Efficient Semantic Features for Automated Reasoning over Large Theories

Cezary Kaliszyk[*]
University of Innsbruck

Josef Urban[†]
Radboud University

Jiří Vyskočil[‡]
Czech Technical University in Prague

Abstract

Large formal mathematical knowledge bases encode considerable parts of advanced mathematics and exact science, allowing deep semantic computer assistance and verification of complicated theories down to the atomic logical rules. An essential part of automated reasoning over such large theories are methods learning selection of relevant knowledge from the thousands of proofs in the corpora. Such methods in turn rely on efficiently computable features characterizing the highly structured and inter-related mathematical statements.

In this work we (i) propose novel semantic features characterizing the statements in such large semantic knowledge bases, (ii) propose and carry out their efficient implementation using deductive-AI datastructures such as substitution trees and discrimination nets, and (iii) show that they significantly improve the strength of existing knowledge selection methods and automated reasoning methods over the large formal knowledge bases. In particular, on a standard large-theory benchmark we improve the average predicted rank of a mathematical statement needed for a proof by 22% in comparison with state of the art. This allows us to prove 8% more theorems in comparison with state of the art.

1 Introduction: Reasoning in Large Theories

In the conclusion of his seminal paper on AI [Turing, 1950], Turing suggests two alternative ways how to eventually build learning (AI) machines: (i) focusing on an abstract activity like chess, and (ii) focusing on learning through physical senses. Both these paths have been followed with many successes, but so far without producing AI competitive in the most advanced application of human intelligence: scientific thinking. The approach we follow is to try to learn that from large bodies of computer-understandable scientific reasoning.

In the last decade, large corpora of complex mathematical (and scientific) knowledge and reasoning have been encoded in a fully computer-understandable form. In such encodings, the mathematical knowledge consisting of definitions, theorems, proofs and theories is explained in complete detail, allowing the computers to fully understand the semantics of such complicated objects and to verify correctness of the long reasoning chains with respect to the formal inference rules of the chosen logical framework (set theory, type theory, etc.).

Recent highlights of this development include the formal encoding and verification of two graduate textbooks leading to the proof of the Odd Order theorem ("every finite group of odd order is solvable") [Gonthier *et al.*, 2013], the formal verification of the 300-page book leading the proof of the Kepler conjecture [Hales, 2012], and verification of the seL4 operating system microkernel [Klein *et al.*, 2010].

This means that larger and larger parts of mathematics and mathematical thinking can now be analyzed, explored, assisted, and further developed by computers in ways that are impossible in domains where complete semantics is missing. The computers can not only use inductive AI methods (such as learning) to extract ideas from the large corpora, but they can also combine them with deductive AI tools such as automated theorem provers (ATPs) to attempt formal proofs of new ideas, thus further growing the body of verified scientific knowledge (which can then again be further learned from, and so on ad infinitum). In fact, this has started to happen recently. Large formal corpora built in expressive logics of interactive theorem provers (ITPs) such as Isabelle [Nipkow and Klein, 2014], Mizar [Grabowski *et al.*, 2010] and HOL Light [Harrison, 1996] have been translated to first-order logic, and ATPs such as Vampire [Kovács and Voronkov, 2013], E [Schulz, 2002] and Z3 [de Moura and Bjørner, 2008] are used to prove more and more complicated lemmas in the large theories.

Since existing ATP calculi perform poorly when given thousands to millions of facts, a crucial component that makes such ATP assistance practical are heuristic and learning AI methods that select a small number of most relevant facts for proving a given lemma [Kühlwein *et al.*, 2012]. This means that we want to characterize all statements in the knowledge bases by *mathematically relevant features* that will to a large extent allow to pre-compute the most promising combinations of formulas for proving a given conjecture. In some sense, we are thus trying to make a high-level approximation of the proof-search problem, and to restrict the fragile local decisions taken by the underlying ATP search by such

[*]Supported by the Austrian Science Fund (FWF): P26201.
[†]Supported by NWO grant nr. 612.001.208.
[‡]Supported by the Czech Grant Agency, GACR P103/12/1994.

high-level knowledge about what makes sense globally and what is likely a blind alley.[1] The question is how to design efficient features that will make such global approximative methods as good as possible. This is the subject of this paper.

Contributions

1. Semantic features for characterizing mathematical statements. We propose matching, abstraction and unification features and their combinations as a suitable means for characterizing statements in large mathematical corpora written in expressive logical frameworks (Section 4).

