Clause Features for Theorem Prover Guidance[*]

Jan Jakubův and Josef Urban

Czech Technical University in Prague, Prague, Czech Republic

1 Given Clause Guidance in Theorem Proving

State-of-the-art saturation-based automated theorem provers (ATPs) for first-order logic (FOL), such as E [10], are today's most advanced tools for general reasoning across a variety of mathematical and scientific domains. Many ATPs employ the *given clause algorithm*, translating the input FOL problem $T \cup \{\neg C\}$ into a refutationally equivalent set of clauses. The search for a contradiction is performed maintaining sets of *processed* (P) and *unprocessed* (U) clauses. The algorithm repeatedly selects a *given clause* g from U, extends U with all clauses inferred with g and P, and moves g to P. This process continues until a contradiction is found, U becomes empty, or a resource limit is reached. The search space of this loop grows quickly and it is a well-known fact that the selection of the right given clause is crucial for success.

ENIGMA [5, 6] is an *efficient* learning-based method for guiding given clause selection in saturation-based ATPs. ENIGMA is based on a simple but fast *logistic regression* algorithm effectively implemented by the LIBLINEAR open source library [4]. In order to employ logistic regression or a similar machine learning method, all generated first-order clauses need to be translated to fixed-length numeric *feature vectors*. This is done by translating each clause to a multi-set of *features* which is in turn translated to a numeric vector.

Various possible choices of efficient clause features for theorem prover guidance have been experimented with [5, 6, 7, 8]. So far our goal has been to develop with fast and practically usable methods, allowing E users to directly benefit from our work. Related research in developing neural approaches [2, 9, 11] is of great interest, but so far less practically usable in the context of a fast saturation-style prover. The original ENIGMA [5] uses term-tree walks of length 3 as features, while the second version [6] reaches better results by employing various additional features. Which of the additional features are the main cause of the improvement was however, not investigated. The main goal of this work is to employ experiments with various combinations of features, in order to estimate the value of every possible features choice. Moreover, we propose to use another popular machine learning method instead of logistic regression, namely *decision trees* and their gradient boosted ensembles, effectively implemented by XGBoost library [3]. In this way, we shall be able to estimate, which of the features are useful across various machine learning methods.

2 Experiments with Selection of Clause Features

The following types of features are used in the recent ENIGMA (see [6, Sec.2]):

Vertical Features (V) are (top-down-)oriented term-tree walks of length 3. For example, the unit clause $P(f(a, b))$ contains only features (P, f, a) and (P, f, b).

Horizontal Features (H) are horizontal cuts of a term tree. For every term $f(t_1, \ldots, t_n)$ in the clause, we introduce the feature $f(s_1, \ldots, s_n)$ where s_i is the top-level symbol of t_i.

Symbol Features (S) are various statistics about symbols used in the clause, namely, the

[*]Supported by the ERC Consolidator grant no. 649043 *AI4REASON*.

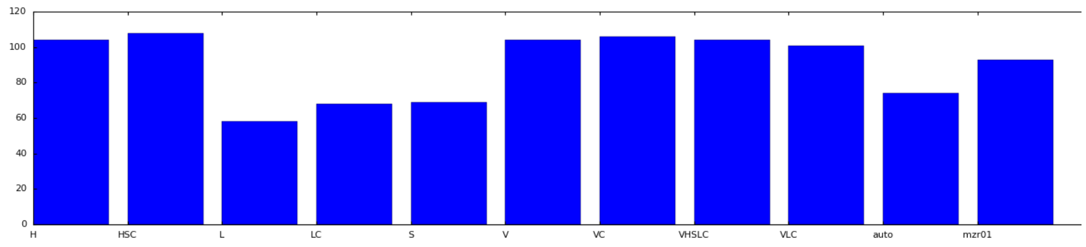

Figure 1: Number of training problems solved by selected variants.

number of occurrences and the maximal depth for each symbol.
Length Features (L) count the clause length and the numbers of positive and negative literals.
Conjecture Features (C) embed information about conjecture being proved into the feature vector. In this way, ENIGMA can provide conjecture-dependent predictions.

ENIGMA uses these features to translate clauses to features vectors. See [6] for how these feature vectors are used for given clause guidance. We experiment with all the possible combinations of features. This gives us 30 possible combinations, for example, "VHS" denotes the variant with vertical, horizontal, and symbol features. The original ENIGMA [5] uses only "V", while the enhanced ENIGMA [6] uses "VHSLC".

We evaluate all possible feature combination on the Mizar MPTP2078 [1] benchmark with 2078 problems as follows. We take a random subset consisting of 200 training problems and a single good-performing E Prover strategy (denoted "mzr01"), we run E with mzr01 on the training problems to obtain training samples, and we create a separate ENIGMA predictor for every possible feature combination. The number of training problems solved by selected variants are presented in Figure 1. From these preliminary results, we can conclude that there are variants that perform slightly better on the training problems than "VHSLC".

