Noname manuscript No.
(will be inserted by the editor)

MizAR 40 for Mizar 40

Cezary Kaliszyk · Josef Urban

the date of receipt and acceptance should be inserted later

Abstract As a present to Mizar on its 40th anniversary, we develop an AI/ATP system that in 30 seconds of real time on a 14-CPU machine automatically proves 40% of the theorems in the latest official version of the Mizar Mathematical Library (MML). This is a considerable improvement over previous performance of large-theory AI/ATP methods measured on the whole MML. To achieve that, a large suite of AI/ATP methods is employed and further developed. We implement the most useful methods efficiently, to scale them to the 150000 formulas in MML. This reduces the training times over the corpus to 1–3 seconds, allowing a simple practical deployment of the methods in the online automated reasoning service for the Mizar users (Miz$\mathbb{AR}$).

1 Introduction and Motivation

Since 2003 the Mizar Mathematical Library[1] (MML) has been used as a repository for developing AI/ATP methods for solving formally stated (computer-understandable) conjectures in general large-theory mathematics [28–30]. By *large theories* we mean theories with many concepts, definitions, theorems and lemmas, over which many related conjectures are posed, and where it is not immediately clear which of the previous facts will (not) be useful for a proof of a conjecture. The number and strength of the methods developed has been growing, however the methods were often developed and evaluated on smaller benchmarks such as the MPTP Challenge[2] and MPTP2078 [1]. Recently, we have tried to develop a strong suite of AI/ATP methods that scale to the whole June 2012 version of the Flyspeck [7, 8] development [14, 15], containing more than 14000 theorems. The best methods

Cezary Kaliszyk
University of Innsbruck, Austria

Josef Urban, funded by NWO grant *Knowledge-based Automated Reasoning*
Radboud University, Nijmegen

[1] `http://www.mizar.org`
[2] `http://www.cs.miami.edu/~tptp/MPTPChallenge/`

using the accumulated data have been recently deployed in an online ("cloud-based") AI/ATP service for HOL Light [9] formalizations [12, 16]. When running the 14 strongest methods in parallel, 47% of the Flyspeck theorems can be proved in 30 seconds without any user interaction. To a significant extent, this performance is achieved by learning from the large number of previous proofs various high-level [22, 32] and low-level [14, 33] guiding methods for state-of-the-art ATP and SMT systems such as Vampire [19], E [25] and Z3 [3]. A similar work has been recently undertaken with Isabelle [21].

We believe that this performance is a milestone on the way to John McCarthy's AI and QED dream of "Heavy Duty Set Theory", i.e., a sufficiently smart AI/ATP/ITP system that can without forcing mathematicians to struggle with various current technologies of explicit "proof programming" automatically understand and check reasoning steps done on the level that is commonly used in mathematical proofs. If such a system is developed, the current barrier preventing computer understanding of common mathematical proofs will to a large extent disappear, and mathematics (and thus all human exact thinking) may enter an era of ubiquitous computer understanding and strong AI assistance.

In this work we employ, further develop, and evaluate a suite of scalable AI/ATP methods on the whole Mizar library, containing nearly 58000 theorems. The main experimental result (Section 3) is that the 14 strongest methods run in parallel for 30 seconds prove 40.6% of the 58000 Mizar theorems without any user interaction. If users are also allowed to manually select the relevant premises used then by the ATPs, the performance grows to 56.2%. Our hope is that this performance may significantly lower the barrier to formalizing mathematics in the Mizar system [6], which has a long history of targeting mathematicians with its standard logical foundations, intuitive proof style, and linguistic closeness to mathematical vernacular. The various methods used to achieve this performance are described in Section 2. Section 3 discusses the experiments and results obtained with the methods, Section 4 takes a brief look at the data obtained from the automatically found proofs, Section 5 briefly describes the first integration of the methods in the Miz$\mathbb{AR}$ online service [34], and Section 6 discusses future work and concludes.

2 Learning Proof Guidance From the MML

The general idea behind the large-theory ATP-for-ITP systems that started to be developed in the last decade is to combine (i) translations between the ATP and ITP formalisms with (ii) high-level premise selection methods [1] and (iii) state-of-the-art ATP systems which can be further strengthened and tuned in various ways for the large-theory setting. For the translation from the Mizar logic to TPTP we re-use the existing MPTP translation [29, 30]. After several initial experiments with various ATPs and their versions, we have decided to limit the set of ATPs to Vampire 3.0, Z3 4.0, and E 1.8 run using the Epar scheduler and strategies [33]. This combination also worked well for the experiments with Flyspeck. In this work, the main focus is on (ii), i.e., on deploying and improving for MML the suite of scalable high-level premise selection methods which we have recently developed for the whole Flyspeck corpus, containing over 20000 formulas. Here *premise selection* is the task of choosing a subset of all available facts (premises), which is most likely to lead to a successful automated deduction proof of a given conjecture.

