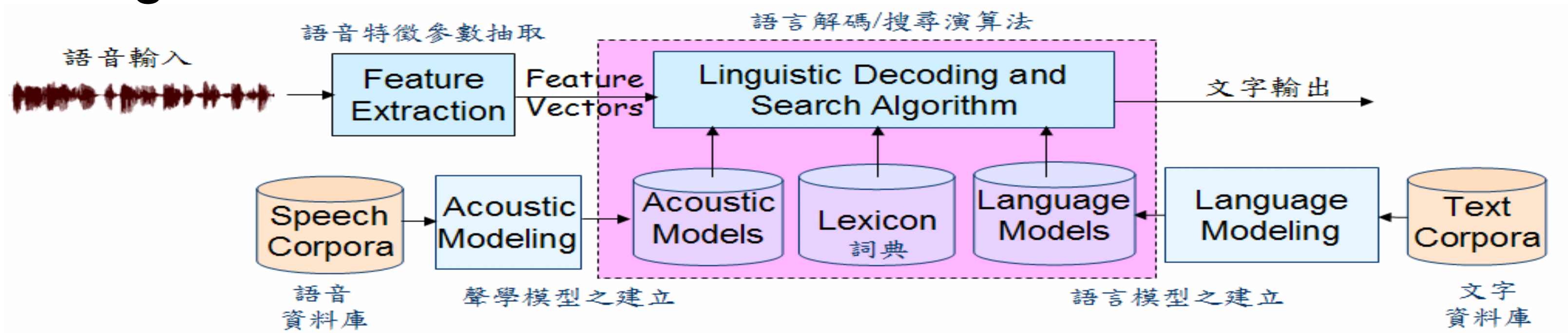
Large Vocabulary Continuous Speech Recognition

Berlin Chen 2003

Why LVCSR Difficult ?

