Learning from Multiple Proofs: First Experiments

Daniel Kühlwein and Josef Urban*
Radboud University Nijmegen
Nijmegen, Netherlands
firstname.lastname@gmail.com

Abstract

Mathematical textbooks typically present only one proof for most of the theorems. However, there are infinitely many proofs for each theorem in first-order logic, and mathematicians are often aware of (and even invent new) important alternative proofs and use such knowledge for (lateral) thinking about new problems.

In this paper we start exploring how the explicit knowledge of multiple (human and ATP) proofs of the same theorem can be used in learning-based premise selection algorithms in large-theory mathematics. Several methods and their combinations are defined, and their effect on the ATP performance is evaluated on the MPTP2078 large-theory benchmark. Our first findings are that the proofs used for learning significantly influence the number of problems solved, and that the quality of the proofs is more important than the quantity.

1 Introduction

Automated Theorem Provers (ATPs) struggle when a problem has a large number of unnecessary premises [9]. Premise selection in large theories seems to be an important instance of the general *proof guidance* problem. It has been shown that proper design and choice of knowledge selection heuristics can change the overall success of large-theory ATP techniques by tens of percents [1]. This paper continues our work on machine learning algorithms for premise selection [1, 5]. We investigate how the knowledge of different proofs can be integrated in the machine learning algorithms for premise selection, and how it influences the performance of the ATPs.

In our earlier experiments [5] we tested and evaluated several premise selection algorithms on a subset of the Mizar Mathematical Library (MML), the MPTP2078 large-theory benchmark,[1] using the (human) proofs from the MML as training data for the learning algorithms. We have found that (i) learning from such human proofs helps a lot, but (ii) alternative proofs can quite often be successfully constructed by ATPs, making heuristic methods like SInE surprisingly strong and orthogonal to learning methods. Thanks to these experiments we now also have (possibly several) ATP proofs for most of the problems. This gives us an opportunity to learn from different (and in particular not just human) proofs and their combinations, and observe the influence on the ATP performance.

The rest of the paper is organized as follows. Section 2 introduces the necessary machine learning terminology and explains how different proofs can be used in the algorithms. In Section 3, we define several possible ways to use the additional knowledge given by the different proofs. The different proof combinations are evaluated and discussed in Section 4, and Section 5 concludes.

*The authors were supported by the NWO projects "MathWiki a Web-based Collaborative Authoring Environment for Formal Proofs" and "Learning2Reason".

[1] Available at http://wiki.mizar.org/twiki/bin/view/Mizar/MpTP2078.

2 The Machine Learning Framework and the Data

We start with the setting introduced in [1, 5]. Γ denotes the set of all first order formulas (usually axioms, definitions and theorems) that appear in a given (fixed) large mathematical corpus (MPTP2078 in this paper). The corpus is assumed to use notation (symbols) and formula names consistently, since they are used to define the features and labels for the machine learning algorithms. Given a conjecture c and a large number of *allowed premises* $\mathcal{A}_c \subset \Gamma$, the *premise selection problem* is to predict those premises from $\mathcal{A}_c$ that are likely to be useful for automatically constructing a proof of c. We typically experiment with proving theorems from Γ, so not all formulas in Γ can be allowed as premises for a given $c \in \Gamma$. In practice, Γ is typically ordered by the chronological growth of ITP libraries, and we use this ordering to define which premises are allowed for a given conjecture.

We say that a *proof P is a proof over* Γ if the conjecture and all premises used in P are elements of Γ. Given a set of proofs Δ over Γ in which every formula has at most one proof, the (Δ-based) *proof matrix* $\mu_\Delta : \Gamma \times \Gamma \to \{0, 1\}$ is defined as

$$\mu_\Delta(c, p) \coloneqq \begin{cases} 1 & \text{if } p \text{ is used to prove } c \text{ in } \Delta, \\ 0 & \text{otherwise.} \end{cases}$$

In other words, μ_Δ is the adjacency matrix of the graph of the direct proof dependencies from Δ. This proof matrix, together with the formula features, is used for training machine learning algorithms in [1, 5], using MML proofs as Δ. [1] and [5] also contain more details on using machine learning for premise selection.

In [5] we compared several different premise selection algorithms on the MPTP2078 dataset. Thanks to this comparison we now have ATP proofs for 1328 of the 2078 problems, found by Vampire 0.6 [7]. For some problems we found several different proofs, meaning that the sets of premises used in the proofs differ. Figure 1 shows the number of different ATP proofs we have for each problem. The maximum number is 49. We found 6.71 proofs per solvable problem on average.

