Indexing Transducer For Spoken Term Detection

Lucas Ondel

December 11, 2012

1. Theoretical background
    - Semirings
    - Weighted Finite-State Transducer
2. Indexing algorithm
    - Preprocessing
    - Factor generation
    - Factor merging
    - Factor disambiguation
    - Optimization
    - Global index and search

Definition

A *monoid* is a 3-tuple $(\mathbb{K}, \otimes, \bar{1})$ where $\otimes$ is a closed associative binary operator on the set $\mathbb{K}$, and $\bar{1}$ is the identity element. A monoid is commutative if $\otimes$ is commutative.

Definition

A *semiring* is a 5-tuple $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$, where $(\mathbb{K}, \oplus, \bar{0})$ is a commutative monoid, $(\mathbb{K}, \otimes, \bar{1})$ is a monoid, $\otimes$ distributes over $\oplus$ and $\bar{0}$ is an annihilator for $\otimes$.

Definition

The *log* semiring :

$$\mathcal{L} = (\mathbb{R} \cup \{-\infty, +\infty\}, \oplus_{log}, +, +\infty, 0)$$

Where

$$\forall a, b \in \mathbb{R} \cup \{-\infty, +\infty\},\ a \oplus_{log} b = -\log(e^{-a} + e^{-b})$$

And by conventions : $e^{-\infty} = 0$ and $-\log(0) = +\infty$.

Definition

The *tropical* semiring :

$$\mathcal{T} = (\mathbb{R} \cup \{-\infty, +\infty\}, min, +, +\infty, 0)$$

Where the *min* operation is defined as

$$\forall a, b \in \mathbb{R} \cup \{-\infty, +\infty\}, min(a, b) = a \iff a \leq b \tag{1}$$

Definition

The *product semiring* of two semirings $\mathcal{A} = (\mathbb{A}, \oplus_{\mathbb{A}}, \otimes_{\mathbb{A}}, \bar{0}_{\mathbb{A}}, \bar{1}_{\mathbb{A}})$ and $\mathcal{B} = (\mathbb{B}, \oplus_{\mathbb{B}}, \otimes_{\mathbb{B}}, \bar{0}_{\mathbb{B}}, \bar{1}_{\mathbb{B}})$ is defined as

$$\mathcal{A} \times \mathcal{B} = (\mathbb{A} \times \mathbb{B}, \oplus_{\times}, \otimes_{\times}, \bar{0}_{\mathbb{A}} \times \bar{0}_{\mathbb{B}}, \bar{1}_{\mathbb{A}} \times \bar{1}_{\mathbb{B}})$$

Where $\oplus_{\times}$ and $\otimes_{\times}$ are component-wise operator, e.g.

$$\forall a_1, a_2 \in \mathbb{A},\ \forall b_1, b_2 \in \mathbb{B},\ (a_1, b_1) \otimes_{\times} (a_2, b_2) = (a_1 \otimes_{\mathbb{A}} a_2, b_1 \otimes_{\mathbb{B}} b_2)$$

Definition

The *lexicographic semiring* of two semirings $\mathcal{A} = (\mathbb{A}, \oplus_{\mathbb{A}}, \otimes_{\mathbb{A}}, \bar{0}_{\mathbb{A}}, \bar{1}_{\mathbb{A}})$ and $\mathcal{B} = (\mathbb{B}, \oplus_{\mathbb{B}}, \otimes_{\mathbb{B}}, \bar{0}_{\mathbb{B}}, \bar{1}_{\mathbb{B}})$ is defined as

$$\mathcal{A} * \mathcal{B} = (\mathbb{A} \times \mathbb{B}, \oplus_*, \otimes_*, \bar{0}_{\mathbb{A}} \times \bar{0}_{\mathbb{B}}, \bar{1}_{\mathbb{A}} \times \bar{1}_{\mathbb{B}})$$

Where $\otimes_*$ is a component-wise operator and $\oplus_*$ is a lexicographic priority operator
$\forall a_1, a_2 \in \mathbb{A},\ \forall b_1, b_2 \in \mathbb{B},$

$$(a_1, b_1) \oplus_* (a_2, b_2) = \begin{cases} (a_1, b_1 \oplus_{\mathbb{B}} b_2) & a_1 = a_2 \\ (a_1, b_1) & a_1 = a_1 \oplus_{\mathbb{A}} a_2 \neq a_2 \\ (a_2, b_2) & a_1 \neq a_1 \oplus_{\mathbb{A}} a_2 = a_2 \end{cases}$$

