# Machine Learning for Theorem Proving

Dobrik Georgiev, dgg30

November 27, 2019

## 1 Introduction

Machine Learning (ML) has found its application in many areas (natural language processing, computer vision, etc) and it has succeeded in solving challenging tasks where classical deterministic approaches have performed poorly. ML was also successfully integrated into theorem proving, both interactive and automatic.

This survey focuses some of the ML techniques that have been applied for theorem proving. It is organised as follows: In section 2 I give a bit of background on theorem proving (both automated and interactive). In section 3 the focus falls on how ML is used in Interactive Theorem Proving and premise selection in particular. Section 4 will investigate how ML can be used to guide Automated Theorem Provers to improve their performance. Section 5 summarises the survey.

## 2 Background

Automated Theorem Provers (ATP) usually work by trying to prove a conjecture using techniques similar to an algorithm. Many approaches exist – resolution with unification, tableaux, DPLL (Davis et al., 1962), etc. Since these approaches can take exponential time to check a formula, various heuristic (even some ML ones) have been developed. Such provers, however, have been unable to make use of high-level abstractions and reasoning that humans usually do, when developing a complex proof or theory.

Proving complicated theories in higher-order logic in a 'one click' manner has turned out difficult to achieve and automate. Thus, a human factor has been integrated in the process of theorem proving and Interactive Theorem Provers (ITP) emerged (Harrison et al., 2014). In interactive theorem proving, users specify high-level definitions and proofs using a sequence of so-called *tactics*[1]. Low level details are then translated to ATP formalisms and handed to an automated theorem prover, such as E (Schulz, 2002), VAMPIRE (Riazanov and Voronkov, 2002) or similar. As programming languages, ITPs also have

[1]In an analogy to classic programming, these can be viewed as the instructions.

"libraries" (called *theories* or *theory sets*), which contain various proofs, which can be imported and reused.

## 3 Interactive Theorem Proving – Machine Learning for Premise Selection

One of the problems when handling the proof of a conjecture to a lower-level ATP arises when there are a lot of premises and facts (called *axioms* in theorem proving terminology), but very few of them are actually used in the proof of the conjecture. Consequently, the ATP may attempt to use the irrelevant facts which can in turn slow down the proving process or even lead to a timeout[2]. This occurs especially when a user is using a large theory or a set of several theories. Ideally, we would want to pick *only* those facts that are most relevant to what we want to prove/necessary to the proof. This section presents how this was approached using ML.

Some of the first experiments were preformed when translating Mizar[3] Mathematical Library, which consisted of many formalised mathematical theories, to first-order logic (Urban, 2004). In an attempt to automatically select useful axioms when proving theorems the Mizar Proof Advisor (MPA) was created. In the implementation of MPA the symbols used in a proof of a theorem $T$ were used as features and those were associated with the relevant proofs and facts used for proving $T$. A naive Bayes was used for evaluating whether an axiom would be useful for proving $T$.

Building on the Mizar Proof Advisor, MaLARea (Urban, 2007) was created using the same idea as in Urban (2004) that features of a conjecture can be associated with proofs which were successfully used when these features are present. The goal of MaLARea is again to give a ranking of the axioms according to their usefulness. What distinguishes this system from the above, is that it implements *deduce, learn from it, and loop* policy (Urban, 2007, p. 3). The policy's main loop consists of attempting to prove a set of problems (using an ATP) with no more than a fixed number of axioms which were selected by a Bayes classifier and under a fixed time limit. If nothing new is proven, the axiom and time limits are increased. Upon a successful proof, they are reset to their minimal values again and the proof is 'learned'. It was shown that with learning a relevance filter, the number of problems that can be solved with an ATP has increased drastically (cf. §3 of Urban (2007)) compared to passing all premises to the ATP.

The policy outlined above has been reused in the future as well, signifying the contribution of Urban (2007). Some of the examples follow:

- **MaLARea SG** Urban et al. (2008) – The system itself was improved

[2]As proving a conjecture can be NP-complete in some logics, most ATPs are run only for a fixed amount of time.

