Results of Neural-Checker Toolbox in Taysir 2023 Competition

Franz Mayr MAYR@ORT.EDU.UY
Sergio Yovine YOVINE@ORT.EDU.UY
Matías Carrasco MATIAS.CARRASCO@FI365.ORT.EDU.UY
Alejo Garat
Martín Iturbide
Juan da Silva
Federico Vilensky
Facultad de Ingeniería, Universidad ORT Uruguay, Montevideo, Uruguay

Editors: François Coste, Faissal Ouardi and Guillaume Rabusseau

Abstract

This paper presents the results obtained with the Neural-Checker toolbox in the Taysir 2023 challenge. It briefly describes the two tracks of the competition and the specific techniques that yielded the best results with respect to the corresponding scoring metrics.

Keywords: Active learning. Neural language models. DFA. PDFA. MAT-PAC.

1. Description of the competition

TAYSIR is an on-line competition about model inference form Neural Networks (NN). The 2023 challenge was divided in two tracks. The goal of Track 1 was to learn language acceptors from Recurrent NN and Transformer classifiers. Track 2 focused on learning surrogate models from Neural Language Models (NLM), also RNN and Transformers, that process sequences and output a distribution over the next symbols. For both tracks the evaluation score was defined as follows: Score $= 1/2 \times \mathrm{ER} + 1/4 \times \mathrm{MU} + 1/4 \times \mathrm{CT}$ where ER in the first track represents the error rate the learnt model has when comparing to the original NN, that is $\mathrm{ER}_1 = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[y_i \neq \hat{y}_i]$, and in the second represents the MSE $\times 10^6$, $\mathrm{ER}_2 = \frac{10^6}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$. MU represents the memory used by the model and CT represents the CPU time the submitted model lasts processing a given sequence.

The first track contained 11 NNs while the second one contained 10 NNs. The score was computed independently for each of the networks.

2. Description of the Tools Used

The tool we used is Neural-Checker Mayr et al. (2021). Neural-Checker is a repository that provides two libraries, pythautomata 0.38.1 and pyModelExtractor 0.35.2. Pythautomata's main goal is to provide implementations for the structures needed for working in the Model

Extraction Framework. PyModelExtractor's main goal is to enable the explainability and checking of complex systems in a black box context through the use of active learning techniques. For the first track of the competition we used a representation of a DFA (deterministic finite automata). A DFA provides a Regular Language that can differentiate which sequences belong to the Language. A DFA over an alphabet Σ can be represented as a tuple $(Q, \tau, q_{\text{in}}, \mathcal{F})$ consisting of: a finite set of states (Q), a transition function ($\tau : Q \times \Sigma \to Q$), an initial state ($q_{\text{in}}$) and a finite set of final states ($\mathcal{F}$). The DFA can process a sequence by starting in q_{in}, for each symbol on the sequence the states are traversed using τ given the symbol and the actual state. When the sequence is fully processed the DFA will return True if, and only if, the last state belongs to $\mathcal{F}$ or not.

For the second track the models were PDFA (probabilistic deterministic finite automata). A PDFA over Σ is a tuple $(Q, q_{\text{in}}, \pi, \tau)$, where Q is a finite set of states, $q_{\text{in}} \in Q$ is an initial state, $\pi : Q \to \Delta(\Sigma_{\$})$ maps each state to a probability distribution over $\Sigma_{\$}$, and $\tau : Q \times \Sigma \to Q$ is the transition function. Both π and τ are total functions.

3. Extraction Approach

PyModelExtractor provides several active black-box model extraction algorithms such as L* Angluin (1987) for DFA, and more novel ones like QuaNT Mayr et al. (2022) for PDFA. These algorithms construct regular models by interacting with a Minimally Adequate Teacher (MAT) using two operations: membership queries (**MQ**) and equivalence queries (**EQ**). **MQ** responds the output of querying the underlying target model inside the MAT, while **EQ** is a function that compares the target model with the constructed automaton. These algorithms require some considerations when working with NNs as there is no direct way of computing **EQ**, and there is no termination guarantees, as the target languages may be more complex than automata. In this scenario we followed Mayr and Yovine (2018) approach, that is, to use a sampling technique, like the one presented by the PAC framework Valiant (1984) and establish bounds to the learning processes.

EQ needs to define a distribution over sequences to sample from. We evaluated the following cases: *uniform length*, sampling from *length distribution in validation data*, and using the *full prefix set* of words up to some given length. The uniform length sampling consists in uniformly sampling a random length between a minimum and a maximum for each word and then creating a word with the chosen length using an uniform distribution over the symbols. Sampling from length distribution in validation, implies first choosing a length following a random sample from validation and then choosing every symbol of the sequence uniformly.

To guarantee termination, a maximum running time was set. This implies stopping an extraction if the run surpasses a given duration. When the extraction process is stopped by a bound instead of returning the last built model, we create a new model based on the final observation table (or tree). This idea, that we called *partial model*, allowed us to use

all the information gathered in the extraction and not only the information until the last **EQ**.

Finally to lower the cpu time and the memory usage of the submitted model we used new automata implementations, *FastDFA* and *FastPDFA*, which reduced the models' inference time and memory usage. However, these structures are not as general as the type of models provided by pyhtautomata, since symbols are only restricted to integer type and do not allow for more complex data structures.