Definition

A *weighted finite-state automata* A over a semiring $\mathbb{K}$ is an 7-tuple $A = (\Sigma, \mathrm{Q}, \mathrm{I}, \mathrm{F}, \mathrm{E}, \lambda, \rho)$

Where

- Σ is the finite input alphabet
- Q is the finite set of states
- $\mathrm{I} \subseteq \mathrm{Q}$ is the set of initial states
- $\mathrm{F} \subseteq \mathrm{Q}$ is the set of final states
- $\mathrm{E} \subseteq \mathrm{Q} \times (\Sigma \cup \{\epsilon\}) \times \mathbb{K} \times \mathrm{Q}$ is the finite set of arcs
- $\lambda : \mathrm{I} \rightarrow \mathbb{K}$ is the initial weight function
- $\rho : \mathrm{F} \rightarrow \mathbb{K}$ is the final weight function

Definition

A *weighted finite-state transducer* T over a semiring $\mathbb{K}$ is an 8-tuple $T = (\Sigma, \Delta, \mathrm{Q}, \mathrm{I}, \mathrm{F}, \mathrm{E}, \lambda, \rho)$

Where

- Σ is the finite input alphabet
- Δ is the finite output alphabet
- Q is the finite set of states
- $\mathrm{I} \subseteq \mathrm{Q}$ is the set of initial states
- $\mathrm{F} \subseteq \mathrm{Q}$ is the set of final states
- $\mathrm{E} \subseteq \mathrm{Q} \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times \mathbb{K} \times \mathrm{Q}$ is the finite set of arcs
- $\lambda : \mathrm{I} \to \mathbb{K}$ is the initial weight function
- $\rho : \mathrm{F} \to \mathbb{K}$ is the final weight function

Given an arc $e \in E$, we note

- $i[e]$ its input label
- $o[e]$ its output label
- $w[e]$ its weight
- $p[e]$ its previous state
- $n[e]$ its next state

A path $\pi = e_1 e_2 ... e_k$, $\pi \in E^*$ where $n[e_{i-1}] = p[e_i]$ $i = 2, ..., k$ By generalization of the previous notation,

- $n[\pi] = n[e_k]$
- $p[\pi] = p[e_1]$
- $i[\pi] = i[e_1]...i[e_k]$
- $o[\pi] = o[e_1]...o[e_k]$
- $w[\pi] = w[e_1] \otimes ... \otimes w[e_k]$

The weight of a set of path $\prod$ is defined as

$$w\left[\prod\right] = \bigoplus_{\pi \in \prod} w[\pi]$$

and the weight of the set of *successful path* of the string pair $(x, y) \in \Sigma^* \times \Delta^*$ is given by

$$[T](x, y) = \bigoplus_{\pi \in \prod(I,x,y,F)} \lambda(p[\pi]) \otimes w[\pi] \otimes \rho(n[\pi])$$

The *shortest distance* from the initial states I_i to a state q denoted by $\alpha_i[q]$ is given by

$$\alpha_i[q] = \bigoplus_{\pi \in \prod(\mathrm{I_i}, q)} (\lambda_i(p[\pi]) \otimes w[\pi])$$

In a similar way, the *reverse shortest distance* from a state q to the final states F_i denoted by $\beta_i[q]$ is given by

$$\beta_i[q] = \bigoplus_{\pi \in \prod(q, \mathrm{F_i})} w[\pi] \otimes (\rho_i(n[\pi])$$

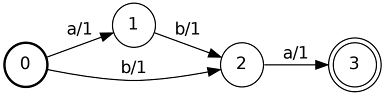
Thus, we can write the weight of a pair-factor as

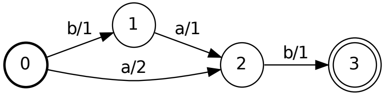
$$w(x, y) = \bigoplus_{i[\pi]=x, o[\pi]=y, \pi \in \prod} \alpha_i[p[\pi]] + w[\pi] + \beta_i[n[\pi]]$$

Some useful algorithms on weighted finite-state transducers :

- ϵ-removal: produces an equivalent transducer with no ϵ-transitions
- determinization: produces an equivalent deterministic transducer (if the input transducer is determinizable)
- minimization: transform a deterministic transducer into an equivalent mininal deterministic transducer
- pushing: allows to distribute the weight along a path without changing the global weight of this path

