Overview and Evaluation of Premise Selection Techniques for Large Theory Mathematics

Daniel Kühlwein, Twan van Laarhoven, Evgeni Tsivtsivadze, Josef Urban and Tom Heskes*

Radboud University Nijmegen

Abstract. In this paper, an overview of state-of-the-art techniques for premise selection in large theory mathematics is provided, and new premise selection techniques are introduced. Several evaluation metrics are introduced, compared and their appropriateness is discussed in the context of automated reasoning in large theory mathematics. The methods are evaluated on the MPTP2078 benchmark, a subset of the Mizar library, and a 10% improvement is obtained over the best method so far.

1 Introduction: Formal Mathematics and its AI Methods

In recent years, more and more mathematics is becoming available in a computer-understandable form [12]. A number of large formalization projects are progressing [11,13,15], formal mathematics is considered by influential mathematicians,[1] and new approaches and proof assistants are discussed and developed by interested newcomers [3,8,22,39].

As this happens, the users and developers of formal mathematics are increasingly faced with the problem of searching for relevant formal knowledge, analogous to the search problems started since the early days of the Internet. Web search has led to a large body of research of robust and scalable non-semantic methods in fields like information retrieval, machine learning, and data mining. On the other hand, formal mathematics has been traditionally focusing on exhaustive and precise deductive search methods, typically used on small, carefully manually pre-arranged search space. In nutshell, the difference between the former and the latter methods is that the former focus on heuristically finding knowledge that could be most relevant for solving semantically underspecified (typically natural language) queries, while the latter methods try to find a precise answer and a proof for a conjecture that is expressed with full semantic precision. The former methods are largely *inductive* and *data-driven* [29]: the "solutions" are unconfirmed suggestions derived from heuristics and previous evidence, and essential parts of the algorithms are typically obtained by learning from large corpora. The latter methods have so far been largely *deductive* and *theory-driven*: the solutions are deduced in a logically correct way, and the

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

[1] http://gowers.wordpress.com/2008/07/28/more-quasi-automatic-theorem-proving

algorithms are to a large extent specified by their programmers without inducing major parts of the programs from large datasets.

There are a number of interesting ways how existing deductive methods can be improved in the presence of previous knowledge. Some of them are mentioned below. An early research done by the Munich group [10] has already produced a number of ideas and advanced implementations, like for example the pattern-based proof guidance in E prover [26]. It seems that the recently appeared large corpora of formal knowledge allow AI combinations of inductive and deductive thinking (e.g., the MaLARea metasystem [37]) that can hardly be tried in other AI domains that lack precise semantics and the notion of formal proof.

On the other hand, the first and lasting necessity obvious since 2003, when the first large formal mathematical corpora became available to Automated Theorem Provers (ATPs), has been good selection of relevant premises for a given new conjecture. 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]. Such large improvements provide an incentive for further research, evaluations, and benchmarks developing the field of knowledge-based automated reasoning.

This paper develops this field in several ways. First, in Section 2 an overview of state-of-the-art techniques for premise selection in large-theory mathematics is provided, focusing on premise ranking. In Section 3 we present several relevant machine learning metrics developed for feasible training and evaluation of ranking algorithms. The premise selection methods are evaluated on the MPTP2078 benchmark in section 4, using the machine learning metrics as well as several different ATPs. The learning and ATP evaluation methods are compared, and the relevance of the machine learning metrics based on human-proof data is discussed, together with the performance of different methods and ATPs. Based on the findings, use of ensemble methods for aggregating premise selectors is proposed and initially tested in Section 5, and shown to further raise the overall ATP performance by 10% in comparison to the best method so far. Section 6 concludes and proposes directions for further research.

2 Premise Selection Algorithms

2.1 Premise Selection Setting

The typical setting for the task of premise selection is a large developed library of formally encoded mathematical knowledge, over which mathematicians attempt to prove new lemmas and theorems [5, 32, 36]. The actual mathematical corpora suitable for ATP techniques are only a fraction of all mathematics (e.g. about 50000 lemmas and theorems in the Mizar library) and started to appear only recently, but they already provide a corpus on which different heuristic methods can be defined, trained, and evaluated. Premise selection can be useful as a standalone service for the formalizers (suggesting relevant lemmas), or in conjunction with ATP methods that can attempt to find a proof from the relevant premises.

2.2 Learning-based Ranking Algorithms

Learning-based ranking algorithms have a training and a testing phase and typically represent the data as points in pre-selected feature spaces. In the training phase the algorithm tries to fit one (or several) prediction functions to the data it is given. The result of the training is the best fitting prediction function which can then be used in the testing phase for evaluations.

In the typical setting presented above, the algorithms would train on all existing proofs in the library and be tested on the new theorem the mathematician wants to prove. We compare three different algorithms.

SNoW: SNoW (Sparse Network of Winnows) [6] is an implementation of (among others) the naive Bayes algorithm that has already been successfully used for premise selection (see e.g. [1, 32, 33]).

