Zulu: an Interactive Learning Competition*

David Combe, Colin de la Higuera and Jean-Christophe Janodet

Université de Lyon, F-42023 Saint-Étienne,
CNRS UMR5516, Laboratoire Hubert Curien,
Université de Saint-Etienne - Jean Monnet
david.combe@gmail.com, {cdlh,janodet}@univ-st-etienne.fr

Abstract. Active language learning is an interesting task for which theoretical results are known and several applications exist. In order to better understand what the better strategies may be, a new competition called Zulu (http://labh-curien.univ-st-etienne.fr/zulu/) is launched: participants are invited to learn deterministic finite automata from membership queries. The goal is to obtain the best classification rate from a fixed number of queries.

1 Introduction, motivations, history and background

When learning language models, techniques usually make use of huge corpora that are unavailable in many under-resourced languages. One possible way around this problem of lack of data is to be able to interrogate an expert with a number of chosen queries, in an interactive mode, until a satisfying language model is reached. In this case, an important indicator of success of the learning algorithm is the amount of energy the expert has to spend in order for learning to be successful. A nice learning paradigm covering this situation is that of Active Learning, introduced by Dana Angluin [1, 2]. The setting is formal and has been nicely studied over the years by a number of researchers, with a number of publications mainly in journals and conferences in machine learning or theoretical computer science.

Typically, the formalism involves the presence of a learner (he) and an Oracle (she).

There has to be an agreement on the type of queries one can make. Typical queries are:

- Membership queries. Learner presents a string to the Oracle and gets back the label.
- Strong equivalence queries. The learner presents a machine and gets back YES or a counterexample.
- Weak equivalence queries. The learner presents a machine and gets back YES or NO.

* This work was partially supported by the IST Programme of the European Community, under the Pascal 2 Network of Excellence, Ist–2006-216886.

– Correction queries. The learner presents an example and gets back YES or a close element from the target language ('close' means that a topology is defined).

A number of variants exist for the Oracle. She can be probabilistic, have a worse case policy, return noisy examples, etc. There are several issues to be considered, depending on what the Oracle really represents: a worse case situation, an average situation or even a helpful situation (the Oracle may then be a teacher).

A typical task on which active learning has been investigated is grammatical inference [3]. In this case, the task is to learn a grammatical representation of a language, while querying the Oracle. There are a number of negative results that have been obtained, tending to prove that some class of languages/grammars could not be efficiently learned by using queries of one type or another. Deterministic finite automata (DFA) were thoroughly investigated in this setting: as negative results, it was proved that they could not be learned from just a polynomial number of membership queries [4] nor from just a polynomial number of strong equivalence queries [5].

On the other hand, algorithm L* designed by Angluin [6], was proved to be able to learn DFA from a polynomial number of membership queries and equivalence queries: this combination is called a *minimally adequate teacher*. Extensions or alternative presentations of L* can be found in [7, 8] and there has been further theoretical work aimed at counting the number of queries really necessary [9, 10], on identifying the power of the equivalence queries [11], or relating the query model to other ones [12, 13]. Several open problems related to learning grammars and automata in this setting have been proposed [14].

Algorithm L* has been adapted to learn probabilistic automata [15, 16], multiplicity automata [17], NFA, tree automata, transducers [18], etc, [19, 20] with, in each case subtle variations.

An interesting extension has been studied recently, concerning correction queries [21–23].

On the other hand one may believe that the setting is purely formal and ill adapted to practical situations. For example, the fact that there is more and more data available might mean the end of interactive learning. On the contrary, the fact of being able to choose what data needs labelling, during the actual learning, offers many algorithmic advantages.

As far as applications go, typical situations in which Oracle learning makes sense are described in [24].

The earliest task addressed by L* inspired algorithms was that of map building in robotics: the (simplified) map is a graph and the outputs in each state are what the robot may encounter in a state. A particularity here is that the robot cannot be reset: the learning algorithm is to learn from just one very long string and all its prefixes [25, 26].

A task somehow related is that of discovering the rational strategy followed by an adversary. This line was looked into in a number of papers related to agent technologies [27, 28].

One task on which grammatical inference is proving to be particularly useful is that of wrapper induction: the idea is to find in a web page (or various of the same type) all arguments of a special sort. In this context, the role of the Oracle is played by the human user [29].

In different tasks linked with checking if the specifications of a system or a piece of hardware are met, the item to be checked is used as an Oracle. Queries are made in order to build a model of the item, and then the learned model can be checked against the specifications. Obviously, there is an issue with the distribution used to simulate the equivalence query [30–32].

