Keyword Spotting & Utterance Verification

Berlin Chen
Department of Computer Science & Information Engineering
National Taiwan Normal University

References:

1. R.C. Rose, “Word spotting – extracting partial information from continuous utterance”, in the book “Automatic Speech and Speaker Recognition,” edited by C.H. Lee et al.
2. M. G. Rahim et al., “Discriminative utterance verification for continuous digits recognition,” IEEE Tans. Speech and Audio Processing, May 1997
3. B. Chen et al., “A*-Admissible Key-Phrase Spotting with Sub-Syllable Level Utterance Verification," ICSLP 1998

Introduction (1/3)

- Keyword spotting is to spot semantically significant fragments (keywords) from the utterances and reject irrelevant sounds
  – Irrelevant sounds can include “out-of-vocabulary” (OOV) words, background acoustic events and noise, etc.

- Most keyword spotting systems comprise two major constituents (two-stage or one-stage approaches)
  – A mechanism (search strategy) allows for generating keyword hypotheses from a continuous utterance
  – A mechanism (utterance verification or hypothesis test) allows for verifying the occurrence of keyword hypotheses

Introduction (2/2)

- Keyword spotting have been applied to a broad array of problems
  - Commend control (or name/phone dialing)
  - Telecommunications services (keyword detection in customer responses to automated operator queries)
  - Information retrieval from stored speech messages (spoken documents)
  - etc.

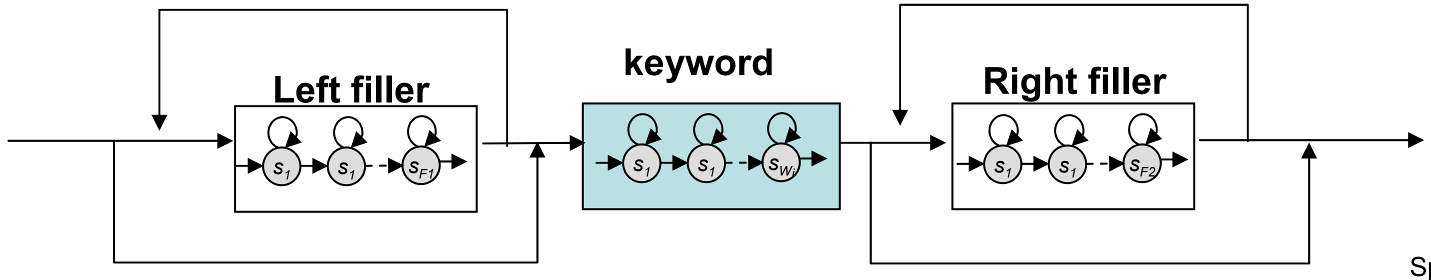
Filler pauses,
Hesitation,
Repetition,
Out-of-vocabulary words (OOV)

"Mm,...,"
"I wanna talk ..talk to.."
"What?"
幫我找台..台灣銀行的乁電話

Search Paradigms (1/2)

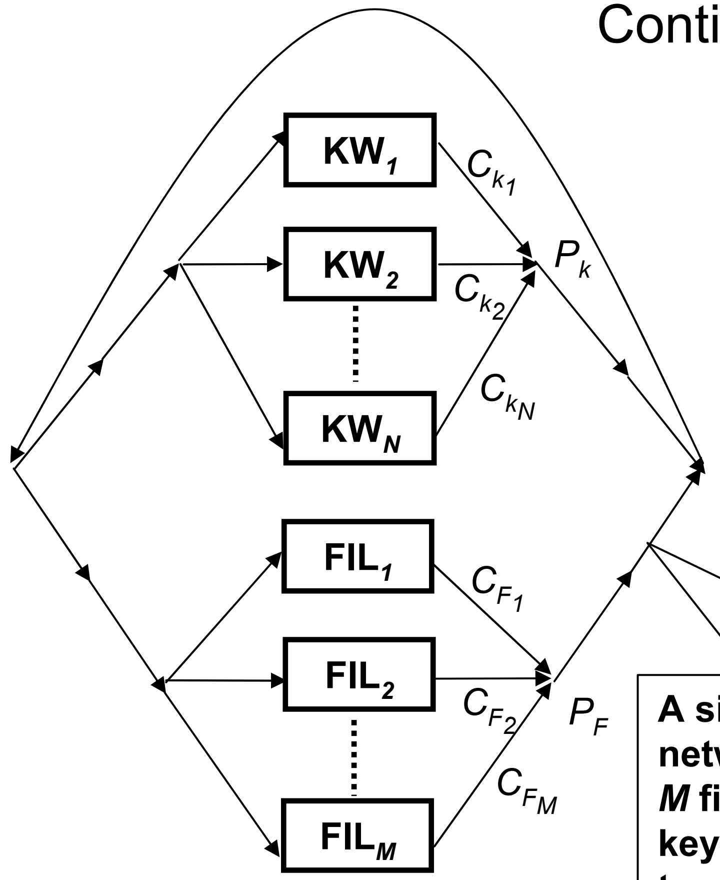
- Detection and segmentation of speech utterances into keyword hypotheses
  - Keyword hypotheses are often generated by incorporating explicit models of OOV words and non-speech sounds (usually, filler HMMs) that compete in a search procedure with models of the keywords (usually, keyword HMMs)
- Two paradigms