This database of proofs allows us to start considering in this paper multiple proofs for a $c \in \Gamma$. For each conjecture c, let Θ_c be the *set of all ATP proofs of* c in our dataset, and let n_c denote the *cardinality of* Θ_c. Due to the nature of our learning algorithms, we still use the (generalized) proof matrix setting, where the multiple proofs of c are all in one row of μ represented using *weights*. Therefore μ will rather be called the (premise) *weight matrix* in the rest of the paper, and the general interpretation of $\mu_X(c, p)$ is the relevance (*weight*) of a premise p for a proof of c determined by X, where X can either be a set of proofs defining μ as above, or a particular algorithm (typically in conjunction with the data to which it is applied) defining the values of μ. For a single proof σ, let $\mu_\sigma \coloneqq \mu_{\{\sigma\}}$, i.e.,

$$\mu_\sigma(c, p) \coloneqq \begin{cases} 1 & \text{if } \sigma \in \Theta_c \text{ and } p \text{ is used to prove } c \text{ in } \sigma, \\ 0 & \text{otherwise.} \end{cases}$$

We end this section by introducing the (so far a bit fuzzy) concept of (premise) *redundancy*, which seems to be at the heart of the problem that we are exploring. Let c be a conjecture and σ_1, σ_2 be proofs for c ($\sigma_1, \sigma_2 \in \Theta_c$) with used premises $\{p_1, p_2\}$ and $\{p_1, p_2, p_3\}$ respectively. In this case, premise p_3 can be called *redundant* since we know a proof of c that does not use p_3.[2]

[2]For this we assume some similarity between the efficiency of the proofs in Θ_c, which is the case for our experiments based on the 5-second time limit.

Redundant premises appear quite frequently in ATP proofs, for example, due to exhaustive equational normalization that can turn out to be unnecessary for the proof. Now imagine we have a third proof of c, σ_3 with used premises $\{p_1, p_3\}$. With this knowledge, p_2 could also be called redundant (or at least unnecessary). But one could also argue that at least one of p_2 and p_3 is not redundant. In such cases, it is not yet clear what a meaningful definition of redundancy should be (see also [6] for related topics in machine learning). Hence for the remainder of the paper, we use the term *redundancy* as a (partially) vague concept to talk about premises that might not be necessary for a proof.

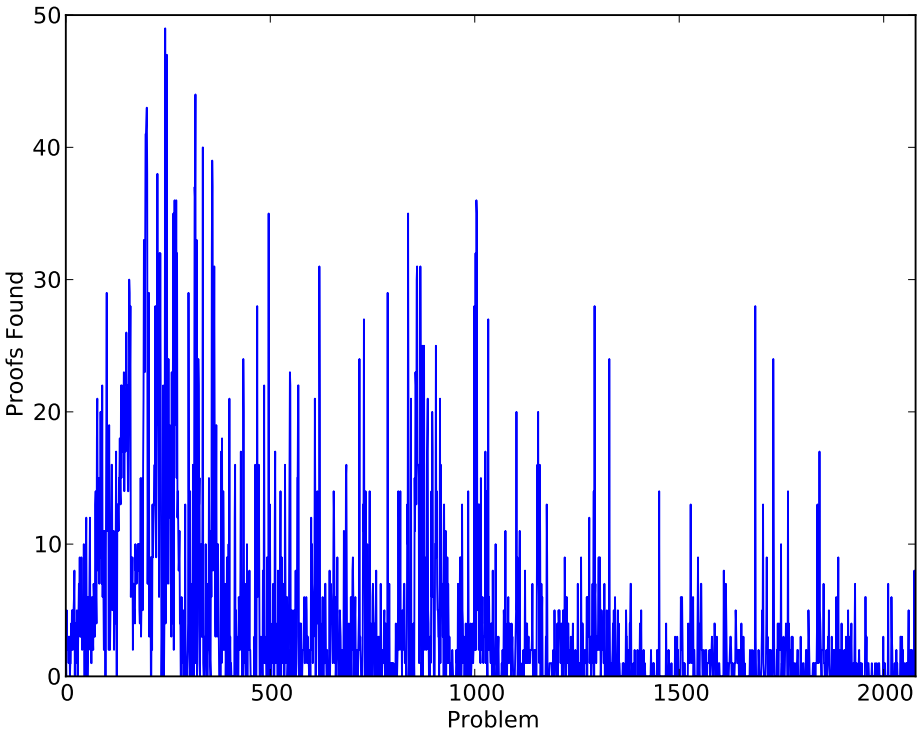


Figure 1: Number of different ATP proofs for each of the 2078 problems. The problems are ordered by their appearance in the MML.

3 Using Multiple Proofs

We define several different combinations of MML and ATP proofs and their respective premise weight matrices. Keep in mind that there are many problems for which we do not have any ATP proofs. For those problems, we will always just use the MML proof. I.e., for all premise weight matrices μ_X defined below, if there is no ATP proof of a conjecture c, then $\mu_X(c, p) = \mu_{\text{MML}}(c, p)$.

3.1 Substitutions and Unions

The simplest way to combine different proofs is to either only consider the used premises of one proof, or take the union of all used premises. We consider five different combinations.

Definition 3.1.1 (MML Proofs)**.**

$$\mu_{\mathrm{MML}}(c,p) \coloneqq \begin{cases} 1 & \textit{if } p \textit{ is used to prove } c \textit{ in the MML proof,} \\ 0 & \textit{otherwise.} \end{cases}$$

This dataset will be used as baseline throughout all experiments. It uses the human proofs from the Mizar library.