2. Fast semantic feature-extraction mechanisms. The crucial idea making the use of semantic features feasible is that such features often correspond to the nodes of fast deductive-AI data-structures such as substitution trees and discrimination nets. We implement and optimize such feature extraction mechanisms and demonstrate that they scale very well even on the largest formal corpora, achieving extraction times below 100 seconds for over hundred thousand formulas (Sections 5 and 6).

3. Improved Premise-Selection Performance. We evaluate the performance of the semantic features when selecting suitable premises for proofs and compare them to the old features using standard machine-learning metrics such as Recall, Precision, AUC, etc. The newly proposed features improve the average predicted rank of a mathematical statement needed for a proof by 22% in comparison with the best old features.

4. Improved Theorem-Proving Performance. We compare the overall performance of the whole feature-characterization/learning/theorem-proving stack for the new and old features and their combinations. The improved machine-learning performance translates to 8% more theorems proved automatically over the standard MPTP2078 large-theory benchmark, getting close to the ATP performance obtained by using human-selected facts (Section 7).

2 The Large-Theory Setting: Premise Selection and Learning from Proofs

The object of our interest is a large mathematical corpus, understood as a set Γ of formally stated theorems, each with zero or more proofs.[2] Examples of such corpora are the Mizar Mathematical Library (MML),[3] the Isabelle Archive of Formal Proofs (AFP),[4] and the Flyspeck (Formal Proof of the Kepler Conjecture) development[5] done in HOL Light. Such large corpora contain tens to hundreds of thousands of proved statements. To be able to do many ATP experiments,[6] smaller benchmarks have been defined as meaningful subsets

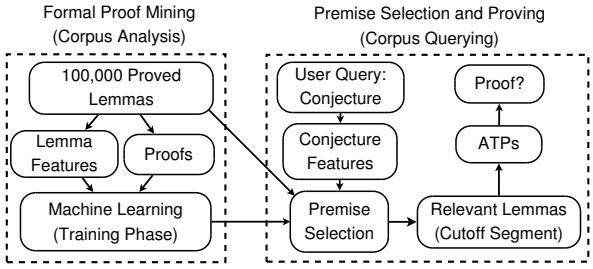

Figure 1: Theorem proving over large formal corpora. The corpus contains many lemmas that can be used to prove a new conjecture (user query). The corpus comes with many formal proofs that can be mined, i.e., used to learn which lemmas (premises) are most relevant for proving particular conjectures (queries). The strength of the learning methods depends on designing *mathematically relevant features* faithfully characterizing the statements. When a new conjecture is attempted, the learners trained on the corpus rank the available lemmas according to their estimated relevance for the conjecture, and pass a small number (cutoff segment) of the best-ranked lemmas to ATP systems, which then attempt a proof.

of the large corpora. Here we rely on the MPTP2078 benchmark [Alama *et al.*, 2014] used in the 2012 CASC@Turing ATP competition [Sutcliffe, 2013] of the Alan Turing Centenary Conference.[7]

In this setting, the task that we are interested in is to automatically prove a new conjecture given all the available theorems. Because the performance of existing ATP methods degrades considerably [Urban *et al.*, 2010; Hoder and Voronkov, 2011] when given large numbers of redundant axioms, our research problem is to estimate the facts that are most likely to be useful in the final proof. Following [Alama *et al.*, 2014] we define:

Definition 1 (*Premise selection problem*)**.** Given an ATP A, a corpus Γ and a conjecture c, predict those facts from Γ that are likely to be useful when A searches for a proof of c.

The currently strongest premise selection methods use machine learning on the proofs of theorems in the corpus, such as naive Bayes, distance-weighted k-nearest neighbor, kernel methods, and basic ensemble methods [Kühlwein *et al.*, 2012; Kühlwein *et al.*, 2013; Kaliszyk and Urban, 2013b; 2013a; 2014]. It is possible that a particular fact is useful during a proof search without being used in the final formal proof object. Such cases are however rare and hard to detect efficiently. Therefore the relation of *being useful during the proof search* is usually approximated by *being used in the final proof*. Additionally, the learning setting is usually simplified by choosing at most one ("best") proof for each theorem c [Kuehlwein and Urban, 2013], and representing the proof of c as a set of theorems $P(c)$ used in the proof. The query and update speed is important: in the ITP setting there are a number of alternative (usually less automated) theorem-proving techniques that can be used if full automation is weak or slow. Likewise, the learning systems should quickly digest and adapt to new theorems and proofs. The overall large-theory setting is shown in Figure 1.