Naive Bayes is a statistical learning method based on Bayes' theorem with a strong (or naive) independence assumption. Given a new conjecture c and a premise p, SNoW computes the probability of p being needed to prove c, based on the previous use of p in proving conjectures that are similar to c. The similarity is in our case typically expressed using symbols and terms of the formulas. The independence assumption says that the (non-)occurrence of a symbol/term is not related to the (non-)occurrence of every other symbol/term.

MOR-CG: MOR-CG (Multi-Output Ranking Conjugate Gradient) is a kernel-based learning algorithm (see [29]) that is a new variation of our MOR algorithm described in [1]. The difference between MOR and MOR-CG are that MOR-CG uses a linear kernel instead of a Gaussian. Furthermore, MOR-CG uses conjugate-gradient descent to speed up the time needed for training.

Kernel-based algorithms do not aim to model probabilities, but instead try to minimize the expected loss of the prediction functions on the training data. For each premise p MOR-CG tries to find a function C_p such that for each conjecture c, $C_p(c) = 1$ iff p was used in the proof of c. Given a new conjecture c, we can evaluate the learned prediction functions C_p on c. The higher the value $C_p(c)$ the more relevant p is to prove c.

BiLi: BiLi (Bi-Linear) is a new algorithm that is based on a bilinear model of premise selection, similar to the work of Chu and Park [7]. Like MOR-CG, BiLi aims to minimize the expected loss. The difference lies in the kind of prediction functions they produce. In MOR-CG the prediction functions only take the features[2] of the conjecture into account. In BiLi, the prediction functions use the features of both the conjectures and the premises.[3] The bilinear model learns a weight for each combination of a conjecture feature together with a premise feature. Together, this weighted combination determines whether or not a premise is relevant to the conjecture.

[2] In our experiments each feature indicates the presence or absence of a certain symbol or term in a formula.

[3] This makes BiLi a bit closer to methods like SInE that symbolically compare conjectures with premises.

When the number of features becomes large, fitting a bilinear model becomes computationally more challenging. Therefore, in BiLi the number of features is first reduced to 100, using random projections [4]. To combat the noise introduced by these random projections, this procedure is repeated 20 times, and the averaged predictions are used for ranking the premises.

2.3 Other Algorithms Used in the Evaluation

SInE: The SInE (SUMO Inference Engine) is a heuristic state-of-the-art premise selection algorithm by Kryštof Hoder [14], recently also implemented in the E prover.[4] The basic idea is to use global frequencies of symbols in a problem to define their global *generality*, and build a relation linking each symbol S with all formulas F in which S is has the lowest global generality among the symbols of F. In common-sense ontologies, such formulas typically *define* the symbols linked to them, which is the reason for calling this relation a *D-relation*. Premise selection for a conjecture is then done by recursively following the D-relation, starting with the conjecture's symbols. For the experiments described here the E implementation of SInE has been used, because it can be instructed to select exactly N most relevant premises. This is compatible with the way how other premise rankers are used here, and it allows to compare the premise rankings produced by different algorithms for increasing values of N.[5]

Aprils: The Automated Prophesier of Relevance Incorporating Latent Semantics (APRILS) [24] is a signature-based premise selection method that employs Latent Semantic Analysis (LSA) [9] to define symbol and premise similarity. Latent semantics is a machine learning method that has been successfully used for example in the Netflix Prize,[6] and in web search. Its principle is to automatically derive "semantic" equivalence classes of words (like *car, vehicle, automobile*) from their co-occurrences in documents, and to work with such equivalence classes instead of the original words. In APRILS, formulas define the symbol co-occurrence, each formula is characterized as a vector over the symbols' equivalence classes, and the premise relevance is its dot product with the conjecture.

2.4 Techniques Not Included in the Evaluation

As a part of the overview, we also list important or interesting algorithms used for ATP knowledge selection that for various reasons do not fit the evaluation done here. Because of space contraints we refer readers to [34] for their discussion.

- The default premise selection heuristic used by the Isabelle/Sledgehammer export [17]. This is an Isabelle-specific symbol-based technique similar to SInE that would need to be evaluated on Isabelle data.

[4] http://www.mpi-inf.mpg.de/departments/rg1/conferences/deduction10/slides/stephan-schulz.pdf

[5] The exact parameters used for producing the E-SInE rankings are at https://raw.github.com/JUrban/MPTP2/master/MaLARea/script/filter1.

