Statistical Structured Prediction

3. Making Prediction: Decoding and Inference
   - 3.1. Linear generative models: $\Rightarrow$ PFSAs
     - ➢ Decoding: **Viterbi,**
     - ➢ Inference: **Fordward**
   - 3.2. Linear discriminative models: $\Rightarrow$ CRFs
     - ➢ Decoding: **Viterbi,**
     - ➢ Inference: **Fordward**
   - 3.3. Non-linear generative models: $\Rightarrow$ PCFGs
     - ➢ Decoding: probability of most likely parse tree: **Viterbi**
     - ➢ Inference: probability of a string: **Inside**
   - 3.4. Word Graph based N-best search
     - ➢ Word graphs
     - ➢ N-best search

Probabilistic Finite State Automata

➢ Probability of a path

$$P_A(x,\pi) \;=\; I(s_0)\cdot\left(\prod_{i=1}^{k} P(s_{i-1},x_i,s_i)\right)\cdot F(s_k)$$

➢ Probability of a string $\quad P_A(x) \;=\; \sum_{\pi\in\Pi_A(x)} P_A(x,\pi)$

➢ Probability of the best path $\quad \widehat{P}_A(x) = \max_{\pi\in\Pi_A(x)} P_A(x,\pi)$

➢ Best path $\quad \widehat{\pi}(x) = \arg\max_{\pi\in\Pi_A(x)} P_A(x,\pi)$

PFSA: Viterbi algorithm

➢ **Definition**: The probability of generating the prefix x_1^t through the best path and reaching state q is:

$$\gamma_x(t,q) \stackrel{\text{def}}{=} \max_{\substack{s_0^t\in\Pi_A(x_1^t):\\ s_t=q}} I(s_0)\cdot\left(\prod_{i=1}^{t} P(s_{i-1},x_i,s_i)\right) \qquad \forall q\in Q,\ 0\le t\le |x|$$

➢ **Initialization**: $\quad \gamma_x(0,q) \;=\; I(q) \qquad \forall\ q\in Q$

➢ **Recursion**: $\quad \forall q\in Q,\quad 1\le t\le |x|$

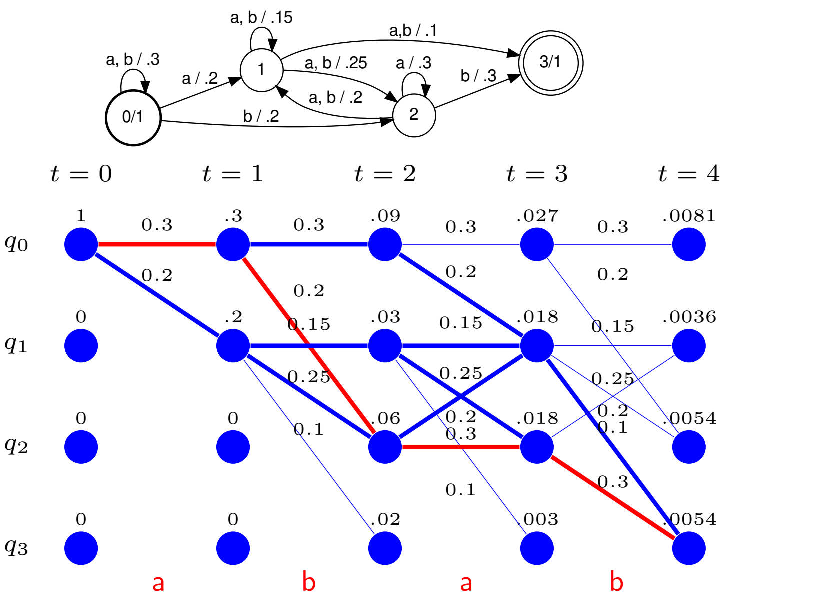
$$\gamma_x(t,q) \;=\; \max_{q'\in Q}\ \{\ \gamma_x(t-1,q')\ \cdot\ P(q',\ x_t,\ q)\ \}$$