- The software complexity of a search algorithm is considerable

- The effort required to build an efficient decoder is quite large



可能詞句　語音輸入

$$\hat{W} = \arg\max_{W} P(W|X)$$

$$= \arg\max_{W} \frac{P(X|W)P(W)}{P(X)}$$

貝氏定理

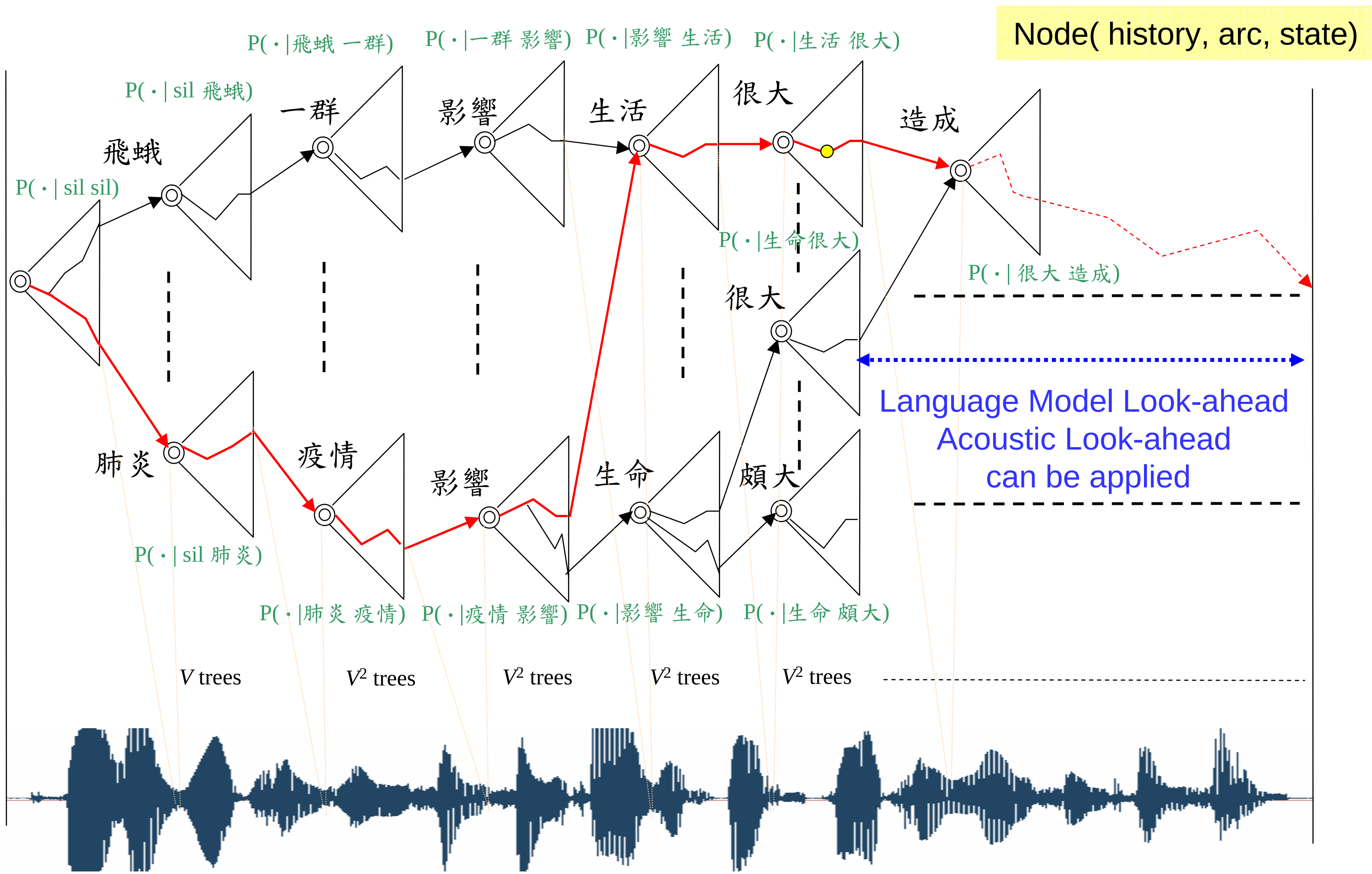
$$= \boxed{\arg\max_{W} P(X|W)P(W)}$$

詞彙網路搜尋

聲學模型機率　語言模型機率

One-Pass Tree-Copy Search

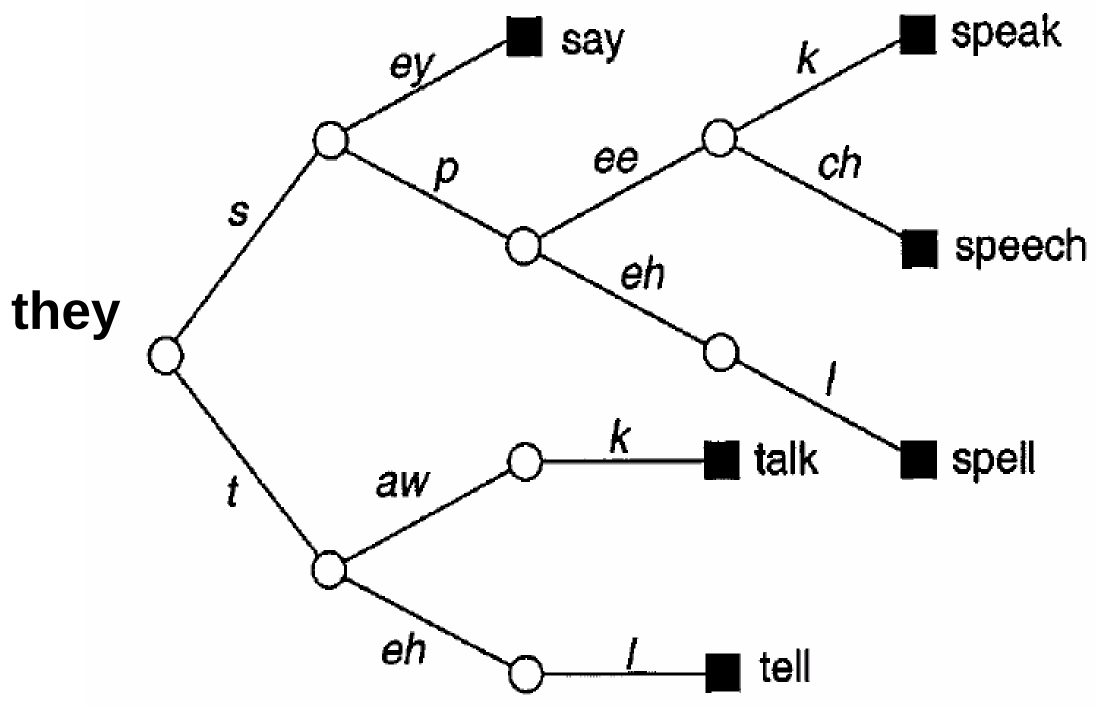
- Trigram language modeling used here



Lexical/Phonetic Tree

- Each arc stands for a phonetic unit
- Each leaf node is shard by words having the same pronunciation
- The application of language modeling is delayed until leaf nodes are reached

$$P(say \mid they)$$
$$P(tell \mid they)$$

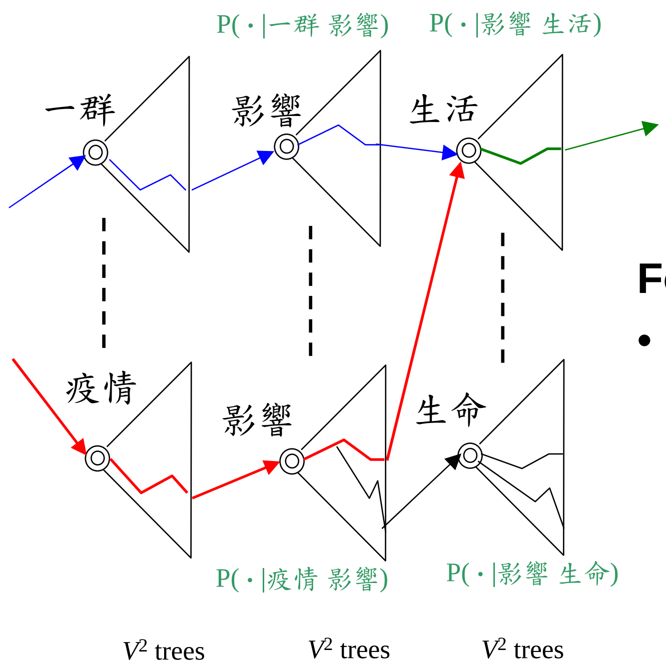


Lexical/Phonetic Tree

- Reasons for using the lexical/phonetic tree
  - States according to phones that are common to different words are shared by different hypotheses
    - A compact representation of the acoustic-phonetic search space
  - The uncertainty about the word identity is much higher at its beginning than its beginning
    - More computations required at the beginning of a word than toward its end

One-Pass Tree-Copy Search

- Word (history)-conditioned Search
  - **A virtual/imaginary tree copy** explored for linguistic context of active search hypotheses
  - Search hypotheses recombined at tree root nodes according to language modeling (or the history)



For n-gram language modeling:

- Retain distinct n-1-gram word histories

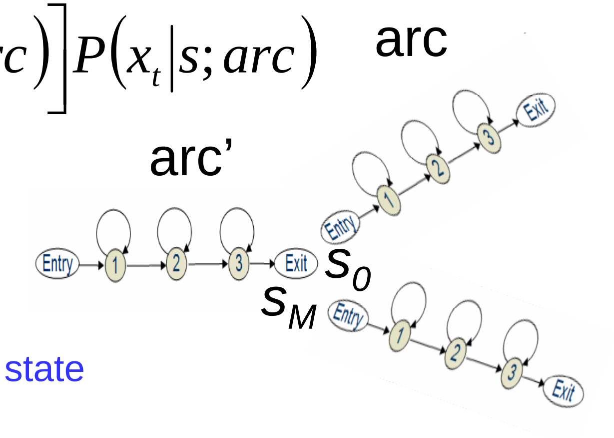
One-Pass Tree-Copy Search

- Integration of **acoustic** and **linguistic** knowledge
    - A network (dynamically) built to describe sentences in terms of words
        - Language models for network transition probabilities
    - A network (statically) built to describe words in terms of phone
        - The pronunciation dictionary (organized as a phonetic tree)
        - Transition penalties are applied
    - A network (statically) built to describe a phone unit in terms of sequences of HMM states
        - Spectral vectors derived from the speech signal are consumed

One-Pass Tree-Copy Search

- Three basic operations performed
  - Acoustic-level recombinations within tree arcs
    - Viterbi search

$$Q_{v_1^{n-1}}(t,s;arc)=\max_{s'}\left[Q_{v_1^{n-1}}(t-1,s';arc)P(s|s';arc)\right]P(x_t|s;arc)$$



Backtracking Information should be manipulated

  - Tree arc extensions

$$Q_{v_1^{n-1}}(t,S_0;arc)=Q_{v_1^{n-1}}(t-1,S_M;arc')$$

The beginning state

The ending state

  - Language-model-level recombination
    - Word end hypotheses sharing the same history were recombined

$$H(v_2^n;t)=\max_{v_1}\left[Q_{v_1^{n-1}}(t,S_{v_n};arc_E)\cdot P(v_n|v_1^{n-1})^{\alpha}\right]$$

$$Q_{v_2^n}(t,S_0;arc_B)=H(v_2^n;t)$$

$$Q_{v_1^{n'-1}}(t,S_0;arc_B)$$

n-1

一群 影響 生活

v_1 v_2 v_3

疫情 影響 生活

n=3

P(生活|一群 影響)

