Automatic Speech Recognition: A Whirlwind Tour

Brian Kingsbury
IBM Research
Outline

- A brief history of speech recognition
- The probabilistic approach
- Feature extraction
- Acoustic modeling
- Language modeling
- Search
- Where do we stand?
A brief history of speech recognition
In the beginning there was Radio Rex

A speaker-independent, one-word recognizer in the 1920’s! If it detected sufficient energy at 500 Hz (from “e” in “Rex”), the dog popped out of the house.
The early years: 1920–1960’s

Ad hoc methods, limited by available processing power.

- Small vocabulary (digits, yes/no, vowels)
- Small number of speakers (1–10).
- Simple signal processing approaches:
  - Detect energy in certain frequency bands.
  - Find dominant frequencies.
- But we see many ideas central to modern ASR:
  - Statistical training
  - Language modeling
The turning point: “Whither Speech Recognition?”


- “Speech recognition has glamour. Funds have been available. Results have been less glamorous.”
- “[Speech recognition has a] scarcity in the field of people who behave like scientists and of results that look like science.”
- “It would seem appropriate for people to ask themselves why they are working in the field and what they can expect to accomplish.”

Killed speech recognition research at Bell Labs for many years.
“Whither Speech Recognition?” also provided partial impetus for the first (D)ARPA program (1971–1976) on speech recognition.

- Program goal: integrate speech knowledge, linguistics, and AI to make a breakthrough in ASR.
- Introduced the use of *competitive evaluation* on a common task:
  - Connected speech
  - Many (cooperative) speakers
  - 1000-word vocabulary
  - Artificial syntax
  - < 10% “semantic error”
  - A few times real time on a 100 MIPS computer.
The demise of rule-based systems

- DARPA evaluation tested four systems.
- 3 used hand-derived rules with scoring based on knowledge of speech and language.
- CMU’s HARPY system integrated all knowledge sources in a finite-state network and trained parameters in a statistical, data-driven manner.

HARPY won hands down.
The probabilistic approach
The fundamental equation of speech recognition

\[ \hat{W} = \arg \max_W P(W|X; \Theta) \]

\[ = \arg \max_W \frac{P(X|W; \Theta)P(W; \Theta)}{P(X)} \quad \text{Bayes’ Rule} \]

\[ = \arg \max_W P(X|W; \Theta)P(W; \Theta) \quad P(X) \text{ irrelevant.} \]

Where \( W \) is a word sequence,
\( \hat{W} \) is the best word sequence,
\( X \) is a sequence of acoustic features, and
\( \Theta \) denotes the model parameters.
Implementing the fundamental equation

$$\hat{W} = \arg \max_W P(X|W; \Theta)P(W; \Theta)$$
Implementing the fundamental equation

$$\hat{W} = \arg \max_W P(X|W; \Theta) P(W; \Theta)$$
Implementing the fundamental equation

\[ \hat{W} = \arg \max_W P(X|W; \Theta)P(W; \Theta) \]
Implementing the fundamental equation

\[ \hat{W} = \arg \max_{W} P(X|W; \Theta)P(W; \Theta) \]
Implementing the fundamental equation

$$\hat{W} = \arg \max_W P(X|W; \Theta)P(W; \Theta)$$
Feature extraction

\[
\hat{W} = \arg \max_\mathbf{W} \ P(\mathbf{X}|\mathbf{W}; \Theta) P(\mathbf{W}; \Theta)
\]
Feature extraction has four main goals

**Characterize** signal aspects conveying linguistic information.

**Suppress** signal aspects conveying nonlinguistic information.

**Concise** representation.

**Conform** to the constraints of the acoustic model.
Feature extraction uses a handful of methods to meet these goals.

1. Extract a smoothed spectral envelope.
2. Warp the envelope such that it has high resolution at low frequencies and low resolution at high frequencies.
3. Compress the envelope to limit its dynamic range.
4. Decorrelate the spectral representation.
5. Include information about the local temporal evolution of the features.
   - Use first- and (optionally) second-order temporal differences.
   - Linear projection of concatenated features.
Acoustic modeling

\[ \hat{W} = \arg \max_W P(X|W; \Theta) P(W; \Theta) \]
Most large vocabulary ASR systems model words as sequences of phones.

