Relative Prefix Distance Between Languages

Timothy Ng David Rappaport Kai Salomaa

School of Computing, Queen's University, Kingston, Canada

DLT 2017, Liège, Belgium

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.

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.

STARTING

STARLIGHT

We can extend a distance on words to a distance between a word and a language

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

We can extend a distance on words to a distance between a word and a language

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

We can extend this further to a distance between two languages

$$d(L_1, L_2) = \min_{w_1 \in L_1} d(w_1, L_2)$$

We can extend a distance on words to a distance between a word and a language

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

We can extend this further to a distance between two languages

$$d(L_1 | L_2) = \sup_{w_1 \in L_1} d(w_1, L_2)$$

We can extend distances between words to distances between two languages in a different way.

$$d(L_1|L_2) = \sup_{w_1 \in L_1} d(w_1, L_2)$$

This is called the relative distance from L_1 to L_2.

We can extend distances between words to distances between two languages in a different way.

$$d(L_1|L_2) = \sup_{w_1 \in L_1} d(w_1, L_2)$$

This is called the relative distance from L_1 to L_2.

- This distance is not symmetric.
- This distance can be unbounded.

Prior Work

Prior Work

- Almost-reflexivity of word relations (Choffrut, Pighizzini 2002)

Prior Work

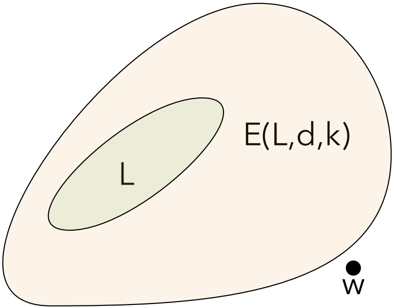
- Almost-reflexivity of word relations (Choffrut, Pighizzini 2002)
- Repair of regular specifications (Benedikt, Puppis, Riveros 2011)

Prior Work

- Almost-reflexivity of word relations (Choffrut, Pighizzini 2002)
- Repair of regular specifications (Benedikt, Puppis, Riveros 2011)
- Edit distance of pushdown automata (Chatterjee et al. 2015)

The neighbourhood of a language L is the set of words that are close to L.

$$E(L, d, k) = \{w \in \Sigma^* \mid d(w, L) \leq k\}$$



We say L_1 is contained in L_2 if $L_1 \subseteq L_2$. Similarly, if $d(L_1|L_2) \leq \infty$, then we can say that L_1 is approximately contained in L_2.

$$d(L_1|L_2) \leq k \text{ if and only if } L_1 \subseteq E(L_2, d, k)$$

REGULAR LANGUAGES

How to compute the distance from L_1 *to* L_2

Theorem

Let L_1, L_2 be regular languages recognized by NFAs A_1 and A_2 with n_1 and n_2 respectively. Suppose $d_p(L_1|L_2)$ is bounded. Then

$$d_p(L_1|L_2) \leq n_1 + n_2 - 2.$$

Theorem

Let L_1, L_2 be regular languages recognized by NFAs A_1 and A_2 with n_1 and n_2 respectively. Suppose $d_p(L_1|L_2)$ is bounded. Then

$$d_p(L_1|L_2) \leq n_1 + n_2 - 2.$$

- By the Pumping Lemma

Theorem

Let A_1 and A_2 be DFAs. Then $d_p(L(A_1)|L(A_2))$ is computable in polynomial time.

Theorem

Let A_1 and A_2 be DFAs. Then $d_p(L(A_1)|L(A_2))$ is computable in polynomial time.

- We just need to check

$$L(A_1) \subseteq E(L(A_2), d_p, n_1 + n_2 - 2).$$

Theorem

Let A_1 and A_2 be DFAs. Then $d_p(L(A_1)|L(A_2))$ is computable in polynomial time.