The currently strongest premise selection methods for large-theory mathematics are *data-driven* [26]: instead of explicit programming of all aspects of knowledge selection, data-driven methods extract (learn) significant parts of such complicated algorithms from the existing large libraries of solutions (proofs). This shift from explicit (*theory-driven*) programming of AI heuristics to learning AI algorithms from data is to a large extent responsible for the recent successes in AI domains such as web search, consumer choice prediction, autonomous car driving, etc. But this also means that extracting good training data from the MML is equally important as the methods that learn premise selection on such data. Interestingly, in large-theory ATP there is a full positive feedback loop [31] between the amount/quality of the data and the strength of the methods: not only more/better data produce stronger methods (which is the standard data-driven argument), but also stronger proving methods produce more/better data in the form of proofs. This is quite a unique property of this very expressive and fully semantic AI domain, born quite recently thanks to the development on large formal mathematical libraries such as the MML [36]. The main body of our work thus consists of the following steps:

1. Obtaining from the MML suitable data (proof dependencies) on which premise selection methods can be trained.
2. Developing, training and testing such premise selection methods and their parameters on a small random subset of the MML.
3. Iterating steps (1) and (2), i.e., using the most successful methods to get more proofs, and training further methods on them.

2.1 Obtaining Proof Dependencies from the MML

There are 57897 Mizar theorems and unnamed toplevel lemmas in the most recent official MML 4.181.1147. This set is canonically (chronologically) ordered by the MML order of articles, and by the (chronological) order of theorems in the articles. This ordering also applies to the about 90000 other Mizar formulas (typically encoding the type system and other automations known to Mizar) used in the problems. Our goal is to prove automatically as many of the 57897 theorems as possible, using at each point all the available formulas and all information about previous theorems and their proofs.

The human-written Mizar proofs contain explicit information about the theorems and definitions used. This information is however incomplete. For re-playing the Mizar proofs with ATPs, a lot of "background" knowledge (typically about typing of terms) needs to be explicitly added. The MPTP system adds such background formulas heuristically in a fixpoint manner, by looking at the set of symbols in the problem and adding the appropriate typing formulas. The average size of an ATP problem constructed in this way by MPTP is 328 formulas, while the average number of the explicit Mizar proof references is only 12. In [1] we have constructed a computationally expensive method (using the Mizar checker) that reduces the number of the background formulas 2–3 times. However, the measured performance gain from that method when re-proving with ATPs the MPTP2078 problems was only about 4%, and in [2] it was found that the ATPs still typically do not use many of the Mizar-needed background formulas. Since learning from the minimized ATP proofs is typically superior [15, 22], in the current work we decided to skip the expensive Mizar-based proof minimization, and focus on using the ATP proofs of the heuristically constructed re-proving problems coming from the Mizar theorems. This decision was influenced by older preliminary 20-second testing using an Intel Xeon 2.67 GHz server, in which Vampire 1.8 proved 20302 of the MPTP-constructed theorem problems, Epar proved 20324, and together they proved 23141, i.e., 40% of the theorems.

Table 1 shows the gradual growth of the set of ATP-computed proof dependencies that we mainly use for learning. The first set is obtained by running Vampire 3.0 for 300 seconds[3] on the MPTP-constructed re-proving problems. The additional 1677 solutions in the second set are obtained by learning premise selection on the first set, and running ATPs for 120 seconds again on various most relevant slices of the re-proving problems (we always include the explicit Mizar references in such pruned ATP problems). The following passes no longer prune the MPTP-constructed re-proving problems. They just use the premise selectors trained on the previous passes to suggest the most relevant premises, regardless of the original Mizar proofs, and re-learn from the newly obtained proofs in several iterations as in the MaLARea system [31, 35]. Such iterations are also quite expensive to compute for a new development. However, we have shown in [16] that this information can be in large libraries efficiently re-used and does not need to be computed for every version again. As in [15], the difference to the original MaLARea iterations is that at each premise-selection point only the chronologically previous proofs are used for learning. This corresponds to the ultimate deployment scenario, when always only the library proofs written so far are known.