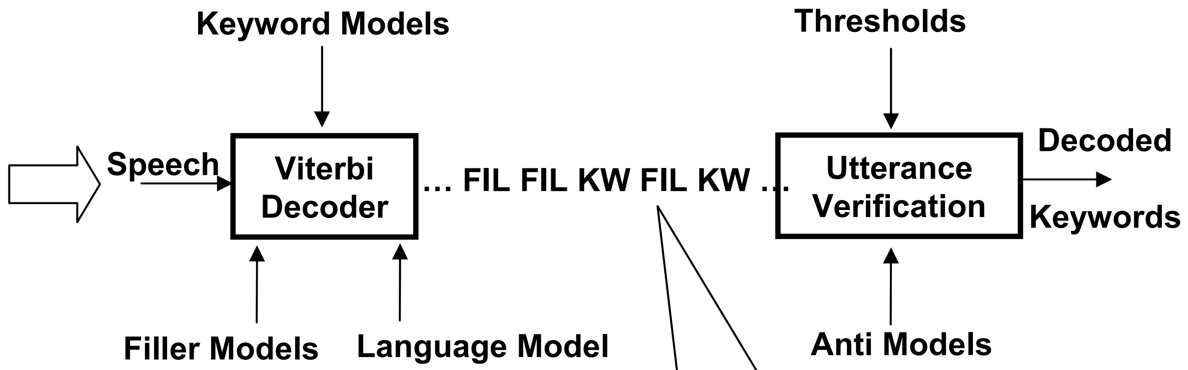
  1. The input utterance contains a single keyword embedded in OOV words, background sounds, or noise
     - A simple example of search network (Linear Lexicon)



Search Paradigms (2/2)

2. The input utterance contains any number of keywords and the input utterance is of indefinite length
   - An example of search network (Tree Structure & Continuous Speech Recognition)





A simple, unconstrained finite state network contains *N* keywords and *M* fillers. Associated with each keyword and filler are word transition penalties.

A continuous stream of keywords and fillers.

Utterance Verification (1/4)