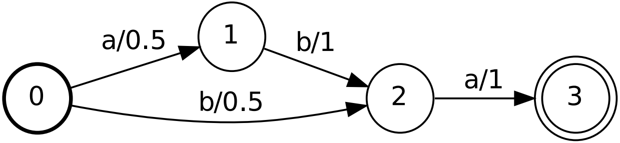
We assume a set of automata A_i over the log-semiring. As an example we suppose a set of two automata A_1, A_2 with timing list $t_1 = t_2 = [0, 1, 2, 3]$. For simplicity, A_1 and A_2 are over the real semiring.





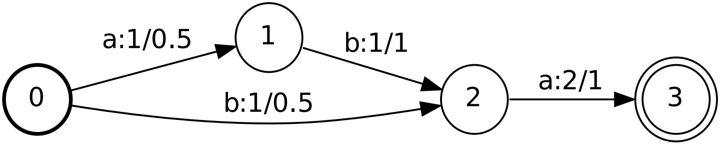
Preprocessing :

- Weight-Pushing



Preprocessing :

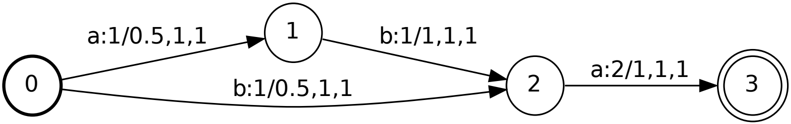
- Clustering



Factor generation :

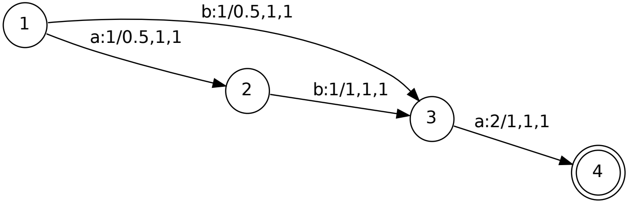
- Map each arc weight

$$w \in \mathcal{L} \rightarrow (w, \bar{1}, \bar{1}) \in \mathcal{L} \times \mathcal{T} \times \mathcal{T}'$$



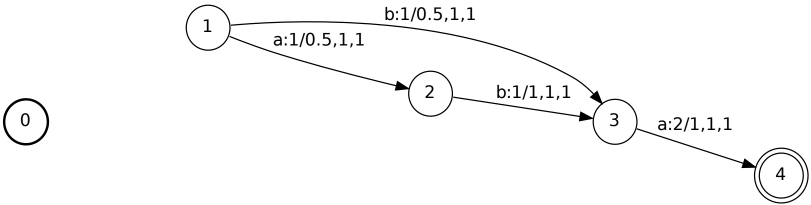
Factor generation :

- Create a unique initial state $q_I \notin \mathrm{Q}_i$
- $\forall q \in \mathrm{Q}_i$ create a new arc $(q_I, \epsilon, \epsilon, (\alpha_i[q], t_i[q], \bar{1}), q)$



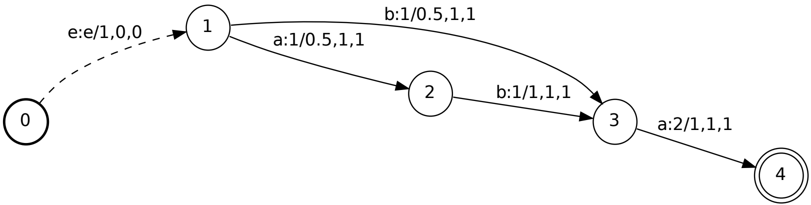
Factor generation :

- Create a unique initial state $q_I \notin \mathrm{Q}_i$
- $\forall q \in \mathrm{Q}_i$ create a new arc $(q_I, \epsilon, \epsilon, (\alpha_i[q], t_i[q], \bar{1}), q)$



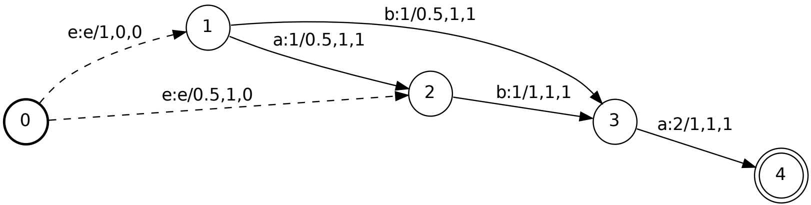
Factor generation :