Table 1: Improving the dependency data used for training premise selection

Pass	ATPs	Premise Selection	Theorems (%)	Dependencies
1	V 300s	MPTP re-proving	27842 (48%)	27842
2	V, E, Z3 120s	trained on (1), premises limited by (1)	29519 (51%)	30024
3	V, E 120s	trained on (2), premises unlimited	30889 (53.4%)	31720
4	V, E 120s	trained on (3), premises unlimited	31599 (54.6%)	32976
5	V, E 120s	trained on (4), premises unlimited	32010 (55.3%)	35870
6	V, E 120s	trained on (5), premises unlimited	32165 (55.6%)	36122

In total, these iterations yield 32165 ATP proofs, and with the final evaluation described in Section 3.2 this number reaches 32557 theorems. This means that when using either human or AI-based premise selection and their combinations, state-of-the-art ATPs are today able to prove 56.23% of the toplevel MML theorems. This is a very good motivation for developing good premise-selection methods.

[3] The time limit of 300 seconds has worked well in the previous experiments done over Flyspeck [15]. Increasing the time limit further does not help significantly and it costs a lot of resources.

2.2 Premise Selection Techniques

The premise selection techniques we start with, are the relatively fast scalable methods used for Flyspeck in [15]: Naive Bayes (nb) and distance-weighted k-nearest neighbor [5] (knn). In particular, a family of differently parametrized k-NNs together with the IDF (inverse document frequency) feature weighting scheme [11] have recently provided quite significant performance improvement [14]. This is here extended to naive Bayes (nb_idf). We are interested both in the strongest possible methods, and also in methods that can be quite weak, but complement well the stronger methods.

Apart from minor implementational modifications, we characterize each formula with the syntactic features used by MaLARea: symbols, terms and subterms of the formula. In the most successful methods, all variables in such features are renamed to just one variable A0 (widening the similarity relation), however to a smaller extent, also the features with original variables are useful. Following the recent successful use by MaLARea 0.5 in the 2013 CASC LTB competition [17] we also add a version of distance-weighted k-NN using the Latent Semantic Indexing [4] (LSI) preprocessing of the feature space done efficiently by the gensim [23] toolkit. We test the LSI preprocessing with 800, 3200, and 6400 topics (lsi_800 .. lsi_6400), and also versions with and without the TF-IDF feature scaling (e.g., (lsi_3200ti)).

The next modification of k-NN are various recursive schemes for weighting the neighbors' dependencies. The geo_1_F version stops the dependency recursion at the first level, weighting each dependence of a neighbor N by the factor $F * \mathit{distance}(N)$ (where $F \in (0,1)$), and taking maximum (instead of sum) of such weights over all neighbors. The geo_r_F version does full dependency recursion, weighting the indirect dependencies by $F^{recursionlevel} * \mathit{distance}(N)$, and again taking maximum over all such factors.

In [22] a linear combination of the strongest learning method with the SInE [10] heuristic produced very good results. This is an instance of *ensemble learning* where multiple base methods are combined into stronger classifiers. We heuristically explore combinations of the various base methods using various weighting schemes. In addition to the linear combination, we try geometric, harmonic, and quadratic average, and also use minimum and maximum of ranks. In particular taking the minimal rank (comb_min) and the geometric average (comb_geo) of ranks (computed as additions of logarithms) turn out to be quite successful. This can be explained in various ways, for example, taking the geometric average is the correct way of averaging ratios. Since ATPs are very (probably exponentially) sensitive to the number of axioms, treating the particular aggregated rankings as ratios is quite likely fitting to our domain (e.g., the ratio between 50th and 60th premise is 1.2, while the ratio between 10th and 20th premise is 2, whereas the linear distance is the same in the two cases).

We also try several methods of *boosting* [24]: using for training of the next method only those proof dependencies that are badly predicted by the previous methods. While we believe that there are good reasons why this approach should help (e.g., our current methods being quite simple and thus hard to fit to more complicated ideas), so far this has not provided significant improvements.

All the tested methods[4] (apart from LSI (gensim) and SInE (E prover)) are now uniformly implemented in OCaml, which gives significant speedup over the initial Perl implementation on the large number of features, labels and examples used when training over the whole MML (the number of features reaches several hundred thousand). The most useful methods are further implemented in C++, making them about twice as fast as their OCaml version. A particularly useful low-level optimization is the use of partial sorting (based on heapsort) of the scores according to the number of premises demanded from a particular premise-selection method. For example, if only 128 premises are needed, the partial sorting is much more efficient than full sorting of the whole array of 150000 MML formulas.

2.3 ATPs and Their Low-Level Guidance

No particular development of ATP strategies was done for this work on the whole MML. However, thanks to the recent CASC competitions containing Mizar divisions (Mizar@Turing12, CASC LTB 2013) the recent versions of Vampire and Epar seem to be tuned well for MPTP2078. In particular, a set of strong strategies for E has been automatically developed by BliStr [33] in 2012 on the 1000 Mizar@Turing12 problems, raising the performance on MPTP2078 over E's auto-mode by 25%. A second round of such strategy evolution on these 1000 problems was done for MaLARea 0.5 in CASC LTB 2013, where additionally a number of strong SInE strategies were evolved. Vampire 3.0 is on MML 16% stronger than Vampire 2.6.