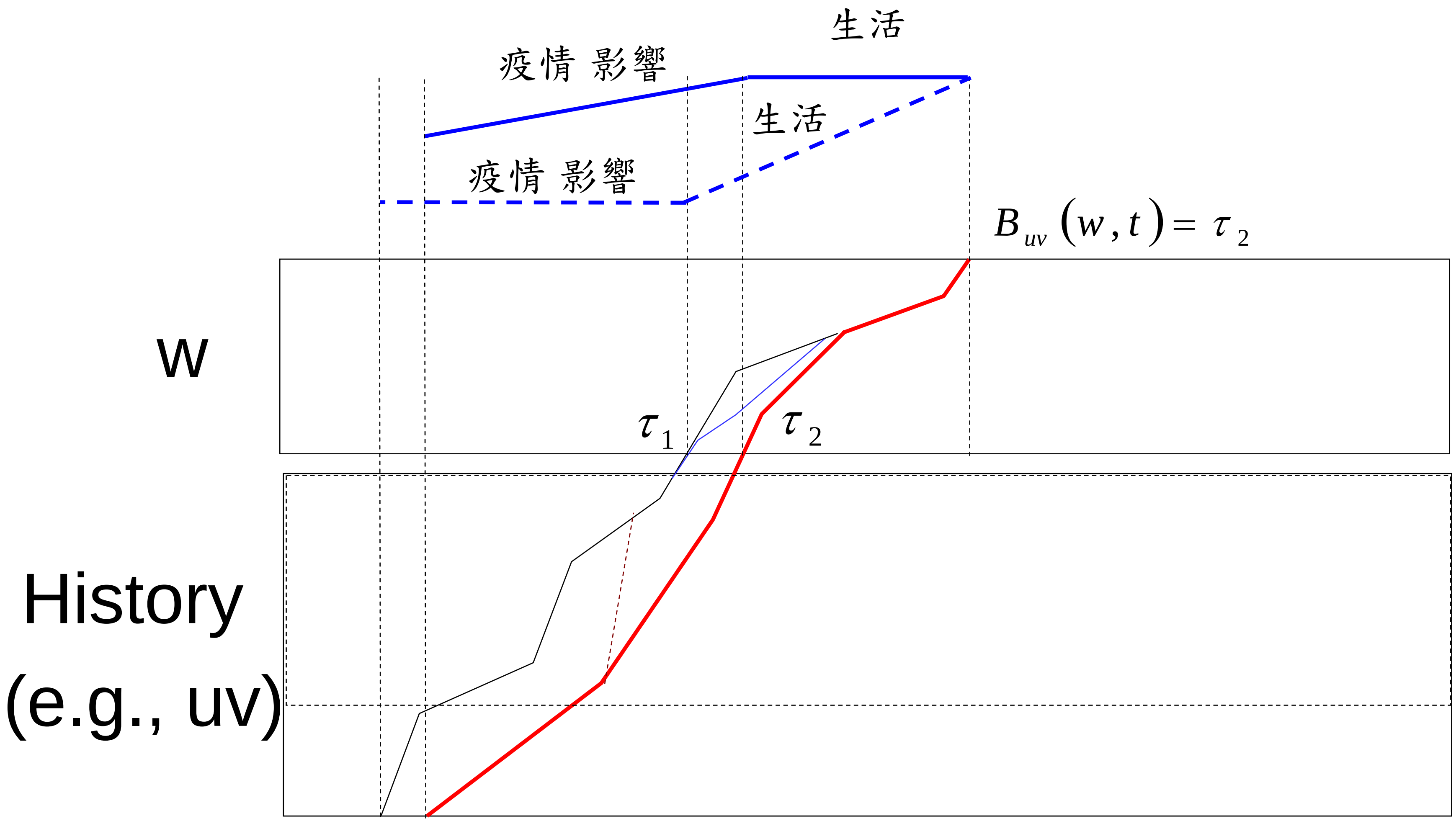
$Q_{一群\ 影響}(-.S_{生活,},-)$

P(生活|疫情 影響)

$Q_{疫情\ 影響}(-.S_{生活,},-)$

One-Pass Tree-Copy Search

- Acoustic-level recombinations



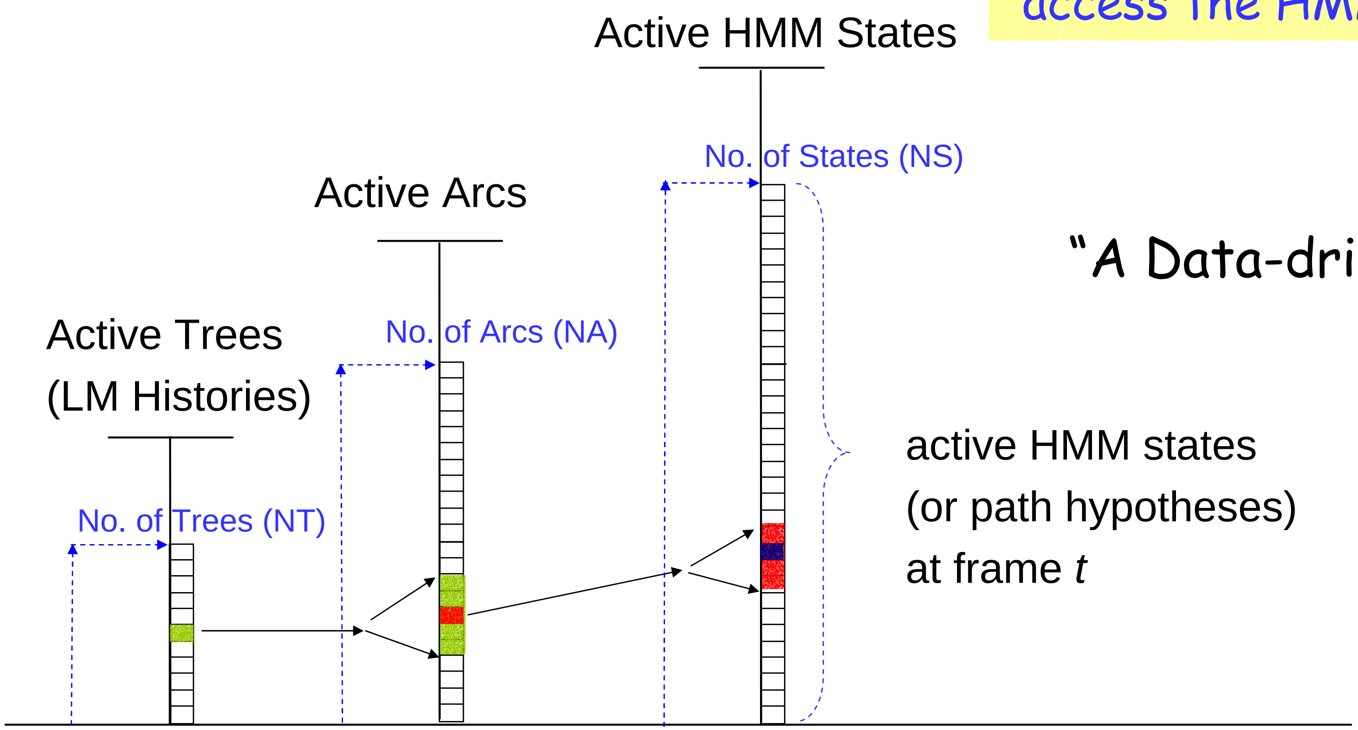
One-Pass Tree-Copy Search

- Different path hypotheses at each time frame are differentiated based on
  - The N-1 word history (for the N-gram LM)
  - The phone unit (or the tree arc)
  - The HMM state

One-Pass Tree-Copy Search

- Organization of active search hypotheses (states)
  - Hierarchical organization

How to directly and efficiently access the HMM states ?

