Tutorial on Speech Recognition

Alex Acero

Microsoft Research

ACL Workshop's Tutorial

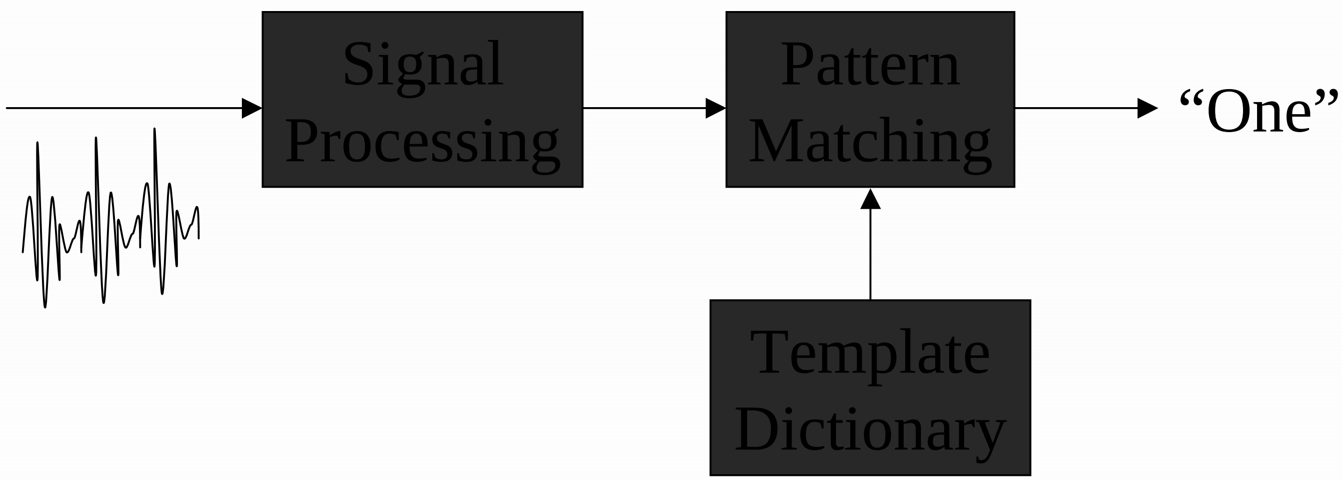
Interactive Spoken Dialog Systems:

Bringing Speech and NLP Together in Real Applications

Outline

- ASR Overview:
  - Signal Processing.
  - Hidden Markov Models Fundamentals.
  - HMMs in Speech Recognition.
  - HMM Training
  - Language Models.
- Tools and Commercial systems.
- Summary.

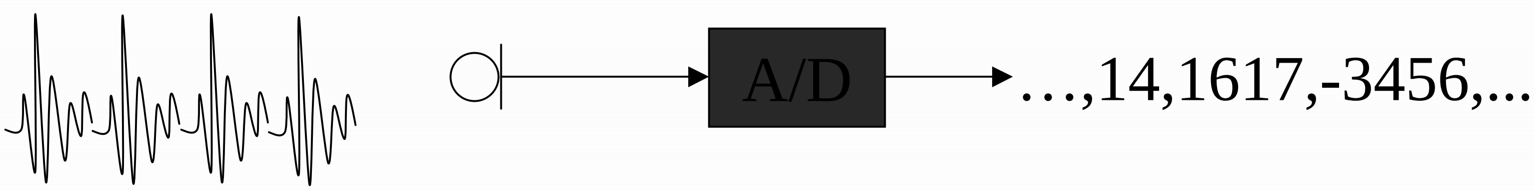
Speech Recognition Basics



Signal Processing

Signal Acquisition

- Speech is captured by a microphone.
- The analog signal is converted to a digital signal by sampling it 16000 times a second.
- Each value is quantized to 16 bits, a number between -32768 and 32767.



Spectral Analysis

- Take a **window** of 20-30 ms (320 to 480 samples).
- **FFT** (Fast Fourier Transform) used to compute energy in several frequency bands, typically 20 to 40.
- **Cepstrum** computed, 12 or so coefficients.

Spectral Analysis

- A **frame**, a snapshot of the input's spectrum, is represented by a vector of 12 cepstrum coefficients.
- To describe the spectral change with time, one frame is computed every 10 ms or so, *i.e.* 100 frames per second.
- A word that lasts 1 second is described by 1200 numbers.