[6] http://www.netflixprize.com

- Goal directed ATP calculi including the *Conjecture Symbol Weight* clause selection heuristics in E prover [27] giving lower weights to symbols contained in the conjecture, the Set of Support (SoS) strategy in resolution/superposition provers, and tableau calculi like leanCoP [19] that are in practice goal-oriented.
- Model-based premise selection, as done by Pudlák's semantic axiom selection system for large theories [21], by the SRASS metasystem [30], and in a different setting by the MaLARea [37] metasystem.
- MaLARea [37] is a large-theory metasystem that loops between deductive proof and model finding (using ATPs and finite model finders), and learning premise-selection (currently using SNoW or MOR-CG) from the proofs and models to attack the conjectures that still remain to be proved.
- Abstract proof trace guidance implemented in the E prover by Stephan Schulz for his PhD [26]. Proofs are abstracted into clause patterns collected into a common knowledge base, which is loaded when a new problem is solved, and used for guiding clause selection. This is also similar to the *hints* technique in Prover9 [16].
- The MaLeCoP system [38] where the clause relevance is learned from all closed tableau branches, and the tableau extension steps are guided by a trained machine learner that takes as input features a suitable encoding of the literals on the current tableau branch.

3 Machine Learning Evaluation Metrics

Given a database of proofs, there are several possible ways to evaluate how good a premise selection algorithm is without running an ATP. Such evaluation metrics are used to estimate the best parameters (e.g. regularization, tolerance, step size) of an algorithm. We want to use the evaluation metrics that are the best indicators for the final ATP performance. The input for each metric is a ranking of the premises for a conjecture together with the information which premises where used to prove the conjecture (according to the training data).

Recall *Recall@n* is a value between 0 and 1 and denotes the fraction of used premises that are among the top n highest ranked premises.

$$\text{Recall@}n = \frac{|\{\text{used premises}\} \cap \{n \text{ highest ranked premises}\}|}{|\{\text{used premises}\}|}$$

Recall@n is always less than *Recall@*$(n+1)$. As n increases, *Recall@n* will eventually converge to 1. Our intuition is that the better the algorithm, the faster its *Recall@n* converges to 1.

AUC The AUC (Area under the ROC Curve) is the probability that, given a randomly drawn used premise and a randomly drawn unused premise, the used premise is ranked higher than the unused premise. Values closer to 1 show better performance.

Let $x_1, .., x_n$ be the ranks of the used premises and $y_1, .., y_m$ be the ranks of the unused premises. Then, the AUC is defined as

$$\text{AUC} = \frac{\sum_i^n \sum_j^m 1_{x_i > y_j}}{mn}$$

where $1_{x_i > y_j} = 1$ iff $x_i > y_j$ and zero otherwise.

100%Recall *100%Recall* denotes the minimum n such that Recall@$n = 1$.

$$100\%\text{Recall} = \min\{n \mid \text{Recall@}n = 1\}$$

In other words *100%Recall* tells us how many premises (starting from the highest ranked one) we need to give to the ATP to ensure that all necessary premises are included.

4 Evaluation

4.1 Evaluation Data

The premise selection methods are evaluated on the large (*chainy*) problems from the MPTP2078 benchmark[7] [1]. These are 2078 related large-theory problems (conjectures) and 4494 formulas (conjectures and premises) in total, extracted from the Mizar Mathematical Library (MML). The MPTP2078 benchmark was developed to supersede the older and smaller MPTP Challenge benchmark (developed in 2006), while keeping the number of problems manageable for experimenting. Larger evaluations are possible,[8] but not convenient when testing a large number of systems with many different settings. MPTP2078 seems sufficiently large to test various hypotheses and find significant differences.

MPTP2078 also contains (in the smaller, *bushy* problems) for each conjecture the information about the premises used in the MML proof. This can be used to train and evaluate machine learning algorithms using a chronological order emulating the growth of MML. For each conjecture, the algorithms are allowed to train on all MML proofs that were done up to that conjecture.[9] For each of the 2078 problems, the algorithms predict a ranking of the premises.

4.2 Machine Learning Evaluation – Comparison of Predictions with Known Proofs

We first compare the algorithms introduced in section 2 using the machine learning evaluation metrics introduced in section 3. All evaluations are based on the training data, the human-written formal proofs from the MML. They do not take the possibility of alternative proofs into account.[10]

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

[8] See [2, 35] for recent evaluations spanning the whole MML.

[9] This in particular means that the algorithms do not train on the data they were asked to predict.

[10] This could be improved in the future by adding alternative proofs, as discussed in section 6.

Recall Figure 1 compares the average Recall@n of MOR-CG, BiLi, SNoW, SInE and Aprils for the top 200 premises over all 2078 problems. Higher values denote better performance. The graph shows that MOR-CG performs best, and Aprils worst. Note that here is a sharp distinction between the learning algorithms,