What has not been hardly studied is how to optimise the learning task by trying to make easy queries, or queries for which the Oracle's answer is simple.

These are strong motivations for stemming research in the direction of developing new interactive learning strategies and algorithms.

2 A brief overview of Zulu

Zulu is both a web based platform simulating an Oracle in a DFA learning task and a competition.

As a web platform, Zulu allows users to generate tasks, to interact with the Oracle in learning sessions and to record the results of the users. It provides the users with a baseline algorithm written in JAVA, or the elements allowing to build from scratch a new learning algorithm capable of interacting with the server.

The server (http://labh-curien.univ-st-etienne.fr/zulu) can be accessed by any user/learner who can open an account. The server acts as an Oracle for membership queries. A player can log in and ask for a target DFA. The server then computes how many queries it needs to learn a reasonable machine (reasonable means less than 30% classification errors), and invites the player to interact in a learning session in which he can ask up to that number of queries. At the end of the learning process the server gives the learner a set of unlabelled strings (a test set). The labels the learner submits are used to compute his score.

As a starting point the baseline algorithm, which is a simple variation of L*, with some sampling done to simulate equivalence queries, is given to the user, who can therefore play with some simple JAVA code for a start (*i.e.* he doesn't have to develop from scratch).

The competition itself will be held in the spring of 2010. In this case, the competing algorithms will all have to solve new tasks. The exact design of the competition is still to be decided, with some open questions concerning fairness and avoiding collusion still to be solved.

The reasons for launching this challenge are that there seems to be a renewed interest in L*, or more generally in active learning of discrete structures. People have started to apply or adapt L* in computational linguistics but also in web applications or in formal specification checking. And interestingly, in all cases, the crucial resource is not the computation time but the number of queries.

Basically we believe that there is a lot of room for improvement, and real need for new ideas.

The hope is that in between researchers active in the field, others more interested in DFA, and perhaps students that have studied (or are studying) L* in a machine learning course, the competition will be a success.

3 Other related competitions

There have been in the past competitions related with learning finite state machines or grammars.

- Abbadingo (http://abbadingo.cs.nuim.ie/) was organised in 1999 : the goal was to learn DFA of sizes ranging from 64 to 512 states from positive and negative data, strings over a two letter alphabet.
- System Gowachin (http://www.irisa.fr/Gowachin/) was developed to generate new automata for classification tasks: the possibility of having a certain level of noise was introduced.
- Omphalos competition (http://www.irisa.fr/Omphalos/) involved learning context-free grammars, given samples which in certain cases contained both positive and negative strings, in others, just text.
- Tenjinno competition was organised in order to motivate research in learning transducers (http://web.science.mq.edu.au/tenjinno/).
- The Gecco conference organised a competition involving learning DFA from noisy samples (http://cswww.essex.ac.uk/staff/sml/gecco/NoisyDFA.html).

Of course, a number of machine learning competitions have been organised during the past years. A specific effort has been made by the Pascal network (http://pascallin2.ecs.soton.ac.uk/Challenges/).

4 A more complete presentation of Zulu

Let us now describe in more detail the way Zulu works.

4.1 Registration phase

Each participant (learning algorithm) would register and be given an ID. One can admit that a team presents various participants. Registration is done online. An email system ensures a minimum of security.

4.2 Training phase

A participant can choose between creating a new learning task or selecting one created by another player. It receives a number identifying a learning session, the size of the alphabet and the number of membership queries he is allowed. The participant then uses his learning algorithm (which can be a variation of

the baseline algorithm used to measure the difficulty of the task, or anything he chooses and that will communicate with the server with some predefined routines).

The algorithm then submits strings (membership queries) interactively and obtains their label.

When ready to guess (at most after having made the maximum number of queries), the learner receives from the server a set of 1800 new unseen strings, and proposes a labelling of these strings to the server. This is just another string of length 1800. He obtains a score, kept on the server, which can help others know how well the opposition is doing.

An idea of a learning session is represented in the following table. Note that the numbers are here meaningless: of course, you need many more queries than 66 for such a task.

Query : I would like to connect and want a target	Answer: OK, I have generated a new target. It has less than 100 states on a 26 letter alphabet. You are allowed 66 queries
Query: string red	Answer: YES
Query: string $green$	Answer: NO
Query: string t	Answer: YES
Query: string so	Answer: YES
Query: I think I know.	Answer: please label the following 1800 strings
Query: a string of length 1800: 0101000...	Algorithm XYZ did 77% on a 55 state DFA with 66 queries.