[1] Obviously, when developed, such guiding methods can be also tried directly inside the ATP calculi. See for example the hints method [Veroff, 1996], a similar work done for E [Schulz, 2000], and the MaLeCoP system [Urban *et al.*, 2011].

[2] We treat axioms and definitions as theorems with empty proof. In general, a theorem can have several alternative proofs.

[3] `http://mizar.org/`

[4] `http://afp.sourceforge.net/`

[5] `https://code.google.com/p/flyspeck/`

[6] A large number of ATP experiments is expensive. The ATPs are usually run with time limits between 1 second and 300 seconds.

[7] `http://www.turing100.manchester.ac.uk/`

Assuming a method F for extracting mathematically relevant features characterizing the theorems, this setting leads to the following *multi-label learning task*: Each proved theorem $c \in \Gamma$ produces a training example consisting of $F(c)$ and $P(c)$, i.e., given the features $F(c)$ we want the trained predictor to recommend the labels $P(c)$. The learning methods mentioned above are typically used as rankers: given the features of a new conjecture c, the highest ranked (using heuristically determined thresholds) labels for $F(c)$ are given to an ATP, which then attempts a proof. To maximize the ATP performance, usually a portfolio of several most complementary learning methods, feature characterizations and ranking thresholds is used, and the (typically exponentially behaving) ATP is run in a strategy-scheduling mode [Tammet, 1997], i.e., with several shorter time limits over such complementary predictions rather than using the whole time limit for the most promising method.

3 Previously Introduced Features

The most commonly used features for characterizing mathematical statements in large theories are just their *symbols* [Hoder and Voronkov, 2011; Meng and Paulson, 2009]. In addition to that, large-theory ATP systems like HOL(y)Hammer [Kaliszyk and Urban, 2014], Sledgehammer [Kühlwein *et al.*, 2013] and MaLARea [Urban *et al.*, 2008] have so far used features that represent:

- Types, i.e., type constants, type constructors, and type classes [Kaliszyk and Urban, 2014]
- Term walks of length 2 [Kühlwein *et al.*, 2013]
- Subterms [Urban *et al.*, 2008]
- Validity in a pool of finite models [Urban *et al.*, 2008]
- Meta-information such as the theory name and presence in various databases [Kühlwein *et al.*, 2013]

The normalizations for term and type variables that have been tried so far include:

- Replacing variables by their (variable-normalized) types [Kaliszyk and Urban, 2014]
- Using de Bruijn indices [Urban *et al.*, 2008]
- Renaming all variables to a unique common variable [Urban *et al.*, 2008]
- Using the original variable names (this is useful when the same variable names are used for similar purposes)

Except from validity in finite models, all these features can be extracted in a linear time and are thus easy to use. Since their distribution may be quite uneven, normalizing them by methods like TF-IDF[8] is relatively important before handing them over to fast-but-simple learners such as distance-weighted k-NN (used here for evaluation). Since the MPTP2078 is untyped and already comes with variables renamed to de Bruijn indices, we do not use the type enhancements and original variable names here. Validity in a large pool of finite models requires finding a diverse set of models in which the formulas are then evaluated. Even though such features may approximate the semantics really well, they are in general much more expensive to compute than the rest, and for that reason we also avoid them here.

[8]Readers might wonder about *latent semantics* [Deerwester *et al.*, 1990]. So far this does not raise the performance significantly.

4 New Semantic Features for Reasoning

A formal proof is usually defined as a sequence (resp. DAG) of formulas where each of them is either an axiom or is derived from earlier formulas (resp. DAG parents) by an application of an inference rule. In various ITP and ATP systems such inference rules differ, however a crucial and very frequent idiom of (not just) mathematical reasoning is the rule of *Universal Instantiation*, allowing to instantiate a general statement in a concrete context.

Scenario 1. To give the first simple example, the commutativity of addition (*CA*): $X + Y = Y + X$ may be applied to prove that ($L1$): $1 + 2 = 2 + 1$, by matching 1 with X and 2 with Y. Assume that somebody already proved $1 + 2 = 2 + 1$ by referring to $X + Y = Y + X$, and a new conjecture to prove is ($L2$): $3 + 7 = 7 + 3$. When using only the features introduced previously (see Section 3), the feature overlap between these two instances will be the same as with ($L3$): $5 + 8 = 4 + 9$ (which might have been proved very differently). In other words, none of the features captures the fact that $L1$ and $L2$ are closer to each other in the instantiation lattice than to $L3$, and thus have a higher chance of sharing a common proof pattern.