HMM Fundamentals

Isolated digit recognition

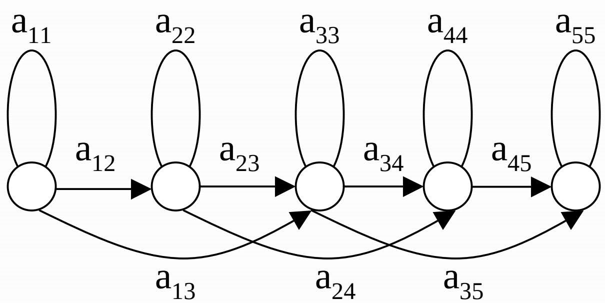
- 10 templates: one template M_i per digit.
- Compare input $\mathbf{x}$ with all templates.
- Select the most similar template *j:*

$$j = \min\{f(\mathbf{x}, M_i)\}$$

where *f* is the comparison function.

Hidden Markov Models (HMM)

HMM defined by:

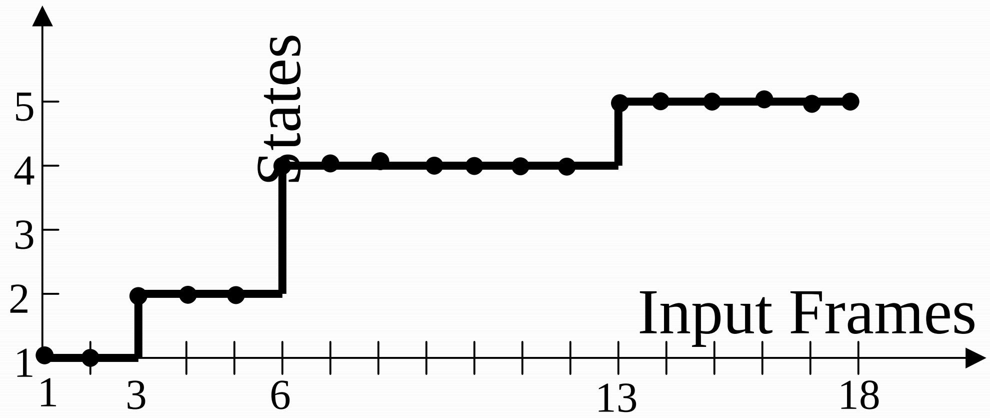


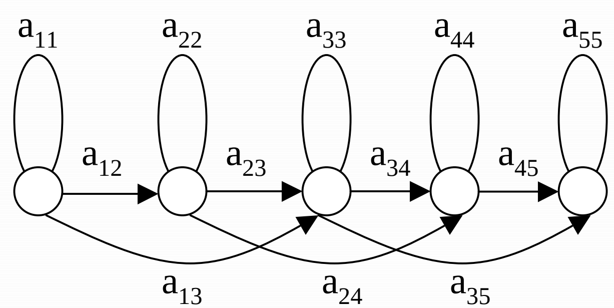
- Number of states
- Transition probabilities a_{ij}
- Output probabilities $b_i(x)$:

$$b_i(x) = \sum_{k=0}^{K-1} c_{ik} N(x, \mu_{ik}, \sigma_{ik})$$

$$N(x, \mu, \sigma) = \frac{1}{(2\pi)^{p/2}} \exp\{-\sum_{m=0}^{p-1} \frac{(x_m - \mu_m)^2}{2\sigma_m^2}\}$$

Probability of a path





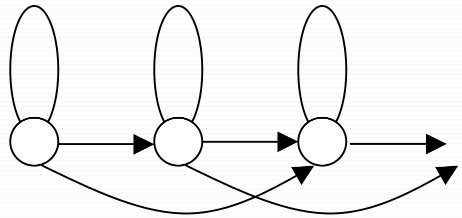
$$P = b_1(x_1)a_{11}b_1(x_2)a_{12}b_2(x_3)a_{22}b_2(x_4)a_{22}b_2(x_5)a_{24}b_4(x_6)$$
$$a_{44}b_4(x_7)a_{44}b_4(x_8)a_{44}b_4(x_9)a_{44}b_4(x_{10})a_{44}b_4(x_{11})a_{44}b_4(x_{12})$$
$$a_{45}b_5(x_{13})a_{55}b_5(x_{14})a_{55}b_5(x_{15})a_{55}b_5(x_{16})a_{55}b_5(x_{17})a_{55}b_5(x_{18})$$

Probability of a model

- Is the probability of the best path.
- Problem: We need to evaluate all possible paths, and there grow exponential with number of states and input frames.
- Solution: The Viterbi algorithm has complexity that grows linearly with number of states and input frames.