A typical score is something like:
Algorithm XYZ did 77% on a 55 state DFA with 66 queries.

4.3 Competition phase

In order to classify the contestants, a two-dimensional grid is used: one dimension concerns the size (in states) of the automata, and the other the size of the alphabet. The exact way of dealing with the classification is still an open issue. If every player is asked to work on the same task, these is a difficulty with collusion: why not share the results of a membership query? On the other hand, there is a real difficulty in comparing players that would not be working on the same tasks, as the variance is unknown.

5 Discussion

The hope is that Zulu will appeal to researchers in a number of fields, even if there will be room for criticism by all:

– Computational linguists are interested in active learning, but not necessarily of formal languages. Since the goal here is to learn about strategies, we feel that a certain control of the target is necessary.
– Researchers using machine learning in robotics (for map building, for example) might disagree with the fact that the structure of the graph underlying the DFA is not planar, or that the robot can be reset. These are some directions that should be investigated.
– Researchers closer to applications will disagree with the fact that there is no noise nor 'real' probabilities. Our feeling in this case is that too little is known about the 'simple' task for now.
– Researchers in active learning would surely like to be able to use alternative types of queries: this is in project, and we hope to be able to add the possibility of querying the Oracle in different ways (possibly with different costs depending on the 'difficulty' of the query) in a near future.
– The more orthodox theoreticians will argue (correctly) that we are treating the equivalence query replacement issue in an unsatisfying manner. This is true and we welcome better suggestions.
– Anyone who has looked into the problem of generating random automata will certainly be able to propose improvements on the procedure described on the website. We look forward to such discussions.
– A number of people would convincingly argue in favour of learning transducers (using translation queries) instead of DFA. This indeed seems to us to be the next step.

Acknowledgement

Zulu makes use of many pieces from the Gowachin engine developed by François Coste.

References

1. Angluin, D.: Queries and concept learning. Machine Learning Journal **2** (1987) 319–342
2. Angluin, D.: Queries revisited. Theoretical Computer Science **313**(2) (2004) 175–194
3. de la Higuera, C.: A bibliographical study of grammatical inference. Pattern Recognition **38** (2005) 1332–1348
4. Angluin, D.: A note on the number of queries needed to identify regular languages. Information and Control **51** (1981) 76–87
5. Angluin, D.: Negative results for equivalence queries. Machine Learning Journal **5** (1990) 121–150
6. Angluin, D.: Learning regular sets from queries and counterexamples. Information and Control **39** (1987) 337–350
7. Balcázar, J.L., Diaz, J., Gavaldà, R., Watanabe, O.: An optimal parallel algorithm for learning Dfa. In: Proceedings of the 7th Colt, New York, Acm Press (1994) 208–217

