ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Department of Electrical and Computer Engineering

Automatic Speech Recognition: An Introduction to Basic Techniques

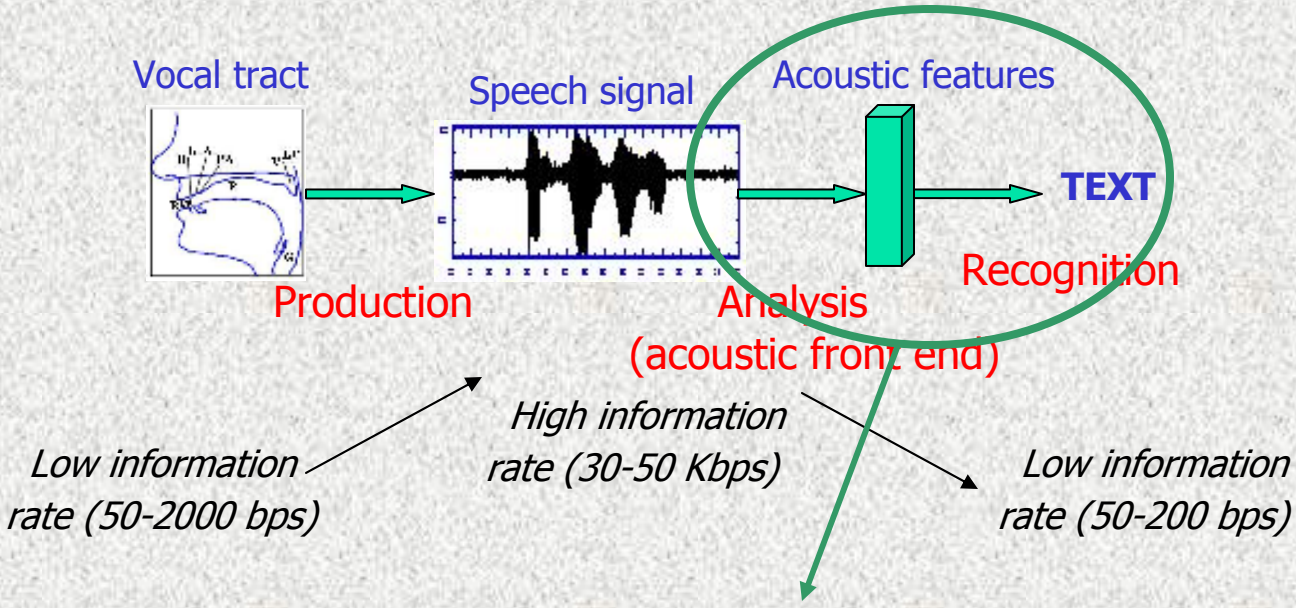
Gerasimos Potamianos

Associate Professor, Dept. of ECE,
University of Thessaly
Volos, GR

http: //www.inf.uth.gr/~gpotamianos

[1.1-INTRO] The ASR Problem at a Glance (I)

Speech is the main form of **human communication** with information conveyed as:



Automatic Speech Recognition (ASR) or **Speech-to-Text** (STT), refers to the **automatic extraction** of the **uttered word sequence** from the **observed speech signal**.

ASR is a crucial component of natural **human-computer interaction** (HCI) and **data analytics** / **information retrieval** systems.

The ASR Problem at a Glance (II)

- ASR has been an **active research area** since early **70's**. Initial attempts to address it have gone as far back as the 50's.
- Technology has been **maturing** over the past decades, allowing applications of ever **increasing complexity** and **wider deployment**, reaching the wider public:
  - Domain specific dictation (radiology transcripts);
  - Large-vocabulary dictation in clean environments;
  - Telephony applications (call center routing, customer support);
  - Embedded applications (automobiles, etc);
  - Voice search applications in relatively noisy environments; etc…
  - Other applications include education, language teaching, etc.

- Much of the progress is due to heavy **cross-fertilization** with a wide spectrum of research areas, like *machine learning*, *pattern recognition*, *signal processing*, *bioinformatics*, *finance*, *coding*, *text processing*, etc.

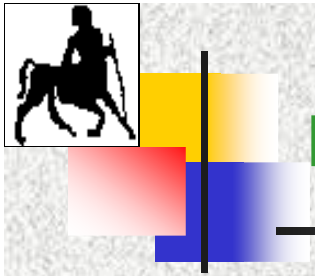
The ASR Problem at a Glance (III)

- ASR nevertheless remains a **challenging problem**, especially in noisy conditions and **unconstrained** or **mismatched** tasks and environments.
- Performance is measured in word error rate (**WER**) that breaks down to *deletions*, *insertions*, and *substitutions*, or (**SER**) in sentence/string error rate.
- Not quite at 0% errors yet ☺. But **acceptable performance** depends on app.:
  - Clearly, little error tolerance in *dictation*, but increased in *speech understanding* / *information retrieval* applications.
- What is possible today? Some **examples** ➔

  Humans perform at **~1/5** the error machines!

Task	WER
Connected digits	~ 0.1%
Read newspaper text	~ 5 %
Broadcast News	~ 10 %
Telephone conversation	~ 15 %
Far-field meetings	~ 35 %

- What **affects performance**?
  - Speaker dependency and characteristics.
  - Read vs. spontaneous speech.
  - Close-talking vs. distant speech and environment.
  - Task complexity (vocabulary size / language model perplexity).



Brief History of ASR (I)

Over **five decades of research** in ASR.

- **Early 20th Century:** Initial work on understanding human speech perception.
  - 1908: Lord Rayleigh's work.
  - 20's onwards: Significant work at Bell Labs on speech perception by Harvey Fletcher and his team (over 30 years).
  - Followed by later work at Haskins Laboratories.
- **50's-60's:** Initial attempts on ASR based on simple processing techniques operating on outputs of filter-banks.
  - 1952: AT&T's David et al. built a single-speaker isolated digit recognizer (analog system).
  - 1956: RCA's Olson and Belar built a single-speaker ten-syllable recognizer.
  - 1962: First all-software vowel recognizer.

Brief History of ASR (II)

- **70's:**
  - First DARPA program (1971–1976). Goal was to integrate speech knowledge, linguistics, and AI, aiming at ASR.
  - Work at Dragon, CMU, BBN, IBM on such ideas.
- **70's – 80's:**
  - Statistical approach to ASR. Development of hidden Markov models and basic language modeling.
  - Work at IBM, Dragon, IDA, Bell Labs.
  - 1984: Jelinek's team at IBM builds the first real-time dictation system (Tangora).
- **80's – 90's:**
  - 1986–1998: Second DARPA program. Common test sets, evaluation competitions.
  - First commercial dictation products (Dragon's *Naturally Speaking*, IBM's *ViaVoice*).

Brief History of ASR (III)

- **00 – 10's:**
  - Large US government research initiatives: EARS, GALES, RATS.
  - Significant EU funding in human language technology research.
  - Major breakthroughs in complexity of systems, training algorithms, data availability, breadth of languages, environmental robustness.
  - Significant commercialization activities (telephony, embedded, distributed).
  - Many R&D / commercial players: Google, Apple, Amazon, Nuance, IBM, AT&T, SRI, MSR.
  - Multitude of university groups active: CLSP/JHU, CMU, ICSI, MIT, IDIAP, Aachen, Cambridge, Edinburgh, USC, …

Some ASR Literature Resources (I)

- **Books:**
  - Benesty, Sondhi, Huang (Eds.), *Springer Handbook of Speech Processing*, Springer 2008.
  - Rabiner, Schafer, *Theory and Applications of Digital Speech Processing*, Prentice-Hall, 2011.
  - Jurafsky, Martin, *Speech and Language Processing*, Prentice Hall, 2009.
- **Journals:**
  - *IEEE Transactions on Audio, Speech and Language Processing.*
  - Elsevier Journals of *Speech Communication* and *Computer Speech and Language*.
  - *Eurasip Journal on Audio, Speech, and Music Processing.*
  - *Speech Technology Magazine*.
- **Conferences:**
  - ICASSP – *International Conference on Acoustics, Speech, and Signal Processing* (Spring)
  - Interspeech – *Conference of the International Speech Communication Association* (Fall) [formerly: Eurospeech, ICSLP]
  - ASRU – *Automatic Speech Recognition and Understanding Workshop*

    SLT – *Spoken Language Technology Workshop* (Winter)

Some ASR Literature Resources (II)