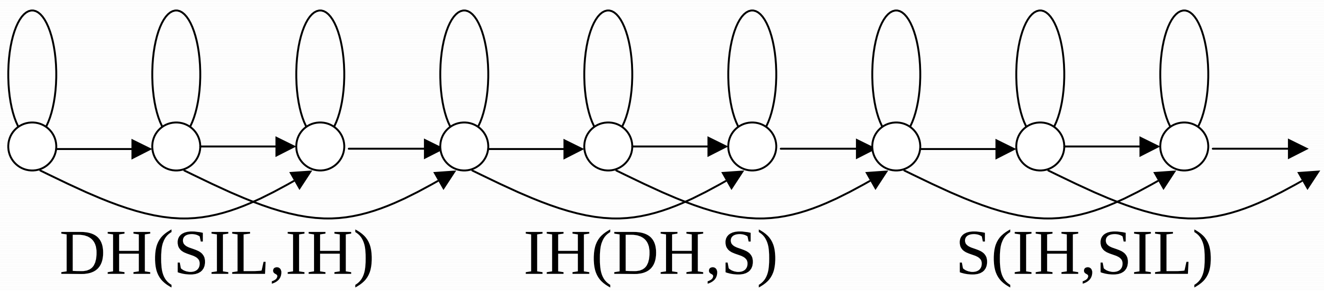
Large vocabulary

- Problem:
  - Huge number of models: 60,000 models
  - Models are hard to "train"
  - Cannot easily add new words.
- Solution: Use phoneme models



Context Dependent Models

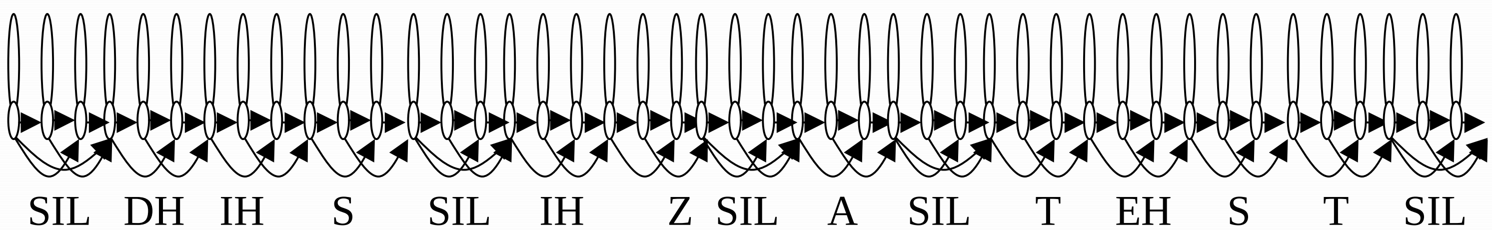
- To improve accuracy, phone models are context-dependent.
- Example: THIS = DH IH S



- 50^3=125,000 possible models are clustered to about 8,000 *generalized triphones*

Continuous speech

- Example: THIS IS A TEST
- SIL DH(SIL,IH) IH(DH,S) S(IH,SIL) SIL IH(SIL,Z) Z(IH,SIL) SIL A(SIL,SIL) SIL T(SIL,EH) EH(T,S) S(EH,T) T(S,SIL) SIL
- SIL DH(SIL,IH) IH(DH,S) S(IH,IH) IH(S,Z) Z(IH,A) A(Z,T) T(A,EH) EH(T,S) S(EH,T) T(S,SIL) SIL

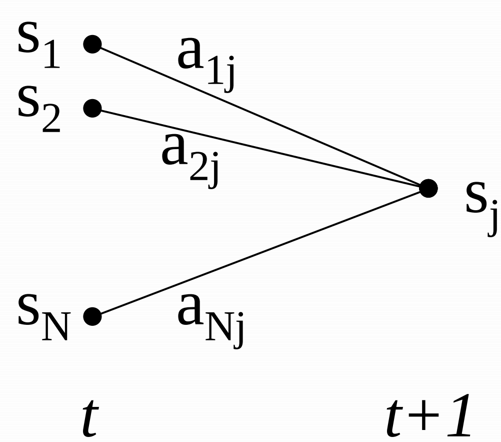


Baum-Welch Algorithm

$$\alpha_t(i) = P(x_1, x_2 \cdots x_t, s_t = i \mid \lambda)$$

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(x_{t+1})$$

$$P(X \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$$



Exact solution.

