Prefix Distance Between Regular Languages

Timothy Ng

School of Computing, Queen's University

CIAA 2016, Seoul, Korea

For words $w_1, w_2 \in \Sigma^*$ and regular languages $L_1, L_2 \subseteq \Sigma^*$,

d	Levenshtein	prefix
$d(w_1, w_2)$	Wagner, Fisher (1974)	LCP
$d(w_1, L_2)$	Wagner (1974)	Bruschi, Pighizzini (2006)
$d(L_1, L_2)$	Mohri (2002)	$\times$
$d(L_1)$	Konstantinidis (2005)	$\times$

A distance is a function $d : \Sigma^* \times \Sigma^* \to [0, \infty)$ such that

1. $d(x, y) = 0$ if and only if $x = y$
2. $d(x, y) = d(y, x)$
3. $d(x, y) \leq d(x, w) + d(w, y)$

We can extend the notion of distance to a distance between a word $x \in \Sigma^*$ and a language $L \subseteq \Sigma^*$ by

$$d(x, L) = \min_{y \in L} d(x, y).$$

We can extend the notion of distance to a distance between a word $x \in \Sigma^*$ and a language $L \subseteq \Sigma^*$ by

$$d(x, L) = \min_{y \in L} d(x, y).$$

We can further generalize this to a distance between two languages L_1 and L_2 by

$$d(L_1, L_2) = \min\{d(w_1, w_2) \mid w_1 \in L_1, w_2 \in L_2\}.$$

The prefix distance of x and y counts the number of symbols which do not belong to the longest common prefix of x and y.

$$d_p(x, y) = |x| + |y| - 2 \cdot \max_{z \in \Sigma^*}\{|z| \mid x, y \in z\Sigma^*\}.$$

Harbord $\rightarrow$ Harbourfront

A nondeterministic finite automaton is a 5-tuples

$$A = (Q, \Sigma, \delta, I, F)$$

where

- Q is a finite set of states,
- Σ is the input alphabet,
- $\delta \subseteq Q \times \Sigma \times Q$ is the transition function,
- $I \subseteq Q$ is the set of initial states,
- $F \subseteq Q$ is the set of accepting states.

A weighted finite transducer is a 6-tuple

$$T = (Q, \Sigma, \Delta, I, F, E)$$

where

- Q is a finite set of states,
- Σ is the input alphabet,
- Δ is the output alphabet,
- $I \subseteq Q$ is the set of initial states,
- $F \subseteq Q$ is the set of accepting states,
- $E \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Delta \cup \{\varepsilon\}) \times \mathbb{K} \times Q$ is the set of transitions with weights in the semiring $\mathbb{K}$.

The size of T, denoted $|T|$, is the sum of the number of states and transitions of T, $|Q| + |E|$.

A path or computation of T is a word π over the alphabet of transitions E

$$\pi = (p_1, u_1, v_1, w_1, q_1) \cdots (p_n, u_n, v_n, w_n, q_n)$$

with $q_i = p_{i+1}$ for $1 \leq i \leq n$.

A path or computation of T is a word π over the alphabet of transitions E

$$\pi = (p_1, u_1, v_1, w_1, q_1) \cdots (p_n, u_n, v_n, w_n, q_n)$$

with $q_i = p_{i+1}$ for $1 \leq i \leq n$. Let $\omega : E^* \to \mathbb{K}$ be a weight function for paths $\pi = \pi_1 \cdots \pi_n$ defined by

$$\omega(\pi) = \sum_{i=1}^{n} \pi_i.$$

The label of a path π, denoted $\ell(\pi)$, is the pair of words (x, y) with $x = u_1 \cdots u_n$ and $y = v_1 \cdots v_n$.

The label of a path π, denoted $\ell(\pi)$, is the pair of words (x, y) with $x = u_1 \cdots u_n$ and $y = v_1 \cdots v_n$. We define a weight function $w : \Sigma^* \times \Sigma^* \to \mathbb{K}$ for labels (x, y) defined as the weight of the minimum weight accepted path labeled by (x, y),

$$w(x, y) = \min_{\pi \in E^*} \{\omega(\pi) \mid \ell(\pi) = (x, y)\}.$$

For an alphabet Σ, let $\mathcal{E}_\Sigma$ be the alphabet of edit operations over Σ,

$$\mathcal{E}_\Sigma = \{(a/b) \mid a, b \in \Sigma \cup \{\varepsilon\}, ab \neq \varepsilon\}.$$

For an alphabet Σ, let $\mathcal{E}_\Sigma$ be the alphabet of edit operations over Σ,

$$\mathcal{E}_\Sigma = \{(a/b) \mid a, b \in \Sigma \cup \{\varepsilon\}, ab \neq \varepsilon\}.$$

The cost of an edit string $e = e_1 e_2 \cdots e_n$ is the sum of the cost of each symbol

$$c(e) = \sum_{i=1}^{n} c(e_i).$$

We define the following sets of edit operations and their costs:

- identities $\mathcal{E}_0 = \{(a/a) \mid a \in \Sigma\}$ with cost 0.
- insertions $\mathcal{I} = \{(\varepsilon/a) \mid a \in \Sigma\}$ with cost 1,
- deletions $\mathcal{D} = \{(a/\varepsilon) \mid a \in \Sigma\}$ with cost 1,
- substitutions $\mathcal{S} = \{(a/b) \mid a \neq b, a, b \in \Sigma\}$ with cost 2,

We define the language of edit strings for the prefix distance L_p by

$$L_p = \mathcal{E}_0^*(\mathcal{E} \setminus \mathcal{E}_0)^*.$$

We define the language of edit strings for the prefix distance L_p by

$$L_p = \mathcal{E}_0^*(\mathcal{E} \setminus \mathcal{E}_0)^*.$$

We define the function $d'_p : \Sigma^* \times \Sigma^* \to \mathbb{N}$ on $x, y \in \Sigma^*$ by

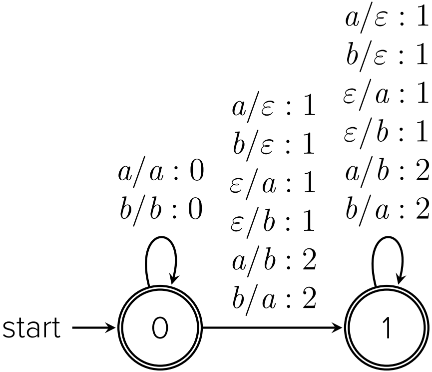
$$d'_p(x, y) = \min_{e \in L_p} \{ c(e) \mid h(e) = (x, y) \}.$$

Theorem

Let $x, y \in \Sigma^*$ be two words. Then $d'_p(x, y) = d_p(x, y)$.

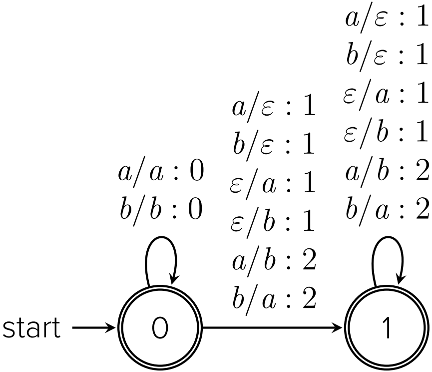
Consider an edit string $e \in L_p$ for two words $x = px'$ and $y = py'$ where p is the longest common prefix of x and y.

- Split e into two parts $e = e_0 e_1$ with $e_0 \in \mathcal{E}_0^*$ and $e_1 \in (\mathcal{E} \setminus \mathcal{E}_0)^*$.
- To minimize $c(e)$, e_0 must be as long as possible and e_1 as short as possible.
- Then e_1 corresponds to an edit string for x' and y'.
- Thus, $c(e) = c(e_1) = |x'| + |y'| = d_p(x, y)$.


a/ε : 1
b/ε : 1
ε/a : 1
ε/b : 1
a/b : 2
b/a : 2
a/a : 0
b/b : 0
a/ε : 1
b/ε : 1
ε/a : 1
ε/b : 1
a/b : 2
b/a : 2
start
0
1

Lemma

The set of accepting paths of the transducer T_p over Σ corresponds to exactly the set of edit strings over Σ belonging to L_p. If π is an accepting path of T_p and e_π is the corresponding edit string, then the weight of π is $c(e_\pi)$.


a/ε : 1
b/ε : 1
ε/a : 1
ε/b : 1
a/b : 2
b/a : 2
a/a : 0
b/b : 0
a/ε : 1
b/ε : 1
ε/a : 1
ε/b : 1
a/b : 2
b/a : 2
start
0
1

Lemma

Let $x, y \in \Sigma^*$. Then the weight $w(x, y)$ of x and y in T_p is exactly $d_p(x, y)$.

Lemma

Let $x, y \in \Sigma^*$. Then the weight $w(x, y)$ of x and y in T_p is exactly $d_p(x, y)$.

- $w(x, y)$ is the weight of the minimal weight accepted path labeled by (x, y)
- The accepting paths of T_p corresponds to edit strings in L_p
- The minimal cost edit string in L_p for x and y has cost $d_p(x, y)$

The composition $T_1 \otimes T_2 = (Q, \Sigma, \Gamma, I, F, E)$ of two weighted transducers $T_1 = (Q_1, \Sigma, \Delta, I_1, F_1, E_1)$ and $T_2 = (Q_2, \Delta, \Gamma, I_2, F_2, E_2)$ is defined by

- $Q = Q_1 \times Q_2$,
- $I = I_1 \times I_2$,
- $F = Q \cap (F_1 \times F_2)$,
- and the transition set E consists of transitions of the form $((q_1, q_1'), a, c, w_1 + w_2, (q_2, q_2'))$ for each transition $(q_1, a, b, w_1, q_2) \in E_1$ and $(q_1', b, c, w_2, q_2') \in E_2$.

