Bálint Kollmann (B.Kollmann@student.tudelft.nl)
Supervisor: Sicco Verwer
EEMCS, Delft University of Technology

Finding distinguishing suffixes in DFA Learning

TU Delft

1. DFA Learning

- Infer a DFA consistent with positive (P) and negative (N) examples.
- Active learning: a learner infers the DFA by querying a teacher.
- iMAT framework with distinguish:
  - Membership queries.
  - **Distinguish queries.**

2. The Distinguish Query Problem

- Input: positive words P, negative words N.
- Query: Given words w_1 and w_2, find a suffix s such that
  $w_1 s \in P$ and $w_2 s \in N$.

Table 1: Examples of distinguish queries.

Input		Query		
P	**N**	**w_1**	**w_2**	**Suffix**
aaab	aa	aa	b	ab
ab	baa	ab	baa	ε
	bab	aaa	a	-

The empty word is ε, the – denotes that no distinguishing suffix is found.

3. Research questions

- How can distinguish queries be supported efficiently?
- What are the trade-offs between these approaches?

4. Double Trie

- Prefix trie + reversed-suffix trie
- For every split $w = ps$, link nodes corresponding to p and s.
- Link set of node = all nodes linked to it.
- **Key insight:** Link-set intersections solve distinguish queries.

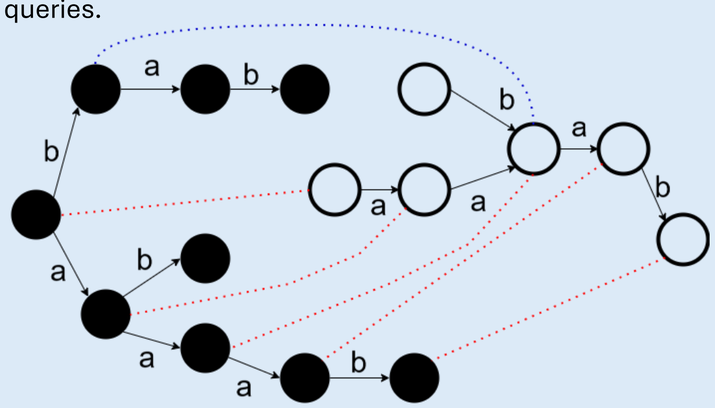

Figure 1: Double trie on (P = {aaab, ab}, N = {bab}).
Prefix trie: full nodes; suffix trie: hollow nodes.
Word aaab induces positive links (red dotted).
The blue dotted line is a negative link induced by bab.

5. Distinguish Query Algorithms

- Naive: trie +DFS
- Precomp: double trie + all distinguishable prefixes pairs
- Sqrt: double trie + sqrt-decomposition

Table 2: Worst-case theoretical time and space complexities (excluding processing of input and output words) .

Version	Preprocessing time	Query time	Space
Naive	O(z)	O(|Σ|z)	O(z)
Precomp	$O(z^2)$	O(1)	$O(z^2)$
Sqrt	$O(z^2/t)$	O(t)	$O(z^2/t^2)$

Σ is the alphabet size. z is the total number of symbols in $P \cup N$. t is the threshold parameter of Sqrt; in this work $t = \sqrt{z}$.

6. Setup of Experiments

- Data: standard benchmarks + hand-crafted datasets
- Evaluation metrics:
  - Preprocessing time.
  - Query time.
  - Memory usage.

7. Experimental Results

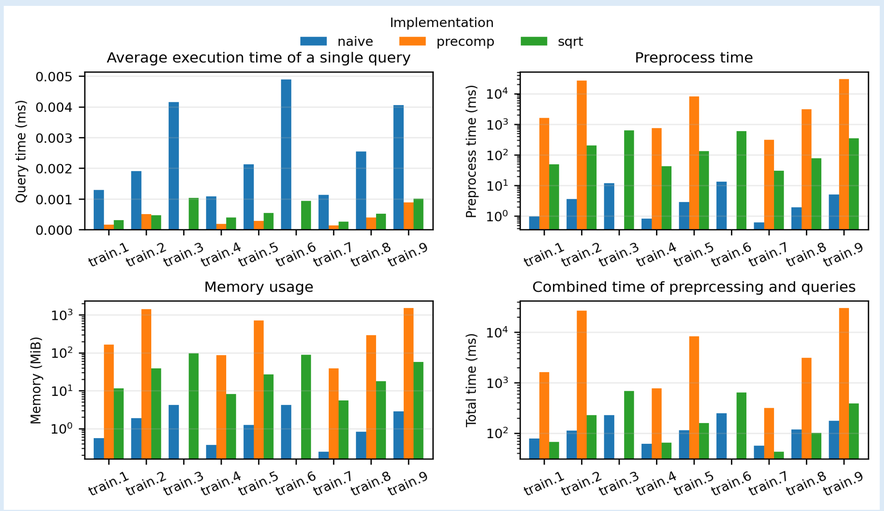

Figure 2: runtime and memory usage on the Abbadingo One datasets. Naïve performs best in terms of preprocess time, memory usage and total time. Sqrt is a close second in total time.

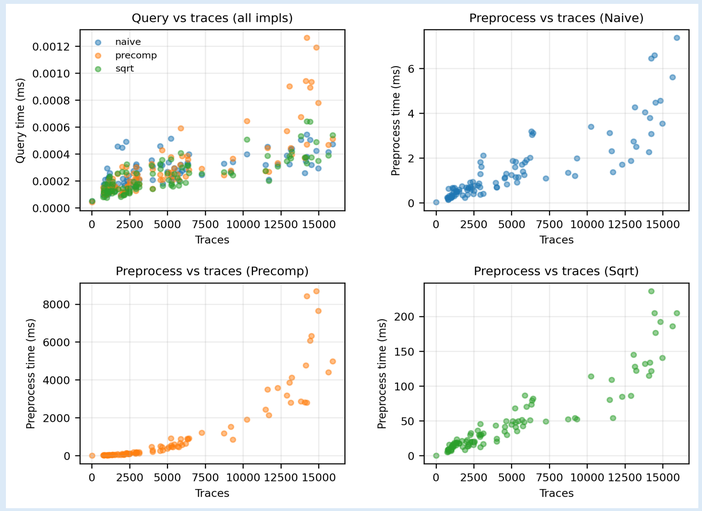

Figure 3: The figure shows the scalability of the implementations.
Query times show no clear trend, while preprocessing scales linearly for Naive and Sqrt and quadratically for Precomp.

8. Conclusion and Future work

- Double trie reduces queries to set intersection.
- Three designs (Naive, Sqrt, Precomp)
  - Theoretical and experimental analysis.
  - Naïve performs best in practice; Sqrt is competitive.
- Future work
  - Support insertion/deletion in P,N.
  - Explore database-based implementation.
  - Use it as a pruning mechanism.

Figure 4: QR code to the experimental pipeline repository.