8. Kearns, M.J., Vazirani, U.: An Introduction to Computational Learning Theory. MIT press (1994)
9. Balcázar, J.L., Diaz, J., Gavaldà, R., Watanabe, O.: The query complexity of learning DFA. New Generation Computing **12** (1994) 337–358
10. Bshouty, N.H., Cleve, R., Gavaldà, R., Kannan, S., Tamon, C.: Oracles and queries that are sufficient for exact learning. Journal of Computer and System Sciences **52** (1996) 421–433
11. Gavaldà, R.: On the power of equivalence queries. In: Proceedings of the 1st European Conference on Computational Learning Theory. Volume 53 of The Institute of Mathematics and its Applications Conference Series, new series., Oxford University Press (1993) 193–203
12. Castro, J., Guijarro, D.: PACS, simple-PAC and query learning. Information Processing Letters **73**(1–2) (2000) 11–16
13. de la Higuera, C., Janodet, J.C., Tantini, F.: Learning languages from bounded resources: the case of the DFA and the balls of strings. [33] 43–56
14. de la Higuera, C.: Ten open problems in grammatical inference. [34] 32–44
15. de la Higuera, C., Oncina, J.: Learning probabilistic finite automata. [35] 175–186
16. Guttman, O., Vishwanathan, S.V.N., Williamson, R.C.: Learnability of probabilistic automata via oracles. In Jain, S., Simon, H.U., Tomita, E., eds.: Proceedings of ALT 2005. Volume 3734 of LNCS., Springer-Verlag (2005) 171–182
17. Bergadano, F., Varricchio, S.: Learning behaviors of automata from multiplicity and equivalence queries. SIAM Journal of Computing **25**(6) (1996) 1268–1280
18. Vilar, J.M.: Query learning of subsequential transducers. In Miclet, L., de la Higuera, C., eds.: Proceedings of ICGI '96. Number 1147 in LNAI, Springer-Verlag (1996) 72–83
19. Saoudi, A., Yokomori, T.: Learning local and recognizable ω-languages and monadic logic programs. In: Proceedings of EUROCOLT. LNCS, Springer-Verlag (1993) 157–169
20. Yokomori, T.: Learning two-tape automata from queries and counterexamples. Mathematical Systems Theory (1996) 259–270
21. Beccera-Bonache, L., Bibire, C., Dediu, A.H.: Learning DFA from corrections. In Fernau, H., ed.: Proceedings of the Workshop on Theoretical Aspects of Grammar Induction (TAGI). WSI-2005-14, Technical Report, University of Tübingen (2005) 1–11
22. Becerra-Bonache, L., de la Higuera, C., Janodet, J.C., Tantini, F.: Learning balls of strings from edit corrections. Journal of Machine Learning Research **9** (2008) 1841–1870
23. Kinber, E.B.: On learning regular expressions and patterns via membership and correction queries. [33] 125–138
24. de la Higuera, C.: Data complexity issues in grammatical inference. In Basu, M., Ho, T.K., eds.: Data Complexity in Pattern Recognition. Springer-Verlag (2006) 153–172
25. Dean, T., Basye, K., Kaelbling, L., Kokkevis, E., Maron, O., Angluin, D., Engelson, S.: Inferring finite automata with stochastic output functions and an application to map learning. In Swartout, W., ed.: Proceedings of the 10th National Conference on Artificial Intelligence, San Jose, CA, MIT Press (1992) 208–214
26. Rivest, R.L., Schapire, R.E.: Inference of finite automata using homing sequences. Information and Computation **103** (1993) 299–347
27. Carmel, D., Markovitch, S.: Model-based learning of interaction strategies in multi-agent systems. Journal of Experimental and Theoretical Artificial Intelligence **10**(3) (1998) 309–332

28. Carmel, D., Markovitch, S.: Exploration strategies for model-based learning in multiagent systems. Autonomous Agents and Multi-agent Systems **2**(2) (1999) 141–172
29. Carme, J., Gilleron, R., Lemay, A., Niehren, J.: Interactive learning of node selecting tree transducer. Machine Learning Journal **66**(1) (2007) 33–67
30. Bréhélin, L., Gascuel, O., Caraux, G.: Hidden Markov models with patterns to learn boolean vector sequences and application to the built-in self-test for integrated circuits. Pattern Analysis and Machine Intelligence **23**(9) (2001) 997–1008
31. Berg, T., Grinchtein, O., Jonsson, B., Leucker, M., Raffelt, H., Steffen, B.: On the correspondence between conformance testing and regular inference. In: Proceedings of Fundamental Approaches to Software Engineering, 8th International Conference, FASE 2005. Volume 3442 of Lncs., Springer-Verlag (2005) 175–189
32. Raffelt, H., Steffen, B.: Learnlib: A library for automata learning and experimentation. In: Proceedings of Fase 2006. Volume 3922 of Lncs., Springer-Verlag (2006) 377–380
33. Clark, A., Coste, F., Miclet, L., eds.: Grammatical Inference: Algorithms and Applications, Proceedings of Icgi '08. In Clark, A., Coste, F., Miclet, L., eds.: Grammatical Inference: Algorithms and Applications, Proceedings of Icgi '08. Volume 5278 of Lncs., Springer-Verlag (2008)
34. Sakakibara, Y., Kobayashi, S., Sato, K., Nishino, T., Tomita, E., eds.: Grammatical Inference: Algorithms and Applications, Proceedings of Icgi '06. In Sakakibara, Y., Kobayashi, S., Sato, K., Nishino, T., Tomita, E., eds.: Grammatical Inference: Algorithms and Applications, Proceedings of Icgi '06. Volume 4201 of Lnai., Springer-Verlag (2006)
35. Paliouras, G., Sakakibara, Y., eds.: Grammatical Inference: Algorithms and Applications, Proceedings of Icgi '04. In Paliouras, G., Sakakibara, Y., eds.: Grammatical Inference: Algorithms and Applications, Proceedings of Icgi '04. Volume 3264 of Lnai., Springer-Verlag (2004)