[3]Mizar is an assistant which mechanically checks proofs written by humans. See Naumowicz and Kornilowicz (2009) for an overview.

with semantic guidance (cf. Sutcliffe and Puzis (2007) for further details on semantic guidance).

- **MaSh** Kühlwein et al. (2013) employed it for improving Isabelle's Sledgehammer, which is a tool used for ATP proving of subgoals of a theory in Isabelle. Similar loop policy is used (Kühlwein et al., 2013, p.3) as well as the same algorithm – still a naive Bayes, but a custom implementation, in order to improve performance (previous works used the off-the-shelf tool SNoW (Carlsonn et al., 1999)). A key difference to the previous work lie in how features of premises/proofs are created – instead of using simply the symbols, the nontrivial first-order patterns up to a given depth are used (Kühlwein et al., 2013, p.6).

- **Learning with Flyspeck** – the policy was also applied to the Flyspeck project (Hales, 2006). One of the key differences here are that Kaliszyk and Urban (2014) claim that translation to ATP formalisms are sound in contrast to previous translations, e.g. Isabelle or Mizar to ATP, where one symbol from the corresponding ITP may be translated in several different ways. The motivation for a consistent translation is that external (heuristical) premise selection tools cannot be used otherwise and in their ranking scheme Kaliszyk and Urban (2014) use them as a 'preprocessing' step of the (re)learning loop. Features used here are a variation of symbol preprocessing (Kaliszyk and Urban, 2014, §4.1), similar to what Urban (2007) have.

Research time has also been devoted to improving the ML methodology for premise selection, since this is a key component of good performance, regardless whether we employ the aforementioned policy or not. Two of the methods that have been tested are:

- **Kernel Methods** – Alama et al. (2011) used a premise selection algorithm based on kernel methods. The benefit is that such classifier can learn non-linear dependencies (it was proved in the paper that NB can learn only linear ones).

- **k Nearest Neighbours (k-NNs)** – Kaliszyk and Urban (2013) also showed that k-NNs with Inverse Document Frequency weighting of the features can give better results than naive Bayes.

More recently, with the increasing popularity of Deep Learning, neural network models have been applied to premise selection (Alemi et al., 2016). The approach taken embeds a pair of (conjecture, axiom) into a feature vector, which is then processed by a neural network, resulting in a metric of how useful this axiom would be. The training loss for a conjecture is defined as the worst ranking of a premise which is known to be a true dependency (i.e. we know we cannot prove the conjecture without it). Many neural network models were tested, inspired both from Computer Vision and from Natural Language Processing. Results showed that the models can give better results than the baseline of k-NN with handcrafted features and in addition can be "complementary", i.e. even better results are obtained when the neural network and hand engineered approaches are combined – 80% of the problems used for testing were solved. Alemi et al. (2016) were the first to apply Deep Learning to theorem proving for premise selection and presented how a variety of neural network models can be used for premise selection.

The need for larger datasets was also identified. One such dataset is HolStep (Kaliszyk et al., 2017), featuring 2M statements (axioms/proofs/etc.) and 10K conjectures making applying data hungry techniques like DL for premise selection much easier. Other tasks together with baseline models (Kaliszyk et al., 2017, §3, 4) were also proposed. This dataset was used in training other models, e.g. the one from Wang et al. (2017).

Currently, the state-of-the-art uses ML/DL approaches for premise selection (e.g. Wang et al. (2017)). DL has also found use in generating full ITP tactics and the focus in the last year or so is more on end-to-end Interactive Theorem Proving proof generation, rather than only premise selection – see Yang and Deng (2019); Huang et al. (2019); Paliwal et al. (2019) for further information.