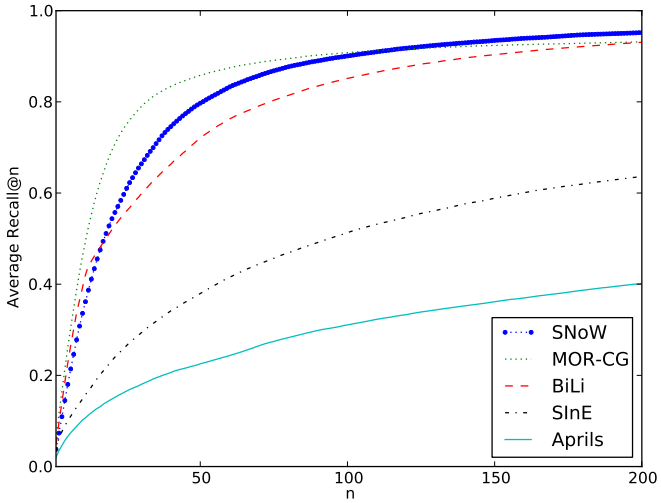

Fig. 1: Recall comparison of the premise selection algorithms

which use the MML proofs and eventually reach a very similar recall, and the heuristic-based algorithms Aprils and SInE.

AUC The average AUC of the premise selection algorithms is reported in table 1. Higher values mean better performance, i.e. a higher chance that a used premise is higher ranked than a unused premise. SNoW (97%) and BiLi (96%) have the best AUC scores with MOR-CG taking the third spot with an AUC of 88%. Aprils and SInE are considerably worse with 64% and 42% respectively. The standard deviation is very low with around 2% for all algorithms.

Table 1: AUC comparison of the premise selection algorithms

Algorithm	Avg. AUC	Std.
SNoW	0.9713	0.0216
BiLi	0.9615	0.0215
MOR-CG	0.8806	0.0206
Aprils	0.6443	0.0176
SInE	0.4212	0.0142

100%Recall The comparison of the 100%Recall measure values can be seen in figure 2. For the first 115 premises, MOR-CG is the best algorithm. From then on, MOR-CG hardly increases and SNoW takes the lead. Eventually, BiLi almost catches up with MOR-CG. Again we can see a big gap between the performance of the learning and the heuristic algorithms with SInE and Aprils not even reaching 400 problems with 100%Recall.

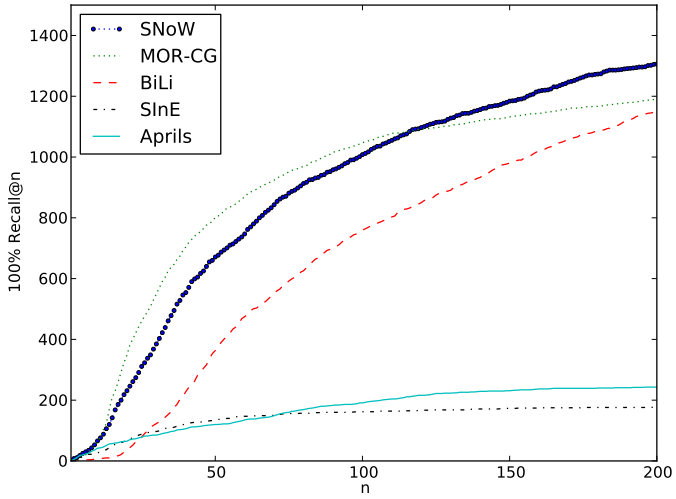

Fig. 2: 100%Recall comparison of the premise selection algorithms

Discussion In all three evaluation metrics there is a clear difference between the performance of the learning-based algorithms SNoW, MOR-CG and BiLi and the heuristic-based algorithms SInE and Aprils. If the machine-learning metrics on the MML proofs are a good indicator for the ATP performance then there should be a corresponding performance difference in the number of problems solved. We investigate this in the following section.

4.3 ATP Evaluation

Vampire In the first experiment we combined the rankings obtained from the algorithms introduced in section 2 with version 0.6 of the ATP Vampire[11] [23]. For each MPTP2078 problem (containing on average 1976.5 premises), we created 20 new problems, containing the $10, 20, ..., 200$ highest ranked premises. The results can be seen in figure 3.

[11] All ATPs are 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. We use Vampire as our default ATP because of its good preformance in the CASC competitions, and because of its good performance on MML reported in [35].