Speeding Search: Pruning

- Viterbi algorithm: replace sum with max.
- Speed-Accuracy Tradeoff: Computations can be reduced by eliminating paths that are "not promising", at the expense of having a chance of eliminating the "best" path.
- Typically, computations can be reduced by more than a factor of 10, without affecting the error rate significantly.

HMM Training

HMM Training

- A **lot** of speech is needed to *train* the models. Done with an iterative algorithm:
  1. Take initial model (*i.e.* uniform probabilities).
  2. Segment Database (Run Viterbi algorithm).
  3. Update models.
  $$\hat{\mu}_j = \frac{\sum_{i=0}^{N-1} x_{ij}}{N} \qquad \hat{\sigma}_j^2 = \frac{\sum_{i=0}^{N-1} (x_{ij} - \hat{\mu}_j)^2}{N}$$
  4. If Converged stop, otherwise go to 2.

HMM Training

- Rule of thumb: Each state has to appear in the training database at least 10 times.
- *Speaker-Independent* systems have lots of data (models well trained), but contain high variability.
- *Speaker-Dependent* systems do not have as much data (models not so well trained) but they are more consistent.

Language Models in ASR

Bayes Rule in ASR

$$P(W \mid A) = \frac{P(A|W)P(W)}{P(A)}$$

$$\hat{W} = \max P(W \mid A) = \max P(A|W)P(W)$$

where *A* is the Acoustics and *W* the sequence of words. *P(W)* is the language model.

Statistical Language Model

$$P(W) = P(W_1 W_2 W_3 \cdots W_N) = P(W_1) P(W_2 \mid W_1)$$
$$P(W_3 \mid W_2 W_1) \cdots P(W_N \mid W_{N-1} \cdots W_2 W_1)$$

$$\approx P(W_1) P(W_2) P(W_3) \cdots P(W_N)$$ Unigram

$$\approx P(W_1) P(W_2 \mid W_1) P(W_3 \mid W_2) \cdots P(W_N \mid W_{N-1})$$
Bigram

$$\approx P(W_1) P(W_2 \mid W_1) P(W_3 \mid W_2 W_1) \cdots P(W_N \mid W_{N-1} W_{N-2})$$
Trigram