- **Some software Toolkits:**
  - HTK – hidden Markov model toolkit, http://htk.eng.cam.ac.uk
  - CMU Sphinx recognizer, http://cmusphinx.sourceforge.net
  - KALDI ASR software, http://kaldi.sourceforge.net/
  - SRILM – the SRI language modeling toolkit, http://www.speech.sri.com/projects/srilm/
  - PRAAT speech analysis toolkit, http://www.fon.hum.uva.nl/praat/
- **Data resources and evaluation:**
  - LDC – Linguistics Data Consortium, https://www.ldc.upenn.edu/
  - ELRA – European Language Resources Association, http://www.elra.info/
  - NIST – National Institute of Standards and Technology, http://nist.gov/itl/iad/mig/
- **Links** to online resources and educational material:
  - http://www.dev.voxforge.org/projects/Main/wiki/TheoryAndAlgorithms

The ASR Statistical Approach

- Since the late 70's, the ASR problem has been formulated as finding the **optimal** sequence of words $\hat{\boldsymbol{\omega}}$ given the acoustic signal (observations) **O**:

$$\hat{\boldsymbol{\omega}} = \arg\max_{\boldsymbol{\omega}} \Pr[\boldsymbol{\omega} \mid \mathbf{O}]$$

- Based on **Bayes' rule**, the following equation is derived that emulates the source / channel equation model:

$$\hat{\boldsymbol{\omega}} = \arg\max_{\boldsymbol{\omega}} \Pr[\boldsymbol{\omega}] \Pr[\mathbf{O} \mid \boldsymbol{\omega}]$$

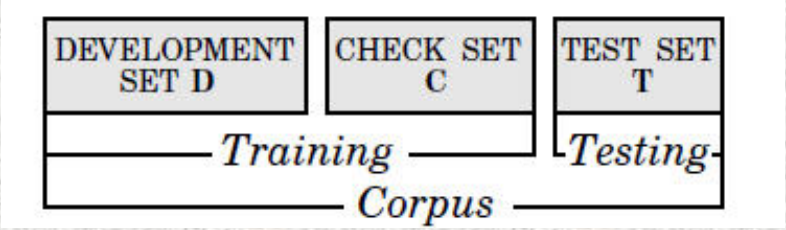
- This immediately highlights the **basic ASR research problems**:
  - **O** : **Feature extraction**.
  - ω : **Vocabulary.**
  - $\Pr[\omega]$: **Language model**.
  - $\Pr[\mathbf{O} \mid \omega]$: **Acoustic model**.
  - $\hat{\omega}$: **Search / decoding**.

Remainder Talk Overview

1. Introduction
2. Language modeling.
3. Speech feature extraction.
4. Acoustic modeling.
5. Search.

[2-LM] Language Modeling

- Aims to provide prior probability for word sequences, thus reducing the "uncertainty" (**perplexity**) in ASR.
- Assumes **causal** model: $\Pr[\boldsymbol{\omega}_1^m] = \prod_{i=1}^{m} \overline{\Pr}[\omega_i \mid \boldsymbol{\omega}_1^{i-1}]$
- Approximation using finite "**history**": $\overline{\Pr}[\omega_i \mid \boldsymbol{\omega}_1^{i-1}] \approx \Pr[\omega_o \mid \Phi(\omega_{-1},\ldots,\omega_{-N+1})] = \Pr[\upsilon \in \mathrm{Voc} \mid \mathbf{h}_{N-1}].$
- LMs are trained and evaluated on **large text corpora**, split into:

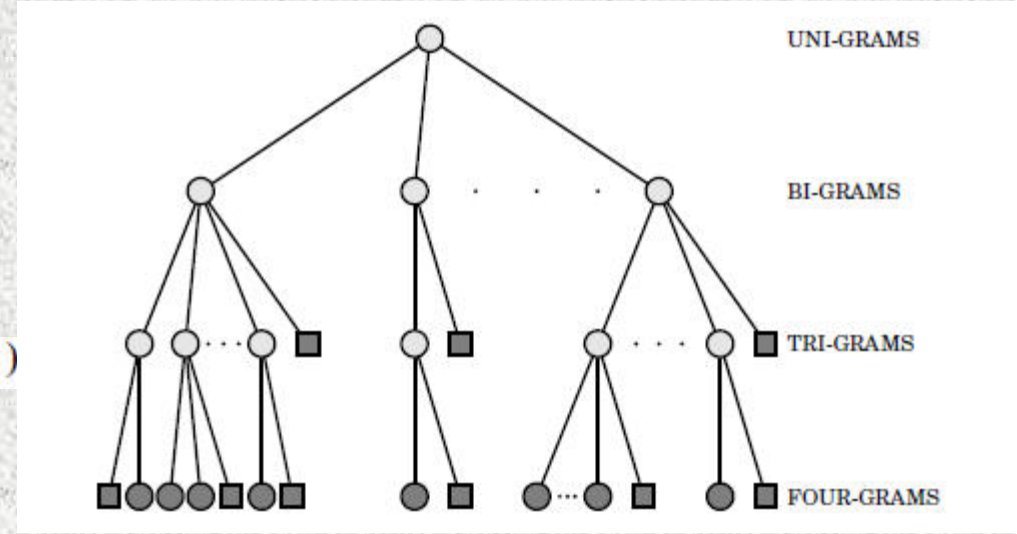


- Evaluation on basis of **perplexity** (PP):

$$LP = -\frac{1}{M}\sum_{i=1}^{M} \log Pr\,[w_i^{(t)} \mid \Phi(w_{i-1}^{(t)} \ldots w_{i-N}^{(t)})]\,. \quad \Rightarrow \quad PP = 2.0^{LP}$$

Language Modeling Using n-grams

Two problems – with their traditional solutions.



Variable-length n-gram as a decision tree

- **History classification**:

$$\Phi(w_{-1}w_{-2}\dots w_{-N}) = (w_{-1}w_{-2}\dots w_{-K})$$

where

$$K = \underset{1 \le k \le N}{\operatorname{argmax}} \{ \#_D(w_{-1}w_{-2}\dots w_{-k}) > 0 \}$$

Typically, *N=1* ➔ bi-gram, *N=2* ➔ tri-gram, or *N*=3 ➔ four-gram

- **Probability estimation**:
  - Maximum likelihood yields:

$$Pr[w \mid \Phi(w_{-1}w_{-2}\dots w_{-N})] = \frac{\#_D(w\, w_{-1}w_{-2}\dots w_{-K})}{\#_D(w_{-1}w_{-2}\dots w_{-K})}$$

  - This creates generalization issues, because unseen data get assigned zero probability! Need of probability estimates **smoothing**.

[2.2-LM] Language Modeling Smoothing (I)

Adopted **notation** for LM **smoothing** approaches:

- Estimate $p_i = Pr[w_i | \Phi(\mathbf{h})]$ from $n_i = \#_D\{w_i, \Phi(\mathbf{h})\}$.
- Denote $k = |V|$, $n = \sum_{i=0}^{k} n_i$, and $q = \#\{w_i : n_i > 0\}$.

Three general approaches.

1. **Laws of succession:** Use current "leaf" ML estimates only.

- Maximum likelihood: $p_i = \frac{n_i}{n}$.
- Laplace's law: $p_i = \frac{n_i + 1}{n + k}$.
- Lidstone's law: $p_i = \frac{n_i + \lambda}{n + k\lambda}$.
- Absolute discounting:

$$p_i = \begin{cases} (n_i - \delta)/n, & \text{if } n_i > 0 \\ q\,\delta/n\,(k-q), & \text{otherwise.} \end{cases}$$

- Linear discounting:

$$p_i = \begin{cases} (1-a)n_i/n, & \text{if } n_i > 0 \\ a/(k-q), & \text{otherwise.} \end{cases}$$

Unknown parameters are estimated on held-out data.

Language Modeling Smoothing (II)

2. **Back-off smoothing:** Use "ancestor" leaf with appropriate law-of-succession.

- **IDEA:** $Pr[w \mid w_{-1} w_{-2} \ldots w_{-N})] = Pr[w \mid w_{-1} w_{-2} \ldots w_{-K}] =$

$$= \begin{cases} LOS\left\{\frac{\#_D(w\, w_{-1} w_{-2} \ldots w_{-K})}{\#_D(w_{-1} w_{-2} \ldots w_{-K})}\right\}, \text{ if } A : \#_D(w\, w_{-1} w_{-2} \ldots w_{-K}) > 0, \\ a\, Pr[w \mid w_{-1} w_{-2} \ldots w_{-K+1}], \qquad \text{if } \#_D(w\, w_{-1} w_{-2} \ldots w_{-K}) = 0, \end{cases}$$

where

$$K = \underset{1 \le k \le N}{\operatorname{argmax}} \left\{ \#_D(w_{-1} w_{-2} \ldots w_{-k}) > 0 \right\},$$

and

$$a = \frac{1 - \sum_{A} LOS\left\{\frac{\#_D(w\, w_{-1} w_{-2} \ldots w_{-K})}{\#_D(w_{-1} w_{-2} \ldots w_{-K})}\right\}}{\sum_{\bar{A}} Pr[w \mid w_{-1} w_{-2} \ldots w_{-K+1}]}.$$

- Probabilities are computed in a **top-down** fashion.
- Unknown parameters of LOS rules are estimated on **held-out** data.
- Popular in **ASR decoding**.