- A *phone* is a basic unit of speech: if you change it, you can change the word, and it cannot be broken down into a sequence of smaller units.

- If you represent words this way, you can recognize words that you never saw during training, provided you saw all of the phones that the word comprises.
Phones can be modeled with hidden Markov models (HMMs)

Model the temporal structure in the phone with separate states: often a begin, middle, and end state.

Note that this is a *generative* model of speech.
In speech recognition, there are 3 key HMM operations

1. The **forward-backward** algorithm, which produces a probability distribution over hidden states, $q$, at each time $t$, given a word sequence, $W$, and an acoustic observation sequence, $X$.

2. The **constrained Viterbi** algorithm (forced alignment), which finds the best state sequence, $\hat{Q} = q_0q_1 \cdots q_n$, given a word sequence, $W$, and an acoustic observation sequence, $X$.

3. The **Viterbi** algorithm, which finds the best state sequence, $\hat{Q} = q_0q_1 \cdots q_n$, given an acoustic observation sequence, $X$. 
Why phones aren’t good enough: An exercise

1. Put your hand in front of your mouth.
2. Say “spin.”
3. Now say “pin.”
Why phones aren’t good enough: An exercise

1. Put your hand in front of your mouth.
2. Say “spin.”
3. Now say “pin.”
4. What is the difference between the p’s?
Why phones aren’t good enough: An exercise

1. Put your hand in front of your mouth.
2. Say “spin.”
3. Now say “pin.”
4. What is the difference between the p’s?
5. The p in “pin” is preceded by a puff of air (it’s *aspirated*).
6. The p in “spin” is not preceded by a puff of air (it’s *unaspirated*).
Why phones aren’t good enough: An exercise

1. Put your hand in front of your mouth.
2. Say “spin.”
3. Now say “pin.”
4. What is the difference between the p’s?
5. The p in “pin” is preceded by a puff of air (it’s *aspirated*).
6. The p in “spin” is not preceded by a puff of air (it’s *unaspirated*).

Phone variants like this are called *allophones*.
Allophones are identified using decision trees

1. Run forced alignment on your training data with an existing ASR system.
2. For every state in your phone models, e.g. AA-b, AA-m, AA-e, ... ZH-e:
   3. Collect all observations assigned to the state by the forced alignment.
   4. Grow a decision tree using recursively binary splits. Stop when the number of samples in any node falls below a threshold.
      - Questions are about the identity of nearby phones.
      - The criterion used to choose the best split is data likelihood, using a single, multivariate Gaussian distribution as the data model.
5. Prune all the trees (using likelihood) to produce a set of context-dependent states of the desired size.
Sample decision tree for AA-b

-1 = Sonorant?

-2 = SIL?  
No  
-1 = Labial?  
Yes  
1 = Retroflex?  
No  
-1 = Nasal?  
Yes  
1 = Sonorant?  
Yes  
No  

Sample questions:
Nasal  N  M  NG
Labial  B  F  M  P  V  W
Retroflex  R  ER
Fricative  DH  F  S  SH  TH  V  Z  ZH
HMM observations are modeled using Gaussian mixtures

\[
p(x_t|q_t = i) = \sum_{j=1}^{K} \frac{w_{ij}}{2\pi \Sigma_{ij}} \frac{1}{d/2} e^{-\frac{1}{2}(x-\mu_{ij})^T \Sigma_{ij}^{-1}(x-\mu_{ij})}
\]

\(w_{ij}\) is the weight of the \(j\)-th component \((\sum_i w_{ij} = 1)\) in state \(i\), \(\mu_i\) is the mean of the \(j\)-th component in state \(i\), and \(\Sigma_i\) is the covariance of the \(j\)-th component in state \(i\).

Usually constrain covariances \(\Sigma_i\) to be diagonal.

Why GMMs?

- Mathematically tractable.
- Can approximate a wide variety of data distributions.
GMMs are trained using the E-M (Expectation-Maximization) algorithm.