## 4 Guiding ATPs

As we saw in the last section, the approach of premise selection gives a generic interface in which any ATP can be substituted and a variety of ML/DL approaches can be employed. Despite its success, this approach has the drawback that it uses the ATP as a black box – it does not utilise the knowledge of successful proofs to guide the ATP itself. In this section I give a summary of some of the work done in using ML to guide an Automated Theorem Prover.

Urban et al. (2011) proposed an 'advisor' architecture to serve queries produced by the ATP and created a Machine Learning Connection Prover (MaLeCoP) which used ML to advise on **the most useful clause for every extension step** in a tableau proof. The advisor system is depicted in Figure 1 – **P** represents a prover, **A** the advisor system, **E** the external system. It can be noted that the idea of *usefulness of axioms* is very similar to what MaLARea does, but here, the actual proof state is also taken into account and the advice is on what axioms are most useful for the next steps, rather than for the *whole* proof.

By substituting leanCoP (Otten and Bibel, 2003), a connection tableaux prover for **P**, SNoW (Carlsonn et al., 1999) for **E** and a hand-implemented script for **A**, MaLeCoP was born. Evaluation showed that MaLeCoP can produce shorter by almost an order of magnitude proofs than those produced without using any ML but there was a very high overhead due to recomputing the features for similar states and the use of and communication to external ML system which made the approach impractical.

Based on the ideas of the above prototype, the system was greatly improved (Kaliszyk and Urban, 2015), making it both much faster than the prototype and better in generalisation (90 unseen problems were proved compared to 7).

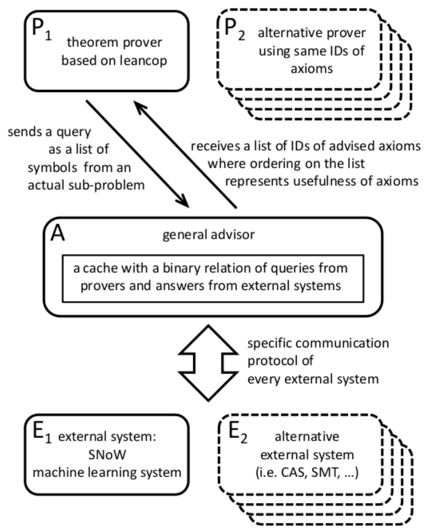
P1 theorem prover based on leancop
P2 alternative prover using same IDs of axioms
sends a query as a list of symbols from an actual sub-problem
receives a list of IDs of advised axioms where ordering on the list represents usefulness of axioms
A general advisor
a cache with a binary relation of queries from provers and answers from external systems
specific communication protocol of every external system
E1 external system: SNoW machine learning system
E2 alternative external system (i.e. CAS, SMT, ...)

Figure 1: General architecture proposed by Urban et al. (2011). Source: Urban et al. (2011)

Some of the differences to MaLeCoP are:

- efficient (re)computation of the features and relevance computation – the feature vector of the proof state (as well as relevance scores) is updated incrementally on a tableau extension.
- additional improvements on preprocessing the data and feature extraction, e.g. hash tables for faster accesses of the sum of occurences for a given features and more
- fast custom implementation of the ML advising system and its integration directly in the leanCoP code

According to Kaliszyk and Urban (2015) and to the best of my knowledge, this was the first time ML was *successfully* used for internally guiding an ATP.

The promising results of Kaliszyk and Urban (2015) inspired other works too and it was applied to other ATPs as well, such as Satallax (Färber and Brown, 2016). However it was not until the work of Jakubuv and Urban (2017) that ML was used for guiding an *state-of-the-art*[4] ATP. This was achieved via efficient feature extraction (term walks of length 3 (Jakubuv and Urban, 2017, §3.2)) and then utilising a large-scale linear classifier. Jakubuv and Urban (2017) again guided on the premise selection on every ATP step, and reused the concept of the feedback loop from MaLARea (Urban, 2007).

Reinforcement Learning algorithms have also been utilised. Kaliszyk et al. (2018) showed that it is possible to apply policy and value learning algorithms to

[4]Such as E and Vampire

guide leanCoP. Although features were extracted in the same hand-engineered way as Jakubuv and Urban (2017), results were very promising – 40% more proofs were obtained, compared to the original prover.