Language Modeling Smoothing (III)

3. **Linear interpolation:** Use all "ancestor" leaf ML estimates up to root note.

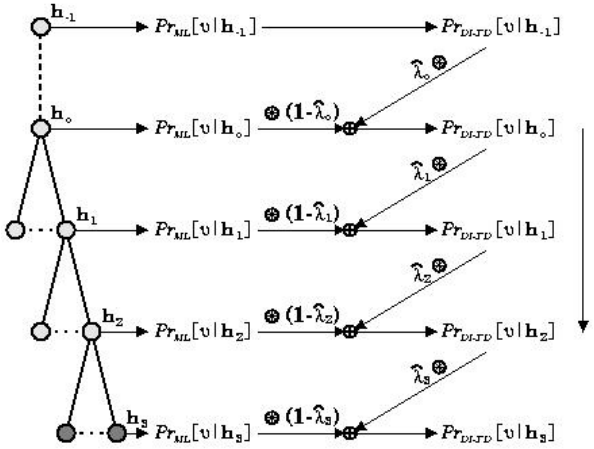
- **IDEA:** Let

$$f^{(o)}(w) = \frac{1}{|V|}, \text{ and } f^{(i)}(w) = \frac{\#_D(w\, w_{-1} w_{-2} \ldots w_{-i+1})}{\sum_w \#_D(w\, w_{-1} w_{-2} \ldots w_{-i+1})}.$$

Then,

$$Pr[w \mid w_{-1} w_{-2} \ldots w_{-N}] = \sum_{i=0}^{K+1} \lambda_i f^{(i)}(w),$$

where λ_i are chosen to minimize the LP of the "check set", $0 < \lambda_i < 1$ and $\sum_{i=0}^{K+1} \lambda_i = 1$.



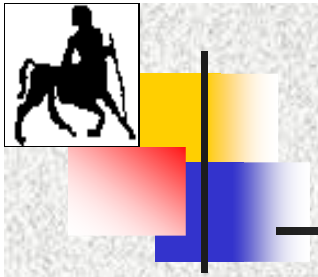
LM probability "smoothing" by top-down linear interpolation

- Estimation of the weights can be performed in various ways, for example sequentially **top-down**, **bottom-up**, or **simultaneously** for all.
- Approach is of interest when also **combining multiple language models**.

[2-LM] Other LM Issues and Approaches.

- Various LM issues / approaches:
  - **Class** LM.
  - **Cache** / Topic LM.
  - LM **interpolation**.
  - LM **pruning**.
  - LM **combination** of grammars and n-grams.
  - Incorporation of additional sources of information (**parts-of-speech** tags).
  - **Language-specific** modeling approaches.

- Other approaches:
  - **Maximum-entropy** language models.
  - **Neural network** language models.
  - **Latent Dirichlet allocation** models.



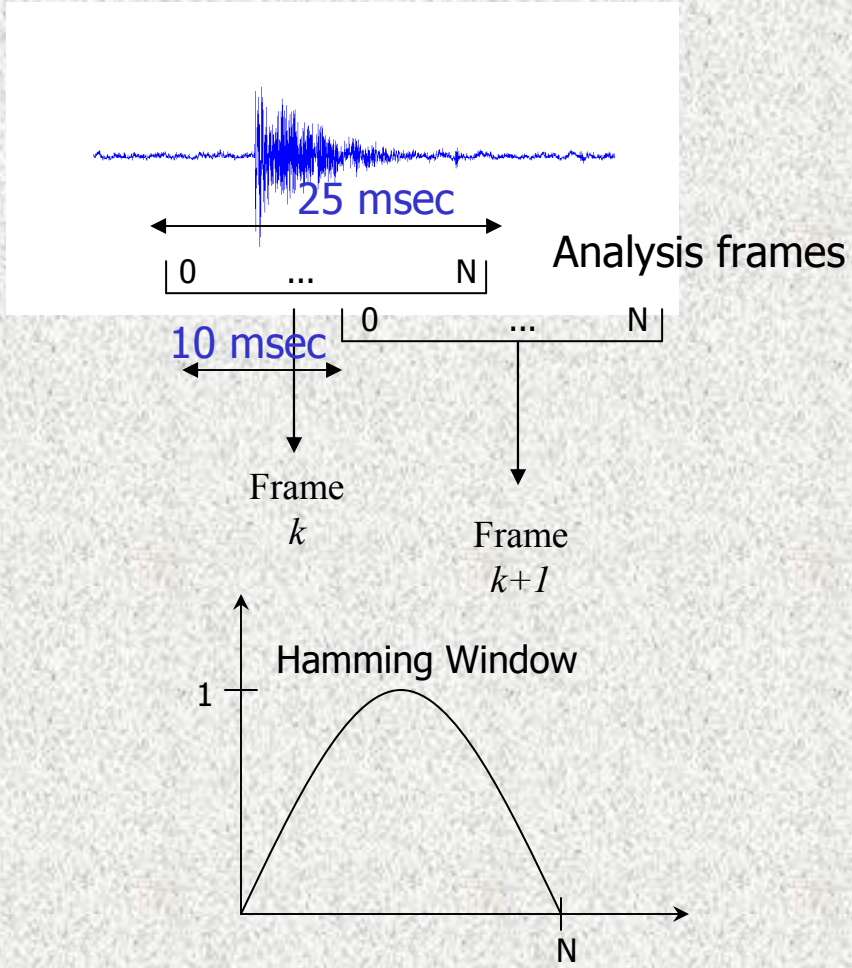
1. Introduction
2. Language modeling.
3. Speech feature extraction.
4. Acoustic modeling.
5. Search.

[3-FE] Speech Feature Extraction

- **Goal:** Extract sequence of features, **O**, from acoustic signal, $\{s_n\}$
- **Main considerations** – Features should be:
  - Informative about what was spoken.
  - Invariant to speaker and environment.
  - Compressing the signal to low dimensional feature vector.
  - Hopefully mimicking human speech perception.
- Various **approaches** exist. Most prevalent ones are low-level, signal based on:
  - The **linear predictive coding** (**LPC**) model of speech.
  - Filter-bank analysis, e.g. **mel-frequency cepstral coefficients** (**MFCC**).
  - Combination of the above, e.g. **perceptual linear prediction** (**PLP**).
- We also discuss:
  - Signal **pre-processing**.
  - Feature **post-processing**.

[3.1-FE] Signal Pre-Processing

- Processing is applied in **short-duration "frames"**, typically of a 25 msec length, with some overlap (typically 10 ms). Signal in frame is $\{s_n, n=1,\ldots,N\}$.



- The following are applied on frame:
  - **DC signal removal**.
  - **Signal pre-emphasis**:

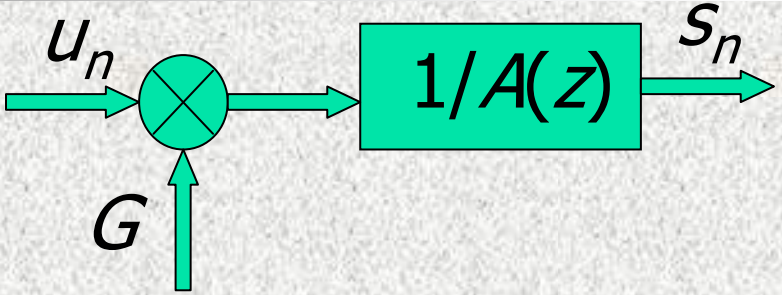
$$s'_n = s_n - 0.97\, s_{n-1}$$

  - **Hamming windowing**:

$$s'_n = \left\{0.54 - 0.46\cos\left(\frac{2\pi(n-1)}{N-1}\right)\right\} \times s_n$$

[3.2-FE] Linear Prediction (LP) Speech Analysis

- Vocal tract is modeled as an all-pole filter, driven by an excitation term:



$$s_n = \sum_{i=1}^{p} a_i s_{n-i} + G u_n$$

- LP analysis aims to minimize the prediction error: $\mathrm{E}\left[s_n - \sum_{i=1}^{p} a_i s_{n-i} \right]^2$ and thus is a MSE problem.
- Efficiently solved using **Durbin**'s algorithm for inverting the *pxp* autocorrelation equation system. Results in **LPC** (linear prediction coefficients): $a_1, a_2, ..., a_p$.
- Superior ASR performance is achieved using the **LPCC** (LP cepstral coefficients):

$$c_m = a_m + \sum_{k=1}^{m-1} \frac{k}{m} c_k a_{m-k};\ m = 1,..., M \le p$$

- Typically, $M = 12$, $p = 14$.