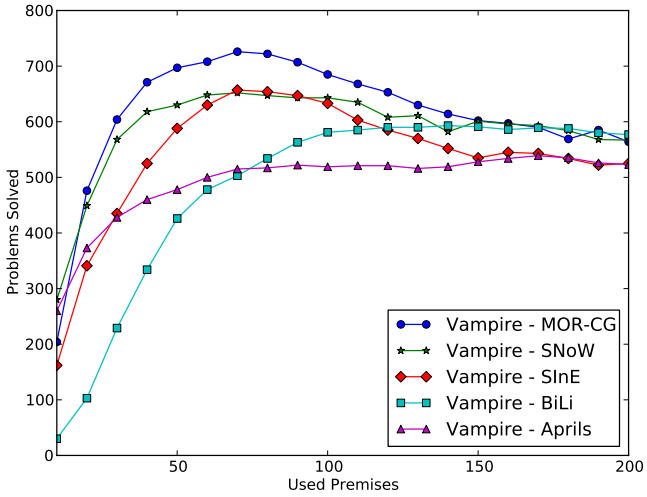

Fig. 3: Problems solved – Vampire

Apart from the first 10-premise batch and the three last batches, MOR-CG always solves the highest number of problems with a maximum of 726 problems with the top 70 premises. SNoW solves less problems in the beginning, but catches up in the end. BiLi solves very few problems in the beginning, but gets better as more premises are given and eventually is as good as SNoW and MOR-CG. The surprising fact (given the machine learning performance) is that SInE performs very well, on par with SNoW in the range of 60-100 premises. This indicates that SInE finds proofs that are very different from the human proofs. Furthermore, it is worth noting that most algorithms have their peak at around 70-80 premises. It seems that after that, the effect of increased premise recall is beaten by the effect of the growing ATP search space.

E, SPASS and Z3 We also compared the top three algorithms, MOR-CG, SNoW and SInE, with three other ATPs: E [27] (version 1.4), SPASS [40] (version 3.7) and Z3 [18] (version 3.2). The results can be seen in figure 4, 5, 6 respectively. In all three experiments, MOR-CG gives the best results. Looking at the number of problems solved by E we see that SNoW and SInE solve about the same number of problems when more than 50 premises are given. In the SPASS evaluation, SInE performs better than SNoW after the initial 60 premises. The results for Z3 are clearer, with (apart from the first run with the top 10 premises) MOR-CG always solving more problems than SNoW, and SNoW solving more problems than SInE. It is worth noting that independent of the learning algorithm, SPASS solves the fewest problems and Z3 the most, and that (at least up to the limit of 200 premises used) Z3 is hardly affected by having too many premises in the problems.

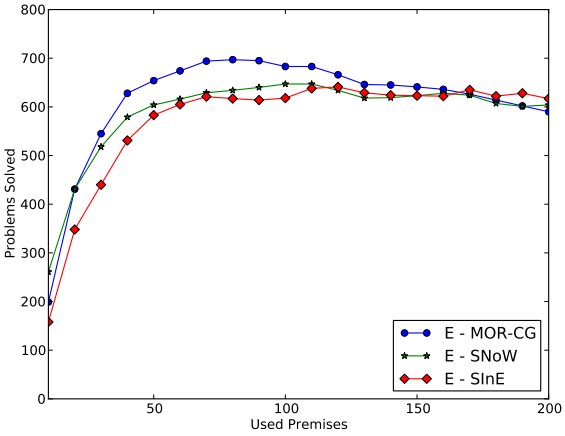

Fig. 4: Problems solved – E

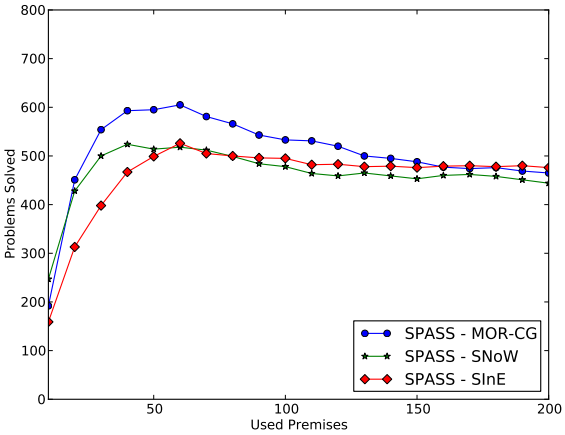

Fig. 5: Problems solved – SPASS

Discussion The ATP evaluation shows that a good ML evaluation performance does not necessarily imply a good ATP performance and vice versa. E.g. SInE performs better than expected, and BiLi worse. A plausible explanation for this is that the human-written proofs that are the basis of the learning algorithms are not the best possible guidelines for ATP proofs, because there are a number of good alternative proofs: the total number of problems proved with Vampire by the union of all prediction methods is 1197, which is more (in 5s) than the 1105 problems that Vampire can prove in 10s when using only the premises used exactly in the human-written proofs. One possible way how to test this hypothesis (to a certain extent at least) would be to train the learning algorithms

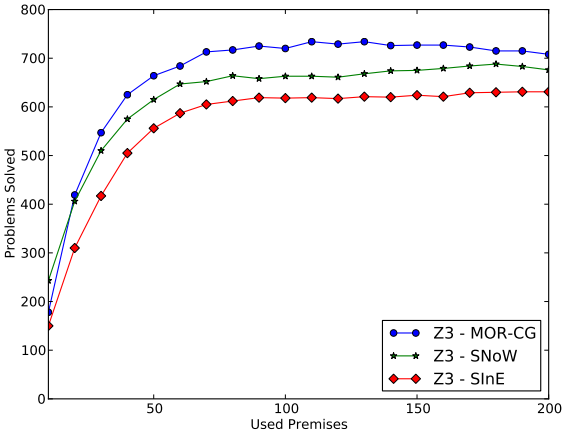

Fig. 6: Problems solved – Z3

on all the ATP proofs that are found, and test whether the ML evaluation performance closer correlates with the ATP evaluation performance.