- Create a unique initial state $q_I \notin \mathrm{Q}_i$
- $\forall q \in \mathrm{Q}_i$ create a new arc $(q_I, \epsilon, \epsilon, (\alpha_i[q], t_i[q], \bar{1}), q)$



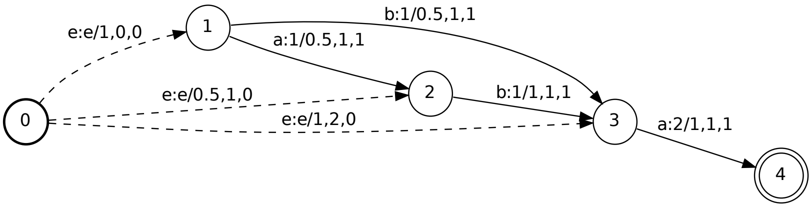
Factor generation :

- Create a unique initial state $q_I \notin \mathrm{Q}_i$
- $\forall q \in \mathrm{Q}_i$ create a new arc $(q_I, \epsilon, \epsilon, (\alpha_i[q], t_i[q], \bar{1}), q)$



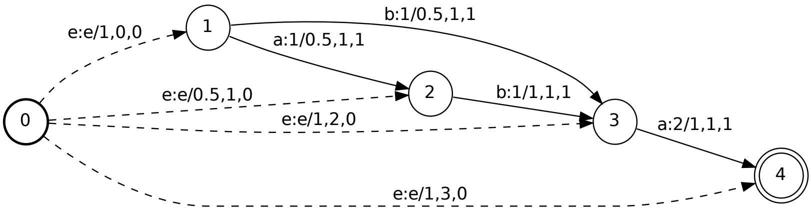
Factor generation :

- Create a unique initial state $q_I \notin \mathrm{Q}_i$
- $\forall q \in \mathrm{Q}_i$ create a new arc $(q_I, \epsilon, \epsilon, (\alpha_i[q], t_i[q], \bar{1}), q)$



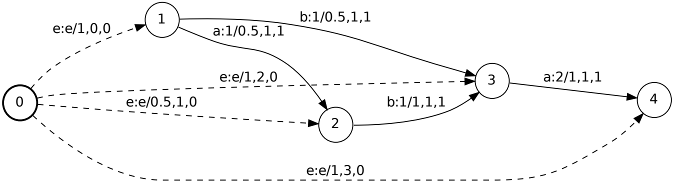
Factor generation :

- Create a unique initial state $q_I \notin \mathrm{Q}_i$
- $\forall q \in \mathrm{Q}_i$ create a new arc $(q_I, \epsilon, \epsilon, (\alpha_i[q], t_i[q], \bar{1}), q)$



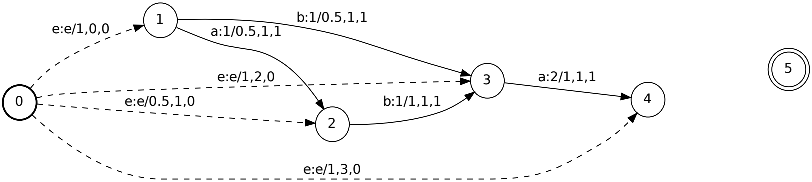
Factor generation :

- Create a unique final state $q_F \notin \mathrm{F}$
- $\forall q \in \mathrm{Q}_i$ create a new arc $(q, \epsilon, i, (\beta_i[q], \overline{1}, t_i[q]), q_F)$



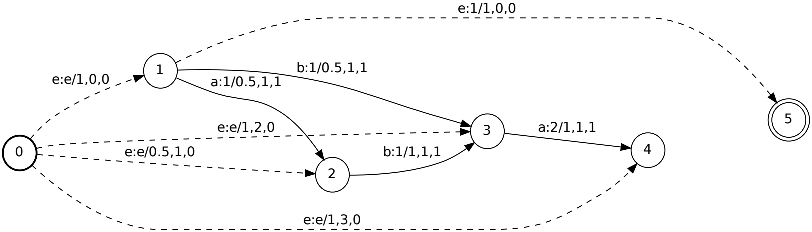
Factor generation :