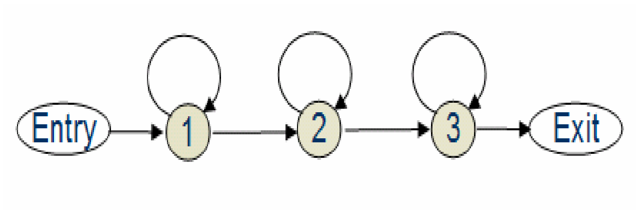


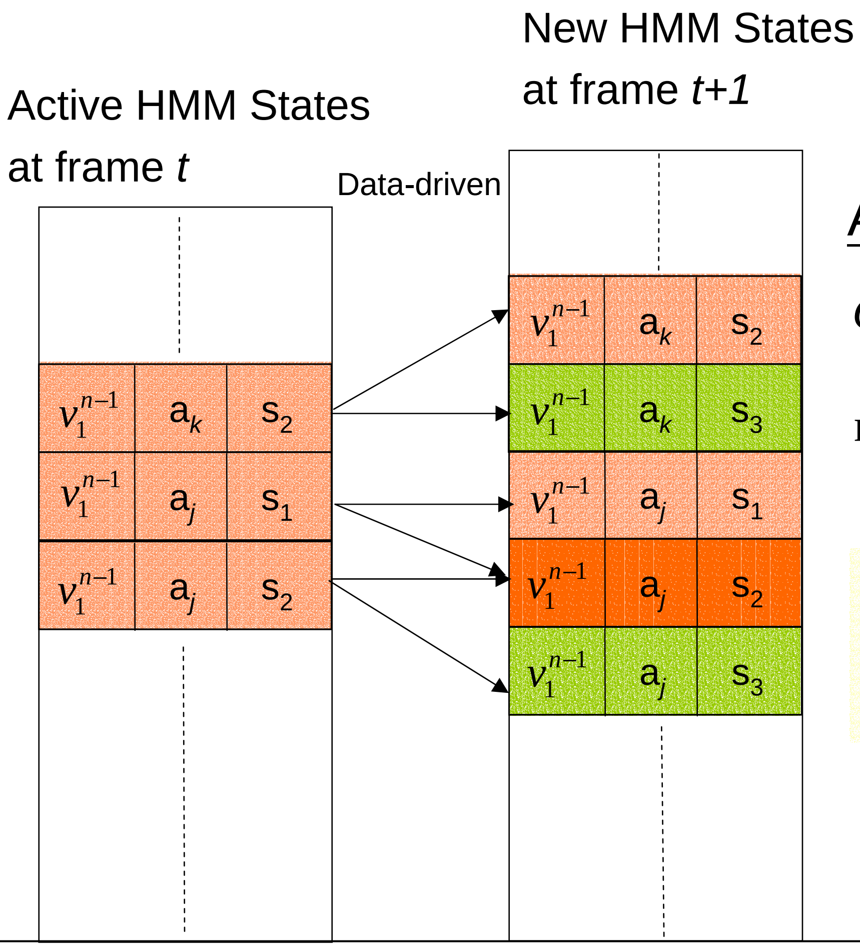
"A Data-driven Approach"

Node(history, arc, state)

One-Pass Tree-Copy Search

- Organization of active search hypotheses (states)
  - Flat organization





<u>Acoustic level recombination</u>

$$Q_{v_1^{n-1}}(t,s;arc) = \max_{s'}\left[Q_{v_1^{n-1}}(t-1,s';arc)P(s|s';arc)\right]P(x_t|s;arc)$$

C++ STL (Standard Template Libraries) is suitable for such an organization

$\log(NS)$ for the access of any HMM state

NS: the number of HMM states

One-Pass Tree-Copy Search

- Viterbi search
  - Belong to a class of breadth-first search
    - *Time-synchronous*
    - Hypotheses terminate at the same point in time
      - Therefore, hypotheses can be compared
  - The search hypotheses will grow *exponentially*
  - Pruning away unlikely (incorrect) paths is needed
    - Viterbi *beam* search
    - Hypotheses with likelihood falling within a fixed radius (or beam) of the most likely hypothesis are retained
    - The beam size determined empirically or adaptively

One-Pass Tree-Copy Search

- Pruning Techniques
  - Standard Beam Pruning (Acoustic-level Pruning)
    - Retain only hypotheses with a score close to the best hypothesis

$$Thr_{AC}(t) = \left[\max_{\left(v_1^{n-1}, s, arc\right)} Q_{v_1^{n-1}}(t, s; arc) \right] \times f_{AC}$$

$$Q_{v_1^{n-1}}(t, s; arc) < Thr_{AC}(t) \;\Rightarrow\; \text{pruned!}$$

  - Language Model Pruning (word-level Pruning)
    - Applied to word-end or tree start-up hypotheses

$$Thr_{LM}(t) = \left[\max_{\left(v_1^{n-1}, S_0, arc_B\right)} Q_{v_1^{n-1}}(t, S_0; arc_B) \right] \times f_{LM}$$

$$Q_{v_1^{n-1}}(t, S_0; arc_B) < Thr_{LM}(t) \;\Rightarrow\; \text{pruned!}$$

  - Histogram Pruning
    - Limit the number of surviving state hypotheses to a maximum number (Need some kind of sorting!)
    - Not Recommended!

One-Pass Tree-Copy Search

- Pruning Techniques (cont.)
  - Stricter pruning applied at word ends
    - The threshold is tightly compared to the acoustic-level one
    - **Reasons**
      - A single path hypothesis is propagated into multiple word ends（同音詞問題）
      - A large number of arcs (models) of the new generated tree copies are about to be activated

Pose severe requirements on the system memory

One-Pass Tree-Copy Search

- Pruning techniques in my system

Acoustic-Level Pruning

```
Acoustic_Penalty=800;
Acoustic_MAX=(float) Min_Delta;
count=0;
for(state_no=0;state_no<NewTreeState;state_no++)
{
  cur_HMM=LEX_STATE[PT2][state_no].TPTR->Model_ID;
  cur_state=LEX_STATE[PT2][state_no].HMM_state;
  if(LEX_STATE[PT2][state_no].Score>Acoustic_MAX)
        Acoustic_MAX=LEX_STATE[PT2][state_no].Score;

}
for(state_no=0;state_no<NewTreeState;state_no++)
{
  cur_HMM=LEX_STATE[PT2][state_no].TPTR->Model_ID;
  cur_state=LEX_STATE[PT2][state_no].HMM_state;
  if(LEX_STATE[PT2][state_no].Score>(Acoustic_MAX-Acoustic_Penalty))
      count++;
}
//20020522
if(count>100000)      Acoustic_MAX=Acoustic_MAX-40;
else if(count>50000) Acoustic_MAX=Acoustic_MAX-80;
else if(count>10000) Acoustic_MAX=Acoustic_MAX-100;
else if(count>5000)  Acoustic_MAX=Acoustic_MAX-150;
else if(count>2000)  Acoustic_MAX=Acoustic_MAX-200;
else if(count>1000)  Acoustic_MAX=Acoustic_MAX-300;
else if(count>400)   Acoustic_MAX=Acoustic_MAX-400;
else                 Acoustic_MAX=Acoustic_MAX-500;
ATreeState=0;
for(state_no=0;state_no<NewTreeState;state_no++)
{
   if(LEX_STATE[PT2][state_no].Score>Acoustic_MAX)
   {
     LEX_STATE[PT1][ATreeState]=LEX_STATE[PT2][state_no];
     ATreeState++;
   }
}
```