Trigrams

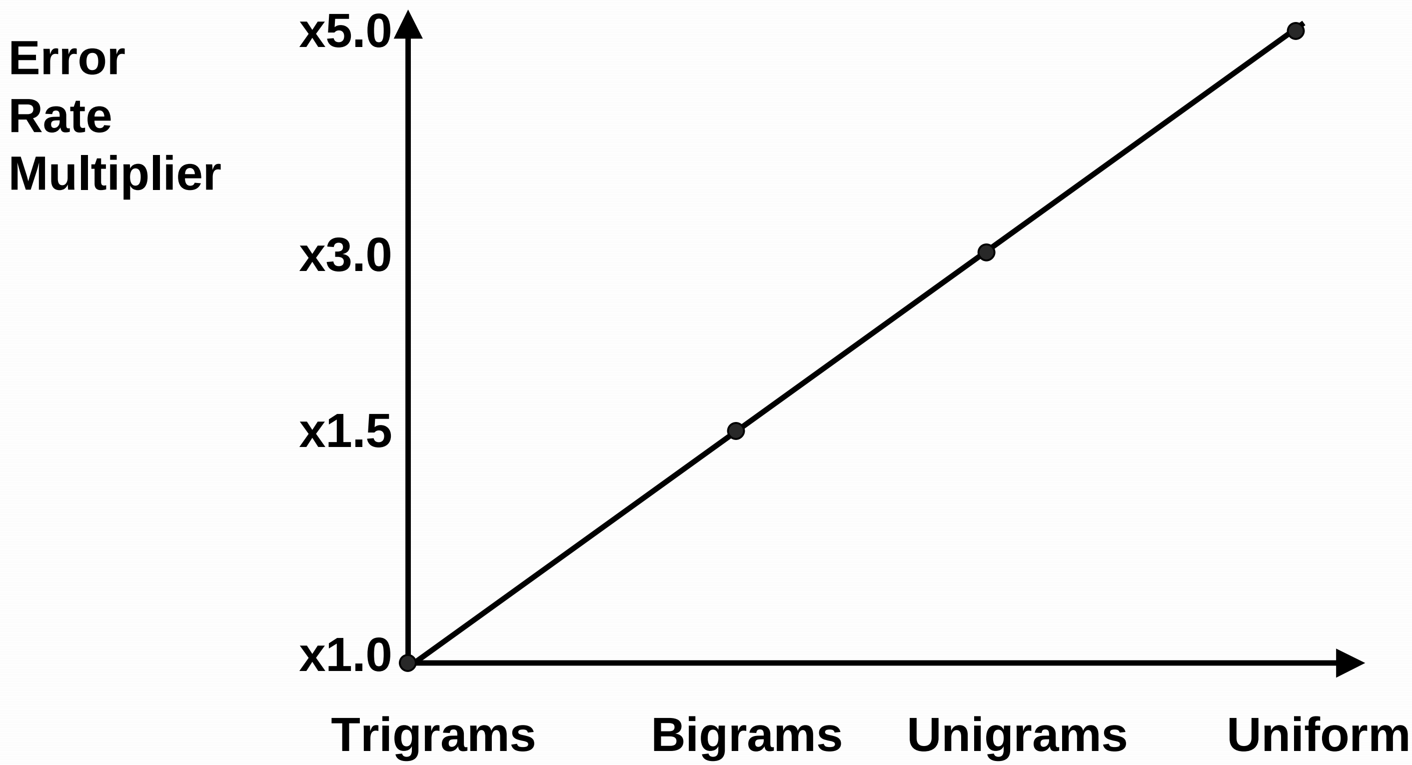
$P(\text{THIS IS A TEST}) =$

$P(\text{THIS})\ P(\text{IS}|\text{THIS})\ P(\text{A}|\text{THIS IS})\ P(\text{TEST}|\text{IS A})$

- These statistical language models **predict** the probability of the current word given the past history.
- While simplistic, they contain a lot of information

$P(\text{IS}|\text{THIS}) >> P(\text{IS}|\text{HAVE})$

N-Gram Performance



Perplexity

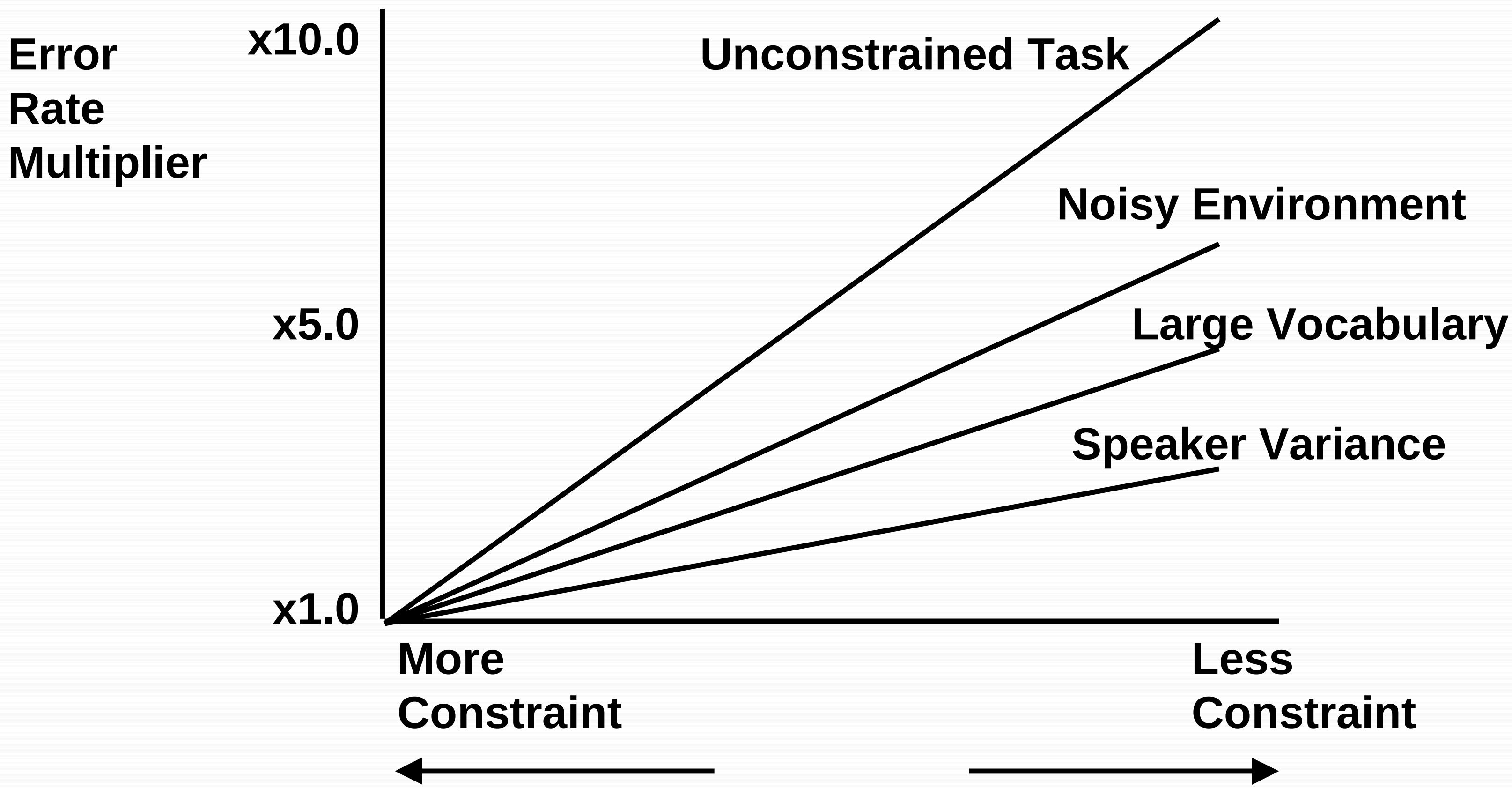
- Perplexity measures the average branching of a text when presented to a language model. $PP = 2^{LP} = \hat{P}(W_1, W_2, \cdots W_n)^{-1/n}$
- An empirical observation $E \approx k\sqrt{P}$ where E is the error rate and P the perplexity.
- A language model with low perplexity is good for ASR.
- Tradeoff between perplexity and coverage.