Definition 3.1.2 (Random ATP Proof)**.** *For each conjecture c for which we have ATP proofs, pick a (pseudo)random ATP proof $\sigma_c \in \Theta_c$. Then we define*

$$\mu_{\mathrm{Random}}(c,p) \coloneqq \begin{cases} 1 & \textit{if } p \textit{ is a used premises in } \sigma_c, \\ 0 & \textit{otherwise.} \end{cases}$$

Definition 3.1.3 (Best ATP Proof)**.** *For each conjecture c for which we have ATP proofs, pick an(y) ATP proof with the least number of used premises $\sigma_c^{min} \in \Theta_c$. We set*

$$\mu_{\mathrm{Best}}(c,p) \coloneqq \begin{cases} 1 & \textit{if } p \textit{ is a used premises in } \sigma_c^{min}, \\ 0 & \textit{otherwise.} \end{cases}$$

Definition 3.1.4 (Random Union)**.** *For each conjecture c for which we have ATP proofs, pick a random ATP proof $\sigma_c \in \Theta_c$. $\mu_{mathrmrandomUnion}$ is defined as*

$$\mu_{\mathrm{RandomUnion}}(c,p) \coloneqq \begin{cases} 1 & \textit{if } p \textit{ is a premise used in } \sigma_c \textit{ or in the MML proof of } c, \\ 0 & \textit{otherwise.} \end{cases}$$

Definition 3.1.5 (Union)**.** *For each conjecture c for which we have ATP proofs, we define*

$$\mu_{\mathrm{Union}}(c,p) \coloneqq \begin{cases} 1 & \textit{if } p \textit{ is a premise used in any ATP or MML proof of } c, \\ 0 & \textit{otherwise.} \end{cases}$$

3.2 Premise Averaging

A more advanced way to combine proofs is to consider some kind of *average* of the used premises. We consider three options, the standard average, a biased average and a scaled average.

Definition 3.2.1 (Average)**.** *The average gives equal weight to each proof.*

$$\mu_{\mathrm{Average}}(c,p) = \frac{1}{n_c+1} \sum_{\sigma \in \Theta_c} \mu_\sigma(c,p) + \mu_{\mathrm{MML}}(c,p)$$

The intuition is that the average gives a better idea of how necessary a premise really is. When there are very different proofs, such average will give a very low weight to every premise. That is why we also tried scaling as follows:

Definition 3.2.2 (Scaled Average)**.** *The scaled average ensures that there is at least one premise with weight 1.*

$$\mu_{\text{ScaledAverage}}(c,p) = \frac{\sum_{\sigma\in\Theta_c}\mu_\sigma(c,p)+\mu_{\text{MML}}(c,p)}{\max_{q\in\Gamma}\sum_{\sigma\in\Theta_c}\mu_\sigma(c,q)+\mu_{\text{MML}}(c,q)}$$

Another experiment is to make the weight of all the ATP proofs equal to the weight of the MML proof:

Definition 3.2.3 (Biased Average)**.**

$$\mu_{\text{BiasedAverage}}(c,p) = \frac{1}{2}\left(\frac{\sum_{\sigma\in\Theta_c}\mu_\sigma(c,p)}{n_c}+\mu_{\text{MML}}(c,p)\right)$$

3.3 Premise Expansion

Consider a situation where $a \vdash b$ and $b \vdash c$. Obviously, not only b, but also a proves c. When we consider the used premises in a proof, we only use the information about the *direct* premises (b in the example), but nothing about the *indirect* premises (a in the example), the premises of the direct premises. Using this additional information might help the learning algorithms. We call this *premise expansion*. We define three different weight functions that try to capture this indirect information. All three penalize the weight of the indirect premises with a factor of $\frac{1}{2}$.

Definition 3.3.1 (MML Expansion)**.** *For the MML expansion, we only consider the MML proofs and their one-step expansions:*

$$\mu_{\text{MMLExp}}(c,p) = \mu_{\text{MML}}(c,p) + \frac{\sum_{q\in\Gamma}\mu_{\text{MML}}(c,q)\mu_{\text{MML}}(q,p)}{2}$$

Note that since $\mu_{\text{MML}}(c,p)$ is either 0 or 1, the sum $\sum_{q\in\Gamma}\mu_{\text{MML}}(c,q)\mu_{\text{MML}}(q,p)$ just counts how often p is a grandparent premise of c.

Definition 3.3.2 (Average Expansion)**.** *The average expansion takes μ_{Average} instead of μ_{MML}:*

$$\mu_{\text{AverageExp}}(c,p) = \mu_{\text{Average}}(c,p) + \frac{\sum_{q\in\Gamma}\mu_{\text{Average}}(c,q)\mu_{\text{Average}}(q,p)}{2}$$

Definition 3.3.3 (Scaled Expansion)**.** *And finally, we consider an expansion of $\mu_{\text{ScaledAverage}}$.*

$$\mu_{\text{ScaledAverageExp}}(c,p) = \mu_{\text{ScaledAverage}}(c,p) + \frac{\sum_{q\in\Gamma}\mu_{\text{ScaledAverage}}(c,q)\mu_{\text{ScaledAverage}}(q,p)}{2}$$

Deeper expansions and different penalization factors are possible, but given the performance of these initial tests shown in the next section we decided to not investigate further.