One-Pass Tree-Copy Search

- Pruning techniques in my system

Word-Level Pruning

```
LM_Penalty=200;
LM_MAX=(float) Min_Delta;
for(j=0;j<LOCAL_ACTIVE_WORD_NO;j++)
  if(LOCAL_ACTIVE_TREE[j].Score>LM_MAX)
       LM_MAX=LOCAL_ACTIVE_TREE[j].Score;

count=0;
for(j=0;j<LOCAL_ACTIVE_WORD_NO;j++)
  if(LOCAL_ACTIVE_TREE[j].Score>(LM_MAX-LM_Penalty))
       count++;
 //berfore 20020522
if      (count>200) LM_MAX=LM_MAX-30;
else if(count>100) LM_MAX=LM_MAX-50;
else if(count>50)  LM_MAX=LM_MAX-70;
else               LM_MAX=LM_MAX-80;

count=0;
for(j=0;j<LOCAL_ACTIVE_WORD_NO;j++)
  if(LOCAL_ACTIVE_TREE[j].Score>=LM_MAX)
       count++;

if((ACTIVE_TREE_WORD[Frame_Num]
      =( struct DEF_ACTIVE_TREE_WORD *)malloc((count+1)*sizeof( struct DEF_ACTIVE_TREE_WORD)))==NULL)
  {
    printf("ACTIVE_TREE_WORD allocation error at FRAME %d!\n",Frame_Num);
    exit(1);
  }

ACTIVE_TREE_WORD_NO[Frame_Num]=0;
for(j=0;j<LOCAL_ACTIVE_WORD_NO;j++)
  if(LOCAL_ACTIVE_TREE[j].Score>=LM_MAX)
    {
       ACTIVE_TREE_WORD[Frame_Num][ACTIVE_TREE_WORD_NO[Frame_Num]]=LOCAL_ACTIVE_TREE[j];
       ACTIVE_TREE_WORD_NO[Frame_Num]++;
    }
```

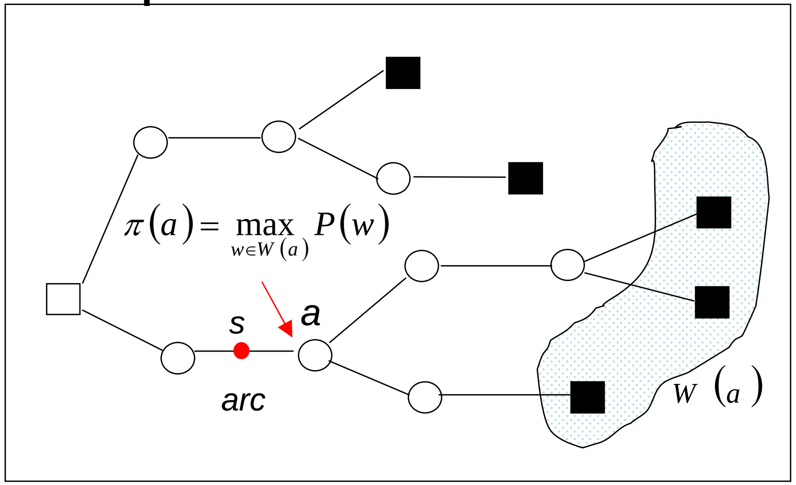
One-Pass Tree-Copy Search

- Language Model Look-ahead
  - Language model probabilities incorporated as early in the search as possible
    - Language model probability incorporated for computing of $Q_{v_1^{n-1}}(t, s; arc)$
- Unigram Look-ahead

$$\pi(a) = \max_{w \in W(a)} P(w)$$

- Bigram Look-ahead

$$\pi_v(a) = \max_{w \in W(a)} P(w|v)$$



- Anticipate the language model probabilities with the state hypothesis

$$\tilde{Q}_{v_1^{n-1}}(t, s; arc) = \pi(a_{s,arc}) Q_{v_1^{n-1}}(t, s; arc)$$

$$\tilde{Q}_{v_1^{n-1}}(t, s; arc) < \overline{Thr}_{AC}(t) \Rightarrow \text{pruned!}$$

$$\overline{Thr}_{AC}(t) = \left[\max_{(v_1^{n-1}, s, arc)} \pi(a_{s,arc}) Q_{v_1^{n-1}}(t, s; arc) \right] \times f_{AC}$$

One-Pass Tree-Copy Search

- Language Model Look-ahead

```
void SpeechClass::Calculate_Word_Tree_Unigram()
{
     if(Root==(struct Tree *) NULL) return;
     Do_Calculate_Word_Tree_Unigram(Root);
}

void SpeechClass::Do_Calculate_Word_Tree_Unigram(struct Tree *ptrNow)
{
  if(ptrNow==(struct Tree *) NULL) return;
  Do_Calculate_Word_Tree_Unigram(ptrNow->Brother);
  Do_Calculate_Word_Tree_Unigram(ptrNow->Child);
  if(ptrNow->Father!=(struct Tree *) NULL)
   if(ptrNow->Unigram > ptrNow->Father->Unigram)
     ptrNow->Father->Unigram=ptrNow->Unigram;
}
```

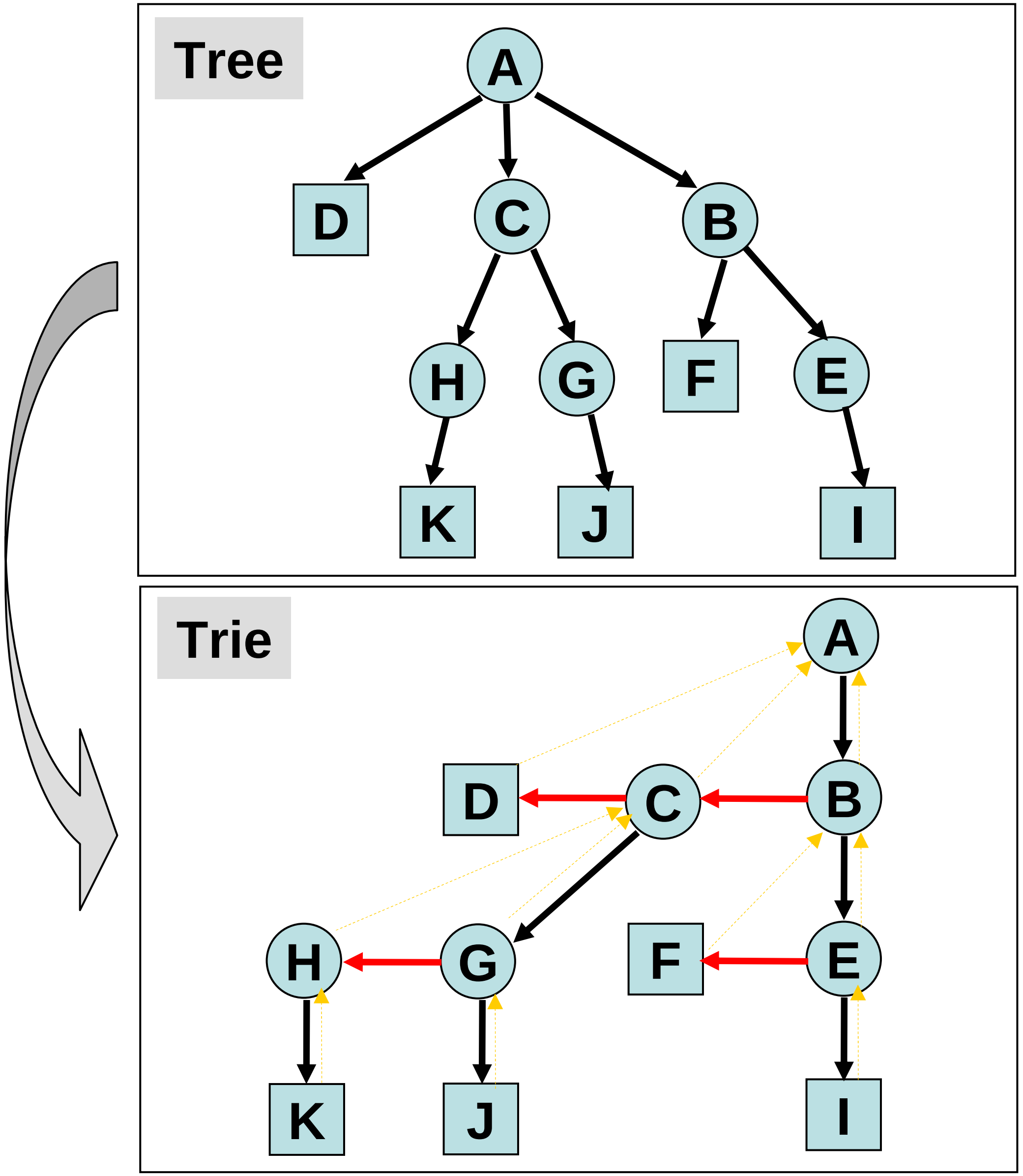
Recursive function for calculating unigram LM look-ahead