A complete way how to remedy this would be to use all (variable-normalized) generalizations of a term as its features. However, the lattice of all generalizations of a term is typically exponentially large wrt. the size of the term (see Figure 2). Fortunately, in this simple scenario, we can replace such full enumeration of generalizations of a given term t with a much smaller set: the set $Gen_\Gamma(t)$ of all terms in our corpus Γ that generalize t. Indeed, since *CA* was used to prove $L1$, it must be in the corpus Γ. Therefore *CA* itself is in $Gen_\Gamma(L1)$ and also in $Gen_\Gamma(L2)$, introducing a new common *matching feature* for $L1$ and $L2$, but not for $L3$. Extraction of all such matching features for large corpora can be efficiently implemented using ATP indexing datastructures called *discrimination trees* (DT_Γ), which we briefly discuss in Section 5.

Scenario 2. Now imagine a more complicated (but still quite common) case: $L1$ is in the corpus, but has a more complicated proof, and *CA* is not in the corpus (yet). For example, 1 and 2 are defined as von Neuman ordinals, where addition is the standard ordinal addition which is not commutative in general. Since *CA* is not in Γ, it is neither in its discrimination tree, and $L1$ and $L2$ will not get the common matching feature corresponding to *CA*. But it is still quite likely that the proof of $L1$ is relevant also for proving $L2$: if we are lucky, the proof is sufficiently general and actually derives the commutativity of ordinal addition for finite ordinals behind the scenes. This means that quite often the common matching feature is still interesting, even if not explicitly present in Γ.

One might again first try to use brute force and attempt to generate all common generalizations for all pairs of terms in Γ. Again, see Figure 2 for an example when such lattice is exponentially large. It is however no longer possible to rely just on the terms that already are in Γ as in the previous case, and some method for generating some common matching features seems needed in this case. We have proposed and implemented two kinds of heuristic solutions. The first inserts for each term t a small number of more general terms

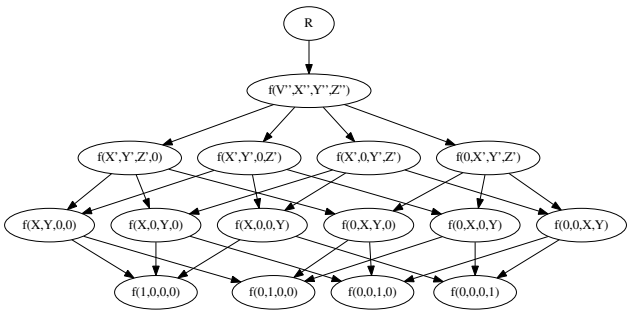

Figure 2: The lattice of least general generalization of four specific ground terms. To calculate the number F of such least general generalizations for some n, we have to count all nodes in every layer using the following equation (counting from the bottom layer to the top): $F(n) = \binom{n}{2} + \binom{n}{3} + \cdots + \binom{n}{n-1} + \binom{n}{n} = \sum_{k=2}^{n} \binom{n}{k}$ from which we get by the Binomial theorem: $F(n) = 2^n - n - 1$. So many generalization cannot be effectively extracted and used as features for more than a hundred of such terms.

into DT_Γ, typically by generalizing the term only linearly-many times, see Section 5 for details. Some of such generalization methods actually corresponds to the internal nodes of DT_Γ. The second solution collects such (differently optimized) generalization nodes explicitly, using another kind of efficient ATP indexing datastructure called *substitution tree* (ST_Γ), described in more detail in Section 6 (see Figure 4). This means that just by building ST_Γ, we naturally obtain for each term t in Γ a set of (some) generalizing features of t: the set $ST_\Gamma^{Anc}(t)$ of the ancestors of t in ST_Γ. Even the substitution tree however cannot (reasonably) guarantee that every two terms in Γ have there their least general generalization (lgg). Such requirement would again result in exponential size for examples such as Figure 2.

Scenario 3. Finally, what substitution trees can guarantee (and are usually used for in ATP) is efficient retrieval of all unifying terms. The easiest semantic motivation for collecting such features comes from the resolution rule, which is the basis of many ATP procedures. Given formulas $p(X, a)$ and $p(a, X) \implies False$, the resolution rule derives $False$ by unifying $p(X, a)$ and $p(a, X)$. Note that the two unifying literals do not match in any direction, however they will always have a nontrivial lgg.