The tediousness of engineering feature extraction and the success of applying Deep Learning to premise selection (Alemi et al., 2016) inspired Loos et al. (2017) to apply Deep Learning based guidance on an ATP. The E clause based prover was chosen for the experiments. The DL guidance consists of taking an unprocessed clause and the negated conjecture, embedding them and calculating the probability that the clause would be used for proving the conjecture. Since calculating the score for all clauses on every step was too expensive and pure DL-based score was not practical, hybrid approaches were proposed (Loos et al., 2017, §5.2.1), which would initially use DL guidance but then would switch to the ATP heuristics, when resources start running out.

Given the success of the last two papers, current research mainly focuses on using RL and DL (Piotrowski and Urban, 2019; Zombori et al., 2019), but the exact approach taken can vary between the papers, depending on the particular problem identified and its solution.

## 5 Conclusion

This assignment surveyed the two main applications of ML to theorem proving – premise selection (usually associated with interactive theorem provers) and internal guidance of automated theorem provers in the presence of large theories. In both cases, that machine learning has been was used for filtering the number of 'step possibilities' (either the premises to use in ITP or the next proof step in ATP). It was observed that the applied algorithms/techniques are advantageous and can even lead to proofs which were impossible to prove before with using heuristics for guiding the ATP.

Due to the emerge of more advanced techniques, such as deep learning, and to the inherent difficulty of feature engineering, research has moved to using neural networks to learn the embeddings of the conjectures/premises and how these correlate. As neural networks can be computationally heavy and slow down the inference speed, various hybrid approaches have been applied instead that trade-off speed (inferences per second) versus smart proof extension choices. Results showed, that when Deep Learning is utilised correctly and the trade-off is balanced, significantly more and also *new* proofs can be obtained, without hurting the performance too much.

## References

Alama, J., Kühlwein, D., Tsivtsivadze, E., Urban, J., and Heskes, T. (2011). Premise selection for mathematics by corpus analysis and kernel methods. *CoRR*, abs/1108.3446.

Alemi, A. A., Irving, G., Szegedy, C., Eén, N., Chollet, F., and Urban, J.

(2016). DeepMath - deep sequence models for premise selection. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2235–2243.

Brown, C. E. (2012). Satallax: An automatic higher-order prover. In *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*, pages 111–117.

Carlsonn, A. J., Cumby, C. M., Rosen, J. L., and Roth, D. (1999). SNoW user guide. Technical report, University of Illinois.

Davis, M., Logemann, G., and Loveland, D. (1962). A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397.

Färber, M. and Brown, C. E. (2016). Internal guidance for satallax. In *Automated Reasoning - 8th International Joint Conference, IJCAR 2016, Coimbra, Portugal, June 27 - July 2, 2016, Proceedings*, pages 349–361.

Hales, T. C. (2006). Introduction to the flyspeck project. In Coquand, T., Lombardi, H., and Roy, M.-F., editors, *Mathematics, Algorithms, Proofs*, number 05021 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

Harrison, J., Urban, J., and Wiedijk, F. (2014). History of interactive theorem proving. In *Computational Logic*, pages 135–214.

Huang, D., Dhariwal, P., Song, D., and Sutskever, I. (2019). Gamepad: A learning environment for theorem proving. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Jakubuv, J. and Urban, J. (2017). ENIGMA: efficient learning-based inference guiding machine. In *Intelligent Computer Mathematics - 10th International Conference, CICM 2017, Edinburgh, UK, July 17-21, 2017, Proceedings*, pages 292–302.

Kaliszyk, C., Chollet, F., and Szegedy, C. (2017). HolStep: A machine learning dataset for higher-order logic theorem proving. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Kaliszyk, C. and Urban, J. (2013). Stronger automation for Flyspeck by feature weighting and strategy evolution. In *Third International Workshop on Proof Exchange for Theorem Proving, PxTP 2013, Lake Placid, NY, USA, June 9-10, 2013*, pages 87–95.