[3.3-FE] Filter-Bank Speech Analysis / MFCCs

- Computes speech energy in a number of bands, after suitable band-pass filtering.
- Following human perception, bands are non-uniform. Typically, triangular filters are used (H_j), with uniform spacing along the **mel** frequency scale:

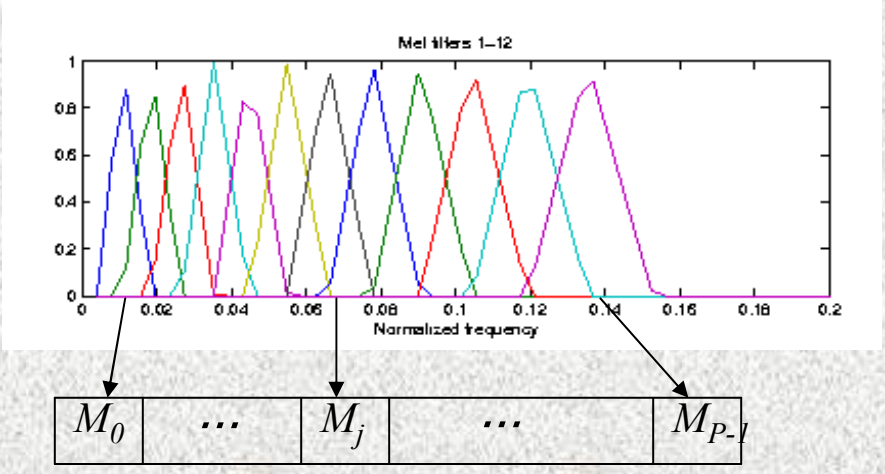
$$\text{mel}(f) = 2595 \log_{10}(1 + \frac{f}{700})$$

- **Mel-frequency cepstral coefficients** (**MFCC**) are obtained by a discrete cosine transform of the log filterbank amplitudes,

$$M_j = 20 \log_{10}\left[\sum_{k=0}^{N-1} |S[k]|\, H_j[k]\right]$$

i.e.,

$$c_i = \sqrt{\frac{2}{p}} \sum_{j=0}^{p-1} M_j \cos\left(\frac{\pi i}{p}(j-0.5)\right)$$



- Occasionally, the $\{M_j\}$'s are used as features (**log Mel FB energies**).

[3.4-FE] Perceptual Linear Prediction (PLP)

- **Aims** to **combine** best features of both LPC and MFCCs
  - Uses perceptual based frequency scale (aka MFCCs).
  - Uses smooth spectral fit (aka LPC)
  - Employs perceptual based amplitude scaling.

- **Basic steps** for PLP feature extraction:
  - Computes **mel-wrapped power spectrum,** $M_j = \left[\sum_{k=0}^{N-1} |S[k]|\, H_j[k] \right]$
  - Takes **cubic root** of above.
  - Takes the **IDFT** of a symmetrized version of the result (ensuring real result).
  - Performs **LPC** assuming that the above result is a signal **autocorrelation**.
  - Obtains **cepstral** LPC coefficients.

[3.5-FE] Other Features Used in ASR

A multitude of other features have been proposed in the literature for ASR, e.g.:

- **AM-FM** speech **signal** representation – instantaneous amplitudes & frequencies:

$$s_n = \sum_{j=0}^{p-1} a_{jn} \cos(\phi_{jn})$$

- **Articulatory** features:
  - Manner, voicing characterizations.
  - Formant locations, formant bandwidths, etc.

- **Statistical Classifier** based features:
  - **Tandem** approach: Features are transformed versions of statistical classifier **posteriors**, typically of **neural networks**.
  - **Bottleneck** features: Features extracted from "narrow" **hidden layer** of multi-layer **neural network**, typically employed as acoustic model.

[3.6-FE] Feature Post-Processing / Normalization

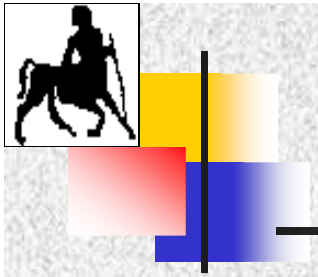
- **Weighting** of the LPC coefficients (also known as "cep-liftering"):

$$c'_n = \left(1 + \frac{L}{2}\sin\frac{\pi n}{L}\right) \times c_n \qquad (\text{e.g., } L = 22).$$

- **Augmentation** of the feature vector (LPCC or MFCC) by log of signal energy:

$$E = \log \sum_{n=1}^{N} s_n^2$$

- **Normalization** by subtracting $E_{\max}$-1 for energy, **mean** for other features.
- **Variance normalization** occasionally employed (typical in NN-based systems).
- Inclusion of "**dynamic**" information, by augmenting features with first and second derivatives, or "learning" dynamic features as a dimensionality-reduction projection of a **concatenation** of features from consecutive, neighboring frames.
- Feature **transformations** (projection / rotation) to other spaces for better statistical modeling (compaction, de-correlation). Examples are **PCA**, **LDA**, **HLDA**, **MLLT**. Many of these require **class label** information.
- Vocal track length normalization (**VTLN**): Frequency axis wrapping.



1. Introduction
2. Language modeling.
3. Speech feature extraction.
4. Acoustic modeling.
5. Search.

[4-AM] Acoustic Modeling

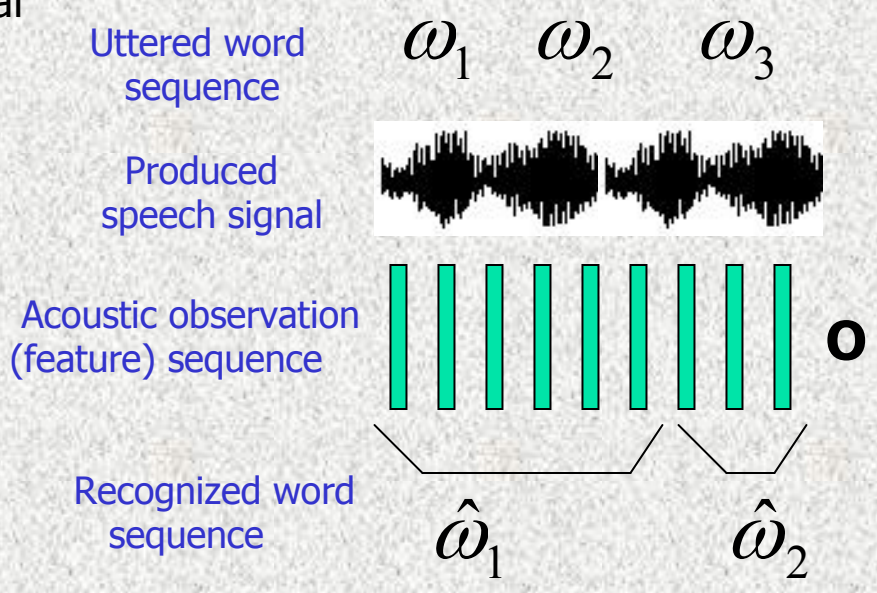
- **Statistical approach** to ASR uses **maximum a-posteriori** (MAP) estimation to obtain optimal word sequence:

$$\hat{\boldsymbol{\omega}} = \arg\max_{\boldsymbol{\omega}} \Pr[\boldsymbol{\omega} \mid \mathbf{O}]$$

  - **"Hidden"** words are **partially observed** through sequence of acoustic features.
- **Two models are needed:**
  - Prior probability of word sequences (**language model**).
  - Generative model of acoustic features from word sequence (**acoustic model**).

$$\Pr[\boldsymbol{\omega} \mid \mathbf{O}] \propto \Pr[\mathbf{O} \mid \boldsymbol{\omega}] \Pr[\boldsymbol{\omega}]$$

AM **LM**



Hidden Markov models (HMMs)