The most successful 10s combination, solving 939 problems, is to run Z3 with the 130 best premises selected by MOR-CG, together with Vampire using the 70 best premises selected by SInE. It is also worth noting that when we consider all provers and all methods, 1415 problems can be solved.

It seems the heuristic and the learning based premise selection methods give rise to different proofs. In the next section, we try to exploit this by considering combinations of ranking algorithms.

5 Combining Premise Rankers

There is clear evidence about alternative proofs being feasible from alternative predictions. This should not be too surprising, because the premises are organized into a large derivation graph, and there are many explicit (and also quite likely many yet-undiscovered) semantic dependencies among them.

The evaluated premise selection algorithms are based on different ideas of similarity, relevance, and functional approximation spaces and norms in them. This also means that they can be better or worse in capturing different aspects of the premise selection problem (whose optimal solution is obviously undecidable in general, and intractable even if we impose some finiteness limits).

An interesting machine learning technique to try in this setting is the combination of different predictors. There has been a large amount of machine learning research in this area, done under different names. *Ensembles* is one of the most frequent, a recent overview of ensemble based systems is given in [20], while for example [28] deals with the specific task of aggregating rankers.

As a final experiment that opens the premise selection field to the application of advanced ranking-aggregation methods, we have performed an initial simple

evaluation of combining two very different premise ranking methods: MOR-CG and SInE. The aggregation is done by simple weighted linear combination, i.e., the final ranking is obtained via weighted linear combination of the predicted individual rankings. We test a limited grid of weights, in the interval of $[0, 1]$ with a step value of 0.25, i.e., apart from the original MOR-CG and SInE rankings we get three more weighted aggregate rankings as follows: $0.25 * \mathrm{CG} + 0.75 * \mathrm{SInE}$, $0.5 * \mathrm{CG} + 0.5 * \mathrm{SInE}$, and $0.75 * \mathrm{CG} + 0.25 * \mathrm{SInE}$. The following Figure 7 shows their ATP evaluation.

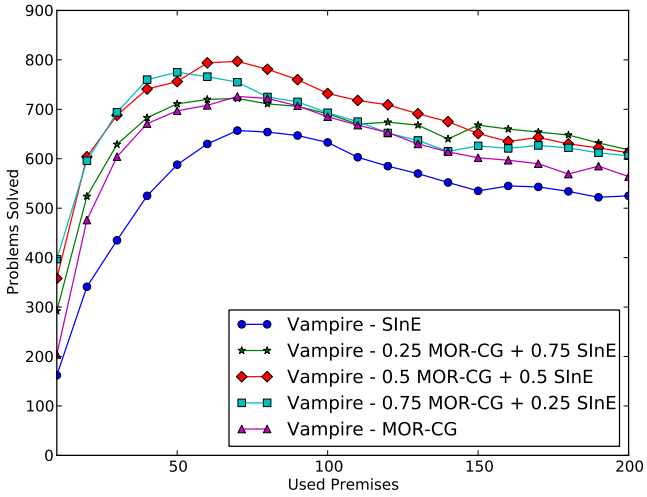

Fig. 7: Combining CG and SInE: Problems solved

The machine learning evaluation (done as before against the data extracted from the human proofs) is not surprising, and the graphs (which we omit due to space constraints) look like linear combinations of the corresponding figures for MOR-CG and SInE. The ATP evaluation (only Vampire was used) is a very different case. For example the equally weighted combination of MOR-CG and SInE solves over 604 problems when using only the top 20 ranked premises. The corresponding values for standalone MOR-CG resp. SInE are 476, resp. 341, i.e., they are improved by 27%, resp. 77%. The equally weighted combination solves 797 when using the top 70 premises, which is a 10% improvement over the best result of all methods (726 problems solved by MOR-CG when using the top 70 premises). Note that unlike the external combination mentioned above, this is done only in 5 seconds, with only one ATP, one premise selector, and one threshold.

6 Conclusion and Future Work

Heuristic and inductive methods seem indispensable for strong automated reasoning in large formal mathematics, and significant improvements can be achieved by their proper design, use and combination with precise deductive methods.

Knowing previous proofs and learning from them turns out to be important not just to mathematicians, but also for automated reasoning in large theories.

The possibility of the ultimate semantic (ATP) evaluation of the proposed premise rankings adds interesting "combined AI" aspects to the standard machine learning methods. Without expressive semantics the methods just try to predict the human proofs as closely as possible. Proposing alternative (and sometimes simpler) proofs is discouraged by the standard machine learning evaluation metrics. This produces interesting questions to AI researchers: given the explicit derivation graph of a large theory, and the precise semantics allowing this graph to grow further, what are good methods and metrics for (reasonably fast) training of premise selection methods? One pragmatic answer that we can give is to develop a growing database of ATP and human proofs, and other results (like counter-models), e.g., in a similar way as in the MaLARea metasystem, and use this growing database for training instead of just the human proofs, testing (and caching for further use) the ATP validity of new predictions on-demand.