4 Results

4.1 Experimental Setup

All experiments were done on the MPTP2078 dataset. Because of its good performance in earlier evaluations, we used the Multi-Output-Ranking (MOR) learning algorithm [1] for the experiments. For each conjecture, MOR is allowed to train on all proofs that were (in the

chronological order of MML) done up to that conjecture. In particular, this means that the algorithms do not train on the data they were asked to predict. Three-fold cross validation on the training data was used to find the optimal parameters. For the combinations in 3.1, the AUC measure was used to estimate the performance. The other combinations used the square-loss error. For each of the 2078 problems, MOR predicts a ranking of the premises.

We again use Vampire 0.6 for evaluating the predictions. Vampire is run with 5s time limit on an Intel Xeon E5520 2.27GHz server with 24GB RAM and 8MB CPU cache. Each problem is always assigned one CPU. [3] For each MPTP2078 problem, we created 20 new problems, containing the $10, 20, ..., 200$ highest ranked premises and ran Vampire on each of them. The graphs show how many problems were solved using the $10, 20, ..., 200$ highest ranked premises. As a performance baseline, Vampire 0.6 in CASC mode (that means also using SInE with different parameters on large problems) can solve 548 problems in 10 seconds [1].

4.2 Substitutions and Unions

Figure 2 shows the performance of the simple proof combinations introduced in 3.1. It can be seen that using ATP instead of MML proofs can improve the performance considerably, in particular when only few premises are provided. One can also see the difference that the quality of the proof makes. The best ATP proof predictions always solved more problems than the random ATP proof predictions. Taking the union of two or more proofs decreases the performance. This can be due to the redundancy introduced by considering many different premises. This would suggest that the ATP search profits most from a simple and clear (one-directional) advice, rather than from a combination of ideas.

4.3 Premise Averaging

Taking the average of the used premises could be a good way to combat the redundant premises. The idea is that premises that are actually important should appear in almost every proof, whereas premises that are redundant should only be present in a few proofs. Hereby, important premises should get a high weight and unimportant premises a low weight. The results of the averaging combinations can be seen in Figure 3.2.

Apart from the scaled average, it seems that taking the average does perform better than taking the union. However, the baseline of only the MML premises is better or almost as good as the average predictions.

4.4 Premise Expansions

Finally, we compare how expanding the premises effects the ATP performance in Figure 3.3. While expanding the premises does add additional redundancy, it also adds further potentially useful information.

However, all expansions perform considerably worse than the MML proof baseline. It seems that the additional redundancy outweighs the usefulness.

[3] We use Vampire as our default ATP because of its good performance in the CASC competitions, and because of its good performance on MML reported in [9]. Version 0.6 was chosen to make the experiments comparable with the results reported in [1] and [5].

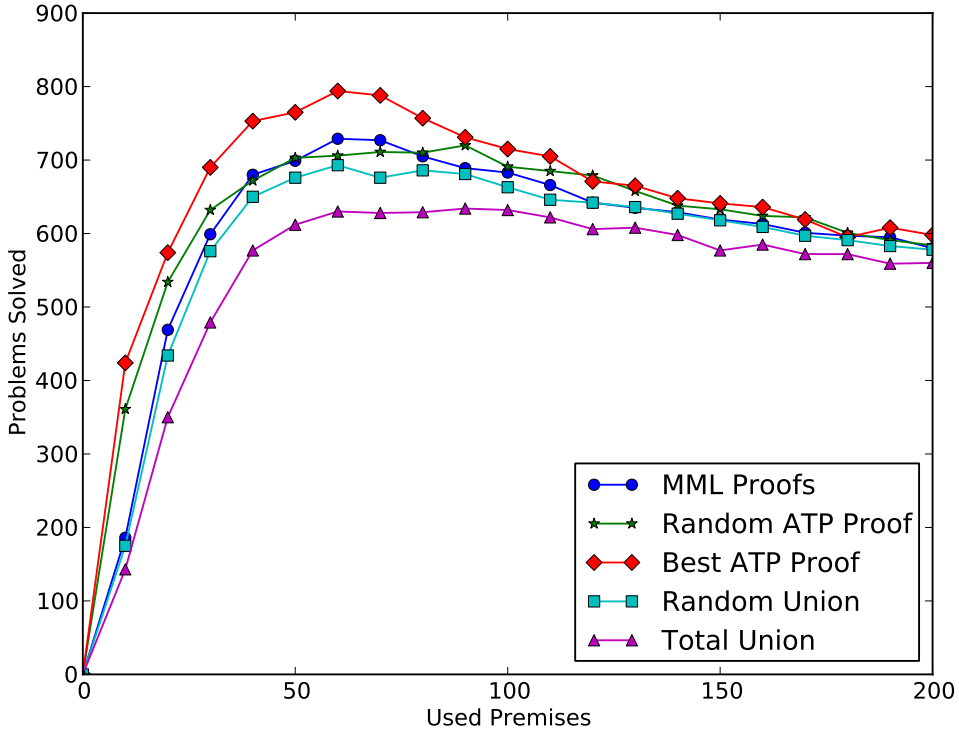


Figure 2: Comparison of the combinations presented in 3.1.