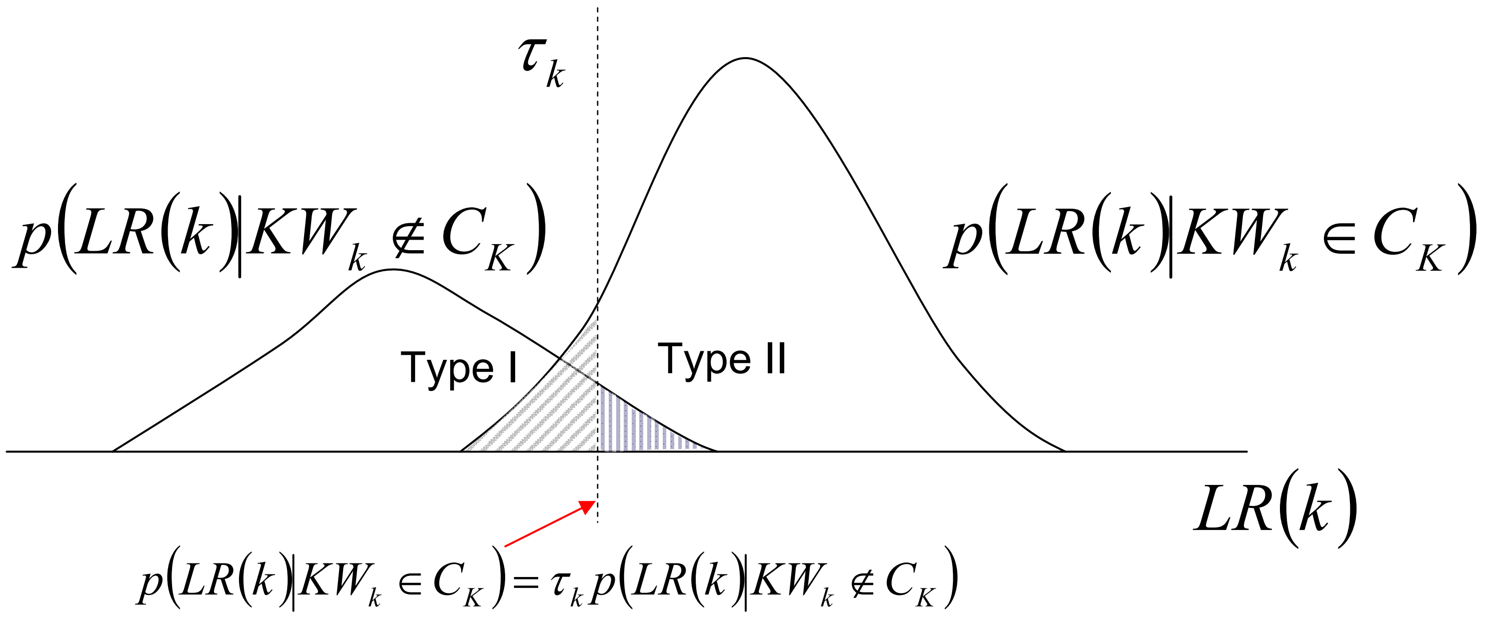
- Attempt to reject or accept part of all of an utterance based on a computed confidence score
- Hypothesis test
  - Given a speech segment O
  - **Null hypothesis (**H_0**)**: a given keyword or a set of keyword exists
  - **Alternative hypothesis (** H_1 **)**: such keyword or keyword set does not exist
  - Test null hypothesis against alternative hypothesis
    - Usually conducted using a **likelihood ratio** statistic; e.g., for a keyword hypothesis k

$$LR(k) = \frac{p_k(O|H_0)}{p_k(O|H_1)} > \tau_k$$

critical threshold

Utterance Verification (2/4)

- Histogram of the likelihood ratio $LR(k)$



τ_k
$p(LR(k)|KW_k \notin C_K)$
$p(LR(k)|KW_k \in C_K)$
Type I
Type II
$LR(k)$
$p(LR(k)|KW_k \in C_K) = \tau_k p(LR(k)|KW_k \notin C_K)$

  – We have to trade-off between Type I and Type II errors
    - Type I: false rejection
      – A null hypothesis (valid keyword) is rejected
    - Type II: false acceptance (false alarm)
      – A alternative hypothesis (invalid keyword), or a keyword that is incorrectly recognized, is accepted

Utterance Verification (3/4)

- Possible outcomes for a keyword spotting system

Decision	Keyword	Non-keyword
Acceptance	(Correctly Classified) K_A_C	NK_A
	(Incorrectly Classified) K_A_IC	
Reject	K_R	NK_R

"putative" errors $\rightarrow$ K_A_IC

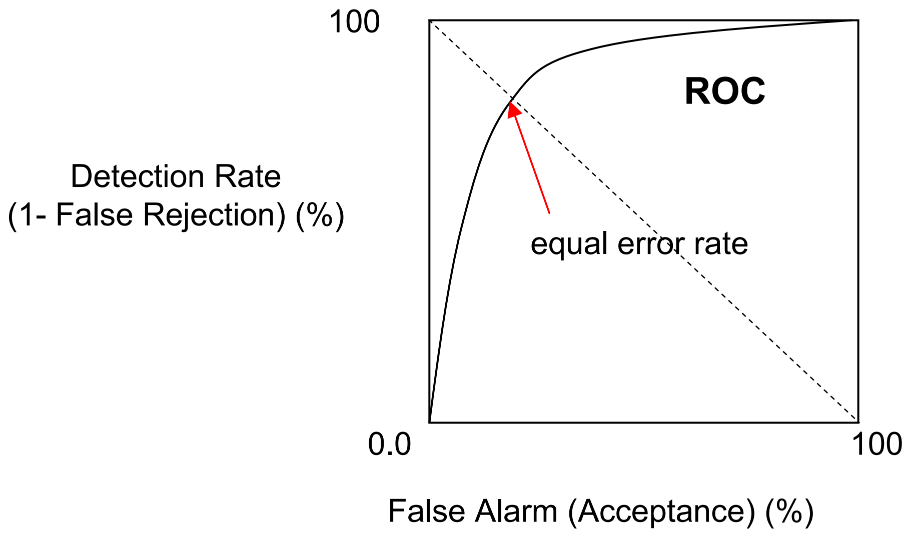
$$\text{Detection Rate} = \frac{\text{K\_A\_C}}{\text{K\_A\_C} + \text{K\_A\_IC} + \text{K\_R}}$$