We have evaluated practically all reasonably fast state-of-the-art premise selection techniques, tried some new ones, and currently experiment with more. This has produced a large amount of data on the most suitable (most orthogonal) combinations of premise selection systems, numbers of premises used, ATPs used, and ATP (currently E prover) strategies used. We further use machine learning in the spirit of the E-MaLeS system to determine optimal (either parallel or lower time-limit) combinations of these. These results are not included here due to space constraints.[12]

There is a trade-off between the precision and speed of the methods, and an interesting future work is to use fast methods as pre-selectors for more expensive methods. This is related to the problems of automated clustering the large theories, that can also be useful by itself for organizing and searching the large formal repositories. Clustering on a finer level is also one of the methods that could be used to further improve premise selection. It is quite likely that there are clusters of theorems that have the same logic power (their conjunctions are equal in the Lindenbaum algebra), and about the same strength when used with ATPs (the same conjecture can be proved from them in a similar number of steps). The current premise selection methods will likely recommend all such equivalent sets, which is blocking other (possibly necessary) premises, so heuristic identification of such (nearly) equivalent sets seems important. Including more semantics (for example evaluation in an evolving set of models as in MaLARea) in the learning and selection process could be one way how to achieve this.

We would like to make our strongest methods useful to as many formal mathematicians as possible. Some of them (like SNoW) have been used for MML and MPTP since 2003, but algorithms like MOR-CG and aggregated rankers are not deployed yet. We also hope to evaluate and deploy the algorithms at least for the Isabelle/Sledgehammer framework in near future.

[12] The fact that Z3 solves largely orthogonal sets of problems to Vampire is probably well known by now. Hence our focus on the differences between the premise selection methods.

References

1. Alama, J., Heskes, T., Kühlwein, D., Tsivtsivadze, E., Urban, J.: Premise Selection for Mathematics by Corpus Analysis and Kernel Methods. CoRR abs/1108.3446 (2011)
2. Alama, J., Kühlwein, D., Urban, J.: Automated and Human Proofs in General Mathematics: An Initial Comparison. In: Bjørner, N., Voronkov, A. (eds.) LPAR. Lecture Notes in Computer Science, vol. 7180, pp. 37–45. Springer (2012)
3. Bem, J.: The Zermelo proof checker, http://zermelo.org/
4. Bingham, E., Mannila, H.: Random projection in dimensionality reduction: applications to image and text data. In: KDD. pp. 245–250 (2001)
5. Blanchette, J.C., Bulwahn, L., Nipkow, T.: Automatic proof and disproof in Isabelle/HOL. In: Tinelli, C., Sofronie-Stokkermans, V. (eds.) FroCos. Lecture Notes in Computer Science, vol. 6989, pp. 12–27. Springer (2011)
6. Carlson, A., Cumby, C., Rosen, J., Roth, D.: The SNoW Learning Architecture. Tech. Rep. UIUCDCS-R-99-2101, UIUC Computer Science Department (5 1999), http://cogcomp.cs.illinois.edu/papers/CCRR99.pdf
7. Chu, W., Park, S.T.: Personalized recommendation on dynamic content using predictive bilinear models. In: Quemada, J., León, G., Maarek, Y.S., Nejdl, W. (eds.) WWW. pp. 691–700. ACM (2009)
8. Cramer, M., Fisseni, B., Koepke, P., Kühlwein, D., Schröder, B., Veldman, J.: The Naproche Project: Controlled Natural Language Proof Checking of Mathematical Texts. In: Fuchs, N.E. (ed.) CNL. Lecture Notes in Computer Science, vol. 5972, pp. 170–186. Springer (2009)
9. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by Latent Semantic Analysis. JASIS 41(6), 391–407 (1990)
10. Denzinger, J., Fuchs, M., Goller, C., Schulz, S.: Learning from Previous Proof Experience. Technical Report AR99-4, Institut für Informatik, Technische Universität München (1999)
11. Gonthier, G.: Advances in the Formalization of the Odd Order Theorem. In: van Eekelen, M.C.J.D., Geuvers, H., Schmaltz, J., Wiedijk, F. (eds.) ITP. LNCS, vol. 6898, p. 2. Springer (2011)
12. Hales, T.C.: Mathematics in the age of the Turing machine. Lecture Notes in Logic (2012), to appear; http://www.math.pitt.edu/~thales/papers/turing.pdf
13. Hales, T.C., Harrison, J., McLaughlin, S., Nipkow, T., Obua, S., Zumkeller, R.: A revision of the proof of the Kepler Conjecture. Discrete & Computational Geometry 44(1), 1–34 (2010)
14. Hoder, K., Voronkov, A.: Sine qua non for large theory reasoning. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) CADE. Lecture Notes in Computer Science, vol. 6803, pp. 299–314. Springer (2011)
15. Klein, G., Andronick, J., Elphinstone, K., Heiser, G., Cock, D., Derrin, P., Elkaduwe, D., Engelhardt, K., Kolanski, R., Norrish, M., Sewell, T., Tuch, H., Winwood, S.: seL4: formal verification of an operating-system kernel. Commun. ACM 53(6), 107–115 (2010)
16. McCune, W.: Prover9 and Mace4 (2005–2010), http://www.cs.unm.edu/~mccune/prover9/
17. Meng, J., Paulson, L.C.: Lightweight relevance filtering for machine-generated resolution problems. J. Applied Logic 7(1), 41–57 (2009)
18. de Moura, L.M., Bjørner, N.: Z3: An Efficient SMT Solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS. Lecture Notes in Computer Science, vol. 4963, pp. 337–340. Springer (2008)