- Create a unique final state $q_F \notin \mathrm{F}$
- $\forall q \in \mathrm{Q}_i$ create a new arc $(q, \epsilon, i, (\beta_i[q], \bar{1}, t_i[q]), q_F)$



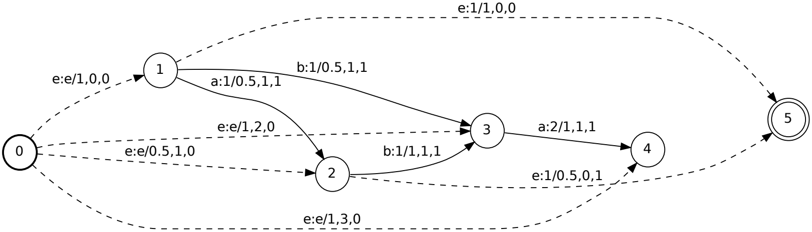
Factor generation :

- Create a unique final state $q_F \notin \mathrm{F}$
- $\forall q \in \mathrm{Q}_i$ create a new arc $(q, \epsilon, i, (\beta_i[q], \bar{1}, t_i[q]), q_F)$



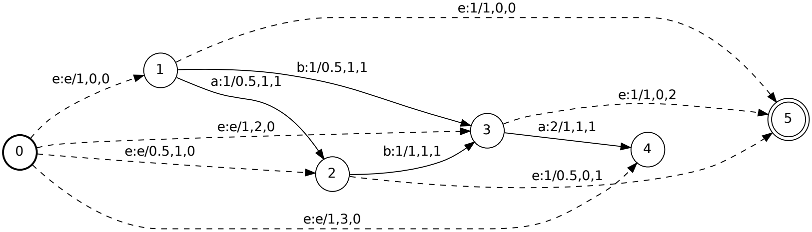
Factor generation :

- Create a unique final state $q_F \notin \mathrm{F}$
- $\forall q \in \mathrm{Q}_i$ create a new arc $(q, \epsilon, i, (\beta_i[q], \bar{1}, t_i[q]), q_F)$



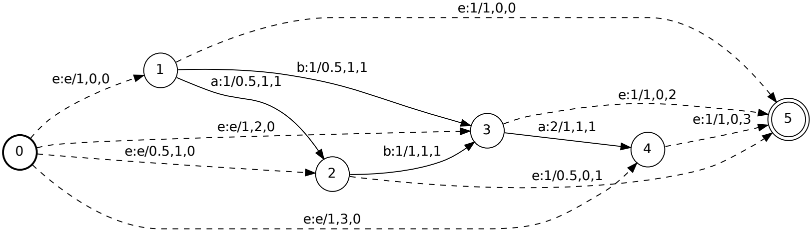
Factor generation :

- Create a unique final state $q_F \notin \mathrm{F}$
- $\forall q \in \mathrm{Q}_i$ create a new arc $(q, \epsilon, i, (\beta_i[q], \bar{1}, t_i[q]), q_F)$



Factor generation :

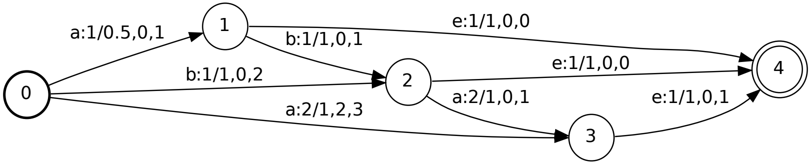
- Create a unique final state $q_F \notin \mathrm{F}$
- $\forall q \in \mathrm{Q}_i$ create a new arc $(q, \epsilon, i, (\beta_i[q], \bar{1}, t_i[q]), q_F)$



Factor merging:

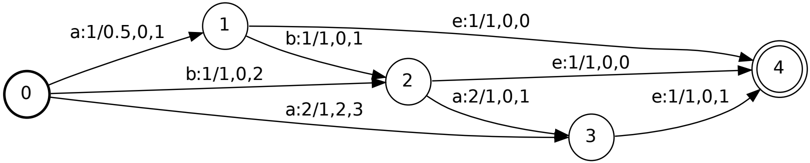
- We merge the path carrying the same factor-pair by viewing the result of the factor generation as an acceptor and applying weighted ϵ-removal, determinization and minimization
- Then, we map each arc weight

$$(w_1, w_2, w_3) \in \mathcal{L} \times \mathcal{T} \times \mathcal{T}' \rightarrow (w_1, w_2, w_3) \in \mathcal{T} * \mathcal{T} * \mathcal{T}$$