➢ **Final result**: The probability of the best state sequence for x given A is:

$$\widehat{P}_A(x) \;=\; \max_{q\in Q}\ \{\ \gamma_x\ (|x|,q)\ \cdot\ F(q)\ \}$$

Backpointers can be used to recover the optimal sequence of states

Algoritmo de Viterbi: ejemplo



PFSA: Forward algorithm

- **Definition**: The probability of generating the prefix x_1^t and reaching state q is:

$$\alpha_x(t,q) \stackrel{\text{def}}{=} \sum_{\substack{s_0^t \in \Pi_A(x_1^t):\\ s_t=q}} I(s_0) \cdot \left(\prod_{i=1}^{t} P(s_{i-1}, x_i, s_i) \right) \qquad \forall q \in Q,\ 0 \le t \le |x|$$

- **Initialization**: $\alpha_x(0,q) = I(q) \qquad \forall\ q \in Q$

- **Recursion**: $\forall q \in Q,\ \ 1 \le t \le |x|$

$$\alpha_x(t,q) = \sum_{q' \in Q} \alpha_x(t-1, q') \cdot P(q',\ x_t,\ q)$$

- **Final result**: The probability of string x is:

$$P_A(x) = \sum_{q \in Q} \alpha_x\ (|x|, q) \cdot F(q)$$

PFSA: Backward algorithm

- **Definition**: The probability of generating the suffix $x_{t+1}^{|x|}$ from the state q is:

$$\beta_x(t,q) \stackrel{\text{def}}{=} \sum_{\substack{s_t^{|x|} \in \Pi_A(x_{t+1}^{|x|}):\\ s_t=q}} \left(\prod_{i=t+1}^{|x|} P(s_{i-1}, x_i, s_i) \right) \cdot F(s_{|x|}) \qquad \forall q \in Q,\ 0 \le t \le |x|$$

- **Initialization**: $\beta_x(|x|,q) = F(q) \qquad \forall\ q \in Q$

- **Recursion**: $\forall q \in Q,\ \ 0 \le t \le |x|-1$

$$\beta_x(t,q) = \sum_{q' \in Q} \beta_x(t+1, q') \cdot P(q,\ x_{t+1},\ q')$$

- **Final result**: The probability of string x is:

$$P_A(x) = \sum_{q \in Q} I(q) \cdot \beta_x\ (0, q)$$

Conditional Random Fields

Given $x = x_1, x_2, \ldots, x_T \in \mathcal{X}^*$ and $y = y_1, y_2, \ldots, y_T \in \mathcal{Y}^*$

Discriminative models: **Conditional Random Fields (CRF)**
[Lafferty, McCallum, Pereira, 2001].

$$p(y|x;\theta) = \frac{\exp\{ \sum_{t=1}^{T} \sum_{k=1}^{K} \theta_k\ f_k(y_{t-1},\ y_t,\ x_t) \}}{Z(x;\theta)} \tag{4}$$

Where

$$Z(x;\theta) = \sum_{y' \in y^*} \exp \left\{ \sum_{t=1}^{T} \sum_{k=1}^{K} \theta_k\ f_k(y'_{t-1},\ y'_t,\ x_t) \right\} \tag{5}$$

CRFs as factor graphs

We can define a transition (factor) as:

$$\Psi_t\,(y_{t-1},\ y_t,\ x_t) \stackrel{\text{def}}{=} \exp\,\{ \sum_{k=1}^{K} \theta_k\ f_k\,(y_{t-1},\ y_t,\ x_t) \}$$

From (4) CRFs can be defined as:

$$p(y|x;\theta) = \frac{1}{Z(x;\theta)} \prod_{t=1}^{T} \Psi_t\,(y_{t-1},\ y_t,\ x_t)$$

Also from (5)

$$Z(x;\theta) = \sum_{y' \in y^*} \prod_{t=1}^{T} \Psi_t\,(y'_{t-1},\ y'_t,\ x_t)$$

CRFs as factor graphs

➢ Decoding: Given θ and x, predict the best output sequence for x.

$$\widehat{y} \;=\; \arg\max_{y}\; p(y|x;\theta) \;=\; \arg\max_{y} \prod_{t=1}^{T} \Psi_t\,(y_{t-1},\; y_t,\; x_t)$$

