

Semantic Graph Kernels for Automated Reasoning

Evgeni Tsivtsivadze* Josef Urban† Herman Geuvers‡ Tom Heskes§

Institute for Computing and Information Sciences
Radboud University Nijmegen, The Netherlands

Abstract

Learning reasoning techniques from previous knowledge is a largely underdeveloped area of automated reasoning. As large bodies of formal knowledge are becoming available to automated reasoners, state-of-the-art machine learning methods can provide powerful heuristics for problem-specific detection of relevant knowledge contained in the libraries. In this paper we develop a semantic graph kernel suitable for learning in structured mathematical domains. Our kernel incorporates contextual information about the features and unlike “random walk”-based graph kernels it is also applicable to sparse graphs. We evaluate the proposed semantic graph kernel on a subset of the large formal Mizar mathematical library. Our empirical evaluation demonstrates that graph kernels in general are particularly suitable for the automated reasoning domain and that in many cases our semantic graph kernel leads to improvement in performance compared to linear, Gaussian, latent semantic, and geometric graph kernels.

1 Background and Motivation: Automated Reasoning and Machine Learning

In the last fifteen years, the body of formally expressed mathematics has grown substantially. Interactive Theorem Provers (ITPs) like Coq, Isabelle, Mizar, and HOL [27] have been used for advanced formal theory developments and verification of non-trivial theorems, like the Four Color Theorem and Jordan Curve Theorem, and also for advanced verification of software and hardware models. The large formal Mizar mathematical library (MML)¹ contains today nearly 1100 formal mathematical articles, covering a substantial part of standard undergraduate mathematical knowledge. The library has about 50000 theorems, proved with about 2.5 million lines of mathematical proofs.

Such proofs often contain nontrivial mathematical

ideas, sometimes precised over decades and centuries of development of mathematics and abstract formal thinking. Having this kind of a “knowledge base of abstract human thinking” in a completely machine-processable and machine-understandable way, presents very interesting opportunities for application and development of novel artificial intelligence methods that make use of the knowledge in various ways. A concrete and pressing task, for which novel machine learning techniques are needed, and on which we focus in this paper, is selection of relevant knowledge from the large formal knowledge bases, when one is presented with a new conjecture that needs to be proved. Providing good solution to this problem is important both for the mathematicians, and also for the existing tools for automated theorem proving (ATP) that typically cannot be successfully used directly with tens or hundreds of thousands of axioms. It has been recently experimentally demonstrated with large theory benchmarks like the MPTP Challenge² and the LTB (Large Theory Batch) division of the CASC competition³ that smart selection of relevant knowledge can significantly boost the performance of existing ATP techniques in large domains [24].

There are a number of different techniques and approaches used for learning (extracting, generalizing) knowledge from large bodies of previous examples. In this paper we focus on the widely applied *kernel-based* and *latent semantics* approaches, and their suitable combination and evaluation on the formal mathematical domain. The available information in formal mathematical domains is frequently represented in structured form (such as a formula graph and a proof graph)⁴.

Kernel-based approaches [19] for learning in structured domains appear to be ideally suited for solving machine learning problems in the domain of computer-assisted reasoning. They sidestep the need for hand-

*E-mail: evgeni@science.ru.nl

†E-mail: josef.urban@science.ru.nl

‡E-mail: herman.geuvers@science.ru.nl

§E-mail: t.heskes@science.ru.nl

¹<http://www.mizar.org>

²<http://www.tptp.org/MPTPChallenge>

³<http://www.tptp.org/CASC>

⁴Generally, a particular formula can have associated to it a number of mathematical structures (models), which are typically (hyper)graphs. Thus, committing treatment of formulas to methods relying, for example, on tree representation could turn out to be limiting.

crafted features and can directly deal with the structures, in particular formula and proof graphs, encountered in this domain. The implicit, non-vectorial representation induced by kernel approaches is potentially much richer and can be further enhanced by designing appropriate kernels to incorporate prior knowledge about the domain (e.g. particular features of logic formulas). Compared to other application domains for structured learning, such as natural language processing and bioinformatics, formal mathematics has the advantage of being indisputable: there is a precise notion of semantics and truth, based on a precise notion of mathematical proof.

Informally, the task we aim to solve in the domain of computer-assisted reasoning is of selecting from a large knowledge base \mathcal{Z} of thousands of theorems those that are most relevant for proving a new formula \mathbf{x} . We can turn this into a learning problem as follows. For each theorem $\mathbf{t} \in \mathcal{Z}$ we construct a dataset $\mathcal{D}_{\mathbf{t}}$ consisting of formulas $\{\mathbf{x}_i\}_{i=1}^m$ as inputs and labels $\{y_i\}_{i=1}^m \in \mathcal{Y} = \{0, 1\}$ as outputs. The label y_i corresponding to a formula \mathbf{x}_i is 1 if the theorem \mathbf{t} is used (possibly also recursively, etc.) to prove formula \mathbf{x}_i and zero otherwise. Based on the dataset $\mathcal{D}_{\mathbf{t}}$, an algorithm can learn a classifier $C_{\mathbf{t}}(\cdot)$ which, given a formula \mathbf{x}_{κ} as input, can predict whether the theorem \mathbf{t} is relevant for proving \mathbf{x}_{κ} .

Typically, classifiers give a graded output. Having learned classifiers for all theorems \mathbf{t} , the classifier predictions $C_{\mathbf{t}}(\mathbf{x}_{\kappa})$ then can be ranked: the theorems that are predicted to be most relevant will have the highest output $C_{\mathbf{t}}(\mathbf{x}_{\kappa})$.