- HMMs are popular generative models for time series of observations. They are characterized by following:
  - States: C={1,2,…,N}. Denote q_t state at t.
  - Initial state distribution:

$$\boldsymbol{\pi} = \{\pi_i = \Pr[q_1 = i],\ i = 1,\ldots,N\}$$

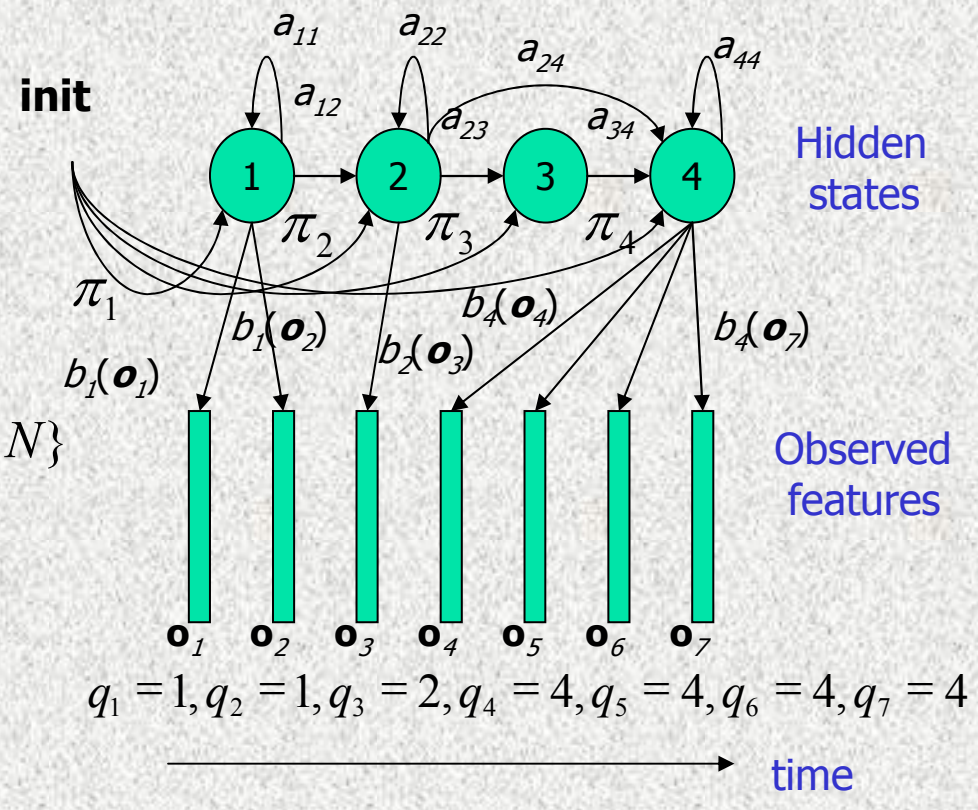
  - State transition probabilities:

$$\mathbf{a} = \{\mathrm{a}_{ij} = \Pr[q_{t+1} = j \mid q_t = i],\ i,j = 1,\ldots,N\}$$

  - State conditional observation probability:

$$\mathbf{b} = \text{parametric representation of}$$
$$\{b_j(\mathbf{o}_t) = \Pr[\mathbf{o}_t \mid q_t = j],\ j = 1,\ldots,N\}$$

- Thus, HMM parameters are: $\boldsymbol{\theta} = [\boldsymbol{\pi}, \mathbf{a}, \mathbf{b}]$



[4.1-AM] HMMs - Cont.

The class-conditional observation probabilities **b** can be:

- **Discrete**, in case that the observation vectors are drawn from a finite set. This can be achieved by vector quantization of the feature space (codebook of size *K*):

$$\mathbf{b} = \{b_j(k) = \Pr[\mathbf{o}_t \approx \mathbf{v}_k \mid q_t = j],\ j = 1,\ldots,N, k = 1,\ldots,K\}$$

- **Continuous**, typically considered as a mixture of multi-dimensional Gaussians:

$$b_j(\mathbf{o}_t) = \sum_{m=1}^{M_j} c_{jm} N(\mathbf{o}_t; \boldsymbol{\mu}_{jm}, \mathbf{U}_{jm}), \quad j = 1,\ldots,N$$

where the d-dimensional Gaussians are

$$N_d(\mathbf{o}; \boldsymbol{\mu}, \mathbf{U}) = \frac{1}{\sqrt{(2\pi)^d |\mathbf{U}|}} \exp\left[-\frac{1}{2}(\mathbf{o}-\boldsymbol{\mu})^{\mathrm{T}} \mathbf{U}^{-1}(\mathbf{o}-\boldsymbol{\mu})\right]$$

and the mixture weights satisfy: $\sum_{m=1}^{M_j} c_{jm} = 1, \quad c_{jm} \geq 0, \quad j = 1,\ldots,N, \quad m = 1,\ldots,M_j$

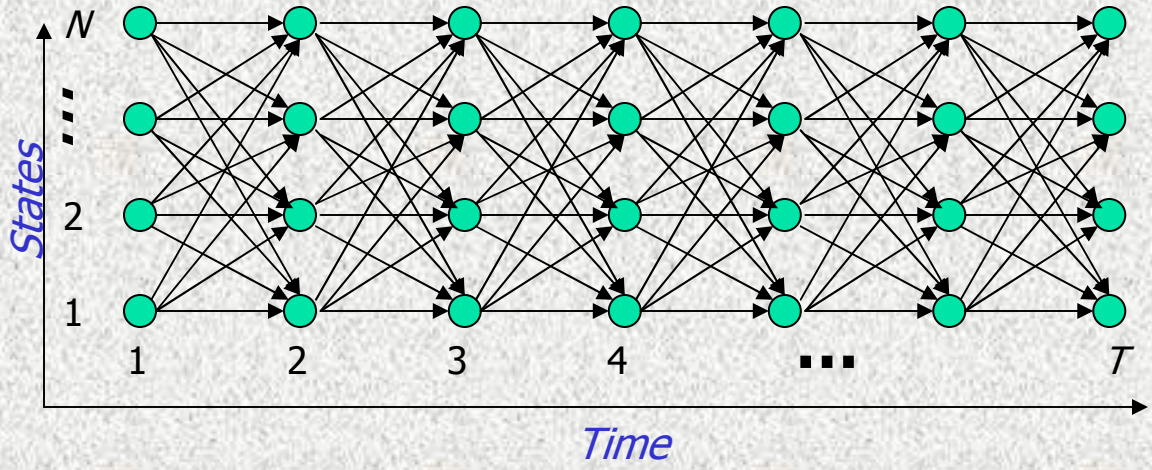
Parameters are then: $\mathbf{b} = \{c_{jm}, \boldsymbol{\mu}_{jm}, \mathbf{U}_{jm},\ j = 1,\ldots,N,\ m = 1,\ldots,M_j\}$

[4.1-AM] HMMs - Cont.

- **The three basic HMM problems.** Recall:
  - Observation sequence of duration T: $\mathbf{O}=[\mathbf{o}_1,\mathbf{o}_2,\ldots,\mathbf{o}_T]$.
  - State sequence: $\mathbf{q}=[q_1,q_2,\ldots,q_T]$.
  - Model parameters: $\boldsymbol{\theta}=[\boldsymbol{\pi},\mathbf{a},\mathbf{b}]$
- **Problem 1:** Given **O** and model parameters, how do we compute $\Pr(\mathbf{O}\,|\,\boldsymbol{\theta})$?
  - "Evaluation" of model fit to the data.
  - Solved by the "forward" or "backward" procedure.
- **Problem 2:** Given **O** & model parameters, what is the optimal state seq. **q**?
  - Uncovers the "hidden" states – used in **recognition**!
  - Solved by the **Viterbi** algorithm.
- **Problem 3:** What are the model parameters that optimize $\Pr(\mathbf{O}\,|\,\boldsymbol{\theta})$?
  - This is the **maximum-likelihood parameter estimation** problem.
  - Solved by the **forward-backward algorithm** (or Baum-Welch), an instance of the expectation-maximization (EM) procedure.

[4.1-AM] HMMs - Cont.

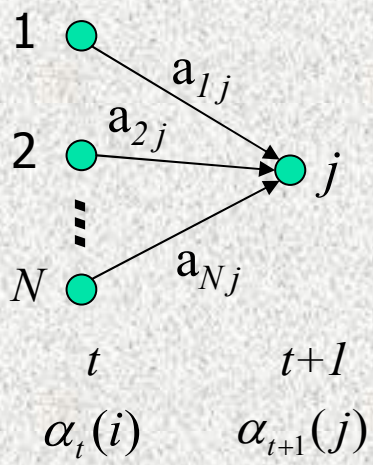
- Brute force solution to these problems is exponential on T, i.e., O(TN^T).
- Luckily, dynamic programming solutions exist!
- They utilize partial computations on the 2-D **lattice** of TxN states in time.
- Complexity of resulting algorithms is O(N^2T).