➢ The probability of an output sequence

$$p(y|x;\;\theta) \;=\; \frac{1}{Z(x;\;\theta)} \prod_{t=1}^{T} \Psi_t\,(y_{t-1},\; y_t,\; x_t)$$

decoding with CRFs: Viterbi algorithm

➢ **Definition**: Score of optimal sequence for $x_1 \dots x_t$ ending at $y_t = s \in \mathcal{Y}$

$$\gamma_t(s) \;\stackrel{\text{def}}{=}\; \max_{y_1^t;\; y_t = s} \prod_{i=1}^{t} \Psi_i\,(y_{i-1},\; y_i,\; x_i)$$

➢ **Initialization**: $\forall\, s \in \mathcal{Y}$

$$\gamma_1(s) \;=\; \Psi_1\,(y_0 = \text{null},\; y_1 = s,\; x_1)$$

➢ **Recursion**: $\forall\, t = 2 \dots T;$ and $\forall\, s \in \mathcal{Y}$

$$\gamma_t(s) \;=\; \max_{s' \in \mathcal{Y}} \left\{ \gamma_{t-1}(s') \,\cdot\, \Psi_t\,(t_{t-1} = s',\; y_t = s,\; x_t) \right\}$$

➢ **Final result**: The optimal score for x is:

$$\max_{s}\; \gamma_T\;(s)$$

Backpointers can be used to recover the optimal sequence $\widehat{y}$

inference with CRFs: Forward algorithm

➢ **Definition**: Score for $x_1 \dots x_t$ ending at $y_t = s \in \mathcal{Y}$

$$\alpha_t(s) \;\stackrel{\text{def}}{=}\; \sum_{y_1^t;\; y_t = s} \prod_{i=1}^{t} \Psi_i\,(y_{i-1},\; y_i,\; x_i)$$

➢ **Initialization**: $\forall\, s \in \mathcal{Y}$

$$\alpha_1(s) \;=\; \Psi_1\,(y_0 = \text{null},\; y_1 = s,\; x_1)$$

➢ **Recursion**: $\forall\, t = 2 \dots T;$ and $\forall\, s \in \mathcal{Y}$

$$\alpha_t(s) \;=\; \sum_{s' \in \mathcal{Y}} \alpha_{t-1}(s') \,\cdot\, \Psi_t\,(y_{t-1} = s',\; y_t = s,\; x_t)$$

➢ **Final result**: The optimal score for x is:

$$\sum_{s} \alpha_T\;(s)$$

CRFs as factor graphs

➢ The probability of an output sequence

$$p(y|x;\theta) \;=\; \frac{1}{Z(x;\theta)} \prod_{t=1}^{T} \Psi_t\,(y_{t-1},\; y_t,\; x_t)$$

Where we can compute $Z(x;\;\theta)$ efficiently using the forward algorithm

$$Z(x;\;\theta) \;=\; \sum_{y'^T_1} \prod_{t=1}^{T} \Psi_t(y'_{t-1},\; y'_t,\; x_t) \;=\; \sum_{s} \alpha_T(s)$$

Probabilistic Context-Free Grammars

Generative models: **Probabilistic Context-Free Grammars**

$$P_\theta(x, t_x) \;=\; \prod_{i=1}^{m} p(r_i) \qquad \text{where} \quad t_x : r_1, \ldots, r_m$$

- Probability of an observation

$$P_\theta(x) \;=\; \sum_{t_x \in \mathcal{T}_x} P_\theta(x, t_x)$$

- Probability of the best parse tree

$$\widehat{P}_\theta(x) \;=\; \max_{t_x \in \mathcal{T}_x} P_\theta(x, t_x)$$

- Finding the best parse

$$\widehat{t}_x \;=\; \arg\max_{t_x \in \mathcal{T}_x} P_\theta(x, t_x)$$

PCFG: Viterbi algorithm

- **Definition**: Given $x = x_1 \ldots x_T \in \Sigma^*$ and $A \in N$

$$\widehat{e}(A, i, i+l) \;\stackrel{\text{def}}{=}\; \widehat{P}_\theta(A \stackrel{*}{\Rightarrow} x_{i+1} \ldots x_{i+l})$$