One-Pass Tree-Copy Search

- Trie Structure

```
struct DEF_LEXICON_TREE
{
        short    Model_ID;
        short    WD_NO;
        int      *WD_ID;
        int      Leaf;
        double   Unigram;
        struct   Tree *Child;
        struct   Tree *Brother;
        struct   Tree *Father;

};
```



One-Pass Tree-Copy Search

- **Linear Lexicon** with Bigram Language Modeling

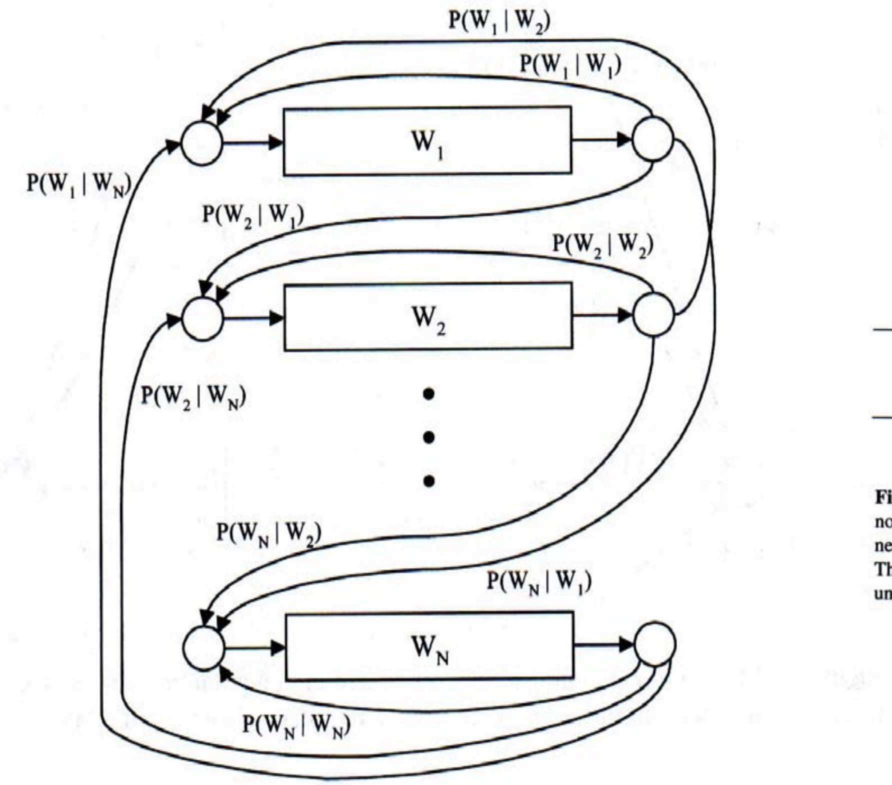


Figure 12.15 A bigram grammar network where the bigram probability $P(w_j \mid w_i)$ is attached as the transition probability from word w_i to w_j [19].

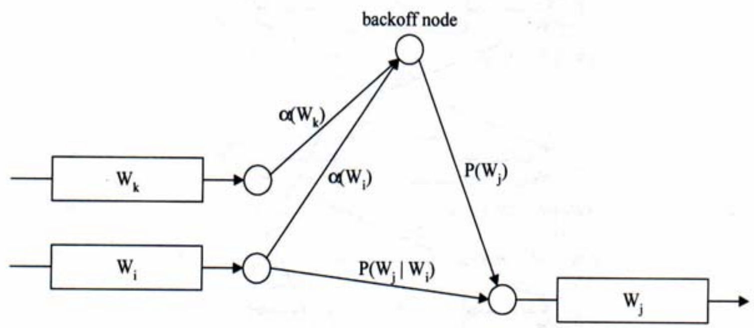
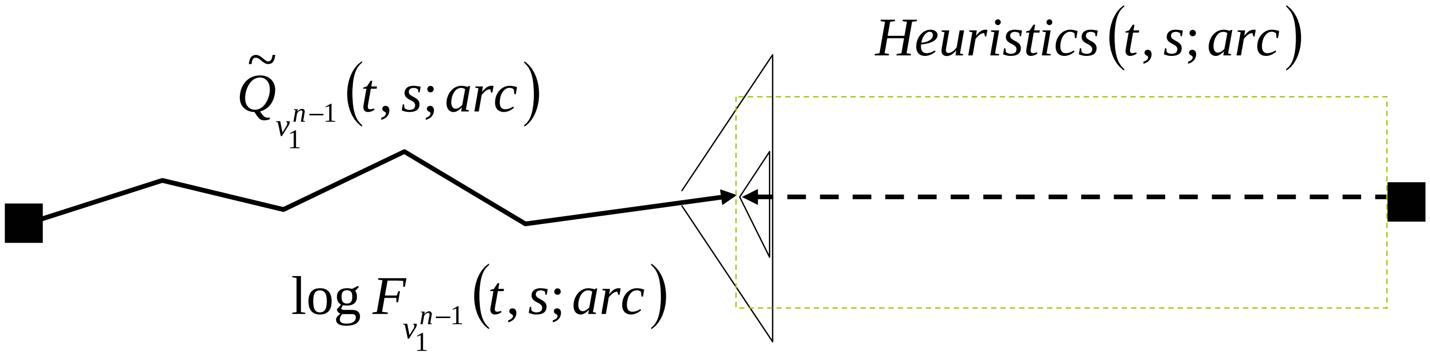


Figure 12.16 Reducing bigram expansion in a search by using the backoff node. In addition to normal bigram expansion arcs for all observed bigrams, the last state of word w_i is first connected to a central backoff node with transition probability equal to backoff weight $\alpha(w_i)$. The backoff node is then connected to the beginning of each word w_j with its corresponding unigram probability $P(w_j)$ [12].

One-Pass Tree-Copy Search

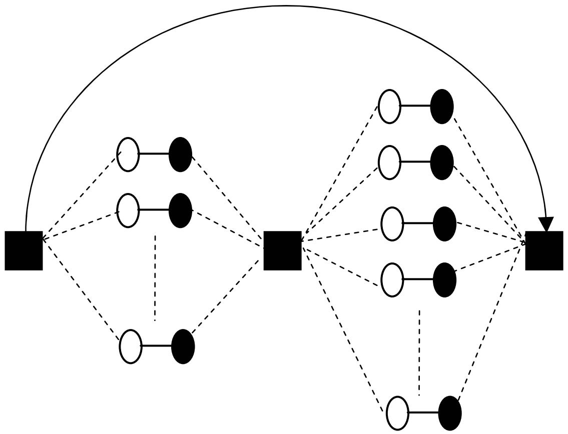
- Acoustic Look-ahead
  - The same idea from A* search
  - The use of acoustic heuristics to speed up the search process
    - Help to make the right decision when pruning
  - How to design the procedure in order to estimate the heuristics ?