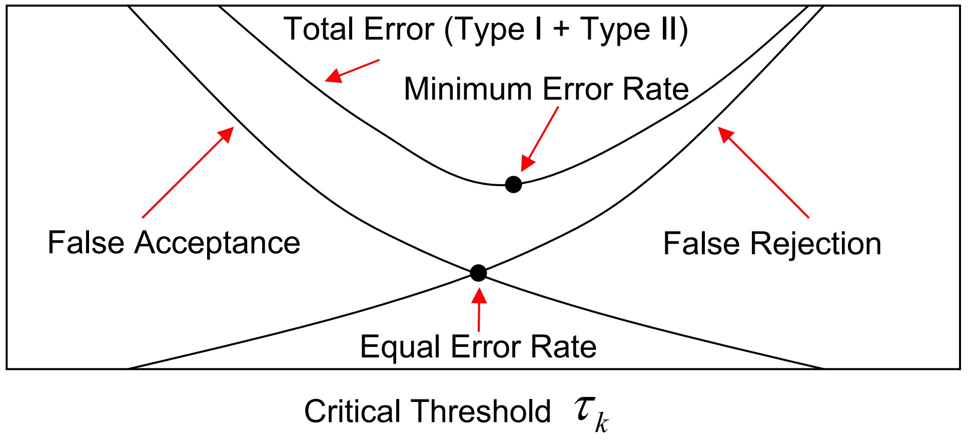
$$\text{False Rejection} = \frac{\text{K\_A\_IC} + \text{K\_R}}{\text{K\_A\_C} + \text{K\_A\_IC} + \text{K\_R}}$$

$$\text{False Acceptance} = \frac{\text{K\_A\_IC} + \text{NK\_A}}{\text{Number of utterances (or Total hours)}} (?)$$

Utterance Verification (4/4)

- Receiver Operating Curve (ROC) curve





A* Search (1/3)

- History of A* Search in AI
  - The most studied version of the best-first strategies (Hert, Nilsson,1968)
  - Developed for **additive cost measures** (The cost of a path = sum of the costs of its arcs)
- Properties
  - Can sequentially generate multiple recognition candidates
  - Need a good heuristic function
- Heuristic
  - A technique (domain knowledge) that improves the efficiency of a search process
  - Inaccurate heuristic function results in a less efficient search
  - The heuristic function helps the search to satisfy admissible condition
- Admissibility
  - The property that a search algorithm guarantees to find an optimal solution, if there is one

A* Search (2/3)

- A Simple Example
  - **Problem**: Find a path with highest score form root node “A” to some leaf node (one of “L1”,”L2”,”L3”,”L4”)

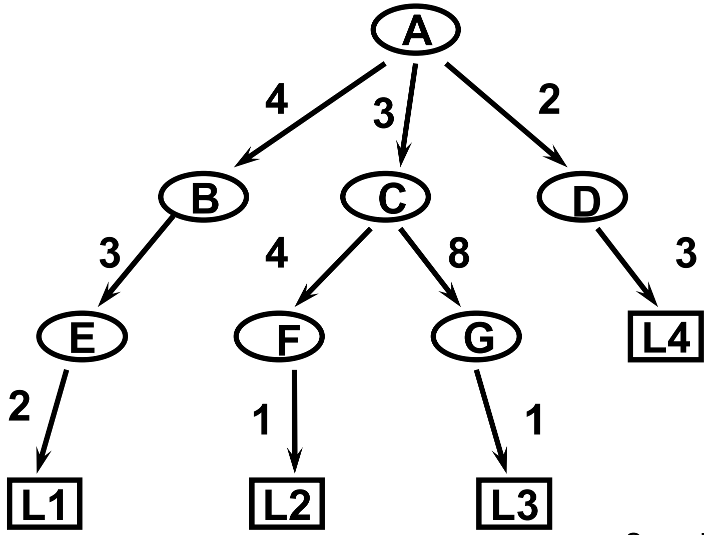
$f(n) = g(n) + h(n)$, evaluation function of node n