- **Initialization**: $\forall\, A \in N; \quad \forall\, i : 0 \ldots T-1;$

$$\widehat{e}(A, i, i+1) \;=\; p(A \to b) \cdot \delta(b, x_{i+1})$$

- **Recursion**: $\forall\, A \in N; \quad \forall\, l : 2 \ldots T; \quad \forall\, i : 0 \ldots T-l;$

$$\widehat{e}(A, i, i+l) \;=\; \max_{B,C \in N} \Big\{ p(A \to BC) \cdot \max_{k=1,\ldots,l-1} \{\; \widehat{e}(B, i, i+k) \cdot \widehat{e}(C, i+k, i+l) \;\} \Big\}$$

- **Final result**: The probability of the best parse tree is:

$$\widehat{P}_\theta(x) \;=\; \widehat{e}(S, 0, T)$$

Backpointers can be used to recover the optimal sequence $\widehat{y}$

PCFG: Inside algorithm

- **Definition**: Given $x = x_1 \ldots x_T \in \Sigma^*$ and $A \in N$

$$e(A, i, i+l) \;\stackrel{\text{def}}{=}\; P_\theta(A \stackrel{*}{\Rightarrow} x_{i+1} \ldots x_{i+l})$$

- **Initialization**: $\forall\, A \in N; \quad \forall\, i : 0 \ldots T-1;$

$$e(A, i, i+1) \;=\; p(A \to b) \cdot \delta(b, x_{i+1})$$

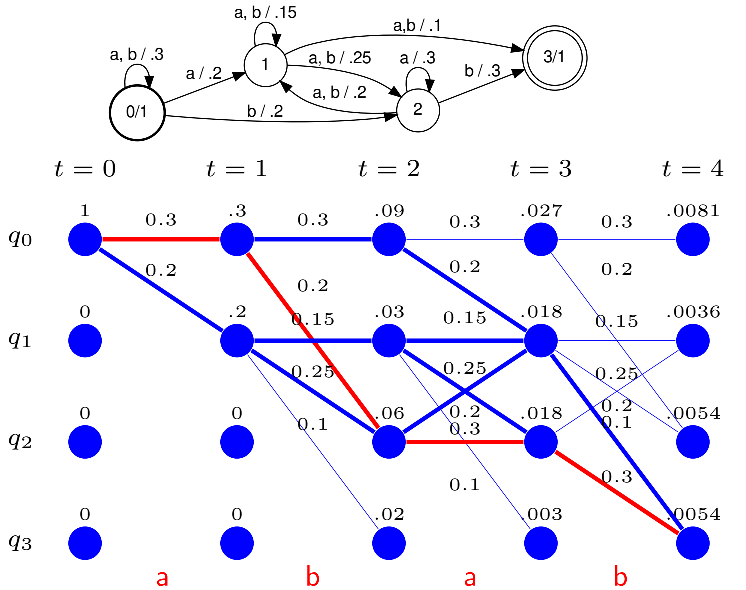
- **Recursion**: $\forall\, A \in N; \quad \forall\, l : 2 \ldots T; \quad \forall\, i : 0 \ldots T-l;$

$$e(A, i, i+l) \;=\; \sum_{B,C \in N} \Big\{ p(A \to BC) \cdot \sum_{k=1,\ldots,l-1} e(B, i, i+k) \cdot e(C, i+k, i+l) \Big\}$$

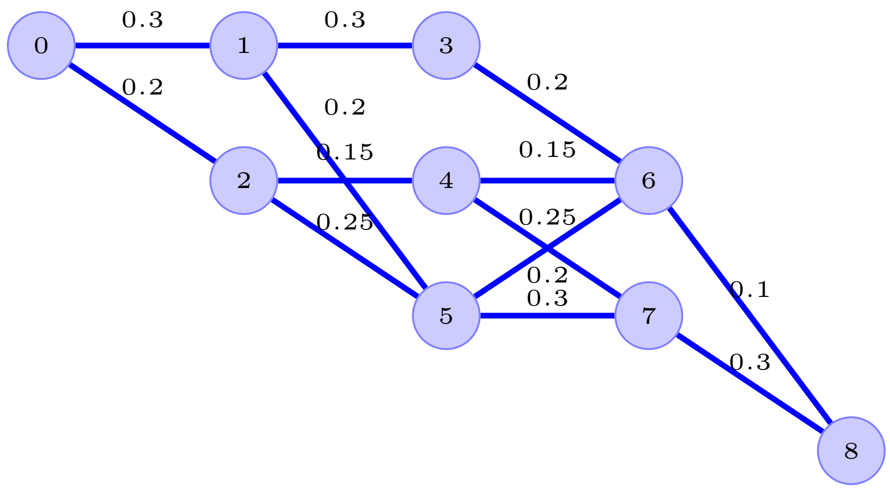
- **Final result**: The sentence probability is:

$$P_\theta(x) \;=\; e(S, 0, T)$$

Word Graph based N-best search



Word Graphs



Word Graphs

- Why do we need word graphs?
  - Trellis-based decoding is very expensive in time and space, even though pruning techniques were used.
  - The Word Graph (WG) allow for a reduction of the search space to the most probable hypotheses.
- Word Graph (WG) definition
  - A WG is a labeled weighted directed acyclic graph.
  - A WG is a data structure that represents a large finite sample of word sequences very efficiently.
  - A WG represents the interpretations with the highest posterior probabilities.
  - The WG is (a pruned version of) the Viterbi search trellis obtained during recognition.

Word Graphs: normalization

- The WG obtained after the search process is not a probabilistic and consistent graph.
- However, for some WG-based operations, it is necessary to normalize the WG.
- There are various graph normalization criteria.

First, we need to define the forward and backward functions:

$$\alpha_x(t,q) = \begin{cases} I(q) & \textbf{if} \quad t=0 \\ \sum_{q'} \alpha_x(t-1,q') \cdot P(q',x_t,q) & \text{otherwise} \end{cases}$$

$$\beta_x(t,q) = \begin{cases} F(q) & \textbf{if} \quad t=|x| \\ \sum_{q'} \beta_x(t+1,q') \cdot P(q,x_{t+1},q') & \text{otherwise} \end{cases}$$

Finally, the following statement can be proven:

$$P_A(x) \;=\; \sum_{q\in Q} \alpha_x\,(|x|,q) \cdot F(q) = \sum_{q\in Q} I(q) \cdot \beta_x\,(0,q)$$

Word Graphs normalization: properties

The normalized weight for an arc (q',q) can be calculated as:

$$\psi(q',q) \;=\; \frac{P(q',q) \cdot \beta(q)}{\beta(q')}$$

- **Property 1:** Well-formed

$$\sum_{q\in Q} \psi(q',q) \;=\; 1 \qquad\qquad \forall q' \in (Q-F);$$

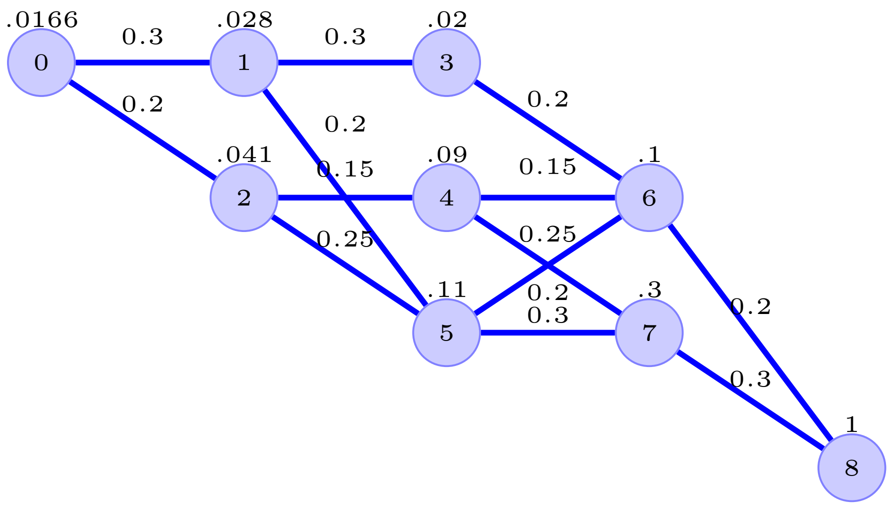
- **Property 2:** Consistency

$$\sum_{\phi\in\Phi} P(\phi) \;=\; 1\,,$$

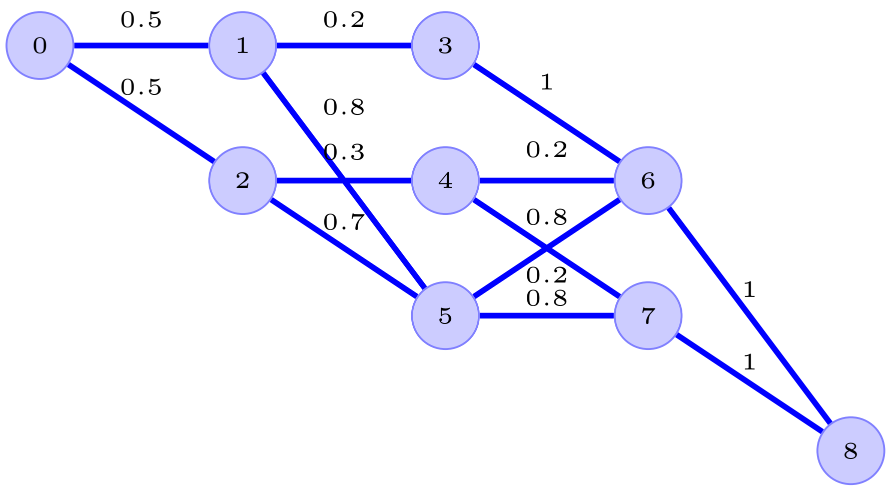
  where Φ are all the paths of the WG.

- **Property 3:** The weight distribution of the sentences in the WG is not changed.

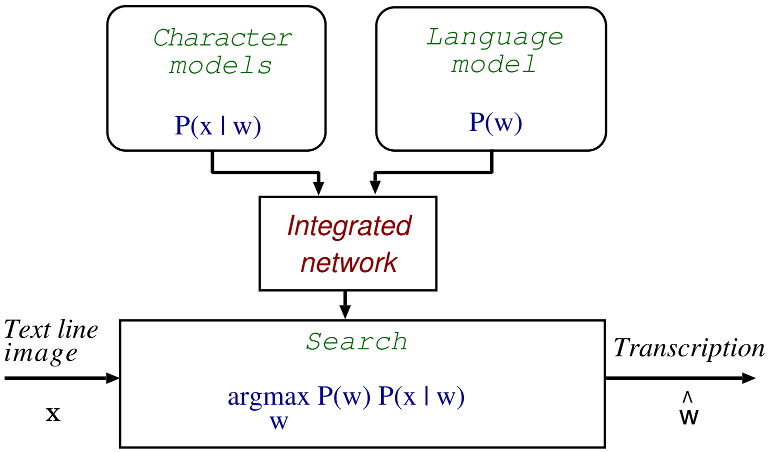
Word Graphs: normalization



Word Graphs: normalization



Word Graphs: HTR integration and decodification



Search engine:

THE VITERBI ALGORITHM (+ beam search + ...)

Word Graphs: HTR integration and decodification

Handwritten Text Recognition (HTR)

Given an input hadwritten text image, find its most likely written transcription.

- Units combine into higher level units: **morphological** $\rightarrow$ **lexical** $\rightarrow$ **syntactics**.
- Relationships between levels can be modeled by weighted graphs.
- Recognition: find the best path in a suitable product graph.
- Morphological, lexical and syntactical are modelled by homogeneous PFS models.
- All these PFS models can be easily integrated into a single global (huge) PFS model.
- This global PFS model accepts sequences of raw feature vectors and outputs strings of recognized words.

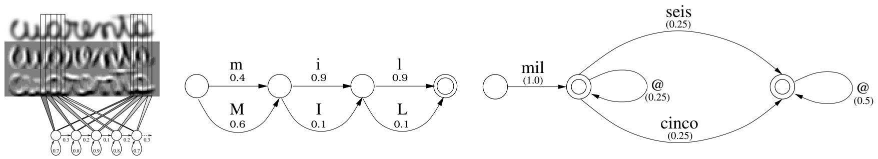
M Connectionist Temporal Classification (CTC) and hybrid NN/HMMs.