3 Experiments and Results

3.1 Experiments and Results on a Random Subset of 1930 Problems

Most of the experimental research was done on a random subset of MML consisting of 1930 theorems (more precisely, every 30th theorem was used). For each of these theorems, the premise selection methods were trained on all the preceding proof data (Section 2.1), and chosen numbers (32, 64, 96, 128, 256, 512 and 1024) of the best-ranked premises were given to the ATPs. Most of the experiments during the development of the premise selection methods were done with Vampire 3.0. The final experiments were extended to Epar and Z3. As in [15], the systems were run with a 30 second time limit on a 48-core server with AMD Opteron 6174 2.2 GHz CPUs, 320 GB RAM, and 0.5 MB L2 cache per CPU. In real time, each evaluation thus took 2–3 hours for one ATP. In total, 70 different premise-selection methods have been tried on the random sample, see our web page for a detailed listing.[5]

Table 2 shows the fourteen methods and their parameters that performed best on the 1930-subset. The Σ-SOTAC (State of the art contribution) is the sum of a system's SOTAC over all problems attempted, where for each problem solved by a system, its SOTAC for the problem is the inverse of the number of systems in our evaluation that solved this problem. This metric shows how useful a particular method is in a collection of other methods (in this case all the 70 methods that have been tried).

[4] For their details see `http://cl-informatik.uibk.ac.at/users/cek/mizAR/legend.txt`

[5] `http://cl-informatik.uibk.ac.at/users/cek/mizAR/5yp.html`

Table 2: 14 best premise selection methods on the 1930-subset

Method	Parameters	Premises	ATP	Theorem (%)	Σ-SOTAC
comb	min_2k_20_20	128	Epar	550 (28.50)	2.41
comb	geo_3k_50_00	96	V	544 (28.19)	1.94
lsi	3200ti_8_80	128	Epar	537 (27.82)	2.17
comb	geo_1k_33_33	96	V	535 (27.72)	1.60
comb	geo_10k_33_33	96	V	533 (27.62)	1.73
comb	geo_3k_20_20	96	V	533 (27.62)	1.71
comb	geo_3k_33_33	96	V	533 (27.62)	1.63
comb	geo_3k_25_25_25	96	V	532 (27.57)	1.62
comb	har_2k_k200_33_33	256	Epar	532 (27.57)	2.19
comb	geo_3k_33_33	96	V	531 (27.51)	1.69
comb	geo_3k_50_00	128	V	531 (27.51)	1.76
comb	geo_1k_33_33	128	V	529 (27.41)	1.68
knn	is040	128	V	528 (27.36)	1.85
knn	is_40	96	Epar	528 (27.36)	2.40

Theorem (%): Number and percentage of theorems proved by a system.
Σ-SOTAC: See the explanatory text for this metric.

Table 3 shows the 14 methods that collectively (when computed in a greedy way)[6] cover as many of the 1930 problems as possible. To be as orthogonal as possible, the methods in this set differ a lot in their parameters, the data trained on, and the number of best premises given to the ATP. Their joint performance on this subset is 44%. All the 70 tested methods together solve 968 of the 1930 problems, i.e., 50.155% .

Table 3: The top 14 methods in the greedy sequence on the 1930-subset

Method	Parameters	Deps	Premises	ATP	Sum %	Sum
comb	min_2k_20_20	ATP6	128	Epar	28.497	550
comb	qua_k200_3k_33_33	ATP4	512	V	32.798	633
comb	geo_3k_33_33	ATP3	64	V	35.959	694
lsi	3200ti_8_80	ATP6	128	Z3	37.461	723
geo	r_99	ATP6	64	V	38.653	746
knn	200	ATP1+Mizar	1024	V	39.741	767
nb	idf010	ATP6	128	Epar	40.725	786
comb	min_20_20	ATP2	128	V	41.347	798
comb	geo_3k_50_00	ATP3	1024	V	41.969	810
knn	is040	ATP1	1024	Epar	42.487	820
knn	is_40	ATP6	96	Z3	43.005	830
geo	1_66	ATP3	1024	V	43.420	838
lsi	6400_8_120	ATP1	64	V	43.782	845
geo	1_33	ATP3	256	V	44.093	851

[6] Such *greedy (covering) sequence* of methods starts with the best method, and each next method in such sequence is the one that greedily adds most solutions to the union of solutions of the previous methods in the sequence.