5 Discrimination Trees for Matching and Generalization Features

Selection of candidate clause-heads that unify, match, or subsume a given goal is a central operation in automated deduction. In order to perform such selection efficiently, all major theorem provers use term indexing techniques [Robinson and Voronkov, 2001]. Discrimination trees, as first implemented by [Greenbaum, 1986], index terms in a trie, which keeps single path-strings at each of the indexed terms (Figure 3 taken from [Robinson and Voronkov, 2001]). A discrimination tree can be constructed efficiently, by inserting each term in the traversal preorder. Since discrimination trees are based on path indexing, retrieval of generalizations is straightforward,

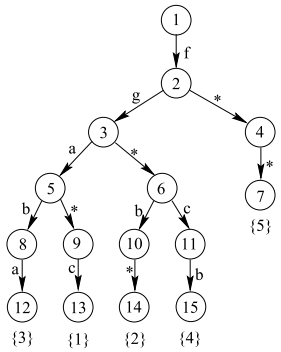

Figure 3: Discrimination Tree containing the terms $f(g(a, *), c)$, $f(g(*, b), *)$, $f(g(a, b), a)$, $f(g(*, c), b)$, and $f(*, *)$.

but retrieval of unifiable terms is more involved.

To address Scenario 1, we create a discrimination net with all the available terms in the corpus inserted in the net. Efficient implementation is needed, since the largest mathematical corpora contain millions of terms. The result of a lookup then corresponds to first order matching. This means that all terms in the corpus that are more general than the queried one are returned. In order to extend this to generalizations that are shared between terms, but not yet in the net (Scenario 2), we do heuristic generalizations. This consists of selecting a subterm of the term and replacing it by a variable. We consider several strategies for selecting the subterms to generalize:

- repeated right-most inner-most
- repeated left-most inner-most
- all positions
- combinations of above (quadratic in the size of the term), including combination of the previous one with itself.

To avoid redundant generation of generalizations of terms that are already in the net, the generation of subterms and their right-most inner-most generalizations is done together. We first iterate over the term top-down, and for each subterm level we try to insert its generalizations iteratively. If a particular generalization is already present in the net, we do not compute any further generalizations, but instead proceed to the next subterm. This optimization brings the sum of feature extraction times for all formulas below 100s for the largest formal mathematical corpus available (MML1147) containing millions of terms, see Section 7 (Table 2).

6 Substitution Trees for Unification and Generalization Features

Substitution trees [Graf, 1995] are a term indexing technique which is based on unifiability checking, rather than simple equality tests. In order to do so, substitution trees keep substitutions in the nodes. This allows for substitution trees to be smaller than other indexing techniques. Since the traversal order is not fixed, substitution trees need to compute the common generalizations of terms during insertion.

Our implementation is using the so called *linear substitution trees* in the same way as described in [Graf, 1996; Robinson and Voronkov, 2001]. The retrieval from a substitution tree may require more backtracking steps than in

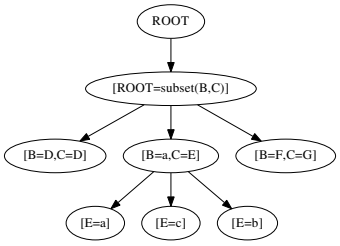

Figure 4: A substitution tree of the terms: subset(A,B), subset(a,b), subset(a,c), subset(C,C), subset(a,a). For term subset(a,c) then there is a path containing the following generalized terms (from the root to the leaf): ROOT, subset(B,C), subset(a,E), subset(a,c).

Name	Description
SYM	Constant and function symbols
TRM_0	Subterms, all variables unified
TRM_α	Subterms, de Bruijn normalized
$\text{MAT}_\varnothing$	Matching terms, no generalizations
MAT_r	Repeated gener. of rightmost innermost constant
MAT_l	Repeated gener. of leftmost innermost constant
MAT_1	Gener. of each application argument
MAT_2	Gener. of each application argument pair
$\text{MAT}_\cup$	Union of all above generalizations
PAT	Walks in the term graph
ABS	Substitution tree nodes
UNI	All unifying terms

Table 1: Summary of all the features used.

other indexing techniques, but the retrieval of unifiable terms is straightforward: it amounts to following all the paths that contain substitutions compatible with the given term. This is simpler than for example implementing unification on discrimination trees [Hoder and Voronkov, 2009].

To get interesting generalizations as features of a term t that is already inserted in the tree, we extract a path from the root to the leaf. Each node on such path represents generalization of the term t (Figure 4).