4. Experimental Results

We will present the best results regarding the competition score, however, they are mainly focused in ER and CT, as MU was highly dependant on auxiliary library versions. In track 1, see Table 1, the datasets 2, 3, 4, 5, 6 and 7 ended up with a perfect submission ER. For these cases **EQ** was implemented with PAC, using a sampling technique that generated words with a fixed length of 22. In this setting we did not fix a maximum running time as L*execution finished with **EQ** passing the PAC test. All PAC tests were performed with $\epsilon = \delta = 0.01$ parameters. For dataset 1, 9, 10 and 11, PAC was not passed, and we resorted to use a a continuous run that stopped every 3 hours, output a partial DFA and then continued running with the same observation table. Best results were obtained on 5 to 7 iterations of this process (15-21 hours runs). For datasets 9, 10 and 11 sampling from length distribution in validation led to better results. For dataset 8 we tried all mentioned techniques but none ended with positive results. The result became more complex the longer we trained the model, making the memory usage bigger and the ER worse. We ended up submitting a trivial model that returns False for every given sequence. This turned out to be the best model.

Dataset	Duration (s)	EQs	MQs	Extracted States	Validation ER	Submission ER	MU (MiB)	CT (ms)
1	1.08E+04	9	1.18E+07	3.84E+04	7.10E-02	8.44E-02	139	6.8E-02
2	6.80E+01	5	1.08E+04	9.00E+00	0.00E+00	0.00E+00	121	7.2E-02
3	1.21E+02	3	3.02E+03	1.00E+01	0.00E+00	0.00E+00	121	7.4E-02
4	1.61E+02	3	2.89E+03	5.00E+00	0.00E+00	0.00E+00	96	5.7E-02
5	6.80E+01	2	1.16E+03	6.00E+00	0.00E+00	0.00E+00	96	5.6E-02
6	4.60E+01	1	1.29E+02	2.00E+00	0.00E+00	1.00E-05	96	5.4E-02
7	1.60E+01	1	1.29E+02	2.00E+00	0.00E+00	0.00E+00	96	5.5E-02
8	-	-	-	1.00E+00	3.42E-01	3.27E-01	108	3.0E-02
9	6.48E+04	3	5.13E+06	3.37E+04	1.34E-02	3.07E-02	121	5.7E-02
10	6.48E+04	1	3.09E+06	1.83E+04	1.82E-01	5.23E-02	126	5.9E-02
11	7.56E+04	0	4.81E+06	1.60E+04	2.49E-02	2.20E-02	186	8.6E-02

Table 1: Track 1 results

For track 2 the final results are presented in Table 2. In contrast to track 1, the validation ER was less correlated to the submission ER and perfect submission ER was

only obtained for datasets 7 and 9. It can be noted that duration times were smaller, this is because despite having been evaluated, running extraction for hours yielded worst results than the ones presented. Regarding **EQ** two sampling strategies proved to be successful, full prefix set and PAC sampling from length distribution in validation. The latter yielded better results in the competition for all datasets except for instance number 9. All PAC tests were performed with $\epsilon = \delta = 0.01$ parameters. None of the presented results passed any of those tests. Also **EQ** number was low, which means, most of the time the algorithm discovered states through **MQ**. Regarding κ, values in the set $\{10^i,\ i \in [1,5]\}$ were evaluated, in most scenarios higher values reported better results.

Dataset	Duration (s)	EQs	MQs	Extracted States	κ	Validation ER	Submission ER	MU (MiB)	CT (ms)
1	3.35E+02	1	5.96E+03	14	10^5	1.04E-04	3.77E-01	118	9.7E-02
2	7.64E+02	1	1.82E+03	32	10^5	6.17E+03	8.57E-03	118	9.7E-02
3	7.47E+02	1	1.26E+03	5	10^5	5.44E+03	4.32E-04	118	9.6E-02
4	4.73E+02	1	3.02E+03	7	10^4	3.40E-08	6.64E-08	118	9.8E-02
5	3.46E+02	1	9.11E+03	8	10^5	1.29E-03	9.66E-08	118	9.6E-02
6	7.21E+02	1	7.22E+03	17	10^5	1.03E-04	4.54E-01	118	9.7E-02
7	4.04E+02	1	2.18E+03	5	10^4	8.06E-34	0.00E+00	119	9.7E-02
8	2.45E+02	1	2.50E+03	4	10^4	1.82E-01	5.35E-03	118	9.7E-02
9	8.27E+02	3	5.63E+05	39	10	6.05E-37	0.00E+00	119	6.8E-02
10	2.23E+03	1	2.45E+05	218	10	1.72E-08	1.83E-01	120	1.2E-01

Table 2: Track 2 results

5. Conclusions

This work presented the results showcasing the performance of the Neural-Checker toolbox in the Taysir 2023 challenge, providing insights into the techniques that were effective in achieving the best results.

Acknowledgments Research reported in this article has been partially funded by ANII-Agencia Nacional de Investigación e Innovación under grant IA_1_2022_1_173516.

References

D. Angluin. Learning regular sets from queries and counterexamples. *Inf. Com.*, 75, 1987.

Franz Mayr and Sergio Yovine. Regular inference on artificial neural networks. In *Machine Learning and Knowledge Extraction*, pages 350–369, Cham, 2018. Springer.

Franz Mayr, Sergio Yovine, Federico Pan, and Federico Vilensky. Neural checker. https://github.com/orgs/neuralchecker/, 2021.

Franz Mayr, Sergio Yovine, Federico Pan, Nicolas Basset, and Thao Dang. Towards efficient active learning of pdfa. *arXiv preprint arXiv:2206.09004*, 2022.

Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.