4.5 Other ATPs

We also investigated how learning from Vampire proofs affects other provers, by running E 1.4 [8] and Z3 3.2[3] on some of the learned predictions.[4] Figure 5 shows the results. The predictions learned from the MML premises serve as a baseline.

E using the predictions based on the best Vampire proofs is not so much improved over the MML-based predictions as Vampire is. This would suggest that the ATPs really profit most from "their own" best proofs. However for Z3 the situation is opposite: the improvement by learning from the best Vampire proofs is at some points even slightly better than for Vampire itself, and this helps Z3 to reach the maximum performance earlier than before. Also, learning from the averaged proofs behaves differently for the ATPs. For E, the MML and the averaged proofs give practically the same performance, for Vampire the MML proofs are better, but for Z3 the averaged proofs are quite visibly better. These effects are probably not so significant to be immediately investigated, but for example a comparison done on the whole MML (or its different parts), together with analysis of the strategies that the ATPs use, could provide more understanding. Another possible next experiment could be to swap the ATPs, and learn from E's proofs (or even Z3 proofs, which seem to be coming soon), and see how they improve the other ATPs.

[4] With the same settings as Vampire.

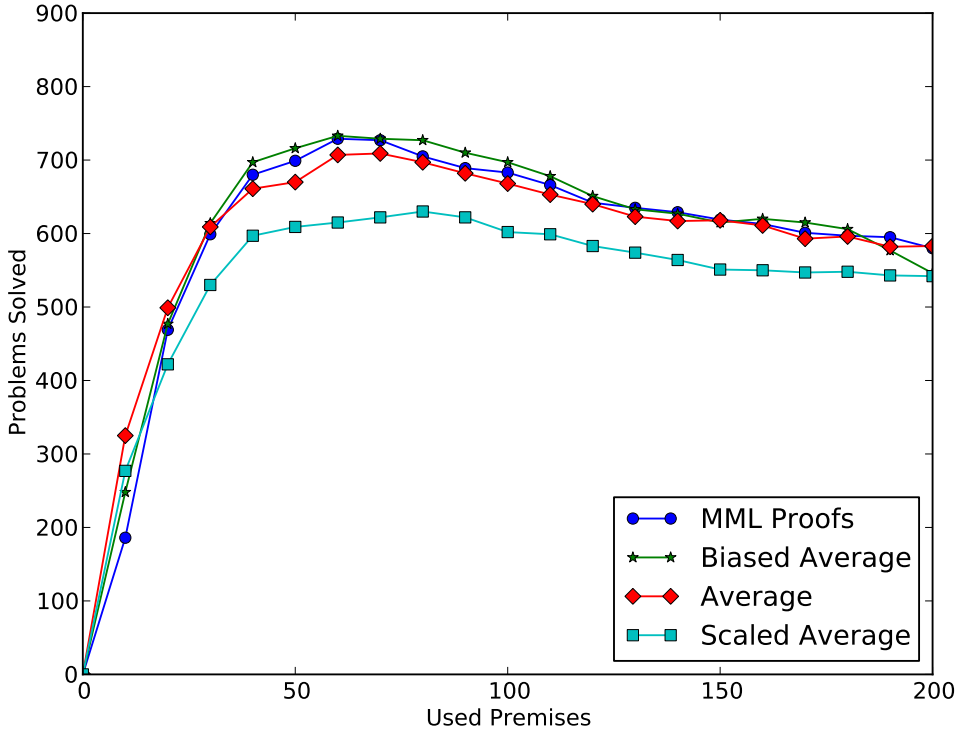


Figure 3: Comparison of the combinations presented in 3.2.

4.6 Comparison With the Best Results Obtained so far

In [5], we found that the a combination of SInE [4] and the MOR algorithm (trained on the MML proofs) has so far best performance on the MPTP2078 dataset. Figure 6 compares the new results from this paper with this combination. Furthermore we also try combining SInE with MOR trained on ATP proofs. For comparison we also include our baseline, the MML Proof predictions, and the results obtained from the SInE predictions.

While learning from the best ATP proofs leads to more problems solved than learning from the MML proofs, the combination of SInE and learning from MML proofs still beats both. However, combining the SInE predictions with the best ATP proof predictions gives even better results with a maximum of 823 problem solved (a 3.3% increase over the former maximum) when given the top 70 premises.

4.7 Machine Learning Evaluation

Machine learning has several methods to measure how good a learned classifier is without having to run an ATP. In general, this is done by comparing the learned prediction with the training data. The 100%Recall measure is an example. It tells us how many premises (starting from the highest ranked one) we need to give to the ATP to ensure that all necessary premises (according to the training data) are included.