7 Experimental Analysis

Benchmark Data and Evaluation Scenario

The MPTP2078 benchmark consists of 2078 related large-theory problems (conjectures) extracted from the Mizar library. These problems contain 4494 unique formulas used as conjectures and axioms. The formulas and problems have a natural linear ordering derived from their (chronological) order of appearance in the Mizar library. As usual in such large-theory evaluations [Kaliszyk and Urban, 2014], we emulate the scenario of using AI/ATP assistance in interactive theorem proving: For each conjecture C we assume that all formulas stated earlier in the development can be used to prove C. This scenario results in *large* ATP problems that have 1877 axioms on average. Such problems are typically difficult to solve without techniques for pre-selection of the most relevant axioms (cf. Table 4).

For each conjecture C we also use its ITP (human-written) proof to extract *only the premises needed for the ITP proof* of C, i.e., $P(C)$.[9] This proof information is used in three ways: (i) to train the machine learners on all previous proofs for each conjecture C, (ii) to compare the premises predicted by such trained machine learners with the actual proof premises $P(C)$, and (iii) to construct the *small* ATP problem for C, which (unlike the *large* version) contains as axioms only the (few) premises $P(C)$ – and in this way, the ATP is very significantly advised by the human author of the ITP proof.

Speed and Machine-Learning Performance

The features that we evaluate are summarized in Table 1. We limit the evaluation to two fast learning methods: distance-weighted k-NN and naive Bayes, however the latter performs significantly worse. Table 2 shows the numbers of features obtained on MPTP2078, the feature-extraction and learning speeds on such MPTP2078 features. To see how the feature extraction process scales, we have also tested the methods on the whole MML library (version 1147) containing 146500 formulas. Note that practically all the extraction methods scale well to this corpus (we did not run UNI, due to its size), typically taking less than 100 seconds to process the whole corpus. This means that computing the features of a new conjecture which is being proved over the (already memory-loaded) corpus will take only miliseconds. Such times are negligible in comparison with standard ATP times (seconds).

Table 3 shows the standard machine learning (k-NN) evaluation, comparing the actual (needed) ITP proof premises of each conjecture C (i.e., $P(C)$) with the predictions of the machine learners trained on all ITP proofs preceding C. The average rank of a needed premise on MPTP2078 decreases from 53.03 with the best old feature method TRM_0 (formulas characterized by all their subterms with all variables renamed to just one) to 43.43 with the best new method $\text{MAT}_\varnothing$ (using the set of all matching terms as features, without any generalizations). This is a large improvement of the standard machine-learning prediction: thanks to better features, the ranking of needed premises is improved by 22.1%. The best combination of features ($\text{SYM}|\text{TRM}_0|\text{MAT}_\varnothing|\text{ABS}$) improves this to 41.97. This is a big difference for the ATP search, evaluated next.

Method	100Cover	Prec	Recall	AUC	Rank
$\text{MAT}_\varnothing$	0.918	17.711	234.12	0.9561	43.43
MAT_1	0.918	17.711	234.69	0.9557	43.83
MAT_l	0.918	17.702	235.04	0.9555	43.91
MAT_r	0.917	17.7	234.31	0.9557	43.84
MAT_2	0.917	17.708	235.37	0.9554	44.06
ABS	0.917	17.686	237.89	0.9542	44.11
PAT	0.916	17.64	235.2	0.9557	44.13
$\text{MAT}_\cup$	0.916	17.672	236.31	0.9551	44.4
TRM_0	0.903	17.425	281.46	0.9447	53.03
UNI	0.891	16.822	257.1	0.9465	51.83
SYM	0.884	17.137	326.67	0.9325	63.21
TRM_α	0.861	16.801	378.9	0.9156	75.52
$\text{SYM}\|\text{TRM}_0\|\text{MAT}_\varnothing\|\text{ABS}$	0.922	17.737	227.7	0.9587	41.97

Table 3: Machine Learning evaluation: 100Cover = average ratio of the proof premises present in the first 100 advised premises; Prec = precision; Recall = average number of premises needed to recall the whole training set; AUC = area under ROC curve; Rank = average rank of a proof premise. The methods are sorted by their 100Cover.

[9] In general, we can also use ATP proofs of the previous theorems alongside with their ITP proofs for the learning, however we will not complicate the evaluation setting here.