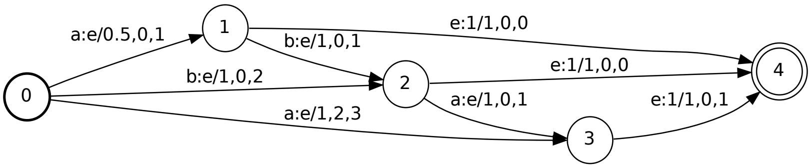
Factor disambiguation:

- Remove cluster identifiers
- Add disambiguation symbol on the final arcs



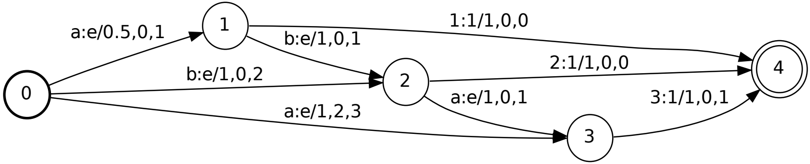
Factor disambiguation:

- Remove cluster identifiers
- Add disambiguation symbol on the final arcs



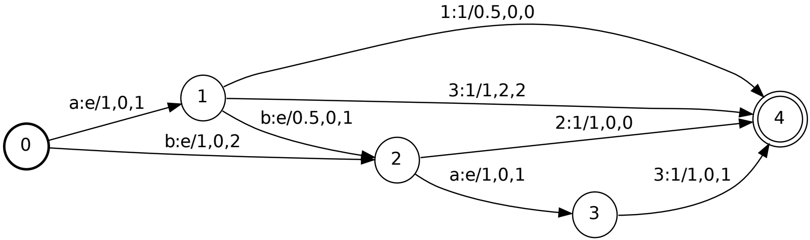
Factor disambiguation:

- Remove cluster identifiers
- Add disambiguation symbol on the final arcs



Optimization:

- By viewing the previous transducer as an acceptor, we optimize it by applying determinization and minimization.
- It yields the final partial index transducer derived from A_1.

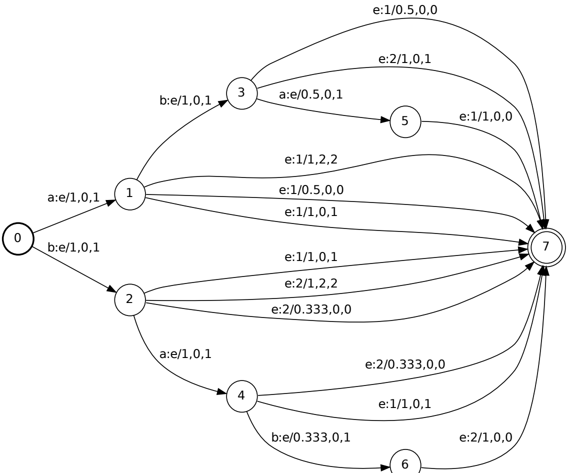


The global index is built by

- taking the union of all partial index transducers T_i:

$$U = \bigcup T_i, \; i = 1, ..., n$$

- applying weighted e-removal, determinization and minimization.
- removing disambiguation symbols


e:1/0.5,0,0
e:2/1,0,1
b:e/1,0,1
a:e/0.5,0,1
e:1/1,0,0
e:1/1,2,2
a:e/1,0,1
e:1/0.5,0,0
e:1/1,0,1
b:e/1,0,1
e:1/1,0,1
e:2/1,2,2
e:2/0.333,0,0
a:e/1,0,1
e:2/0.333,0,0
e:1/1,0,1
b:e/0.333,0,1
e:2/1,0,0
0
1
2
3
4
5
6
7

The search of a factor in this index is done in three steps:

- Convert the query string of the user in a weighted automaton X
- Composing X with the index T
- Removing the ϵ-transition and sorting with the "shortest-path" algorithm.

References:

- *Speech Recognition With Weighted Finite-State Transducers*, Mehryar Mohri, Fernando Pereira, Michael Riley.
- *General Indexation of Weighted-Automata - Application to Spoken Utterance Retrieval*, Cyril Allauzen, Mehryar Mohri, Murat Saraclar.
- *Lattice Indexing for Spoken Term Detection*, Dogan Can, Murat Saraclar.