Given an initial guess of GMM parameters, define the probability that an observation $x_t$ was produced by the $j$-th component as

$$p(j|x_t, q_t = i) = \frac{\frac{w_{ij}}{2\pi \Sigma_{ij}^{d/2}} e^{-\frac{1}{2}(x - \mu_{ij})^T \Sigma_{ij}^{-1}(x - \mu_{ij})}}{p(x_t|q_t = i)}$$

This is the *Expectation* step.
GMMs are trained using the E-M (Expectation-Maximization) algorithm.

We can update the parameters of the $j$-th component as follows:

$$
\mu_{ij} = \frac{1}{N_{ij}} \sum_t p(j|x_t, q_t = i)x_t
$$

and

$$
\Sigma_{ij} = \frac{1}{N_{ij}} \sum_t p(j|x_t, q_t = i)(x_t - \mu_{ij})^2
$$

where

$$
N_{ij} = \sum_t p(j|x_t, q_t = i)
$$

This is the Maximization step.

Then iterate the process.
Practical issues in GMM training

**Initialization**  Two choices:
- Randomly assign samples to components and run E-M, or
- do mixture splitting.

**Regularization**  What if a component gets only one data point?
- Floor the variances.
- Discard mixture components that don’t get a lot of data assigned to them.

**Model selection**  Various options:
- Constant number of components per state.
- Assign components based on counts: $K \propto N_i^{0.2}$
- Bayesian methods
Given initial model parameters,

1. Use the forward-backward or Viterbi algorithm to get assignment of each feature vector, \( x_t \), to an HMM state or states.

2. Given this assignment, run E-M to obtain a new set of GMM parameters (and optionally update the HMM transition parameters).

Iterate this process to convergence.
Where do we get initial model parameters?

If we already have another ASR system,

- Use it to align the training data.
- Using this alignment, train GMMs.
  - Start with random assignment of feature vectors to components.
  - Run E-M to get initial GMMs.

If not, use *flat-start* training.

- Uniformly segment training utterances and assign to context-independent states.
- Use single Gaussians to model states.
- Then run many iterations of forward-backward training.

Iterative training is standard practice in speech recognition.
Language modeling

\[ \hat{W} = \arg \max_W P(X|W; \Theta) P(W; \Theta) \]
Why do we need a language model?

**Mathematician’s answer:** Bayes’ rule requires a prior distribution over word strings, $P(W)$.

**Engineer’s answer:** The language model simplifies the search by penalizing unlikely word sequences.

**User’s answer:** The language model helps to disambiguate homonyms:

“Write a letter to Mrs. Wright, right now.”
The N-gram approximation

The probability of a word sequence, \( \mathbf{W} \), of length \( m \) is

\[
P(\mathbf{W}) = p(\langle s \rangle) p(w_1 | \langle s \rangle) p(w_2 | w_1 \langle s \rangle) p(w_3 | w_2 w_1 \langle s \rangle) \cdots p(\langle /s \rangle | w_m w_{m-1} \cdots w_2 w_1 \langle s \rangle)
\]

where \( \langle s \rangle \) and \( \langle /s \rangle \) are special start-of-sentence and end-of-sentence symbols.

But the chances of observing any sequence of \( m \) words drops exponentially with increasing \( m \).

The solution is to approximate using histories of fixed length:

\[
P(\mathbf{W}) \approx p(w_1 | \langle s \rangle \langle s \rangle) p(w_2 | w_1 \langle s \rangle) p(w_3 | w_2 w_1) \cdots p(\langle /s \rangle | w_m w_{m-1})
\]

The example above, with \( N = 2 \) is called a trigram language model because it considers words in groups of 3.
Training language models

1. Collect as much text as you can get your hands on.
2. Count word co-occurrences up to your maximum desired order.
3. **Smooth** the maximum-likelihood estimates:
   - If you assign 0 probability to an event, the recognizer will never output it.
   - Even if you stick with low-order $N$-grams, you will not see some in the training data.
4. Often get better results by training separate models on different sources, then interpolating estimates from the different models.
Where theory meets practice: The LM scaling factor

The “Fundamental equation of speech recognition” is a lie.