Method	Speed (sec)		Number of features		Learning and prediction (sec)	
	MPTP2078	MML1147	total	unique	knn	naive Bayes
SYM	0.25	10.52	30996	2603	0.96	11.80
TRM_α	0.11	12.04	42685	10633	0.96	24.55
TRM_0	0.13	13.31	35446	6621	1.01	16.70
$\text{MAT}_\varnothing$	0.71	38.45	57565	7334	1.49	24.06
MAT_r	1.09	71.21	78594	20455	1.51	39.01
MAT_l	1.22	113.19	75868	17592	1.50	37.47
MAT_1	1.16	98.32	82052	23635	1.55	41.13
MAT_2	5.32	4035.34	158936	80053	1.65	96.41
$\text{MAT}_\cup$	6.31	4062.83	180825	95178	1.71	112.66
PAT	0.34	64.65	118838	16226	2.19	52.56
ABS	11	10800	56691	6360	1.67	23.40
UNI	25	N/A	1543161	6462	21.33	516.24

Table 2: Feature numbers and learning speed on MPTP2078, and extraction speed on MPTP2078 and MML1147

Evaluation of the Theorem-Proving Performance

First we measure the performance of unaided ATPs running for 60 seconds on the large and small versions of the problems, see Table 4. While Vampire outperforms E, particularly on the large problems, we choose E for the complete evaluation which includes premise selection, because the machine learning advice will not interact with the SInE selection heuristics [Hoder and Voronkov, 2011] used by Vampire very frequently on the large problems.

The complete evaluation then proceeds as follows. For each of the best new and old premise-selection methods we create six versions of ATP problems, by taking the top-ranking segments of 8, 16, 32, 64, 128, and 256 predicted premises, and we run E on such problems for 10 seconds. Each problem thus takes again at most 60 seconds altogether, allowing us to compare the results also with the 60-second unaided ATP runs on the large and small problems. Table 5 then compares the ATP performance (measured as a union of the six slices), also adding the best-performing combination of the old and new features. The best new method $\text{MAT}_\varnothing$ solves 80 problems (8%) more than the best old method TRM_0, and the best-performing combination solves 103 (10%) problems more. This is getting close to the performance of E on the human-advised premises (1210 in Table 4). The behaviour of the ATP with respect to the number of premises used is depicted in Figure 5, showing even higher improvements.

ATP	E 1.8		Vampire 2.6		Z3
Problem set	large	small	large	small	small
Proved	573	1210	907	1319	1065

Table 4: Unaided ATPs on the large and small problems.

Method	Proved (%)	Theorems
$\text{MAT}_\varnothing$	54.379	1130
MAT_r	54.331	1129
MAT_l	54.283	1128
PAT	54.235	1127
$\text{MAT}_\cup$	53.994	1122
MAT_1	53.994	1122
MAT_2	53.898	1120
ABS	53.802	1118
TRM_0	50.529	1050
UNI	50.241	1044
SYM	48.027	998
TRM_α	43.888	912
$\text{SYM}\vert\text{TRM}_0\vert\text{MAT}_\varnothing\vert\text{ABS}$	55.486	1153

Table 5: E 1.8 prover on the large problems filtered by learning-based premise selection using different features. The methods are sorted by the number of theorems proved.

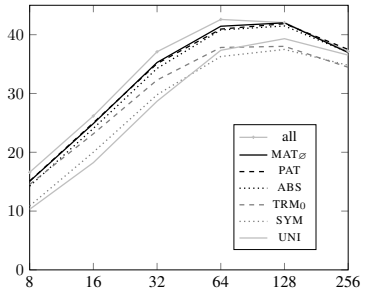

Figure 5: ATP (E 1.8) performance (in % of all large problems) for selected premise numbers and features.

8 Conclusion

We have shown that even when dealing with the largest formal corpora, one can have efficient features that encode important semantic relations and thus make the selection of relevant knowledge much more precise. The newly introduced semantic features significantly improve selection of relevant knowledge from large formal mathematical corpora.

This improvement is 22% in terms of the average predicted rank, while the combination of the new and old features helps to increase the intelligence of the advising algorithms to a level that is nearly equal to that of the human formalizers when comparing the final ATP performance. In particular, the best new method $\text{MAT}_\varnothing$ proves 8% more theorems automatically than the best old method. The cost of manually producing these proofs is very high: The Flyspeck project took about 25 person-years and the Mizar library about 100–150 person-years. So the reported improvements just for these two corpora translate to 2 and 8–12 person-years respectively.

There is a large number of directions for future work, including employing such semantic features also for internal guidance in systems like MaLeCoP and in resolution/superposition ATPs, re-using their efficient term-indexing datastructures.

References

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

[de Moura and Bjørner, 2008] 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.

[Deerwester *et al.*, 1990] 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.

[Gonthier *et al.*, 2013] Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, and Laurent Théry. A machine-checked proof of the Odd Order Theorem. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *ITP*, volume 7998 of *LNCS*, pages 163–179. Springer, 2013.