[4.1.1-AM] HMMs – Solution to Problem 1

- **Problem 1:** Compute the probability of an observed sequence, given model,

$$\Pr(\mathbf{O}\,|\,\boldsymbol{\theta}) = \sum_{q_1,q_2,\ldots,q_T} \pi_{q_1} b_{q_1}(\mathbf{o}_1)\mathrm{a}_{q_1q_2} b_{q_2}(\mathbf{o}_2)\ldots\mathrm{a}_{q_{T-1}q_T} b_{q_T}(\mathbf{o}_T)$$



- Can be efficiently computed by means of the **forward procedure**:
  - Define **forward variable** of time t and state i:

$$\alpha_t(i) = \Pr(\mathbf{o}_1\mathbf{o}_2\ldots\mathbf{o}_t, q_t = i\,|\,\boldsymbol{\theta})$$

  - Initialization:

$$\alpha_1(i) = \pi_i b_i(\mathbf{o}_1), \quad \text{for} \quad i = 1,\ldots,N$$

  - Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)\mathrm{a}_{ij}\right] b_j(\mathbf{o}_{t+1}), \quad \text{for} \quad j = 1,\ldots,N, \quad t = 1,\ldots,T-1$$

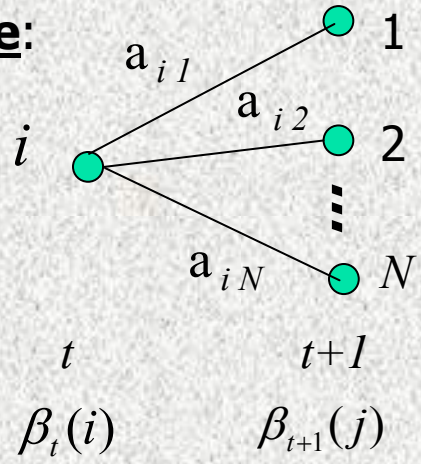
  - Termination:

$$\Pr(\mathbf{O}\,|\,\boldsymbol{\theta}) = \sum_{i=1}^{N} \alpha_T(i)$$

[4.1.1-AM] HMMs – Solution to Problem 1 – Cont.

- Can also be solved by means of the **backward procedure**:
  - Define the backward variable of time t and state i.

$$\beta_t(i) = \Pr(\mathbf{o}_{t+1}\mathbf{o}_{t+2}...\mathbf{o}_T \mid q_t = i, \boldsymbol{\theta})$$



  - Initialization:

$$\beta_T(i) = 1, \quad \text{for all states} \quad i = 1,..., N.$$

  - Induction:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j), \quad \text{for} \quad t = T-1, T-2,...,1, \quad i = 1,..., N$$

  - Termination:

$$\Pr(\mathbf{O} \mid \boldsymbol{\theta}) = \sum_{i=1}^{N} \pi_i b_i(\mathbf{o}_1) \beta_1(i)$$

[4.1.2-AM] HMMs – Solution to Problem 2

- **Problem 2:** Given **O** and model parameters, find the optimal state sequence.
  - This can be obtained via the **Viterbi algorithm**.
  - Define bookkeeping array $\psi_t(j)$ and best state sequence score up to *t:*

$$\delta_t(i) = \max_{q_1,q_2,\ldots,q_{t-1}} \Pr(q_1 q_2 \ldots q_{t-1}, q_t = i, \mathbf{o}_1 \mathbf{o}_2 \ldots \mathbf{o}_t \mid \boldsymbol{\theta})$$

  - Initialization:

$$\delta_1(i) = \pi_i b_i(\mathbf{o}_1), \ \psi_1(i) = 0, \quad \text{for} \quad i = 1,\ldots,N$$

  - Recursion:

$$\delta_t(j) = \max_{i=1,\ldots,N} [\delta_{t-1}(i)\, a_{ij}]\, b_j(\mathbf{o}_t), \ t = 2,\ldots,N, \ j = 1,\ldots,N$$

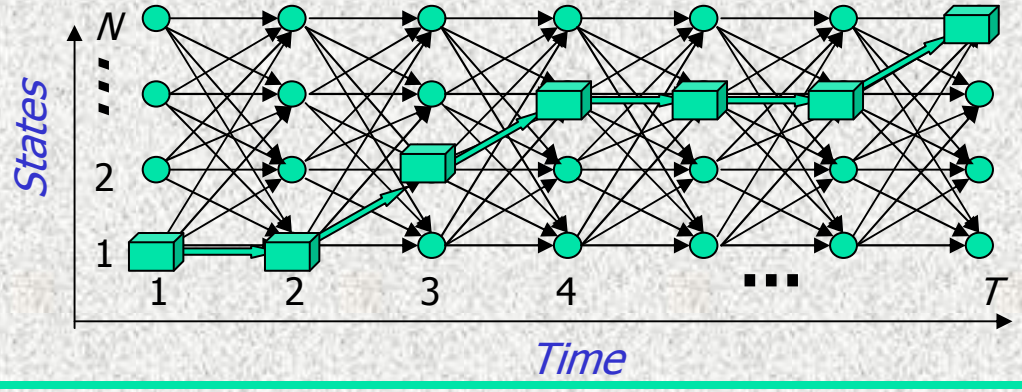
$$\psi_t(j) = \arg\max_{i=1,\ldots,N} [\delta_{t-1}(i)\, a_{ij}], \quad t = 2,\ldots,N, \ j = 1,\ldots,N.$$

  - Termination:

$$\hat{q}_T = \arg\max_{i=1,\ldots,N} [\delta_T(i)].$$

  - Backtracking: $\hat{q}_t = \psi_{t+1}(\hat{q}_{t+1})$,

$$t = T-1, T-2, \ldots, 1.$$



[4.1.3-AM] HMMs – Solution to Problem 3

- **Problem 3:** Maximum likelihood parameter estimation problem.

- Solution via the **expectation-maximization** (**EM**) *iterative* algorithm.
  - Define the **auxiliary function**:

$$Q(\boldsymbol{\theta}', \boldsymbol{\theta}) = \sum_{\mathbf{q}} \Pr(\mathbf{O}, \mathbf{q} \mid \boldsymbol{\theta}') \log \Pr(\mathbf{O}, \mathbf{q} \mid \boldsymbol{\theta})$$

  - **E-step:** Auxiliary function is expectation $E_{\boldsymbol{\theta}'}[\log \Pr(\mathbf{O}, \mathbf{q} \mid \boldsymbol{\theta})]$ under current $\boldsymbol{\theta}'$
  - **M-step:** Obtain new parameters as $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}', \boldsymbol{\theta})$
  - EM results in increased likelihood:

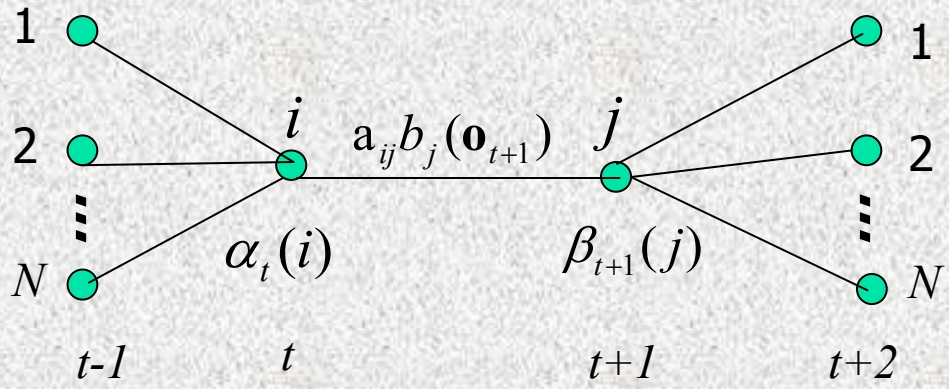
$$Q(\boldsymbol{\theta}', \hat{\boldsymbol{\theta}}) \geq Q(\boldsymbol{\theta}', \boldsymbol{\theta}') \Rightarrow \Pr(\mathbf{O} \mid \hat{\boldsymbol{\theta}}) \geq \Pr(\mathbf{O} \mid \boldsymbol{\theta}')$$

- Its HMM implementation is known as the **Baum-Welch**, or forward-backward algorithm.

[4.1.3-AM] HMMs – Solution to Problem 3 – Cont.