Kaliszyk, C. and Urban, J. (2014). Learning-assisted automated reasoning with Flyspeck. *J. Autom. Reasoning*, 53(2):173–213.

Kaliszyk, C. and Urban, J. (2015). FEMaLeCoP: Fairly efficient machine learning connection prover. In *Logic for Programming, Artificial Intelligence, and Reasoning - 20th International Conference, LPAR-20 2015, Suva, Fiji, November 24-28, 2015, Proceedings*, pages 88–96.

Kaliszyk, C., Urban, J., Michalewski, H., and Olsák, M. (2018). Reinforcement learning of theorem proving. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 8836–8847.

Kühlwein, D., Blanchette, J. C., Kaliszyk, C., and Urban, J. (2013). MaSh: Machine Learning for Sledgehammer. In *ITP*.

Loos, S. M., Irving, G., Szegedy, C., and Kaliszyk, C. (2017). Deep network guided proof search. In *LPAR-21, 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Maun, Botswana, May 7-12, 2017*, pages 85–105.

Naumowicz, A. and Kornilowicz, A. (2009). A brief overview of mizar. In *Theorem Proving in Higher Order Logics, 22nd International Conference, TPHOLs 2009, Munich, Germany, August 17-20, 2009. Proceedings*, pages 67–72.

Otten, J. and Bibel, W. (2003). leanCoP: lean connection-based theorem proving. *J. Symb. Comput.*, 36(1-2):139–161.

Paliwal, A., Loos, S. M., Rabe, M. N., Bansal, K., and Szegedy, C. (2019). Graph representations for higher-order logic and theorem proving. *CoRR*, abs/1905.10006.

Piotrowski, B. and Urban, J. (2019). Guiding theorem proving by recurrent neural networks. *CoRR*, abs/1905.07961.

Riazanov, A. and Voronkov, A. (2002). The design and implementation of VAMPIRE. *AI Commun.*, 15(2-3):91–110.

Schulz, S. (2002). E - a brainiac theorem prover. *AI Commun.*, 15(2-3):111–126.

Sutcliffe, G. and Puzis, Y. (2007). SRASS - A semantic relevance axiom selection system. In *Automated Deduction - CADE-21, 21st International Conference on Automated Deduction, Bremen, Germany, July 17-20, 2007, Proceedings*, pages 295–310.

Urban, J. (2004). MPTP - motivation, implementation, first experiments. *J. Autom. Reasoning*, 33(3-4):319–339.

Urban, J. (2007). MaLARea: a metasystem for automated reasoning in large theories. In *Proceedings of the CADE-21 Workshop on Empirically Successful Automated Reasoning in Large Theories, Bremen, Germany, 17th July 2007.*

Urban, J., Sutcliffe, G., Pudlák, P., and Vyskočil, J. (2008). MaLARea SG1 - machine learner for automated reasoning with semantic guidance. In Armando, A., Baumgartner, P., and Dowek, G., editors, *Automated Reasoning*, pages 441–456, Berlin, Heidelberg. Springer Berlin Heidelberg.

Urban, J., Vyskocil, J., and Stepánek, P. (2011). MaLeCoP machine learning connection prover. In *Automated Reasoning with Analytic Tableaux and Related Methods - 20th International Conference, TABLEAUX 2011, Bern, Switzerland, July 4-8, 2011. Proceedings*, pages 263–277.

Wang, M., Tang, Y., Wang, J., and Deng, J. (2017). Premise selection for theorem proving by deep graph embedding. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 2786–2796.

Yang, K. and Deng, J. (2019). Learning to prove theorems via interacting with proof assistants. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 6984–6994.

Zombori, Z., Csiszárik, A., Michalewski, H., Kaliszyk, C., and Urban, J. (2019). Towards finding longer proofs. *CoRR*, abs/1905.13100.