[Grabowski *et al.*, 2010] Adam Grabowski, Artur Korniłowicz, and Adam Naumowicz. Mizar in a nutshell. *J. Formalized Reasoning*, 3(2):153–245, 2010.

[Graf, 1995] Peter Graf. Substitution tree indexing. In *RTA*, volume 914 of *LNCS*, pages 117–131, 1995.

[Graf, 1996] Peter Graf. *Term Indexing*, volume 1053 of *LNCS*. Springer, 1996.

[Greenbaum, 1986] Steven Greenbaum. *Input transformations and resolution implementation techniques for theorem-proving in first-order logic*. PhD thesis, University of Illinois at Urbana-Champaign, 1986.

[Hales, 2012] Thomas Hales. *Dense Sphere Packings: A Blueprint for Formal Proofs*, volume 400 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 2012.

[Harrison, 1996] 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.

[Hoder and Voronkov, 2009] Krystof Hoder and Andrei Voronkov. Comparing unification algorithms in first-order theorem proving. In Bärbel Mertsching, Marcus Hund, and Muhammad Zaheer Aziz, editors, *KI 2009: Advances in Artificial Intelligence*, volume 5803 of *LNCS*, pages 435–443. Springer, 2009.

[Hoder and Voronkov, 2011] 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.

[Kaliszyk and Urban, 2013a] Cezary Kaliszyk and Josef Urban. Lemma mining over HOL Light. In Kenneth L. McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *LPAR*, volume 8312 of *LNCS*, pages 503–517. Springer, 2013.

[Kaliszyk and Urban, 2013b] 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.

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

[Klein *et al.*, 2010] Gerwin Klein, June Andronick, Kevin Elphinstone, Gernot Heiser, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch, and Simon Winwood. seL4: formal verification of an operating-system kernel. *Commun. ACM*, 53(6):107–115, 2010.

[Kovács and Voronkov, 2013] Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In Natasha Sharygina and Helmut Veith, editors, *CAV*, volume 8044 of *LNCS*, pages 1–35. Springer, 2013.

[Kuehlwein and Urban, 2013] 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.

[Kühlwein *et al.*, 2012] 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*, pages 378–392, 2012.

[Kühlwein *et al.*, 2013] 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, *ITP 2013*, volume 7998 of *LNCS*, pages 35–50. Springer, 2013.

[Meng and Paulson, 2009] Jia Meng and Lawrence C. Paulson. Lightweight relevance filtering for machine-generated resolution problems. *J. Applied Logic*, 7(1):41–57, 2009.

[Nipkow and Klein, 2014] Tobias Nipkow and Gerwin Klein. *Concrete Semantics - With Isabelle/HOL.* Springer, 2014.

[Robinson and Voronkov, 2001] John Alan Robinson and Andrei Voronkov, editors. *Handbook of Automated Reasoning (in 2 volumes)*. Elsevier and MIT Press, 2001.

[Schulz, 2000] Stephan Schulz. *Learning search control knowledge for equational deduction*, volume 230 of *DISKI*. Infix Akademische Verlagsgesellschaft, 2000.

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

[Sutcliffe, 2013] Geoff Sutcliffe. The 6th IJCAR automated theorem proving system competition - CASC-J6. *AI Commun.*, 26(2):211–223, 2013.

[Tammet, 1997] Tanel Tammet. Gandalf. *Journal of Automated Reasoning*, 18:199–204, 1997.

[Turing, 1950] A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[Urban *et al.*, 2008] Josef Urban, Geoff Sutcliffe, Petr Pudlák, and Jiří Vyskočil. MaLARea SG1 - Machine Learner for Automated Reasoning with Semantic Guidance. In *IJCAR*, pages 441–456, 2008.

[Urban *et al.*, 2010] Josef Urban, Krystof Hoder, and Andrei Voronkov. Evaluation of automated theorem proving on the Mizar Mathematical Library. In *ICMS*, pages 155–166, 2010.

[Urban *et al.*, 2011] Josef Urban, Jiří Vyskočil, and Petr Štěpánek. MaLeCoP: Machine learning connection prover. In Kai Brünnler and George Metcalfe, editors, *TABLEAUX*, volume 6793 of *LNCS*, pages 263–277. Springer, 2011.

[Veroff, 1996] Robert Veroff. Using hints to increase the effectiveness of an automated reasoning program: Case studies. *J. Autom. Reasoning*, 16(3):223–239, 1996.