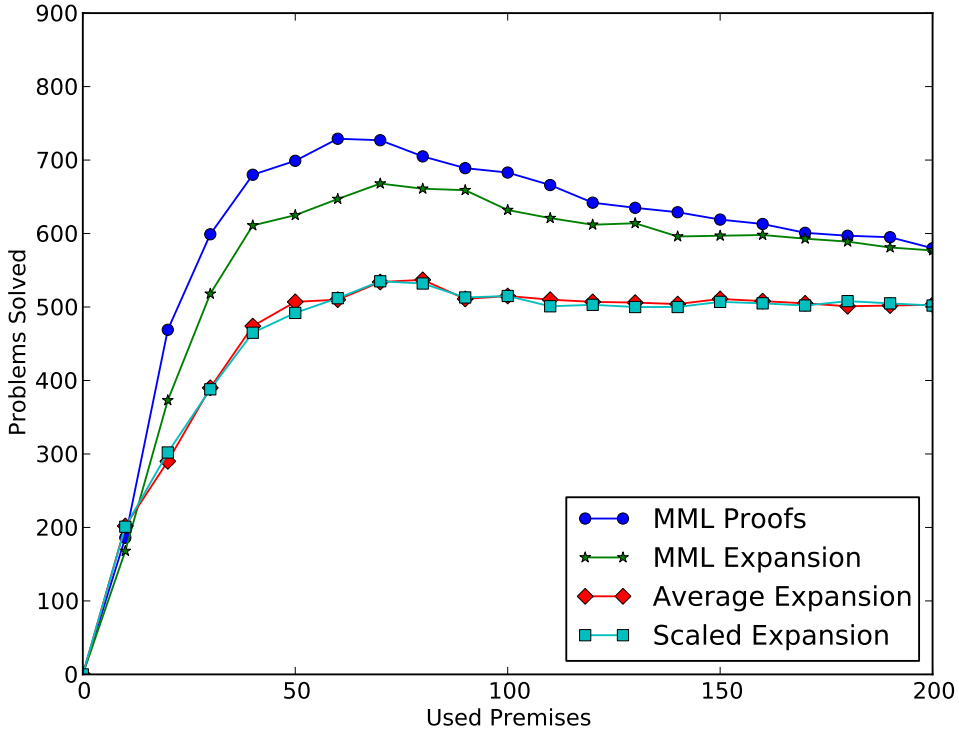


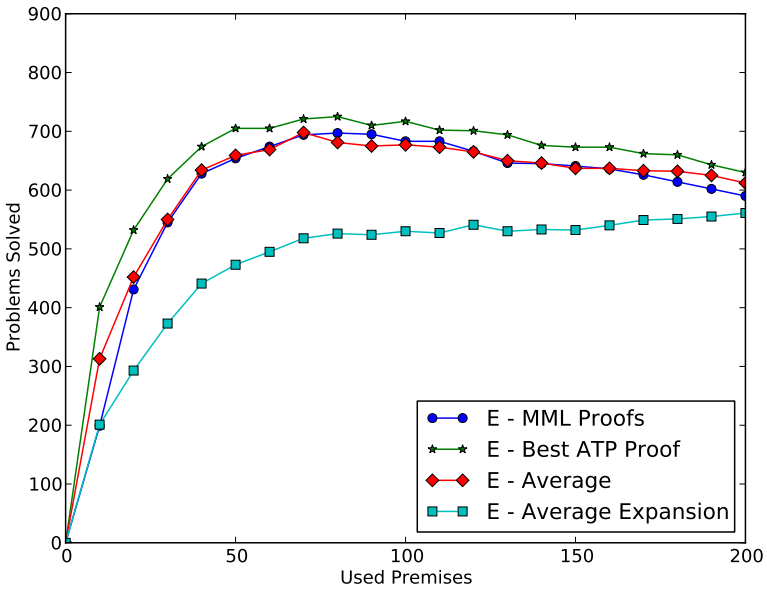
Figure 4: Comparison of the combinations presented in 3.3.

In [5] we found that the machine learning evaluation did not correspond to the ATP evaluation. For example, SInE performed worse than BiLi on the machine learning evaluation but better than BiLi on the ATP evaluation. Our explanation was that we are training from (and therefore measuring) the wrong data. With SInE the ATP found proofs that were very different from the MML proofs.

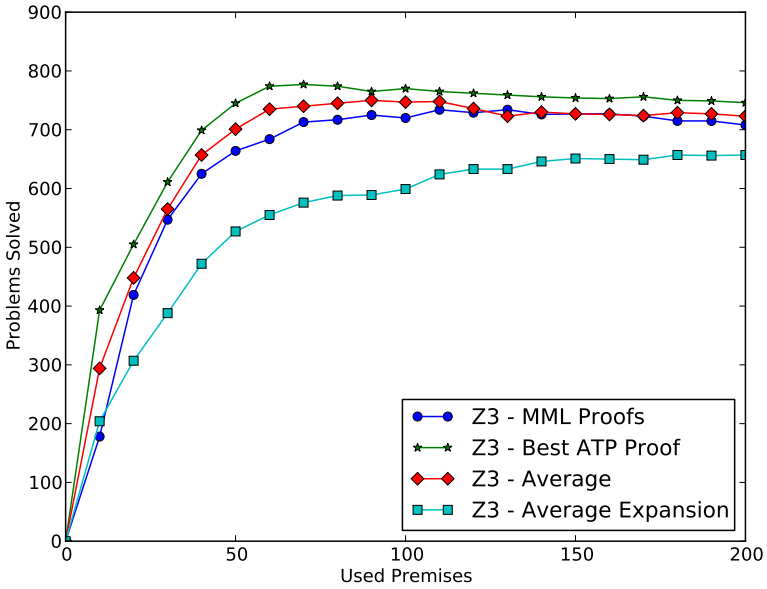
In Figure 7 we see a comparison between a machine learning evaluation (the 100%Recall measure) depending on whether we evaluate on the MML proofs or on the best ATP proofs. Ideally we would like to have that the machine learning performance of the algorithms corresponds to the ATP performance (see Figure 6). This is clearly not the case for the 100%Recall on the MML proofs graph. The best ATP predictions are better than the MML proof predictions, and SInE solves more than 200 problems. With the new evaluation, the 100%Recall on the best ATP proofs graph, the performance is more similar to the actual ATP performance but there is still room for improvement.