This is what we really do:

\[
\hat{W} = \arg \max_W P(X|W; \Theta) P(W; \Theta)^\alpha
\]

where \(\alpha\) is the “language model scaling factor.” Typically \(5 \leq \alpha \leq 25\), and \(\alpha\) is set on development data.

Why? No theoretically satisfying answer, but it works best in practice.
Search

\[ \hat{W} = \arg \max_W P(X|W; \Theta) P(W; \Theta) \]
ASR search is the same as searching for the lowest-cost path in a graph

- Usually perform Viterbi search, for best state sequence.
- Nodes in the graph correspond to entries in a dynamic programming (DP) chart.
- Arcs in the graph correspond to HMM states (observations) and have costs ($\propto$ negative log-likelihoods) from the acoustic and language models.
- At each time step in the search, the DP entries are updated with the lowest cost path that can reach the corresponding state.
The ASR search space: language model
The ASR search space: optional silences
The ASR search space: pronunciation variants

THE: DH AH
THE: DH IY
The ASR search space: context dependence
Pruning the search

For large-vocabulary tasks, the search space can be huge, e.g. 45M nodes and 190M arcs.

To speed up the search, we prune the search, disregarding less likely paths.

- **rank** pruning considers only the $N$ best (lowest cost) nodes in any time step.
- **beam** pruning disregards paths having costs that are some amount (the beam value) higher than the current best path.

Most modern speech recognizers do both.
There are two dominant search methods

**Static** search uses finite-state machine techniques to compile the language model, lexicon, and decision trees into a single, large graph.

- A static decoder can be very, very fast.
- A static decoder is very simple to write.
- Must use a smaller language model in the graph compilation.

**Dynamic** search constructs the relevant portion of the search space on the fly.

- Can use the full language model in the search.
- A dynamic decoder is usually slower than a static decoder.
- A dynamic decoder is usually much more complicated.
Where do we stand?
The standard measure of ASR performance is word error rate

1. Align reference and hypothesized word strings.
2. \( WER(\%) = 100 \frac{N_{sub} + N_{ins} + N_{del}}{N_{ref}} \)
Humans vs. Machines

<table>
<thead>
<tr>
<th>Task</th>
<th>Humans</th>
<th>ASR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connected digits</td>
<td>0.009%</td>
<td>0.7%</td>
</tr>
<tr>
<td>Wall Street Journal (clean)</td>
<td>0.9%</td>
<td>7.2%</td>
</tr>
<tr>
<td>Wall Street Journal (10 dB SNR)</td>
<td>1.1%</td>
<td>12.8%</td>
</tr>
<tr>
<td>Telephone conversations</td>
<td>4.0%</td>
<td>15.0%</td>
</tr>
</tbody>
</table>

A number of factors affect task difficulty

- Task predictability
  - Vocabulary size
  - Language model constraints
- Word confusability
  - It’s hard to wreck a nice beach.
- Audio quality
  - Bandwidth
  - Noise
  - Reverberation
  - Frequency response of the channel
- Speaker characteristics
  - Rate and style of speech
  - Accent
  - Age
  - Vocal tract length
Strengths of the statistical approach to ASR

- Statistical framework provides a solid, mathematical foundation for algorithmic development.
- Lets us compensate for our limited understanding of how human speech recognition works. Instead, we use
  - Very generic models with few constraints,
  - lots of training data, and
  - lots of computation.

“No knowledge is better than wrong knowledge.”
Hynek Hermansky
Weaknesses of the statistical approach to ASR

- Variability, especially mismatch between the training data and what you get when you really use the system, is the Achilles heel of ASR.
- It’s hard to tell where things went wrong when an ASR system fails.
- Development of an ASR system for a new language or new task requires a significant investment in data collection, transcription, and system development by highly skilled people.
For more information

http://www.ee.columbia.edu/~stanchen/e6884/outline.html has lectures for a full semester class on speech recognition taught at Columbia University.

Thanks to Stan Chen for letting me adapt some of the material from the Columbia course for this lecture!