS

- A language model (trained separately on written text) can be used to rate multiple possible outputs of the encoder decoder (e.g., as output by a beam search)
  - This enables us to incorporate language information learned over a large text corpus for which we don't have a corresponding speech corpus
- We use teacher forcing when training the model (as we did in MT as well)

CONNECTIONIST TEMPORAL CLASSIFICATION (CTC)

CONNECTIONIST TEMPORAL CLASSIFICATION (CTC) OVERVIEW

- CTC is an alternative to the encoder-decoder model and works as follows:
  - Step 1: Produce an output letter for each frame in the input (this sequence of output letters is called an alignment since it is aligned to the acoustic signal)
    - Use blank symbol in slots where a letter cannot be detected (denoted with ϵ)
  - Step 2: Collapse sequences of identical letters
    - Collapse letters that are duplicate and next to each other (not separated by a blank symbol)
    - Then remove al blanks

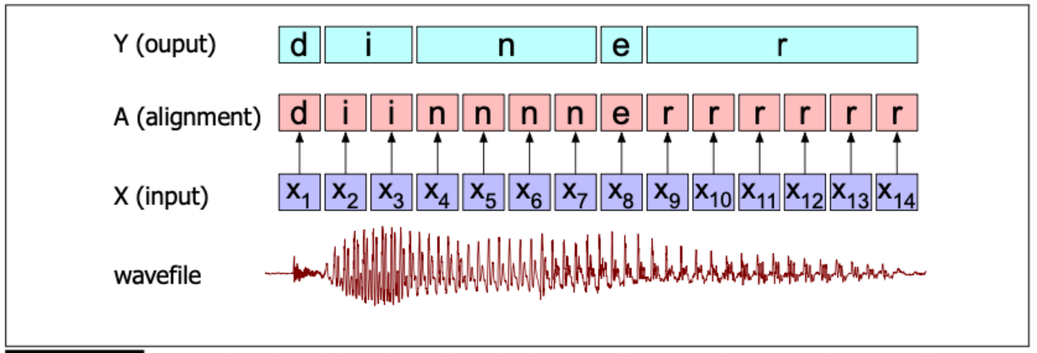

Figure 26.7 A naive algorithm for collapsinging an alignment between input and letters.

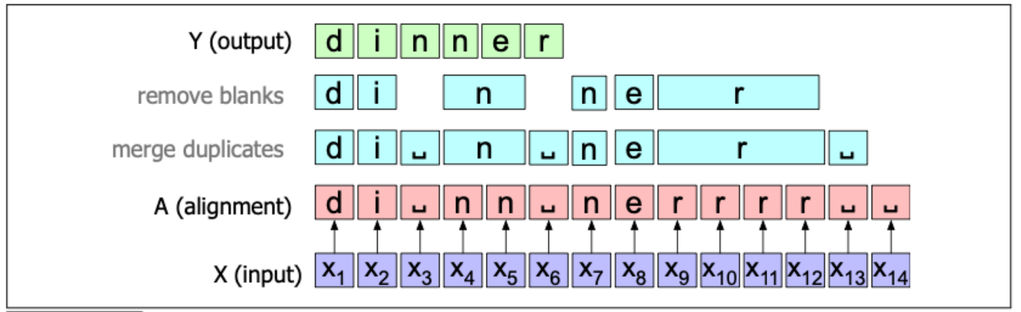

Figure 26.8 The CTC collapsing function B, showing the space blank character ␣; repeated (consecutive) characters in an alignment A are removed to form the output Y.

INFERENCE IN CTC

- CTC assumes conditional independence where given the input X, the computed output α_t at time t is independent of the outputs at any other time α_i:

$$P_{CTC}(A|X) = \prod_{t=1}^{T} p(\alpha_t|X)$$

- So to find the best alignment $\hat{A} = \{\hat{\alpha}_1, \ldots, \hat{\alpha}_T\}$, we greedily choose the character with the max probability at each step t: $\hat{\alpha}_t = arg\max_{c \in C} p_t(c|X)$

  This amounts to using a sequence-modeling task, for which we only need an encoder.