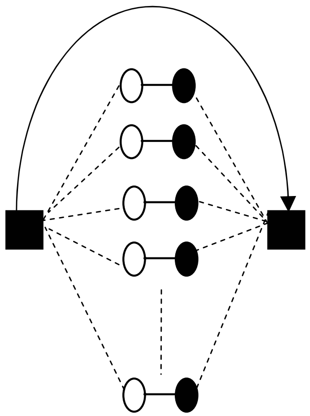
$$\log F_{v_1^{n-1}}(t, s; arc) = \log \tilde{Q}_{v_1^{n-1}}(t, s; arc) + \log Heuristics(t, s; arc)$$

One-Pass Tree-Copy Search

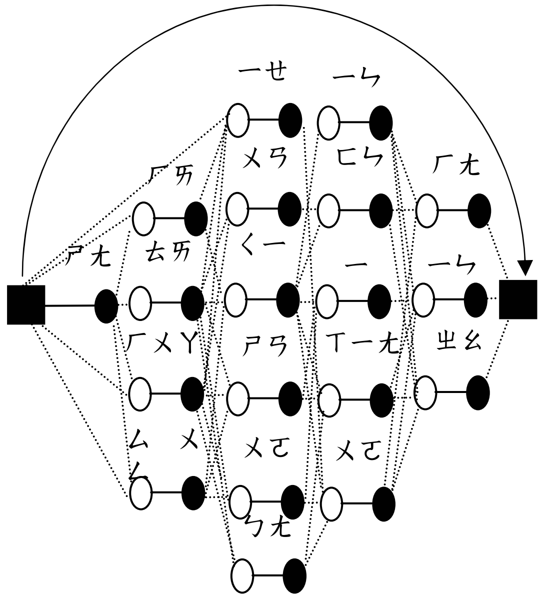
- Acoustic Look-ahead



INITIALs FINALs



Syllables



A lattice based on the lexical tree

Syllable Error Rate		Character Error Rate	
TS	WG	TS	WG
13.1 (1.5%)	12.3 (0.8%)	19.1 (0.0%)	17.2 (0.0%)

	FE	AL	TS	WG	Total
Without Acoustic Look-ahead	0.490	0.000	1.685	0.100	2.275
With Acoustic Look-ahead	0.490	0.003	1.025 (39.2%)	0.085 (15.0%)	1.630 (28.4%)

Word Graph

- If **bigram LM** used in the tree-copy search
  - The beginning time of a word hypothesis w ending at time t