3.2 Experiments and Results on the Whole MML

When the developed methods on the 1930-subset reached sufficiently high joint performance, we evaluated the most useful 14 methods on the whole MML, again with a 30-second time limit. This took about one week of real time on our server. The performance of these methods is shown in Table 4. The methods are ordered there from top to bottom already by their position in the greedy covering sequence for the whole MML. The table says that running these fourteen methods in parallel for 30 seconds gives a 40.6% chance of solving an MML theorem without any user interaction. The best previous result in such fully automated learning/proving over the whole MML was 18%, achieved in [2] by running Vampire 0.6 for 20 seconds (using about twice as fast Intel Xeon machine than our AMD server) on 200 best-ranked premises proposed by the SNoW system using the Naive Bayes learner. Since this was just a single method, a fair comparison is with the best method developed here, which solves 27.3%, i.e. 50% more problems. One of the reasons for this improvement are obviously the better training data developed here by the six MaLARea-style proving/learning passes over the MML.

It should be however noted that much better results than 18% have been achieved on smaller benchmarks such as MPTP2078, where more expensive methods such as kernel-based learning [22] could be applied. Comparison with those results is however possible only in a high-level way: we use different MML version here, different versions of the ATPs, coarser slices of the best premises, and we do not limit the premises only to those available in the 33 articles used for MPTP2078. The best result on MPTP2078 reported in [20] was 823 problems (out of 2078) solved with 70 premises, Vampire 0.6 and 5s on an Intel Xeon machine. The best new performance on the 2061 problems corresponding to the 33 MPTP2078 articles in the current MML is 1059 problems solved in 30 seconds by Epar using 128 best premises.[7] To make a bit closer comparison, we test the current best-performing method on the 2061 problems also with the old Vampire 0.6 and 5 seconds on the old Intel machine, solving 726 problems. This is practically the same result as the performance of the best old kernel-based method (combined with SInE) on the MPTP2078 benchmark when using 128 premises. This seems to be an evidence (modulo all the differences named above) that the methods based on fast scalable learning techniques such as k-NN can with enough care catch up with the existing kernel-based techniques. Quite likely, this is however not the last word, and we hope to get further improvements by scaling up and strengthening the kernel-based and related methods.

4 Proofs

We have briefly compared the shortest ATP proofs found with the corresponding MML proofs. For this, we only consider the 28892 named Mizar theorems for which we have obtained either a human or AI-advised ATP proof. The complexity metric used for a human-written Mizar proof is just the number of proof lines in the Mizar article, while for the ATP proofs we use the number of dependencies.[8]

[7] The detailed results restricted to the 2061 problems are at `http://cl-informatik.uibk.ac.at/users/cek/mizl/mptp2k.html`.

[8] These choices can obviously be questioned, but as a first comparison they are useful enough.

Table 4: 14 most covering methods on the whole MML, ordered by greedy coverage.

Method	Parameters	Prems.	ATP	Σ-SOTAC	Theorem (%)	Greedy (%)
comb	min_2k_20_20	128	Epar	1728.34	15789 (27.3)	15789 (27.2)
lsi	3200ti_8_80	128	Epar	1753.56	15561 (26.9)	17985 (31.0)
comb	qua_2k_k200_33_33	512	Epar	1520.73	13907 (24.0)	19323 (33.4)
knn	is_40	96	Z3	1634.50	11650 (20.1)	20388 (35.2)
nb	idf010	128	Epar	1630.77	14004 (24.2)	21057 (36.4)
knn	is_80	1024	V	1324.39	12277 (21.2)	21561 (37.2)
geo	r_99	64	V	1357.58	11578 (20.0)	22006 (38.0)
comb	geo_2k_50_50	64	Epar	1724.43	14335 (24.8)	22359 (38.6)
comb	geo_2k_60_20	1024	V	1361.81	12382 (21.4)	22652 (39.1)
comb	har_2k_k200_33_33	256	Epar	1714.06	15410 (26.6)	22910 (39.6)
geo	r_90	256	V	1445.18	13850 (23.9)	23107 (39.9)
lsi	3200ti_8_80	128	V	1621.11	14783 (25.5)	23259 (40.2)
comb	geo_2k_50_00	96	V	1697.10	15139 (26.1)	23393 (40.4)
geo	r_90	256	Epar	1415.48	14093 (24.3)	23478 (40.6)

These statistics, sorted by the largest difference between these metrics is available online[9], together with the ATP dependencies used for this comparison[10]. For example the first entry says that the ATP proof of the theorem `REARRAN1:24`[11] has a 534-lines long Mizar proof, while the shortest ATP proof found has only 5 dependencies. Indeed, this greatest AI/ATP-found proof shortening is valid, thanks to a symmetry between the concepts used in this theorem and a previously proved theorem `REARRAN1:17`[12] which can be established quite quickly from the concepts' definitions. The Mizar proof instead proceeds by repeating the whole argument from scratch, modifying it at appropriate places to the symmetric concepts. The AI/ATP toolchain has thus managed to succinctly express the difference between the two theorems in a very explicit and operational way, while the human authors probably were on some level also aware of the symmetry, but were not able to capture it so precisely and succinctly. In some sense, the AI/ATP system has thus managed to find, precisely formulate, and productively use a new mathematical trick.