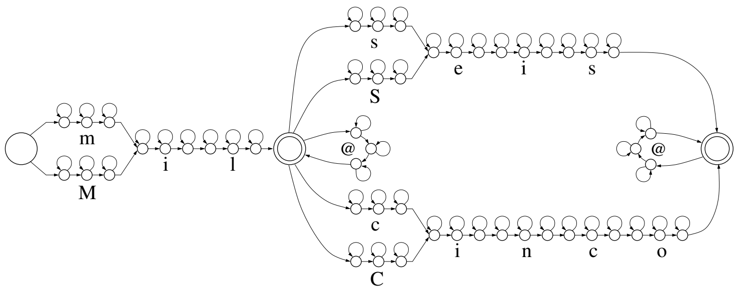
L Pronunciation dictionary transducer mapping caracter transcriptions to sequences.

S Language model weighted automaton.

$$\min\ (\det\ (M \circ L \circ G))$$

Word Graphs: HTR Integration and Decodification





Word Graphs based N-best search

Preliminaries

- The first step consists of calculating the probability of **the best path** from each state $q \in Q$ **to the final state** $f \in F$.

$$\Phi[q] \;=\; \max_{\pi \in P(q,f)} \{P(\pi) \cdot \rho(f)\}$$

- We consider pairs (q, p) of a state $q \in Q$ and a cumulative probability p.
- The algorithm uses a **priority queue** S containing the set of pairs (q, p) to examine next. The queue's ordering is based on Φ and defined by:

$$(q,p) \;>\; (q',p') \quad\Longleftrightarrow\quad (\, p \cdot \Phi[q] \;>\; p' \cdot \Phi[q']\,)$$

- For each state $q \in Q$, $r[q]$ gives the number of times a pair (q, p) with first state q has been extracted from S .
- $\pi[(q, p)]$ defines the predecessor in the path for pair (q, p).
- $E[q]$ provides all the edges that start from state q.

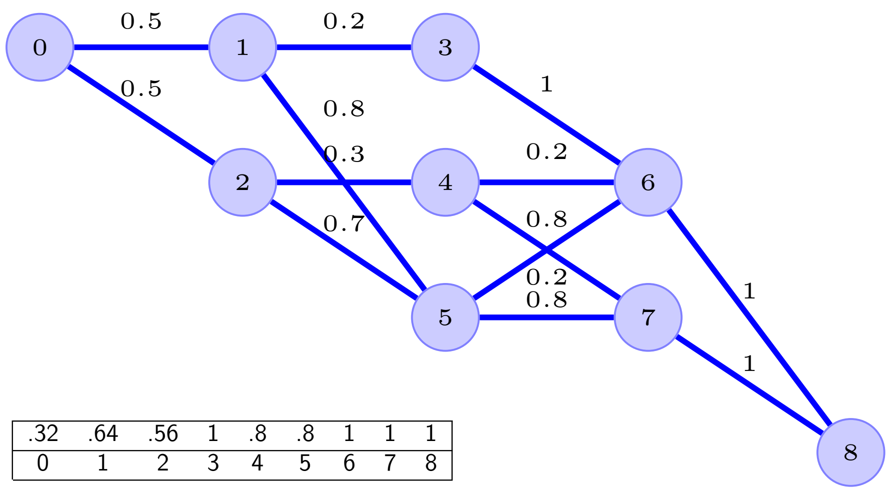
Word Graphs based N-best search

Algorithm 2: N-best algorithm

for $q : 0 \ldots |Q|-1$ **do** $r[q] = 0$;
$\pi[(i,1)] \;=\; \text{NIL}$;
$S \;=\; \{(i,1)\};$;
while $S \neq \emptyset$ **do**
    $(q,p) \;=\; \text{DEQUEUE}(S)$;
    $r[q] \;=\; r[q] + 1$;
    if ($r[q] \;=\; N$ **and** $q \;=\; f \in F$) **then** EXIT;
    if ($r[q] \;\leq\; N$) **then**
        for $e \;\in\; E(q)$ **do**
            $p' \;=\; p \;\cdot\; w[e]$;
            $\pi[(n(e),p')] \;=\; (q,p)$;
            ENQUEUE $(S,(n(e),p'))$;

[M.Mohri and M.Riley: *An efficient algorithm for the N-best-strings problem*, 2002], for more details.