- We just need to check

$$L(A_1) \subseteq E(L(A_2), d_p, n_1 + n_2 - 2).$$

- The DFA for $E(L(A_2), d_p, n_1 + n_2 - 2)$ is at most

$$\frac{n_2(n_2 - 1)}{2} + n_1 + n_2 - 1$$

states (NRS 2015).

Theorem

Let $k \in \mathbb{N}$ *be fixed. For given NFAs* A_1 *and* A_2*, deciding whether or not* $d_p(L(A_1)|L(A_2)) \leq k$ *is PSPACE-complete.*

Theorem

Let $k \in \mathbb{N}$ be fixed. For given NFAs A_1 and A_2, deciding whether or not $d_p(L(A_1)|L(A_2)) \leq k$ is PSPACE-complete.

Lemma

Consider languages L_1 and L_2 over an alphabet Σ. Let $\#$ be a symbol not in Σ and $k \in \mathbb{N}$. Then

$$d_p(L_1\#^k|L_2) \leq k \text{ iff } L_1 \subseteq L_2.$$

Theorem

Let $k \in \mathbb{N}$ be fixed. For given NFAs A_1 and A_2, deciding whether or not $d_p(L(A_1)|L(A_2)) \leq k$ is PSPACE-complete.

Lemma

Consider languages L_1 and L_2 over an alphabet Σ. Let $\#$ be a symbol not in Σ and $k \in \mathbb{N}$. Then

$$d_p(L_1\#^k|L_2) \leq k \text{ iff } L_1 \subseteq L_2.$$

Remark

$$d_p(\Sigma^*\#^k|L) \leq k \text{ iff } \Sigma^* \subseteq L.$$

Corollary

Let A_1 and A_2 be NFAs. Then the problem of deciding whether $d_s(L(A_1)|L(A_2))$ is bounded is PSPACE-complete.

Corollary

Let A_1 and A_2 be NFAs. Then the problem of deciding whether $d_s(L(A_1)|L(A_2))$ is bounded is PSPACE-complete.

- The current best known DFA construction for $E(L(A_2), d_s, n_1 + n_2 - 2)$ has at most $n_1 + 2^{n_2}$ states, and is therefore not known to be polynomial in n_2 (NRS 2017).

NON-REGULAR LANGUAGES

How to determine if the distance from L_1 to L_2 is bounded by k

Proposition

Let $k \in \mathbb{N}$ be fixed. Given a regular language L_1 and a context-free language L_2, determining whether or not $d_p(L_1|L_2) \leq k$ is undecidable.

Proposition

Let $k \in \mathbb{N}$ be fixed. Given a regular language L_1 and a context-free language L_2, determining whether or not $d_p(L_1|L_2) \leq k$ is undecidable.

- We can reduce this to PDA universality

Proposition

Given an NFA A and a PDA P, deciding whether or not $d_p(L(P)|L(A)) \leq k$ is EXPTIME-complete.

Proposition

Given an NFA A and a PDA P, deciding whether or not $d_p(L(P)|L(A)) \leq k$ is EXPTIME-complete.

Proposition (Chatterjee et al. 2015)

Given a PDA P and an NFA A, the inclusion $L(P) \subseteq L(A)$ can be decided in EXPTIME. Given a deterministic PDA P and an NFA A, it is EXPTIME-hard to decide whether or not $L(P) \subseteq L(A)$.

Deterministic context-free languages (DCFL) are a proper subclass of context-free languages and are recognized by deterministic pushdown automata (DPDA).

Deterministic context-free languages (DCFL) are a proper subclass of context-free languages and are recognized by deterministic pushdown automata (DPDA).

- Inclusion of a regular language in a DCFL is decidable

Deterministic context-free languages (DCFL) are a proper subclass of context-free languages and are recognized by deterministic pushdown automata (DPDA).

- Inclusion of a regular language in a DCFL is decidable
- Then we just need to make sure that neighbourhoods of DCFLs are also DCFLs

Lemma

There exist a deterministic context-free language L *and integer* k *for which* $E(L, d_s, k)$ *is not a deterministic context-free language.*

Lemma

There exist a deterministic context-free language L and integer k for which $E(L, d_s, k)$ is not a deterministic context-free language.