This comparison, showing such most striking shortenings, is also useful for heuristic checking of the correctness of the whole translation/AI/ATP toolchain. By random inspection of a dozen of such shortenings, no suspicious proofs were found, i.e., all the inspected ATP proofs could be replayed in Mizar. On the other hand, some of the ATP proofs can get very long, and may be probably already quite hard to understand without further refactoring and presentation methods.

Finally, the Mizar proof length expressed in terms of the lines of code can also serve as another metric for measuring the performance of the ATP methods. The total number of the Mizar source lines used for the proofs of the 52248 named toplevel theorems is 1297926. The sum of the Mizar proof lines of the 28892 named theorems that were proved automatically (either from human or AI-selected premises) is 300914. This means that on average 23.2% of the proof lines can be "written automatically", if such automation is called on the toplevel named theo-

[9] `http://mizar.cs.ualberta.ca/~mptp/mml4.181.1147/html/00prdiff15.html`
[10] `http://mizar.cs.ualberta.ca/~mptp/mml4.181.1147/html/00atpdeps`
[11] `http://mizar.cs.ualberta.ca/~mptp/mml4.181.1147/html/rearran1.html#T24`
[12] `http://mizar.cs.ualberta.ca/~mptp/mml4.181.1147/html/rearran1.html#T17`

rems. This is a metric that in some sense complements the 56.23% ratio obtained in Section 2.1, showing that the ATPs are much better in proving the Mizar-easy theorems. On the other hand, the 23.2% average would be obviously improved a lot if also the proof-local lemmas were included in the experiments, and the number of lines corresponding to such lemmas was appropriately included in the statistics.

5 Integration with Miz$\mathbb{AR}$

The new optimized C++ versions of the premise selectors are sufficiently fast to train on the whole MML in 1–3 seconds. This simplifies the integration of the methods in the Miz$\mathbb{AR}$ service. For each query, the premise selectors are always first trained on the whole MML and also on the features and proof dependencies added from the current article.[13] After such training, the premise selectors are presented with the conjecture features, to which they respond by ranking the available theorems according to their relevance for the conjecture. The several premise selection methods with their corresponding ATPs are run in parallel, and if successful, the result is communicated to the user. The main Miz$\mathbb{AR}$ server (Intel Xeon 2.67 GHz) is considerably faster than the AMD machines used for most of the experiments.

The service thus always updates itself with new data: the conjecture is always a part of a particular Mizar article, which is submitted as a whole to the system. However, in comparison with the recently produced HOL Light service (HOL(y)Hammer), the updating is so far more limited. We do not yet try to get (for better training) the minimized ATP proofs of the article's theorems that precede the current conjecture. One reason is that, unlike in HOL(y)Hammer, the Miz$\mathbb{AR}$ service allows anonymous uploads of whole articles, but does not yet keep such projects persistent. Adding such persistence should make the computing and minimization of ATP proofs less expensive, because such data can then be quite efficiently cached and re-used (see Section 3 of [16]).

Another difference to the HOL Light setting is the very common use of local constants (eigenvariables) in the Jaśkowski-style Mizar proofs. The large-scale experiments (and thus also the training data obtained from them) presented here only deal with the set of toplevel Mizar theorems which do not contain such proof-local constants. This has two different effects when proving the proof-local lemmas that contain such constants:

1. The local assumptions and lemmas about such constants are naturally preferred by the premise selectors (in particular when using weighting schemes such as TF-IDF [14]), because such local constants (which always have a distinct internal name) and the terms containing them are rare. This is good, because such local lemmas are typically quite relevant to the local conjecture.
2. The feature representation of the proof-local lemmas may become quite distant (in the various metrics used by k-NN) from the general theorems that are needed to justify such lemmas, because many terms in the lemmas are instantiated with the local constants. This may be a serious problem, preventing finding the relevant theorems.

[13] In the Miz$\mathbb{AR}$ service, the conjecture is always submitted with the whole Mizar article in which the conjecture is stated.