Context Free Grammar

- Advantages:
  - Low error rate (because of low perplexity).
  - Compact.
  - Easy for application developers.
- Disadvantages:
  - Poor coverage (out of language sentences).

ASR in Real Applications

ASR problem Space



ASR Accuracy in the Lab

- Command &Control (300 words + CFG)
  - 1% word error rate (speaker-independent)
- Discrete Dictation (60K words)
  - 3% word error rate (speaker-dependent)
- Continuous Dictation (60K words)
  - 7% word error rate (speaker-dependent)
- Telephone Digits
  - 0.3% word error rate (speaker-independent)

Commercial ASR Systems

- Requirements:
  - Windows 95 and Windows NT
  - Pentium Processor
  - 16MB (command-and-control), 24MB (discrete dictation), 32MB (continuous dictation)
- For dialogue systems we are interested in C&C (command & control).

Tools

- **Low Level** (Offer more control, but requires more work): HTK (Entropic).
- **High Level** (Offer less control but easier to use):
  - Microsoft Speech SDK
  - Novell's SRAPI
  - Sun's JSAPI

MS SAPI

- http://www.research.microsoft.com/research/srg/
- Supports ASR and TTS.
- An application can use many engines.
- ASR Engines supporting SAPI include:
  AT&T, Dragon, IBM, Kurztweil, Lernout & Hauspie, Microsoft, OMRON, Teles AG, VCS, VPC.
- TTS Engines supporting SAPI include:
  Acuvoice, AT&T, Centigram, DEC, Elan Informatique, Eloquent, First Byte, Lernout & Hauspie, Microsoft, Telia Promotor, Telefonica.

Command & Control (SAPI)

- Application passes active vocabulary to ASR engine.
- Engine finds pronunciation for active vocabulary (from Dictionary or LTS rules).
- Application passes CFG grammar.
- Engine compiles CFG grammar and listens for input, returning top N hypotheses and their probabilities.

C & C Types of Errors

- Pronunciation errors make some words *unrecognizable*.
- User says something outside the vocabulary/grammar.
- User speaks while machine is talking.
- Background noise fires recognizer (false alarm). Rejection needs improvement.
- Spontaneous speech (uhms, ahms)

ASR in Real Applications

- New word addition
- Rejection
- Barge-in.
- Robustness to noise.
- ASR is just a part of an application, User Interface is critical.

Interface NLP/ASR

- Loose coupling: NLP selects correct input from a list of top N candidate sentences:
  - Easy to implement, not optimal
- Tight coupling. NLP provides the language probabilities for the search:
  - Optimal but hard to implement.
  - Can a NLP system reduce perplexity?

Summary

- Current ASR has many limitations
- Nonetheless, it is possible to build useful systems with ASR.
- Application Developer needs to carefully design the dialogue.
- Hard to integrate NLP and Speech. Research is needed here.