- Define state i occupation probability at time t: $\gamma_t(i) = \Pr(q_t = i \,|\, \mathbf{O}, \boldsymbol{\theta})$
- Denote probability of occupying transition from state i to j at time t:

$$\xi_t(i,j) = \Pr(q_t = i, q_{t+1} = j \,|\, \mathbf{O}, \boldsymbol{\theta}) = \frac{\alpha_t(i)\mathrm{a}_{ij} b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)\mathrm{a}_{ij} b_j(o_{t+1})\beta_{t+1}(j)}$$



- Then, estimate new model parameters:

$$\hat{\pi}_i = \gamma_1(i) \;;\; \hat{\mathrm{a}}_{ij} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)}$$

and in case of discrete observations (VQ): $\hat{b}_j(k) = \frac{\sum_{t=1, s.t.:\mathbf{o}_t \to v_k}^{T}\gamma_t(j)}{\sum_{t=1}^{T}\gamma_t(j)}$

[4.1.3-AM] HMMs – Solution to Problem 3 – Cont.

- In the case of continuous observations, modeled as a Gaussian mixture, we denote the probability of being at state j at time t, with mixture k accounting of $\mathbf{o}_t$, as:

$$\gamma_t(j,k) = \frac{\alpha_t(j)\beta_t(j)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(i)} \times \frac{c_{jk} N_d(\mathbf{o}_t;\boldsymbol{\mu}_{jk},\mathbf{U}_{jk})}{\sum_{m=1}^{M_j} c_{jm} N_d(\mathbf{o}_t;\boldsymbol{\mu}_{jm},\mathbf{U}_{jm})}$$

- **Then:**

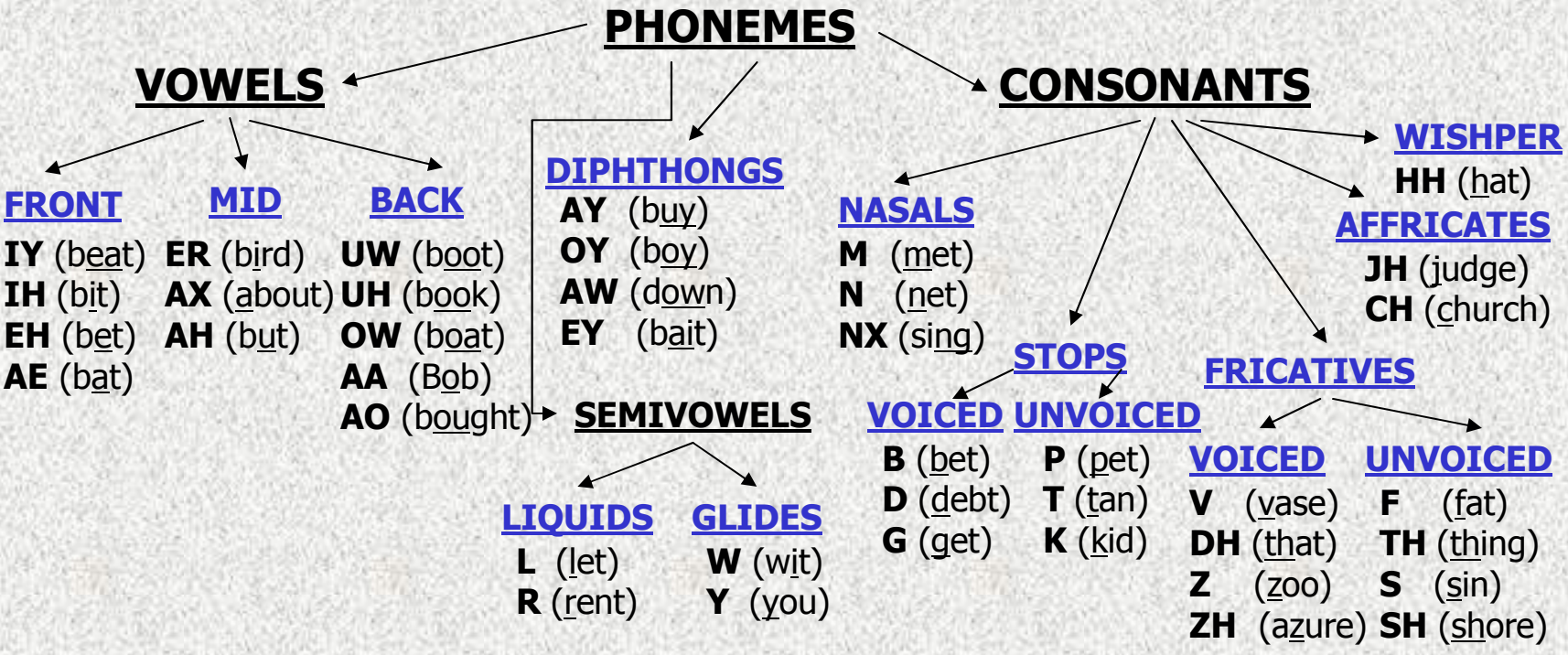
$$\hat{c}_{jk} = \frac{\sum_{t=1}^{T}\gamma_t(j,k)}{\sum_{t=1}^{T}\sum_{m=1}^{M_j}\gamma_t(j,m)} \quad ; \quad \hat{\boldsymbol{\mu}}_{jk} = \frac{\sum_{t=1}^{T}\gamma_t(j,k)\,\mathbf{o}_t}{\sum_{t=1}^{T}\gamma_t(j,k)}$$

and:

$$\hat{\mathbf{U}}_{jk} = \frac{\sum_{t=1}^{T}\gamma_t(j,k)\,(\mathbf{o}_t-\hat{\boldsymbol{\mu}}_{jk})(\mathbf{o}_t-\hat{\boldsymbol{\mu}}_{jk})^{\mathrm{T}}}{\sum_{t=1}^{T}\gamma_t(j,k)}$$

Acoustic modeling using HMMs

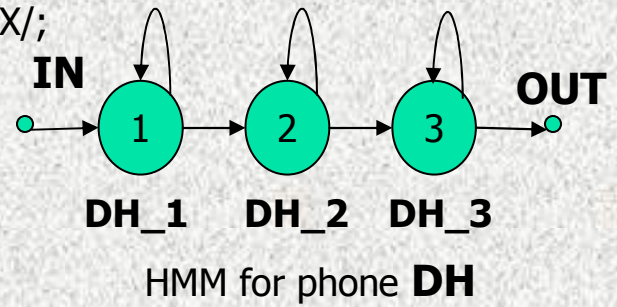
- **Phonemes:** Basic units that describe how speech conveys linguistic information.
- In statistical based ASR (especially large-vocabulary), they constitute the basic **HMM** units.
- **Basic grouping** of the phonemes used in American English (ARPAbet upper case version).



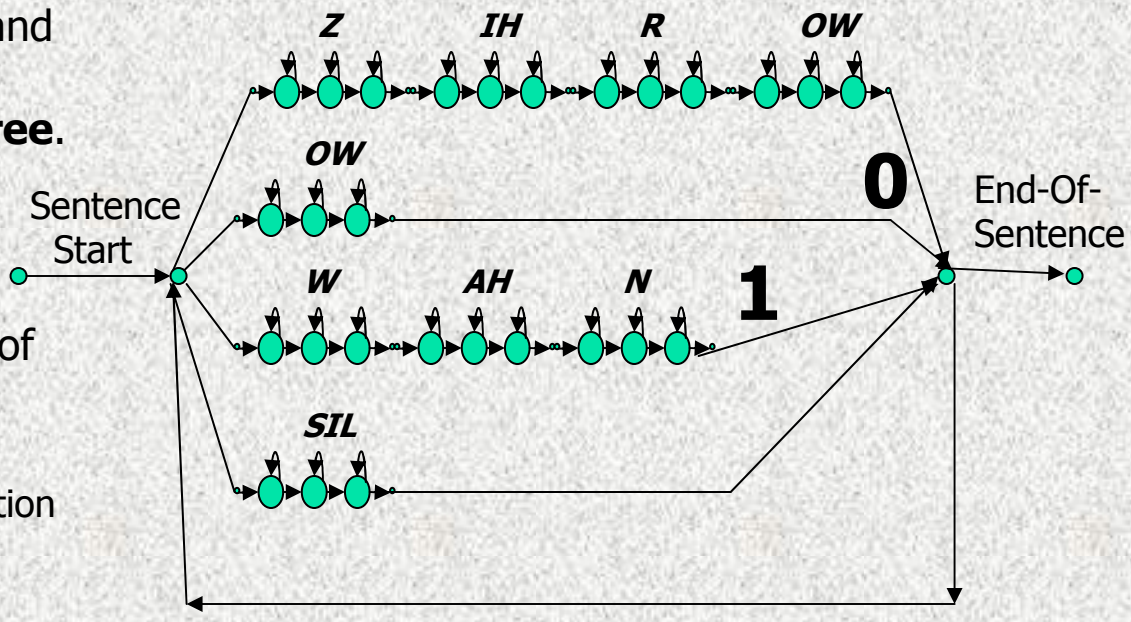
Acoustic modeling using HMMs – Cont.