We use a simple method to counter (2): We generalize all local constants in such proof-local conjectures to variables. The term features of such generalized versions of the lemmas are used together with the standard term features. This is clearly just a first step, the general task of getting features that indicate for example the (lack of) the instantiation relationship between two formulas is quite interesting, and various syntactic and semantic methods are possible [18]. Further experiments and evaluation of such issues, as well as of the user-perceived strengthening of the MizAR service are left as future work.

6 Conclusion, Future Work and Thanks

The main result of this work is the 40.6% success rate in proving the toplevel Mizar theorems fully automatically. This has been achieved by several iterations of implementing better premise selection methods, using them to obtain better training data, and using such data to further improve the performance of the learning methods. The methods were implemented very efficiently, allowing their easy deployment in the MizAR service. We believe that such strong AI/ATP systems are very useful tools that make formal mathematics much more accessible, and their gradual strengthening is today one of the most promising paths towards the eventual adoption of computer-assisted mathematics (and exact science) by mainstream mathematicians (and exact scientists).

The main body of future work is thus further strengthening of the various parts (e.g., features/labels and the whole learning setup, machine-learning techniques, ATPs) of the AI/ATP methods. Also, more advanced proof reconstruction such as [13, 27] is still missing for Mizar. With longer and longer ATP proofs, human-friendly transformations and presentations of such proofs are becoming more and more important tasks that will quite likely also benefit from learning the "human-friendliness" from large repositories of human-oriented proofs such as the MML.

Thus it seems that the forty years of incessant and stubborn designing and building of the human-oriented formal mathematical language and large library by the Mizar team, and in particular by the recently deceased Mizar gurus Andrzej Trybulec and Piotr Rudnicki, have already resulted in one of the most interesting AI corpora currently available to mankind. It will be quite hard for the historians to properly enumerate all their inventions that led to the current state of the art. We would like to thank Andrzej and Piotr for this lifelong Opus Magnum, for their infatuating dreams, their wide and never-ending interest in science (and science fiction), and for their great sense of fun combined with high doses of self-criticism, down-to-earth common sense, caution and modesty, that made them into such great scientists, hackers, teachers, debaters, critics, and friends.

References

1. Jesse Alama, Tom Heskes, Daniel Kühlwein, Evgeni Tsivtsivadze, and Josef Urban. Premise selection for mathematics by corpus analysis and kernel methods. *Journal of Automated Reasoning*, 52(2):191–213, 2014.
2. Jesse Alama, Daniel Kühlwein, and Josef Urban. Automated and Human Proofs in General Mathematics: An Initial Comparison. In Nikolaj Bjørner and Andrei Voronkov, editors, *LPAR*, volume 7180 of *LNCS*, pages 37–45. Springer, 2012.