- However, the most likely alignment may not correspond to the most likely final collapsed output (many alignments can map to one final string), e.g.,

  $X = [x_1 x_2 x_3]$ has the most probable alignment $[a\ b\ \epsilon]$, followed by $[b\ \epsilon\ b]$, and $[\epsilon\ b\ b]$

  The output $Y = [b\ b]$ may be more probable than $Y = [a\ b]$ (e.g., if we consider the top 3 alignment choices as opposed to just the top choice)

  That's why CTC chooses the output sequence with the highest sum over the probability of all possible alignments

- CTC recognizers don't perform as well as encoder-decoders but can be used for streaming: recognizing words on-line rather than having to first process the entire sentence

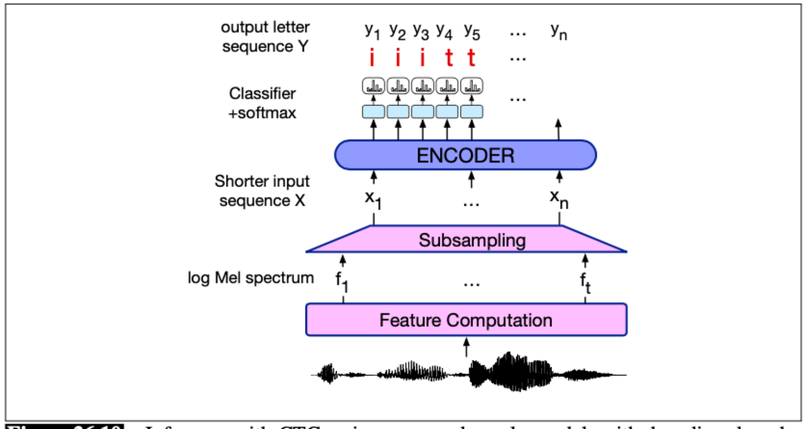

Figure 26.10 Inference with CTC: using an encoder-only model, with decoding done by simple softmaxes over the hidden state h_t at each output step.

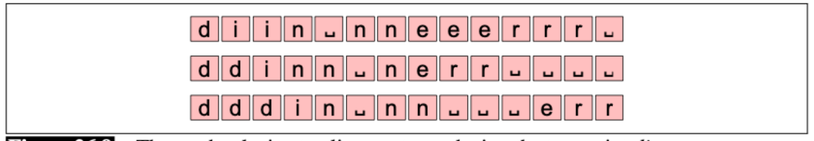

Figure 26.9 Three other legimate alignments producing the transcript *dinner*.

$$P_{CTC}(Y|X) = \sum_{A \in B^{-1}(Y)} P(A|X)$$

$$= \sum_{A \in B^{-1}(Y)} \prod_{t=1}^{T} p(a_t|h_t)$$

$$\hat{Y} = \operatorname*{argmax}_{Y} P_{CTC}(Y|X)$$

ASR EVALUATION

WORD ERROR RATE

- Word error rate measures how different the hypothesized word string (returned by the ASR system) is from the reference transcription
- We compute the minimum edit distance and then the word error rate based on the number of insertions, substitutions and deletions

$$\text{Word Error Rate} = 100 \times \frac{\text{Insertions} + \text{Substitutions} + \text{Deletions}}{\text{Total Words in Correct Transcript}}$$

- Example:

REF:	i	***	**	UM	the	PHONE	IS		i	LEFT	THE	portable	****	PHONE	UPSTAIRS	last night
HYP:	i	GOT	IT	TO	the	*****	FULLEST		i	LOVE	TO	portable	FORM	OF	STORES	last night
Eval:		I	I	S		D	S			S	S		I	S	S	

$$\text{Word Error Rate} = 100\frac{6+3+1}{13} = 76.9\%$$

- Statistical significance tests are used to establish whether an improvement in word error rate is statistically significant

English Tasks	**WER%**
LibriSpeech audiobooks 960hour clean	1.4
LibriSpeech audiobooks 960hour other	2.6
Switchboard telephone conversations between strangers	5.8
CALLHOME telephone conversations between family	11.0
Sociolinguistic interviews, CORAAL (AAL)	27.0
CHiMe5 dinner parties with body-worn microphones	47.9
CHiMe5 dinner parties with distant microphones	81.3
Chinese (Mandarin) Tasks	**CER%**
AISHELL-1 Mandarin read speech corpus	6.7
HKUST Mandarin Chinese telephone conversations	23.5

Figure 26.1 Rough Word Error Rates (WER = % of words misrecognized) reported around 2020 for ASR on various American English recognition tasks, and character error rates (CER) for two Chinese recognition tasks.