Next, we have evaluated all the predictors on all the benchmark (testing) problems with 5 seconds CPU time limit. From these preliminary results, it seems, that there is no clear winner, that is, no single variant outperforms the others. However, different variants solve quite complementary problems. The combined performance of all the variants greatly outperforms both the original strategy and the former ENIGMA. Although for the combined performance we use in general a 30 times higher time limit, we conclude that there is a great potential. Moreover, instead of searching for the best feature variant, it seems better to work simultaneously with several ones. The number of problems solved in time is depicted in graph (1) in Figure 2. Graphs (2) & (3) of the same figure, demonstrate the effect of ENIGMA guidance on the number of processed clauses.[1] ENIGMA guidance heavily shortens the proof search on the training problems and on a large number number of testing problems.

The proposed talk will present extended results, including experiments with additional features (e.g. from [7]), and with decision trees (XGBoost) used instead of logistic regression (LIBLINEAR). Moreover, the talk will present the comparison of the combined performance with the state-of-the-art theorem prover strategies conducted with a comparable time limit.

[1]Point (x, y) in the graph means that x clauses were processed without ENIGMA while y with ENIGMA on the same problem. Hence the more points under the diagonal, the better ENIGMA guides the proof search.

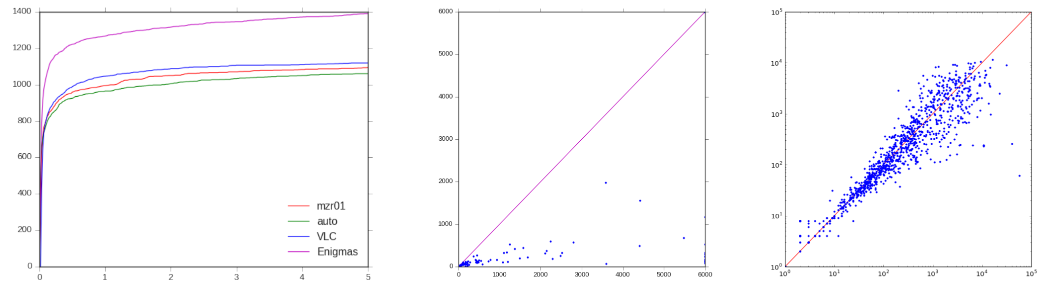

Figure 2: (1) Problems solved by selected strategies in time. Numbers of processed clauses with and without ENIGMA on (2) training and (3) testing problems.

References

[1] Jesse Alama, Tom Heskes, Daniel Kühlwein, Evgeni Tsivtsivadze, and Josef Urban. Premise selection for mathematics by corpus analysis and kernel methods. *J. Autom. Reasoning*, 52(2):191–213, 2014.

[2] Alexander A. Alemi, François Chollet, Niklas Eén, Geoffrey Irving, Christian Szegedy, and Josef Urban. DeepMath - deep sequence models for premise selection. In Daniel D. Lee, Masashi Sugiyama, Ulrike V. Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2235–2243, 2016.

[3] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *KDD*, pages 785–794. ACM, 2016.

[4] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[5] Jan Jakubův and Josef Urban. ENIGMA: efficient learning-based inference guiding machine. In *CICM*, volume 10383 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 2017.

[6] Jan Jakubuv and Josef Urban. Enhancing ENIGMA given clause guidance. In *CICM*, volume 11006 of *Lecture Notes in Computer Science*, pages 118–124. Springer, 2018.

[7] Cezary Kaliszyk, Josef Urban, Henryk Michalewski, and Mirek Olsák. Reinforcement learning of theorem proving. *CoRR*, abs/1805.07563, 2018.

[8] Cezary Kaliszyk, Josef Urban, and Jirí Vyskocil. Efficient semantic features for automated reasoning over large theories. In *IJCAI*, pages 3084–3090. AAAI Press, 2015.

[9] Sarah M. Loos, Geoffrey Irving, Christian Szegedy, and Cezary Kaliszyk. Deep network guided proof search. In Thomas Eiter and David Sands, editors, *LPAR-21, 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Maun, Botswana, May 7-12, 2017*, volume 46 of *EPiC Series in Computing*, pages 85–105. EasyChair, 2017.

[10] Stephan Schulz. E - A Brainiac Theorem Prover. *AI Commun.*, 15(2-3):111–126, 2002.

[11] Mingzhe Wang, Yihe Tang, Jian Wang, and Jia Deng. Premise selection for theorem proving by deep graph embedding. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 2783–2793, 2017.