$g(n)$: cost from root node to node n, decoded partial path score

$h^*(n)$: exact score from node n to a specific leaf node

$h(n)$: estimated score from node n to goal state, heuristic function

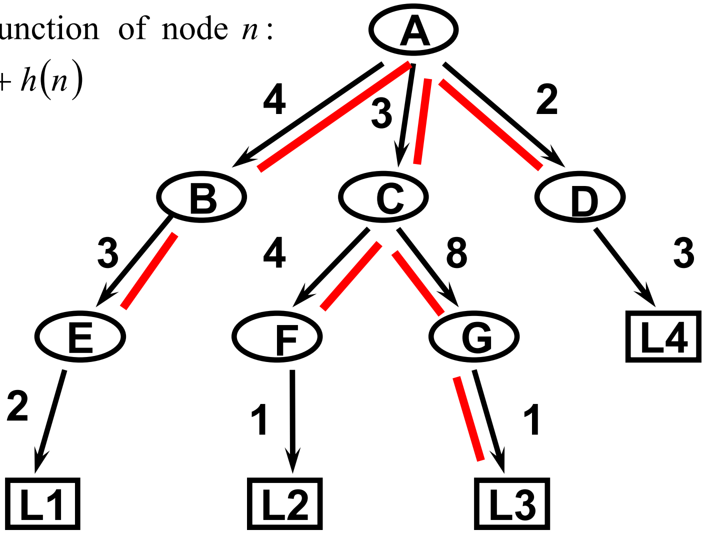
Admissibility: $h(n) \geq h^*(n)$



A* Search (3/3)

- A Simple Example:

Evaluation function of node n:
$f(n) = g(n) + h(n)$



Proving the Admissibility of A* Algorithm:

Suppose when algorithm terminates, *"G "* is a complete path on the top of the stack and *"p"* is a partial path which presents somewhere on the stack.
There exists a complete path *"P"* passing through *"p"*, which is not equal to *"G"* and is optimal.

Proof:
1. *"P"* is a complete which passes through *"p"*, *f(P)<=f(p)*
2.Because *"G"* is on the top of the stack , *f(G)>=f(p)>=f(P)*
3. Therefore, it makes contrariety !!

List or Stack*(sorted)*

Stack Top	*Stack Elements*
A(15)	A(15)
C(15)	C(15), B(13), D(7)
G(14)	G(14), B(13), F(9), D(7)
B(13)	B(13), L3(12), F(9), D(7)
L3(12)	L3(12), E(11), F(9), D(7)