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 *LNCS*, pages 337–340. Springer, 2008.
4. Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by Latent Semantic Analysis. *JASIS*, 41(6):391–407, 1990.
5. Sahibsingh A. Dudani. The distance-weighted k-nearest-neighbor rule. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-6(4):325–327, 1976.
6. Adam Grabowski, Artur Korniłowicz, and Adam Naumowicz. Mizar in a nutshell. *Journal of Formalized Reasoning*, 3(2):153–245, 2010.
7. Thomas Hales. *Dense Sphere Packings: A Blueprint for Formal Proofs*, volume 400 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 2012.
8. Thomas C. Hales, Mark Adams, Gertrud Bauer, Dat Tat Dang, John Harrison, Truong Le Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Thang Tat Nguyen, Truong Quang Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Jason Rute, Alexey Solovyev, An Hoai Thi Ta, Trung Nam Tran, Diep Thi Trieu, Josef Urban, Ky Khac Vu, and Roland Zumkeller. A formal proof of the Kepler conjecture. *CoRR*, abs/1501.02155, 2015.
9. John Harrison. HOL Light: A tutorial introduction. In Mandayam K. Srivas and Albert John Camilleri, editors, *FMCAD*, volume 1166 of *LNCS*, pages 265–269. Springer, 1996.
10. Krystof Hoder and Andrei Voronkov. Sine qua non for large theory reasoning. In Nikolaj Bjørner and Viorica Sofronie-Stokkermans, editors, *CADE*, volume 6803 of *LNCS*, pages 299–314. Springer, 2011.
11. Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.
12. Cezary Kaliszyk and Josef Urban. Automated reasoning service for HOL Light. In Jacques Carette, David Aspinall, Christoph Lange, Petr Sojka, and Wolfgang Windsteiger, editors, *MKM/Calculemus/DML*, volume 7961 of *Lecture Notes in Computer Science*, pages 120–135. Springer, 2013.
13. Cezary Kaliszyk and Josef Urban. PRocH: Proof reconstruction for HOL Light. In Maria Paola Bonacina, editor, *CADE*, volume 7898 of *Lecture Notes in Computer Science*, pages 267–274. Springer, 2013.
14. Cezary Kaliszyk and Josef Urban. Stronger automation for Flyspeck by feature weighting and strategy evolution. In Jasmin Christian Blanchette and Josef Urban, editors, *PxTP 2013*, volume 14 of *EPiC Series*, pages 87–95. EasyChair, 2013.
15. Cezary Kaliszyk and Josef Urban. Learning-assisted automated reasoning with Flyspeck. *Journal of Automated Reasoning*, 53(2):173–213, 2014.
16. Cezary Kaliszyk and Josef Urban. HOL(y)Hammer: Online ATP service for HOL Light. *Mathematics in Computer Science*, 9(1):5–22, 2015.
17. Cezary Kaliszyk, Josef Urban, and Jiří Vyskočil. Machine learner for automated reasoning 0.4 and 0.5. *CoRR*, abs/1402.2359, 2014. Accepted to PAAR'14.
18. Cezary Kaliszyk, Josef Urban, and Jiří Vyskočil. Efficient semantic features for automated reasoning over large theories. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*, 2015. to appear.
19. Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In Natasha Sharygina and Helmut Veith, editors, *CAV*, volume 8044 of *Lecture Notes in Computer Science*, pages 1–35. Springer, 2013.
20. Daniel Kuehlwein and Josef Urban. Learning from multiple proofs: First experiments. In Pascal Fontaine, Renate A. Schmidt, and Stephan Schulz, editors, *PAAR-2012*, volume 21 of *EPiC Series*, pages 82–94. EasyChair, 2013.
21. Daniel Kühlwein, Jasmin Christian Blanchette, Cezary Kaliszyk, and Josef Urban. MaSh: Machine learning for Sledgehammer. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *Proc. of the 4th International Conference on Interactive Theorem Proving (ITP'13)*, volume 7998 of *LNCS*, pages 35–50. Springer, 2013.
22. Daniel Kühlwein, Twan van Laarhoven, Evgeni Tsivtsivadze, Josef Urban, and Tom Heskes. Overview and evaluation of premise selection techniques for large theory mathematics. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *IJCAR*, volume 7364 of *LNCS*, pages 378–392. Springer, 2012.
23. Radim Řehůřek and Petr Sojka. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA.

24. Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.
25. Stephan Schulz. E - A Brainiac Theorem Prover. *AI Commun.*, 15(2-3):111–126, 2002.
26. John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.
27. Steffen Juilf Smolka and Jasmin Christian Blanchette. Robust, semi-intelligible Isabelle proofs from ATP proofs. In Jasmin Christian Blanchette and Josef Urban, editors, *PxTP 2013*, volume 14 of *EPiC Series*, pages 117–132. EasyChair, 2013.
28. Josef Urban. Translating Mizar for first order theorem provers. In *MKM*, volume 2594 of *LNCS*, pages 203–215. Springer, 2003.
29. Josef Urban. MPTP - Motivation, Implementation, First Experiments. *Journal of Automated Reasoning*, 33(3-4):319–339, 2004.
30. Josef Urban. MPTP 0.2: Design, implementation, and initial experiments. *Journal of Automated Reasoning*, 37(1-2):21–43, 2006.
31. Josef Urban. MaLARea: a metasystem for automated reasoning in large theories. In Geoff Sutcliffe, Josef Urban, and Stephan Schulz, editors, *ESARLT*, volume 257 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.
32. Josef Urban. An Overview of Methods for Large-Theory Automated Theorem Proving (Invited Paper). In Peter Höfner, Annabelle McIver, and Georg Struth, editors, *ATE Workshop*, volume 760 of *CEUR Workshop Proceedings*, pages 3–8. CEUR-WS.org, 2011.
33. Josef Urban. BliStr: The Blind Strategymaker. *CoRR*, abs/1301.2683, 2014. Accepted to PAAR'14.
34. Josef Urban, Piotr Rudnicki, and Geoff Sutcliffe. ATP and presentation service for Mizar formalizations. *Journal of Automated Reasoning*, 50:229–241, 2013.
35. Josef Urban, Geoff Sutcliffe, Petr Pudlák, and Jiří Vyskočil. MaLARea SG1 - Machine Learner for Automated Reasoning with Semantic Guidance. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *IJCAR*, volume 5195 of *LNCS*, pages 441–456. Springer, 2008.
36. Josef Urban and Jiří Vyskočil. Theorem proving in large formal mathematics as an emerging AI field. In Maria Paola Bonacina and Mark E. Stickel, editors, *Automated Reasoning and Mathematics: Essays in Memory of William McCune*, volume 7788 of *LNAI*, pages 240–257. Springer, 2013.