The composition $T_1 \otimes T_2$ can be computed in $O(|T_1||T_2|)$ time.

Theorem

Let L_1 and L_2 be regular languages recognized by NFAs A_1 and A_2, respectively. If $x \in L_1$ and $y \in L_2$, then (x, y) is the label of an accepting path of $T = A_1 \otimes T_p \otimes A_2$ and the weight of (x, y) in T is $d_p(x, y)$.

For any accepting path of T,

- the input part must be recognized by A_1,
- the output part must be recognized by A_2,
- the path must correspond to an edit string in L_p.

Thus, there is an accepting path labeled (x, y) with weight $d_p(x, y)$.

Theorem

For given NFAs A_1 and A_2 recognizing the languages L_1 and L_2, respectively, the value $d_p(L_1, L_2)$ can be computed in polynomial time.

Theorem

For given NFAs A_1 and A_2 recognizing the languages L_1 and L_2, respectively, the value $d_p(L_1, L_2)$ can be computed in polynomial time.

- Transducer composition can be done in time $O(|A_1||A_2|)$.
- Shortest path can be computed in time polynomial in the size of $A_1 \otimes T_p \otimes A_2$.

The inner distance of a language L, also called the self distance is the minimal distance between any two distinct words that belong to L,

$$d(L) = \min\{d(w_1, w_2) \mid w_1, w_2 \in L, w_1 \neq w_2\}.$$

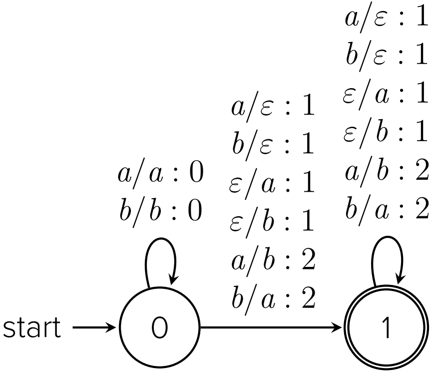
To compute the inner distance, we can use the same approach, but we must exclude all edit strings with cost 0.

To compute the inner distance, we can use the same approach, but we must exclude all edit strings with cost 0.

$$L_p^{(1)} = \mathcal{E}_0^*(\mathcal{E} \setminus \mathcal{E}_0)^+.$$

To compute the inner distance, we can use the same approach, but we must exclude all edit strings with cost 0.

$$L_p^{(1)} = \mathcal{E}_0^*(\mathcal{E} \setminus \mathcal{E}_0)^+.$$

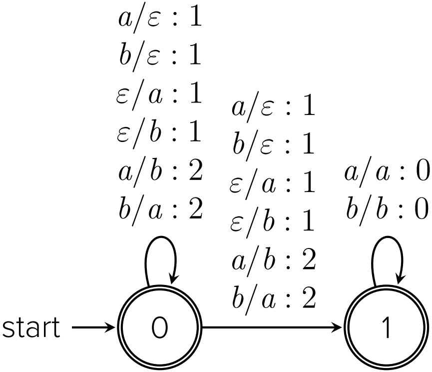


We can apply the same approach to suffix distance.

$$L_s = (\mathcal{E} \setminus \mathcal{E}_0)^* \mathcal{E}_0^*$$

We can apply the same approach to suffix distance.

$$L_s = (\mathcal{E} \setminus \mathcal{E}_0)^* \mathcal{E}_0^*$$

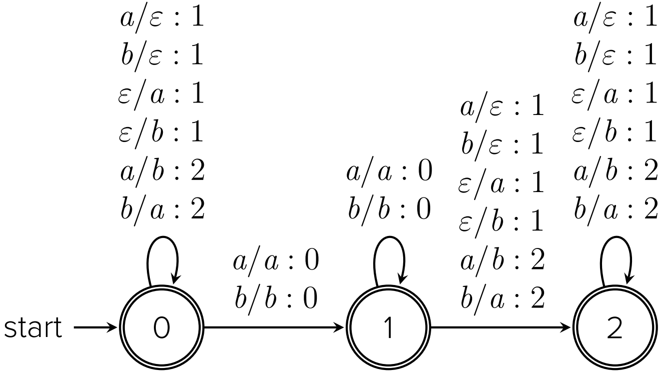


And infix distance.

$$L_f = (\mathcal{E} \setminus \mathcal{E}_0)^* \mathcal{E}_0^* (\mathcal{E} \setminus \mathcal{E}_0)^*$$

And infix distance.

$$L_f = (\mathcal{E} \setminus \mathcal{E}_0)^* \mathcal{E}_0^* (\mathcal{E} \setminus \mathcal{E}_0)^*$$



We have shown

- how to compute the prefix distance between two regular languages
- how to compute the inner prefix distance of a regular language
- how to compute the suffix and infix distances for the above

- What about other distances?
- What about the distance between a regular language and a context-free language?