19. Otten, J., Bibel, W.: leanCoP: lean connection-based theorem proving. J. Symb. Comput. 36(1-2), 139–161 (2003)
20. Polikar, R.: Ensemble based systems in decision making. Circuits and Systems Magazine, IEEE 6(3), 21–45 (2006)
21. Pudlak, P.: Semantic Selection of Premisses for Automated Theorem Proving. In: Sutcliffe et al. [31]
22. Reichelt, S.: The HLM proof assistant, http://hlm.sourceforge.net/
23. Riazanov, A., Voronkov, A.: The design and implementation of VAMPIRE. AI Commun. 15(2-3), 91–110 (2002)
24. Roederer, A., Puzis, Y., Sutcliffe, G.: Divvy: An ATP Meta-system Based on Axiom Relevance Ordering. In: Schmidt [25], pp. 157–162
25. Schmidt, R.A. (ed.): Automated Deduction - CADE-22, 22nd International Conference on Automated Deduction, Montreal, Canada, August 2-7, 2009. Proceedings, Lecture Notes in Computer Science, vol. 5663. Springer (2009)
26. Schulz, S.: Learning search control knowledge for equational deduction, DISKI, vol. 230. Infix Akademische Verlagsgesellschaft (2000)
27. Schulz, S.: E - A Brainiac Theorem Prover. AI Commun. 15(2-3), 111–126 (2002)
28. Sculley, D.: Rank aggregation for similar items. In: SDM. SIAM (2007)
29. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, New York, NY, USA (2004)
30. Sutcliffe, G., Puzis, Y.: SRASS - a semantic relevance axiom selection system. In: Pfenning, F. (ed.) CADE. Lecture Notes in Computer Science, vol. 4603, pp. 295–310. Springer (2007)
31. Sutcliffe, G., Urban, J., Schulz, S. (eds.): Proceedings of the CADE-21 Workshop on Empirically Successful Automated Reasoning in Large Theories, Bremen, Germany, 17th July 2007, CEUR Workshop Proceedings, vol. 257. CEUR-WS.org (2007)
32. Urban, J.: MizarMode—an integrated proof assistance tool for the Mizar way of formalizing mathematics. Journal of Applied Logic 4(4), 414 – 427 (2006)
33. Urban, J.: MaLARea: a metasystem for automated reasoning in large theories. In: Sutcliffe et al. [31]
34. Urban, J.: An Overview of Methods for Large-Theory Automated Theorem Proving (Invited Paper). In: Höfner, P., McIver, A., Struth, G. (eds.) ATE Workshop. CEUR Workshop Proceedings, vol. 760, pp. 3–8. CEUR-WS.org (2011)
35. Urban, J., Hoder, K., Voronkov, A.: Evaluation of automated theorem proving on the Mizar Mathematical Library. In: ICMS. pp. 155–166 (2010)
36. Urban, J., Rudnicki, P., Sutcliffe, G.: ATP and presentation service for Mizar formalizations. CoRR abs/1109.0616 (2011)
37. Urban, J., Sutcliffe, G., Pudlák, P., Vyskocil, J.: MaLARea SG1- Machine Learner for Automated Reasoning with Semantic Guidance. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR. Lecture Notes in Computer Science, vol. 5195, pp. 441–456. Springer (2008)
38. Urban, J., Vyskocil, J., Stepánek, P.: MaLeCoP Machine Learning Connection Prover. In: Brünnler, K., Metcalfe, G. (eds.) TABLEAUX. Lecture Notes in Computer Science, vol. 6793, pp. 263–277. Springer (2011)
39. Voevodsky, V.: Univalent foundations project (2010), manuscript available from http://www.math.ias.edu/~vladimir/Site3/Univalent_Foundations.html
40. Weidenbach, C., Dimova, D., Fietzke, A., Kumar, R., Suda, M., Wischnewski, P.: SPASS Version 3.5. In: Schmidt [25], pp. 140–145