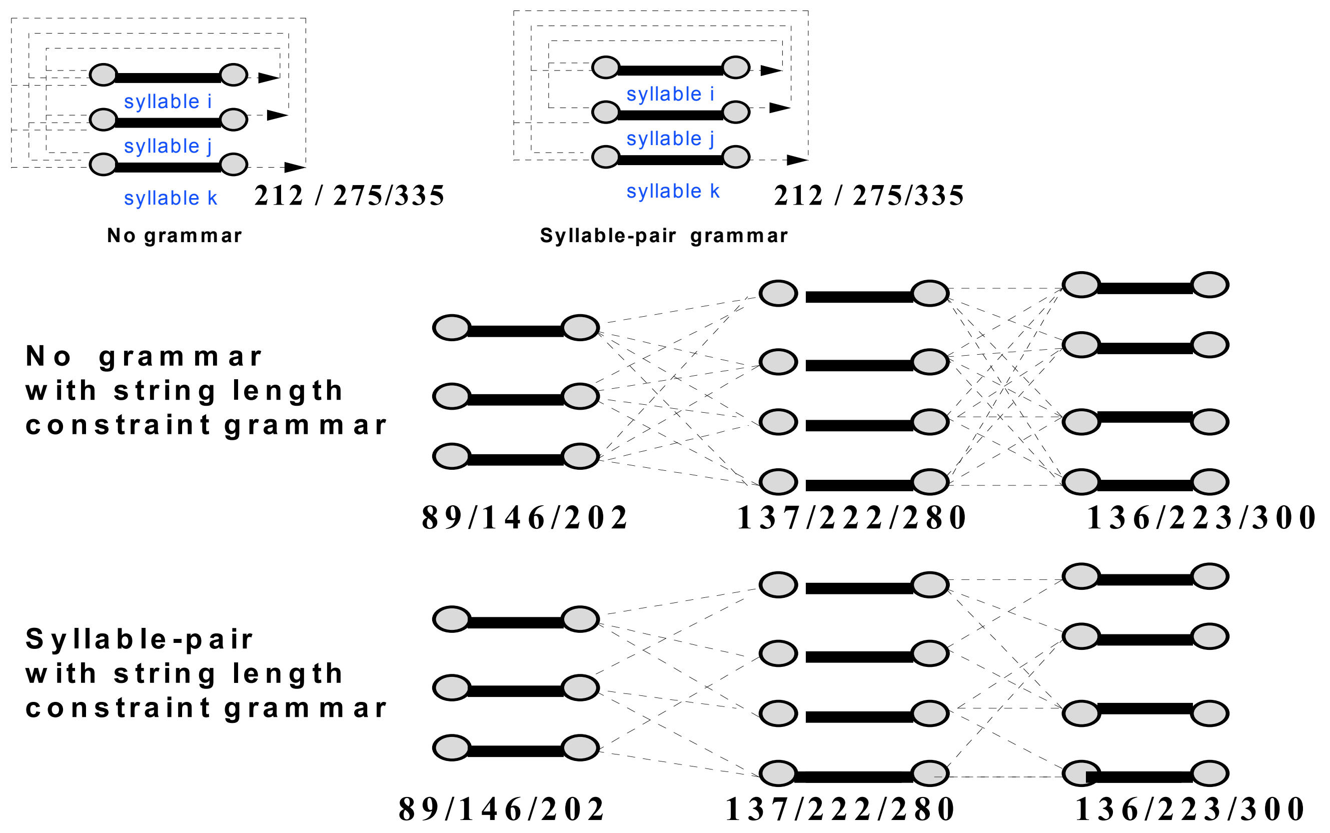
Node	g(n)	h(n)	f(n)
A	0	15	15
B	4	9	13
C	3	12	15
D	2	5	7
E	7	4	11
F	7	2	9
G	11	3	14
L1	9	0	9
L2	8	0	8
L3	12	0	12
L4	5	0	5

Isolated Word Recognition Using A* (1/3)

- Forward Trellis Search (Heuristic Scoring)
  - A forward time-synchronous Viterbi-like trellis search for generating the heuristic score
  - Using a simplified grammar network of different degree grammar type: (Over-generated Grammar)
    - No grammar
    - Syllable-pair grammar
    - No grammar with string length constraint grammar
  - Syllable-pair with string length constraint grammar
- Backward A* Tree Search
  - A backward time-asynchronous viterbi-like A* tree search for finding the “exact” word
  - A backward syllabic tree without over-generating the lexical vocabulary

Isolated Word Recognition Using A* (2/3)

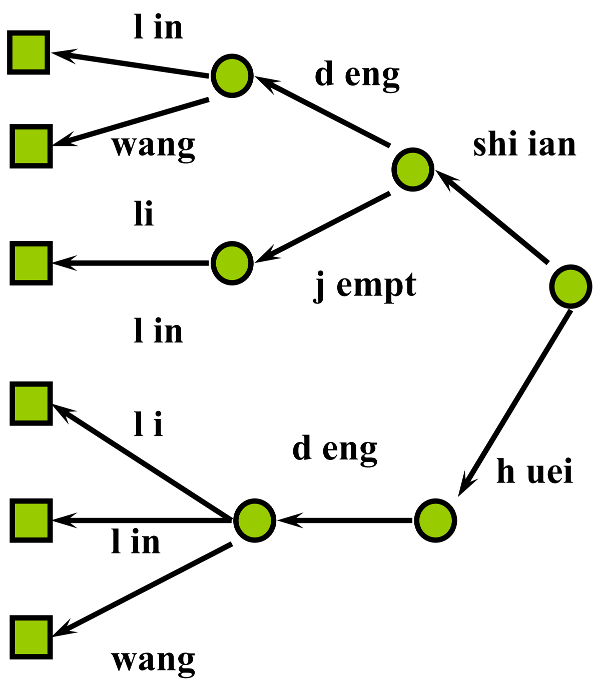
- Search Networks for Heuristic Scoring



Four types of simplified grammar networks used in the tree search.

