Edit Distance of Finite state Transducers

C.Aiswarya[†], Amaldev Manuel[‡], Saina Sunny[‡]

[†]Chennai Mathematical Institute,[†] CNRS Relax IRL 2000, [‡]Indian Institute of Technology Goa

ICALP 2024

Abstract

We present a quantitative approach to distinguishing between two word-to-word functions defined by transducers, moving beyond the binary concept of equivalence. Transducers are machines that map input words to output words, such as spell checkers and grammatical tools. While testing the equivalence of functions defined by transducers is decidable, it alone does not suffice for many applications. For example, when comparing two grammar correction tools, can we ascertain which one aligns more closely with a benchmark based on their outputs? To address this, we define a distance between transducers based on output word metrics, such as edit distance. We prove the computability of the distance of functional transducers for various classical edit distances.

1. Metric on Words

- A *metric on words* over the alphabet A is a function $d : A^* \times A^* \to \mathbb{R} \cup \{\infty\}$ such that for any words u, v and w in A^*,
  1. $d(u, v) = 0 \iff u = v$.
  2. $d(u, v) = d(v, u)$.
  3. $d(u, v) \leq d(u, w) + d(w, v)$.

2. Edit Distances

- Edit distances is a class of metrics on words.
- Edits are operations that transform words, like inserting or deleting a letter.
- For a fixed set of edit operations C, the edit distance with respect to C between words u and v, is the minimum number of edits in C required to transform u to v if it is possible, and ∞ otherwise.
- Table 1 summarizes common edit distances along with their corresponding allowed operations.

Edit Distance	Permissible Operations
Hamming	letter-to-letter substitutions
Conjugacy	left and right cyclic shifts
Transposition	swapping adjacent letters
Longest Common Subsequence	insertions and deletions
Levenshtein	insertions, deletions, and substitutions
Damerau-Levenshtein	insertions, deletions, substitutions and adjacent transpositions

Table 1: Edit Distances.

$$\begin{aligned} d(hello, yellow) &= 2 \text{ (w.r.t. Levenshtein distance - substitute } h \text{ with } y\text{, insert } w\text{)} \\ &= \infty \text{ (w.r.t. Hamming distance)} \end{aligned}$$

3. Word Transducers

- Machines that reads an input word and produces output word(s) using finite memory.
- Transducers define relations between words.
- A transducer is functional if it defines a function.

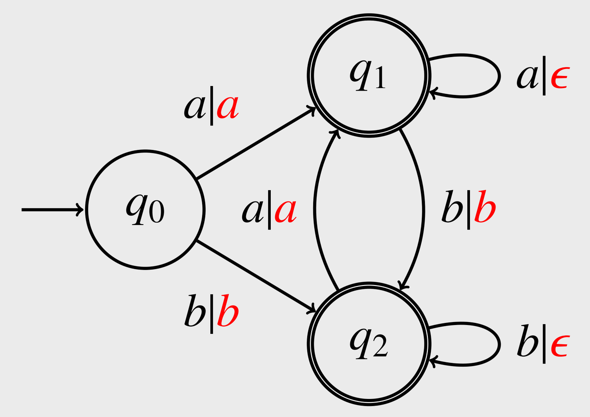
4. Distance between Transducers

- Let d be a metric on words. We can lift it to word-to-word functions.

$$d(f, g) = \begin{cases} \sup\{ d(f(w), g(w)) \mid w \in dom(f)\} & \text{if } dom(f) = dom(g) \\ \infty & \text{otherwise} \end{cases}$$

- The distance between two functional transducers is the distance between the functions defined by them.

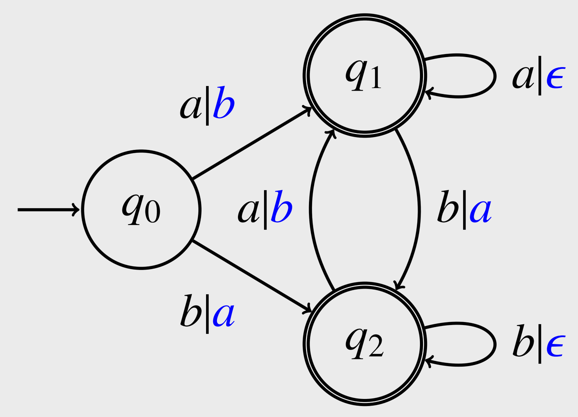
5. An Example



$\mathcal{T}$:

Replace blocks of a's with single a
Replace blocks of b's with single b

$(ab)^n \to (ab)^n$



$\mathcal{S}$:

Replace block of a's with single b
Replace block of b's with single a

$(ab)^n \to (ba)^n$

$$\begin{aligned} d((ab)^n, (ba)^n) &= 2 \text{ (w.r.t. Levenshtein distance - delete (insert) } a \text{ at the beginning (end))} \\ &= 2 * n \text{ (w.r.t. Hamming distance - } 2 * n \text{ substitutions)} \end{aligned}$$

$$\begin{aligned} d(\mathcal{T}, \mathcal{S}) &= 2 \text{ (w.r.t. Levenshtein distance)} \\ &= \infty \text{ (w.r.t. Hamming distance)} \end{aligned}$$

6. Related Questions

Problem	Input	Question
Distance Problem	transducers $\mathcal{T}, \mathcal{S}$	$d(\mathcal{T}, \mathcal{S})$?
Closeness Problem	transducers $\mathcal{T}, \mathcal{S}$	Is $d(\mathcal{T}, \mathcal{S}) < \infty$?
k-Closeness Problem	integer k, transducers $\mathcal{T}, \mathcal{S}$	Is $d(\mathcal{T}, \mathcal{S}) \leq k$?

7. Decidability Results

Theorem 1 *The closeness and k-closeness problem for functional transducers w.r.t. metrics given in Table 1 are decidable.*

For an integer-valued metric d, the distance problem w.r.t. d is computable if and only if k-closeness and closeness problems w.r.t. d are decidable.

Theorem 2 *The distance problem for functional transducers w.r.t. metrics given in Table 1 are computable.*

8. Algorithm for k-Closeness

- Given two transducers, check if their domains are equal.
- Construct product transducer that nondeterministically applies at most k edits to the first output and matches it with the second output and accepts if the matching is successful.
- It suffices to check if this transducer accepts all the inputs in the domain.

9. Algorithm for Closeness

- Given two transducers, check if their domains are equal.
- For Levenshtein family of distances, it is sufficient to check if the outputs generated by the loops of the product transducer are cyclic shifts of each other.
- For Hamming distance, it is sufficient to check if the output words generated by the loops of the product transducer after some shifted delay are identical.