Proof.

Let $L = \{ca^ib^ia^j \mid i, j \geq 0\} \cup \{da^ib^ja^j \mid i, j \geq 0\}$. Then L is a deterministic context-free language but

$$E(L, d_s, 1) \cap a^*b^*a^* = \{a^ib^ia^j \mid i, j \geq 0\} \cup \{a^ib^ja^j \mid i, j \geq 0\},$$

which is a context-free language but is not deterministic. □

Theorem

Let L be a deterministic context-free language. Then for every $k \geq 0$, the neighbourhood $E(L, d_p, k)$ is a deterministic context-free language.

Theorem

Let L be a deterministic context-free language. Then for every $k \geq 0$, the neighbourhood $E(L, d_p, k)$ is a deterministic context-free language.

- Whether the input word is within a distance of k can be determined by the current state and the top k symbols on the stack

Theorem

Let L be a deterministic context-free language. Then for every $k \geq 0$, the neighbourhood $E(L, d_p, k)$ is a deterministic context-free language.

- Whether the input word is within a distance of k can be determined by the current state and the top k symbols on the stack
- Keep track of the top k symbols of the stack in memory

Theorem

Let L be a deterministic context-free language. Then for every $k \geq 0$, the neighbourhood $E(L, d_p, k)$ is a deterministic context-free language.

- Whether the input word is within a distance of k can be determined by the current state and the top k symbols on the stack
- Keep track of the top k symbols of the stack in memory
- $O(nk|\Gamma|^k)$ states

A visibly pushdown automaton (VPA) is a PDA with the restriction that stack operations are determined by input symbols.

A visibly pushdown automaton (VPA) is a PDA with the restriction that stack operations are determined by input symbols.

The input alphabet Σ is partitioned into three sets

- call actions Σ_c; the VPA must push a symbol onto the stack
- return actions Σ_r; the VPA must pop a symbol from the stack
- internal actions Σ_i; the VPA cannot perform any stack operations

A visibly pushdown automaton (VPA) is a PDA with the restriction that stack operations are determined by input symbols.
The input alphabet Σ is partitioned into three sets

- call actions Σ_c; the VPA must push a symbol onto the stack
- return actions Σ_r; the VPA must pop a symbol from the stack
- internal actions Σ_i; the VPA cannot perform any stack operations

VPAs recognize the class of visibly pushdown languages.

Theorem

Let L be a visibly pushdown language. Then $E(L, d_p, k)$ is a visibly pushdown language for all $k \geq 0$.

- Modify the DPDA construction
- Dummy symbols are pushed onto the stack in order to satisfy the condition that symbols are pushed and popped from the stack when the correspodning symbols are read.

Proposition

Let $k \in \mathbb{N}$ be fixed. For given VPAs A_1 and A_2, deciding $d_p(L(A_1)|L(A_2)) \leq k$ is EXPTIME-complete.

Proposition

Let $k \in \mathbb{N}$ *be fixed. For given VPAs* A_1 *and* A_2*, deciding* $d_p(L(A_1)|L(A_2)) \leq k$ *is EXPTIME-complete.*

- Inclusion for VPAs is EXPTIME-complete (Alur, Madhusudan 2004)

The computational complexity of deciding $d_p(L(A_1)|L(A_2)) \leq k$ is summarized as follows.

A_2	DFA	NFA	VPA	DPDA	PDA
A_1					
DFA	P	PSPACE	EXPTIME	P	$\times$
NFA	P	PSPACE	EXPTIME	P	$\times$
VPA	P	EXPTIME	EXPTIME	$\times$	$\times$
DPDA	P	EXPTIME	$\times$	$\times$	$\times$
PDA	P	EXPTIME	$\times$	$\times$	$\times$

Open Questions

- How to decide when the distance is bounded for non-regular languages.
- How to compute the distance for non-regular languages.