Isolated Word Recognition Using A* (3/3)

- Backward A* Tree Search

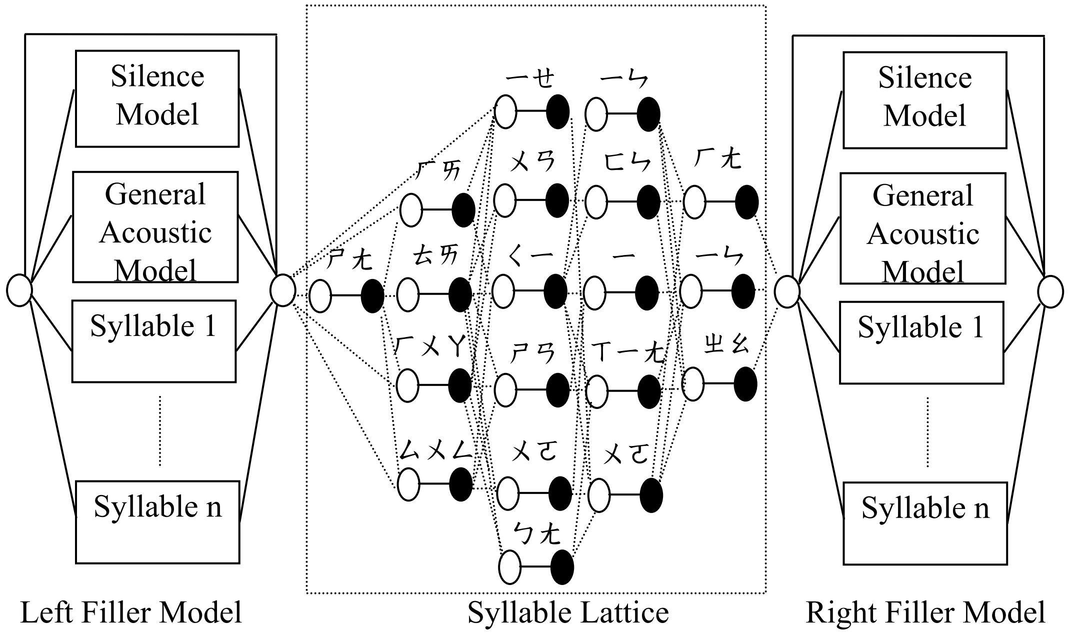


<u>Steps:</u>

- At each iteration of the algorithm-
  - A sorted list (or stack) of partial paths, each with a evaluation function
- The partial path with the highest evaluation function -
  - Expanded
  - For each one -phone(or one syllable or one arc) extensions permitted by the lexicon, the evaluation functions of the extended paths are calculated
  - And the extended partial paths are inserted into the stack at the appropriate position (sorted according to " evaluation function ")
- The algorithm terminates -
  - When a complete path (or word) appears on the top of the stack

Keyword Spotting Using A* (1/2)