5 Conclusion and Future Work

The fact that there is never only one proof makes premise selection an interesting machine learning problem. Since it is in general undecidable to know the “best prediction”, the domain



(a) E



(b) Z3

Figure 5: Performance of other ATPs when learning from Vampire proofs.

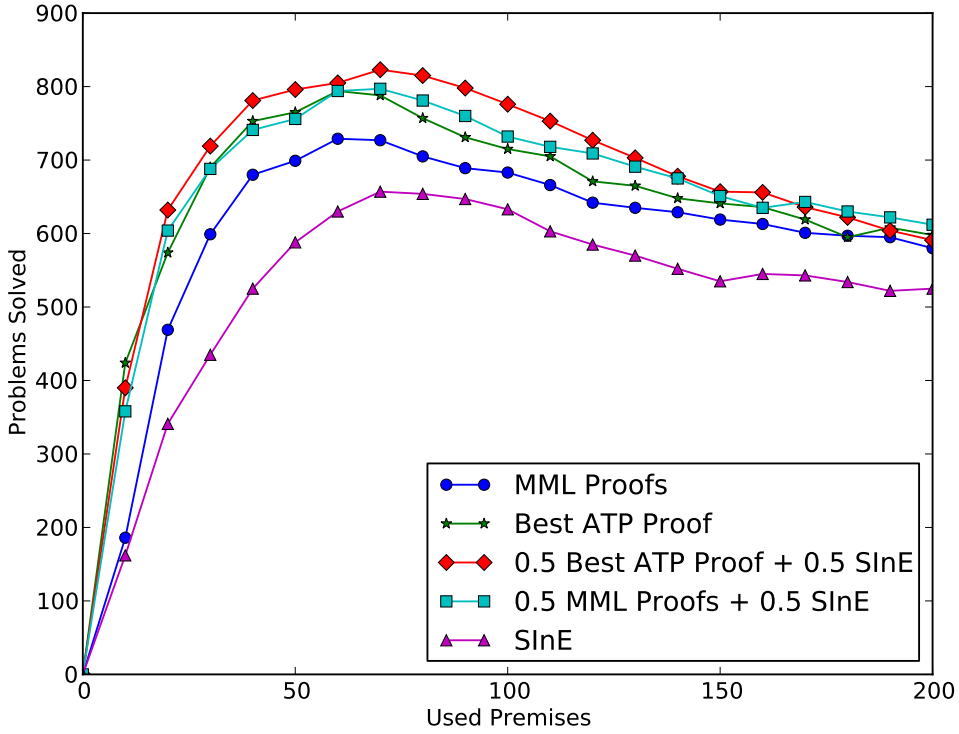


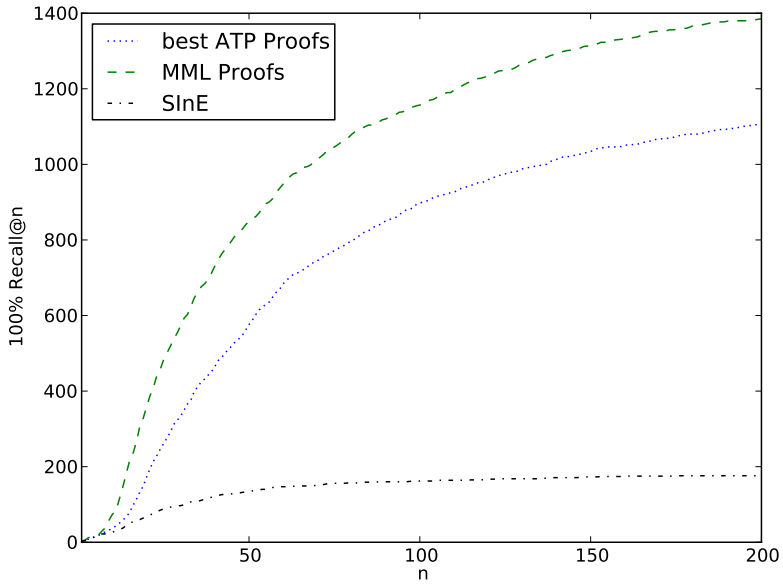
Figure 6: Comparison of the best performing algorithms.

has a randomness aspect that is quite unusual (Chaitin-like [2]) in AI.

