Effect of changing the sequence orders on DFA ensembles learned via EDSM

Wiktor Cupiał (w.m.cupial@student.tudelft.nl)

Background

Deterministic Finite Automaton (DFA) learning is a problem of finding the smallest automaton consistent with a given language sample. The problem is proven to be NP-hard. However, to solve it, a heuristic called state merging [2] exists. It starts by constructing a structure called a prefix tree acceptor. Then, nodes in the tree are iteratively merged, aiming to produce the smallest automaton consistent with the given language sample.

(a) Initial prefix tree acceptor. (b) Automaton after the merge.

Figure 1. Example of performing a merge.

However, the order of merges is crucial because they introduce constraints on future merges. One of the methods to order the merges is Evidence Driven State Merging (EDSM) [1]. It is a proven and widely used method. To improve on this idea, we propose creating an ensemble of models learned using the EDSM algorithm, where each model is learned on modified input data. This change is made to introduce variety in the evidence computed by the algorithm and thus providing variety and potentially better results.

Problem Description

Research questions:

- What is an effective way to adjust sequence orders for different models in the ensemble?
- What is a good metric for inter-model variety for DFAs?

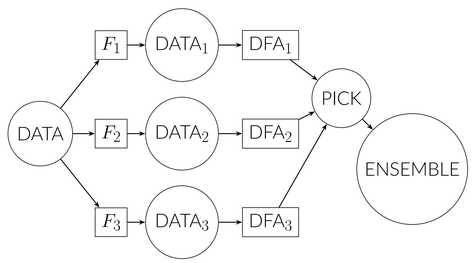


Figure 2. Visualization of the pipeline.

The main idea of the proposed approach is to consider some transformation of the input data (F_i in figure 2) and learn a DFA from it using the EDSM algorithm. Then, pick the best-suited models for an ensemble. To obtain predictions, apply the same transformation to the query and ask each model. Accept the trace if the number of models accepting it is above some given threshold.

Contribution

To create an ensemble of DFAs, we proposed changing the sequence orders. The goal of modifying the sequence order before running the EDSM algorithm is to highlight different aspects of data and thus inject some variety into the models produced by it.

The first method that we propose is changing the reading direction from prefix to suffix (or vice versa) after a fixed number of steps.

Figure 3. An example reading order achieved with this method.

The second method that we propose is reversing the substring in the input data. The motivation for this change comes from the fact that the part we want to highlight can be located in the middle of the trace.

Figure 4. An example processing order achieved with this method.

Both methods apply the same transformation to all traces in the training data. Since certain transformations may generate unreliable models, we use a validation set to pick a fixed number of the best-performing models into the ensemble. However, multiple models may be similar to each other. By similar, we mean they have languages that have a strong overlap.

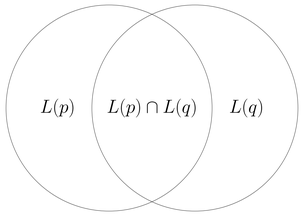


Figure 5. Languages of two automata and their intersection.

To mitigate this problem, we propose a random walk based similarity metric. It works by using a random walk to sample from languages of both automata, and then it computes the intersection. To add models into the ensemble, we again use the validation set, but we only add an automaton if the ensemble does not contain a model with similarity greater than some fixed threshold.

References

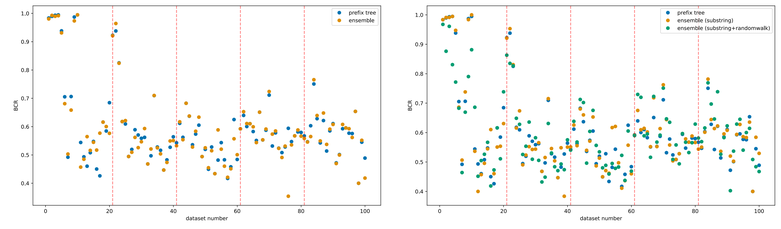
[1] Kevin J. Lang, Barak A. Pearlmutter, and Rodney A. Price. Results of the abbadingo one dfa learning competition and a new evidence-driven state merging algorithm. In Vasant Honavar and Giora Slutzki, editors, *Grammatical Inference*, pages 1–12, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[2] José Oncina and Pedro Garcia. Identifying regular languages in polynomial time. In *Advances in structural and syntactic pattern recognition*, pages 99–108. World Scientific, 1992.

[3] Neil Walkinshaw, Bernard Lambeau, Christophe Damas, Kirill Bogdanov, and Pierre Dupont. Stamina: a competition to encourage the development and assessment of software model inference techniques. *Empirical software engineering*, 18(4):791–824, 2013.

Results

The proposed approaches were evaluated on the dataset from the StaMinA competition [3]. They were compared against a baseline of DFA learned using the EDSM algorithm starting with a prefix tree acceptor.



(a) Reverse after k steps compared to the baseline. (b) Reverse substring methods compared to the baseline.

Figure 6. Results of the proposed approaches compared to the baseline on the StaMinA dataset.

Alphabet size	2	5	10	20	50	combined
Reverse after k steps	+7 =2 -11	+6 =4 -10	+9 =3 -8	+10 =4 -6	+9 =3 -8	+41 =16 -43
Reverse substring	+7 =2 -11	+9 =2 -9	+10 =0 -10	+13 =2 -5	+15 =1 -4	+54 =7 -39
Compression	+6 =1 -13	+11 =0 -9	+13 =0 -7	+13 =0 -7	+14 =0 -6	+57 =1 -42

Table 1. The score distribution and RMS for the *reverse after k steps* method.

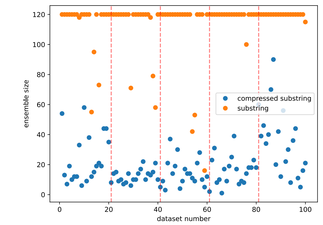


Figure 7. Sizes of the ensembles with and without using the similarity metric.

Conclusions

In the table 1 and figure 6, we can observe that the first proposed method was inconsistent, but achieved improvement over the baseline in some cases. Both the complete and reduced substring methods achieved worse results on smaller alphabet problems, but had substantial improvement on harder problems. This shows that the ensemble constructed by manipulating the sequence orders can be successfully used for DFA learning. However, further research is needed to explain why the ensemble performs worse on certain test cases.

Responsible professor: **Sicco Verwer**
Supervisor: **Simon Dieck**

TUDelft