- Forward Heuristic Scoring



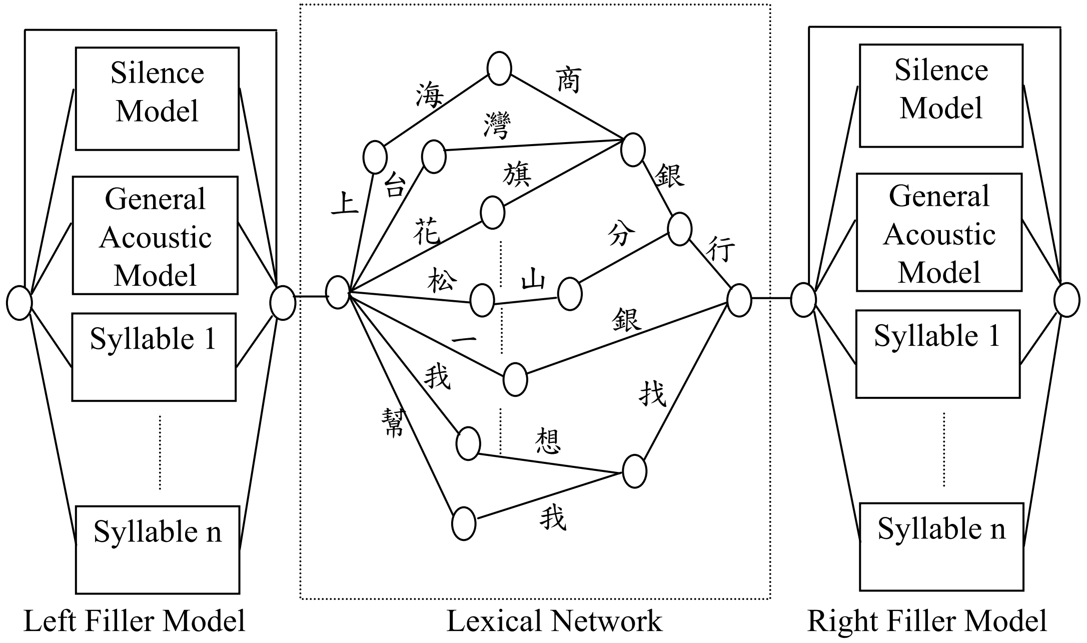
$$f(t) = a \cdot sil(t) + b \cdot \overline{syl}(t) + (1-a-b) \cdot fil(t)$$

$$h^*(n_k, t) = \underset{0 \le t_1 < t}{MAX} \left[f_L(t_1) + h(n_k, t_1 + 1, t) \right]$$

The structure of the compact syllable lattice and the filler models in the first pass

Keyword Spotting Using A* (2/2)

- Backward Time-Asynchronous A* Search



$$E_p(n_k) = \underset{0<t<T}{MAX}\left[d_p(n_k,t) + h^*(n_k,t-1)\right]$$

$$d_p(n_k,t) = \underset{t<t_2<T}{MAX}\left[g_p(n_k,t,t_2-1) + f_R(t_2)\right]$$

The search framework of keyword spotting

Sub-syllable Level Utterance Verification (1/2)

- For each spotted keyword
  - Sub-syllable level verification is then performed
  - The sub-syllable level verification score of a specific sub-syllabic unit (INITIAL/FINAL) S is defined as follows

$$LLR(S) = \log \frac{P(O|\lambda_S)}{P(O|\lambda_{\overline{S}})}$$

filler model

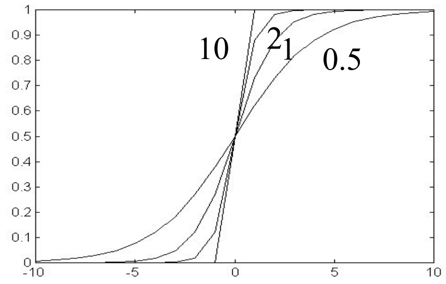
$$P(O|\lambda_{\overline{S}}) = P(O|\lambda_{Filler})$$

$$\text{or } P(O|\lambda_{\overline{S}}) = \frac{1}{N-1}\sum_{S' \neq S} P(O|\lambda_{S'})$$

anti model

  - Then the score is transformed into a range between 0 and 1 by a Sigmoid function

$$\zeta(LLR(S)) = \frac{1}{1+\exp(-\alpha \cdot (LLR(S)-\beta))}$$



Log Likelihood Ratio (LLR)

Sub-syllable Level Utterance Verification (2/2)

- Furthermore, the confidence measure of a keyword hypothesis KW is represented as

$$CM\left(KW\right) = \frac{1}{N_{KW}} \sum_{S \in KW} \zeta\left(LLR\left(S\right)\right)$$

  - N_{KW}: number of sub-syllabic units involved in KW

HW#1 Keyword Spotting

- Implement an A* keyword spotter (detecting whether a keyword embedded in an utterance) with a sub-syllable-level utterance verification mechanism
  - 150 RCD INITIAL/FINAL models
  - 1 silence model
  - 2 filler models (fil_ini & fil_fin) modeling the general syllabic structure of the Chinese language
    - We can also train your sub-syllable-specific anti-models
- The associated evaluation data will be ready shortly
  - You have to try different heuristic scoring approaches
  - You have to plot the ROC curve
- Due: 3/31