Word Graphs based N-best search



	.32	.64	.56	1	.8	.8	1	1	1
Φ	0	1	2	3	4	5	6	7	8

Word Graphs based N-best search

	.32	.64	.56	1	.8	.8	1	1	1
Φ	0	1	2	3	4	5	6	7	8
r	1	1	1	0	0	2	0	2	2

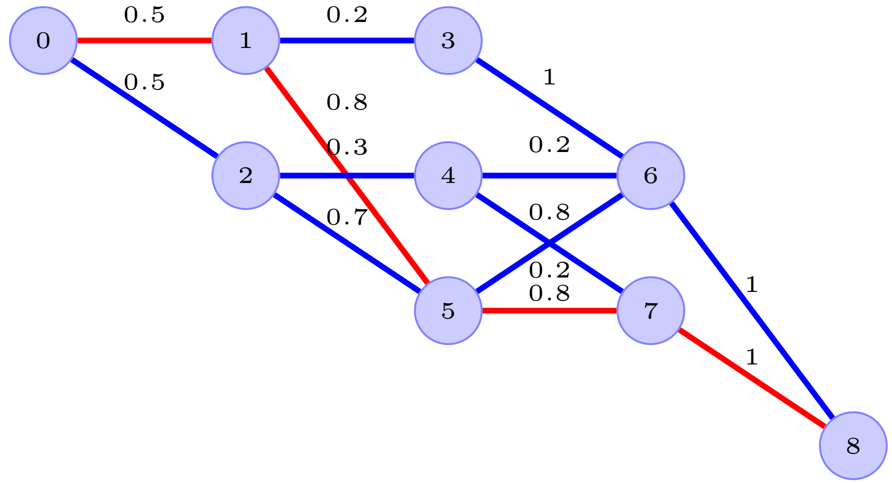
N=2

(0, 1)	1	1 $\cdot$ 0.5	π(1, 0.5) $\leftarrow$ (0, 1)
	2	1 $\cdot$ 0.5	π(2, 0.5) $\leftarrow$ (0, 1)
(1, 0.5)	3	0.5 $\cdot$ 0.2	π(3, 0.1) $\leftarrow$ (1, 0.5)
	5	0.5 $\cdot$ 0.8	π(5, 0.4) $\leftarrow$ (1, 0.5)
(5, 0.4)	6	0.4 $\cdot$ 0.2	π(6, 0.08) $\leftarrow$ (5, 0.4)
	7	0.4 $\cdot$ 0.8	π(7, 0.32) $\leftarrow$ (5, 0.4)
(7, 0.32)	8	0.32 $\cdot$ 1	π(8, 0.32) $\leftarrow$ (7, 0.32)
(8, 0.32)			
(2, 0.5)	4	0.5 $\cdot$ 0.3	π(4, 0.15) $\leftarrow$ (2, 0.5)
	5	0.5 $\cdot$ 0.7	π(5, 0.35) $\leftarrow$ (2, 0.5)
(5, 0.35)	6	0.35 $\cdot$ 0.2	π(6, 0.07) $\leftarrow$ (5, 0.35)
	7	0.35 $\cdot$ 0.8	π(7, 0.28) $\leftarrow$ (5, 0.35)
(7, 0.28)	8	0.28 $\cdot$ 1	π(8, 0.28) $\leftarrow$ (7, 0.28)
(8, 0.28)			

S

(0, 1)	(1, 0.5)	(2, 0.5)
	0.32	0.28
(3, 0.1)	(5, 0.4)	(6, 0.08)
0.1	0.32	0.08
(7, 0.32)	(8, 0.32)	(4, 0.15)
0.32	0.32	0.12
(5, 0.35)	(6, 0.07)	(7, 0.28)
0.28	0.07	0.28
(8, 0.28)		
0.28		

Word Graphs based N-best search



Word Graphs based N-best search