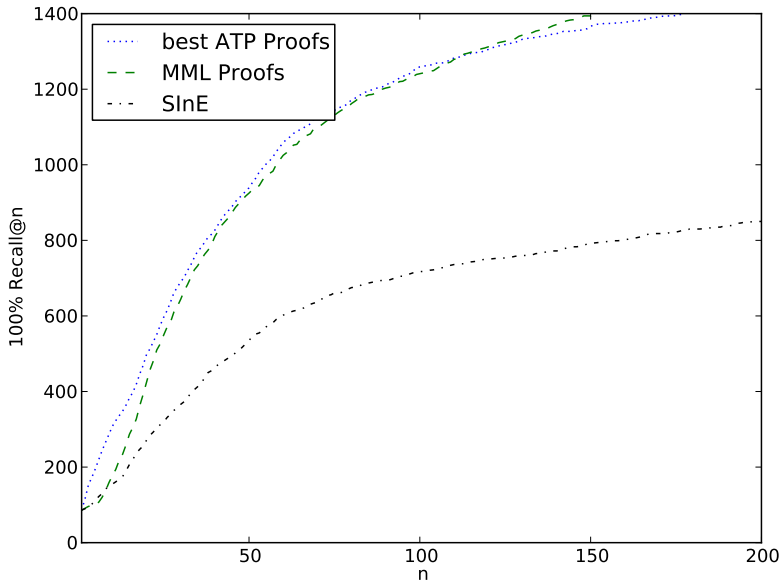
In this paper we started to explore how to combine different proofs to obtain better information for high-level proof guidance by premise selection. We found that it is easy to introduce so much redundancy that the predictions created by the learning algorithms are not good for existing ATPs. On the other hand we saw that learning from proofs with few premises (and hence probably less redundancy) increases the ATP performance. It seems that we should look for a measure of how 'good' or 'simple' a proof is, and only learn from the best proofs (A similar problem appears in machine learning under the name *instance selection* [6]). Such measures could be for example the number of inference steps done by the ATP during the proof search, or the total CPU time needed to find the proof.

Another question that was (at least initially) answered in this paper is to what extent learning from human proofs can help an ATP, in comparison to learning from ATP proofs. We saw that while not optimal, learning from human proofs seems to be approximately equivalent to learning from suboptimal (for example random, or averaged) ATP proofs. Learning from the best ATP proof is about as good as combining SInE with learning from the MML proofs. Combining SInE with learning from the best ATP proof still outperforms all methods, but the improvement is not as strong as in [5].

In the future we want to explore better methods for combining heuristics like SInE with machine learning methods. Such methods could operate (e.g., using SInE-like symbol expan-



(a) 100%Recall on the MML proofs.



(b) 100%Recall on the best ATP proofs.

Figure 7: 100%Recall comparison between evaluating on the MML and the best ATP proofs. The graphs show how many problems have all necessary premises (according to the training data) within the n highest ranked premises.

sion) on the graph of symbol definitions (and similarities) in an analogous way to how the proof graph expansion was used in Section 3, but enriching the set of learning features instead of the set of learning labels. This, together with learning from simpler proofs could lead to even better results. We are also interested in using thresholds (instead of just rankings) in our algorithms.

6 Acknowledgments

Tom Heskes, Evgeni Tsivtsivadze, and Elena Marchiori helped us with their machine learning knowledge and pointed us to literature that might be relevant for this task. We thank the PAAR 2012 referees for their thorough reviews and comments that helped to improve the final presentation of this paper.

References

[1] Jesse Alama, Tom Heskes, Daniel Kühlwein, Evgeni Tsivtsivadze, and Josef Urban. Premise Selection for Mathematics by Corpus Analysis and Kernel Methods. *CoRR*, abs/1108.3446, 2011.

[2] Gregory J. Chaitin. The Omega Number: Irreducible Complexity in Pure Math. In Jonathan M. Borwein and William M. Farmer, editors, *MKM*, volume 4108 of *Lecture Notes in Computer Science*, page 1. Springer, 2006.

[3] Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: An Efficient SMT Solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *TACAS*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.

[4] Krystof Hoder and Andrei Voronkov. Sine Qua Non for Large Theory Reasoning. In Nikolaj Bjørner and Viorica Sofronie-Stokkermans, editors, *CADE*, volume 6803 of *Lecture Notes in Computer Science*, pages 299–314. Springer, 2011.

[5] Daniel Kühlwein, Twan van Laarhoven, Evgeni Tsivtsivadze, Josef Urban, and Tom Heskes. Overview and Evaluation of Premise Selection Techniques for Large Theory Mathematics. In *IJCAR*, 2012. To appear.

[6] Huan Liu and Hiroshi Motoda. *Instance Selection and Construction for Data Mining*. Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[7] Alexandre Riazanov and Andrei Voronkov. The Design and Implementation of Vampire. *AI Commun.*, 15(2-3):91–110, 2002.

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

[9] Josef Urban, Krystof Hoder, and Andrei Voronkov. Evaluation of Automated Theorem Proving on the Mizar Mathematical Library. In *ICMS*, pages 155–166, 2010.