$$\tau(t; v, w) = B_v(t, S_w; arc_E)$$

  - The acoustic score of a word hypothesis w

$$AC_v(w; \tau, t) = Q_v(t, S_{v_n}; arc_E) / H(v, \tau)$$

Not only the word hypothesis with the best predecessor word were recorded

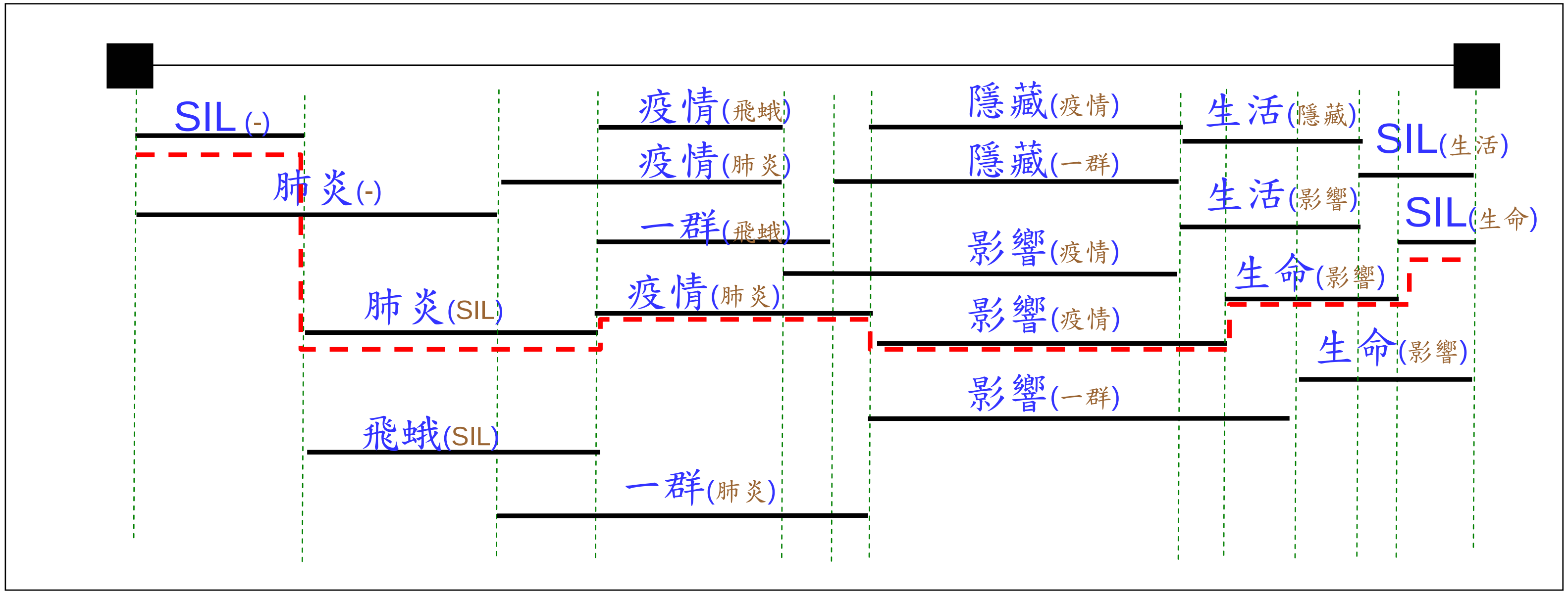
$$AC_{v_0}(w; \tau_0, t)$$

$$AC_{v_1}(w; \tau_1, t)$$

$$AC_{v_2}(w; \tau_2, t)$$

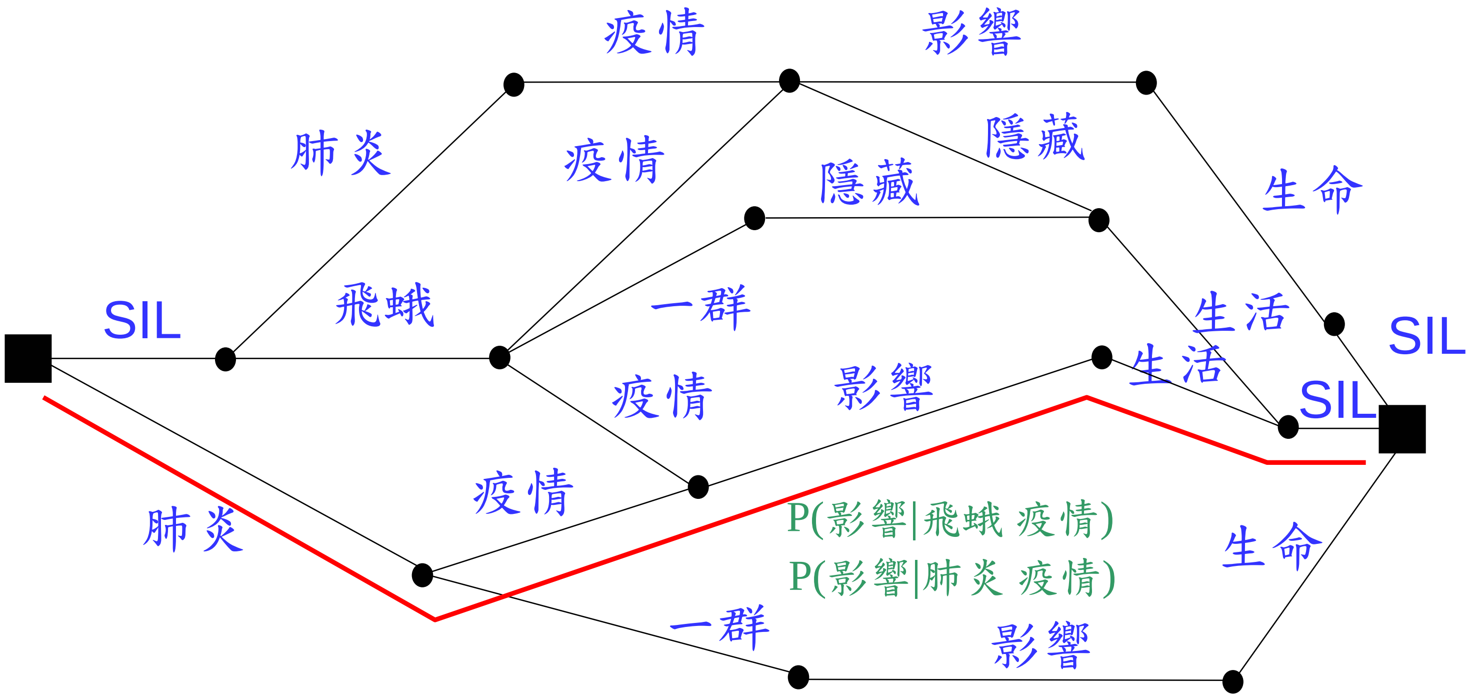
Word Graph

- Bookkeeping at the word level
  - When word hypotheses were recombined into one hypothesis to start up the next tree
    - Not only the word hypothesis with the best predecessor word were recorded
    - But for the hypotheses that have the same LM history, only the best one was kept



Word Graph

- A word graph built with word-pair approximation



- Each edge stands for a word hypothesis
- The node at the right side of an edge denotes the **word end**
  – There is a maximum of incoming word edges for a node (?)
  – There is no maximum of the num. of outgoing edges for a node (?)

One-Pass Tree-Copy Search

- Time-Conditioned Search
  - Acoustic-level recombinations within tree arcs
    - Viterbi search

$$Q_\tau(t,s;arc)=\max_{s'}\left[Q_\tau(t-1,s';arc)P(s|s';arc)\right]P(x_t|s;arc)$$

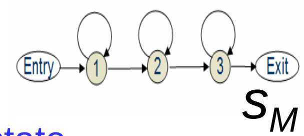
The starting time of the lexical tree

  - Tree arc extensions

$$Q_\tau(t,S_0;arc)=Q_\tau(t-1,S_M;arc')$$

The beginning state

The ending state



  - Language-model-level recombination
    - Word end hypotheses sharing the same history were recombined

$$h(v_n;\tau,t)=Q_\tau(t,S_{v_n};arc_E)/H_{\max}(\tau)$$

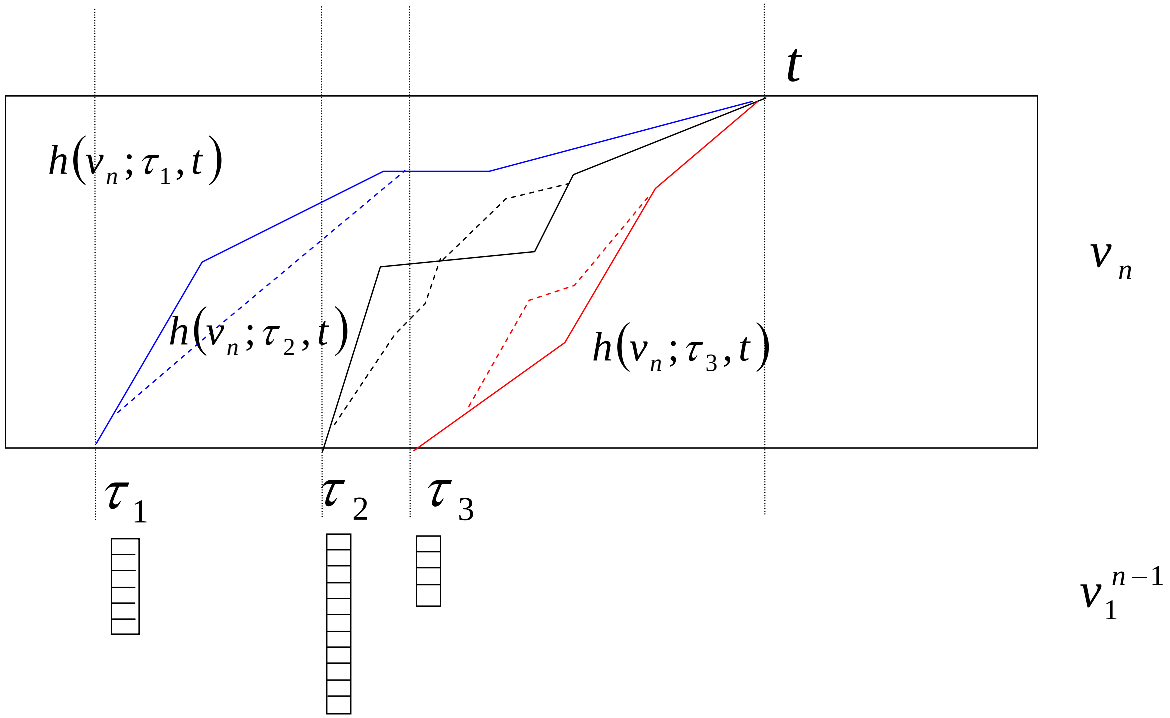
$$H(v_2^n;t)=\max_{(v_1,\tau)}\left[H(v_1^{n-1};\tau)\cdot h(v_n;\tau,t)\cdot P(v_n|v_1^{n-1})^\alpha\right]$$

$$H_{\max}(t)=\max_{v_2^n} H(v_2^n;t)$$

$$Q_t(t,S_0;arc_B)=H_{\max}(t)$$

One-Pass Tree-Copy Search

- Time-Conditioned Search



$$H\left(v_2^n;t\right)=\max_{(v_1,\tau)}\left[H\left(v_1^{n-1};\tau\right)\cdot h\left(v_n;\tau,t\right)\cdot P\left(v_n\middle|v_1^{n-1}\right)^{\alpha}\right]$$

$$=\max_{(v_1,\tau)}\left[H\left(v_1^{n-1};\tau\right)\cdot\frac{Q_{\tau}\left(t,S_{v_n};arc_E\right)}{H_{\max}(\tau)}\cdot P\left(v_n\middle|v_1^{n-1}\right)^{\alpha}\right]$$