- Words are modeled as phone sequences (phonetic dictionary).
- Phones are typically modeled as 3-state **left-2-right** HMMs.
- To improve performance, states have **context-dependent** observation pdfs. Contexts are clusters of left and right phonetic sequences (1-5 in length), obtained by a **decision tree**.
- Training and recognition is then performed utilizing the HMM algorithms discussed previously (problems 2 and 3), on a network of HMM states, composed by words, phones, and sub-phonetic units.
  - Example of 0-1 connected recognition using context-independent units.

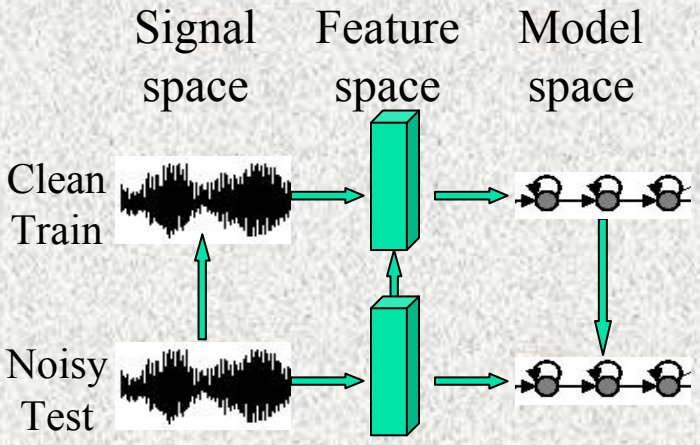
THE = /DH IY/;/DH AX/;
THEME= /TH IY M/;



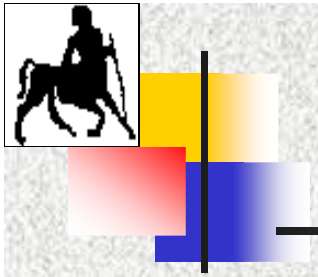
HMM for phone **DH**



ASR robustness / adaptation



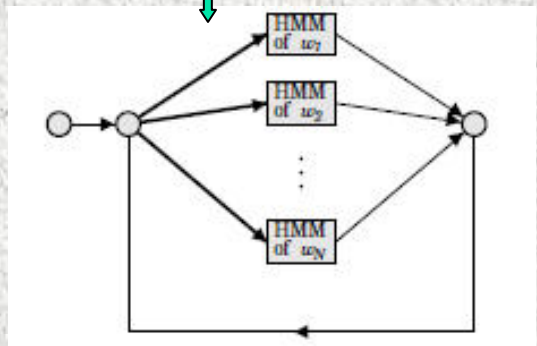
- Typically, ASR performance **degrades** in noisy environments, and **mismatched** conditions and **unseen** speakers in training (lack of **robustness**).
  - Performance can be improved by **noise compensation**, or in case available sample of the new condition / subject, by **adaptation**.
- Three categories of techniques:
  - **Signal** space, **feature** space, **model** based.
  - E.g.: Spectral subtraction, Wiener filtering, vocal tract length normalization (**VTLN**), noise adaptive prototypes, parallel model combination (**PMC**), maximum-a-posteriori adaptation (**MAP**), maximum likelihood linear regression (**MLLR**), speaker-adaptive training (**SAT**), feature-space MLLR (**FMLLR**), etc.
- These techniques are moderately only successful. Lack of robustness remains an issue.

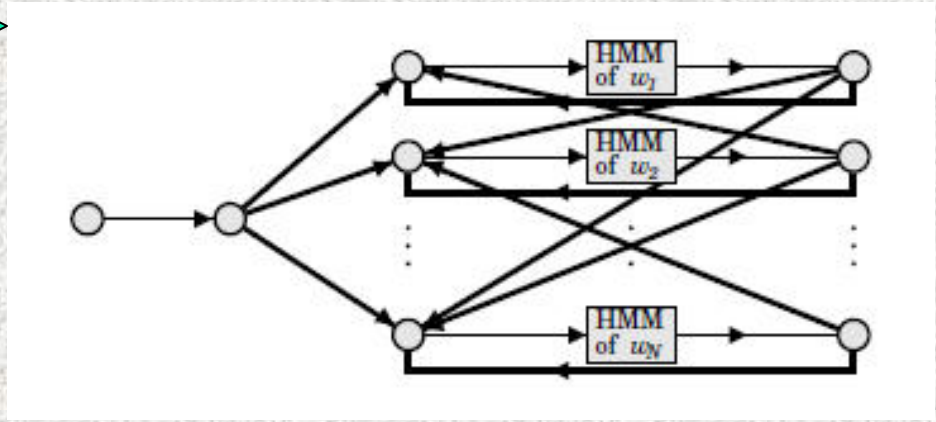


1. Introduction
2. Language modeling.
3. Speech feature extraction.
4. Acoustic modeling.
5. Search.

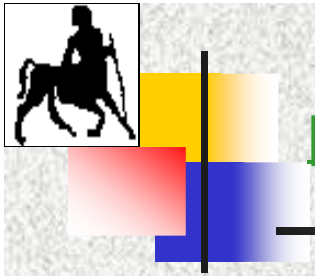
[5-DEC] Decoding/Search – Graph Composition

- **So far** we have discussed: Language model, dictionary, HMMs.
- All these need to be "**composed**" together into a "**graph**", over which Viterbi decoding will be performed.
- **n-gram LMs** can be easily converted into a graph.
  - Trivial for uni- and bi-grams.





  - Needs thought for tri-grams and above, so that state-space does not "explode".
  - Solution: Can be implemented using **back-off LM** approach.
  - LM pruning can further reduce the size of the graph.



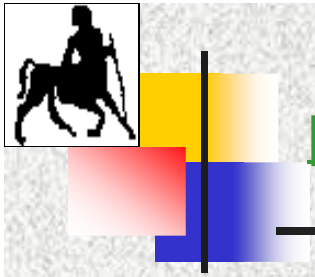
[5.1-DEC] Graph Composition (II)

Following the LM graph creation …

- Each **word** can be expanded to one or more **phone sequences** based on the dictionary.
- Then, each **phone** can be expanded with into its **HMM** states and transitions.
  - Easy for word- or context-independent models.
  - Difficult for context-dependent HMMs employing cross-word modeling.

Graph composition is facilitated by the **theory of finite state machines** (**FSMs**): weighted finite state acceptors (**FSAs**), weighted finite state transducers (**WFSTs**).

- Pioneered at AT&T-Labs in late 90's.
- FSM toolkit (http://www2.research.att.com/~fsmtools/fsm/)
- Operations of interest:
  - Weighted **composition** creates the final graph.
  - Weighted **determinization** and **minimization** compact it.



[5.2-DEC] Static vs. Dynamic Decoding

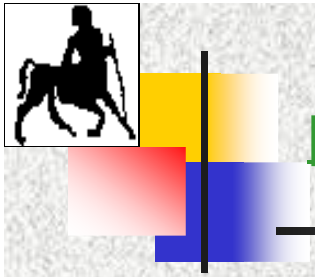
Decoding graph can be:

1. **Static:** All components are integrated into a single network, before ASR is run.

- *Approach:* FSM-based composition of all system components (decision tree, dictionary, language model).
- *Pros:*
  - Expansion is performed off-line, does not affect decoding run-time.
  - Allows better optimization.
- *Cons:*
  - Large CPU and memory required during graph buildup.
  - Cannot use adaptive vocabulary.
  - LM must be FSM-representable.

2. **Dynamic:** At least some system components get integrated during run-time.

- *Approach:* Varies by R&D group.
- *Pros:* Provides flexibility and ASR performance improvements, e.g., when dialog-state based language models can be utilized.
- *Cons:* Slows down ASR process.



[5.3-DEC] Decoding Strategies / Other Issues

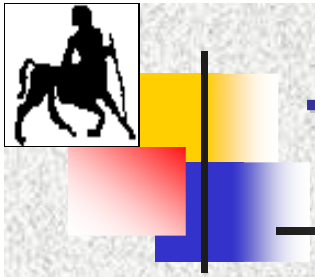
So far we have discussed **Viterbi decoding** (synchronous).

There is also an alternative decoding strategy called **A*search** / **stack decoding**.
Uses "fast match" scores to decide which paths to extend, exploring best-looking paths. Approach is rather complicated and becoming obsolete.

Finally, there is the **2-pass decoding strategy**, where ASR output is rescored by typically more complex LMs. Typically operates on **lattices** or **n-best** lists.

Additional points in Viterbi decoding:

- Path **pruning**.
- LM / AM **re-weighting**.
- **Word insertion penalty**.



Thank you for your attention!