1.1 Notations The set of all $n \times m$ matrices with real coefficients is denoted by $\mathcal{M}_{n \times m}(\mathbb{R})$. Given a matrix $M \in \mathcal{M}_{n \times m}(\mathbb{R})$, we denote the element in the i -th row and j -th column by $[M]_{i,j}$. As a shorthand notation for the set $\{1, \dots, n\}$ we use $[n]$. For two sets $\mathcal{R} = \{i_1, \dots, i_r\} \subseteq [n]$ and $\mathcal{S} = \{j_1, \dots, j_s\} \subseteq [m]$ of indices, we use $M_{\mathcal{R}, \mathcal{S}}$ to denote the matrix that contains only the rows and columns of M that are indexed by \mathcal{R} and \mathcal{S} , respectively. At last, we use y_i to denote the i -th coordinate of a vector $\mathbf{y} \in \mathbb{R}^n$.

2 Methods and Technical Solutions: Graph Kernels for Mathematical Domain

Recently, kernels for structured domains have received significant attention in machine learning (see e.g. [9]). Many successful applications have been reported, for example, predicting the toxicity of chemical molecules [26], protein function prediction [3], etc. The domain of formal mathematics provides its own challenges, such as the abundance of structured and symbolic information, the existence of many related tasks, and the need to

deal with abstractions. All these should be taken into account when designing and/or using appropriate kernel function.

2.1 Geometric Graph Kernels Graph kernels are usually applied in situations where a graph-based structure/annotation is naturally present. For a thorough overview of graph kernels and their efficient computation we refer to [26]. Here we are concerned with geometric graph kernels (frequently applied kernel functions for computing similarity between graphs) [10]. In the following subsection we formulate a semantic graph kernel for the automated reasoning domain.

Let us define $\mathcal{L} = \{l_r\}, r \in \mathbb{N}$ to be the index set of all possible labels that could occur in the graph. Also, let $G = (\mathcal{V}, E, h)$ be a graph consisting of the ordered set of vertices $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$, the set of directed edges $E \subseteq \mathcal{V} \times \mathcal{V}$, and a function $h : \mathcal{V} \rightarrow \mathcal{L}$ that assigns a label to each vertex of a graph. A vertex v_i is a neighbour of another vertex v_j if they are connected by an edge, namely $(v_i, v_j) \in E$. A walk of length n on graph G is a sequence of indices i_0, i_1, \dots, i_n such that $(v_{i_{r-1}}, v_{i_r}) \in E$ for all $1 \leq r \leq n$. We suppose that the function h is represented as a label allocation matrix $L \in \mathcal{M}_{|\mathcal{L}| \times |\mathcal{V}|}(\mathbb{R})$ so that $[L]_{i,j} = 1$ if the label of v_j is l_i and 0 otherwise. The adjacency matrix $A \in \mathcal{M}_{|\mathcal{V}| \times |\mathcal{V}|}(\mathbb{R})$ having the rows and columns indexed by \mathcal{V} and where

$$[A]_{i,j} = \begin{cases} 1, & \text{if } (v_i, v_j) \in E \\ 0, & \text{otherwise} \end{cases}$$

corresponds to the edge set of the G . In our definitions we do not allow self loops, that is the diagonal entries of A are always zeros.

We will consider two different kernel functions to compute the similarity between graphs G and G' . The first one, called the direct product kernel [10] is constructed as follows. The vertex set of the graph G_{\times} that would take into account common walks between the vertices of the G and G' is $\mathcal{V}_{\times} \subseteq \mathcal{V} \times \mathcal{V}'$. The graph G_{\times} has a vertex iff the labels of the vertices in the corresponding graphs G and G' have the same label. Furthermore, there is an edge between the vertices in the graph G_{\times} iff there are edges between the corresponding vertices in both graphs G and G' . Let us denote the adjacency matrix of the graph G_{\times} as A_{\times} . The direct product kernel [10] then reads

$$(2.1) \quad k_{\times}(G, G') = \sum_{i,j=1}^{|\mathcal{V}_{\times}|} \left[\sum_{n=0}^{\infty} w_n A_{\times}^n \right]_{i,j},$$

where $w_n \in \mathbb{R}, w_n > 0$, is a typically monotonically decreasing sequence of weights. This kernel can be

computed efficiently using exponential or geometric series, if the limit in (2.1) exists.

As a second example we consider the kernel which measures similarity based on the start and the end vertices of the random walk. We use the fact that $[A^n]_{i,j}$ is the number of walks of length n from vertex v_i to vertex v_j , where A^n denotes the n th power of the adjacency matrix of the graph G . Moreover, if the labels of the graph vertices are taken into account, $[LA^nL^T]_{s,t}$ corresponds to the number of walks of length n between vertices labelled l_s and l_t .

We denote by $\langle M, M' \rangle_F$ the Frobenius product of matrices M and M' , that is, $\langle M, M' \rangle_F = \sum_{i,j} [M]_{i,j} [M']_{i,j}$. Further, let $\gamma \in \mathcal{M}_{n \times n}(\mathbb{R})$ be a positive semidefinite matrix containing coefficients penalizing long walks. The kernel k_n between the graphs G and G' can be defined as follows:

$$(2.2) \quad k_n(G, G') = \sum_{i,j=0}^n [\gamma]_{i,j} \langle LA^iL^T, L'A'^jL'^T \rangle_F = \sum_{s,t=1}^{|\mathcal{L}|} \sum_{i,j=0}^n [\gamma]_{i,j} [LA^iL^T]_{s,t} [L'A'^jL'^T]_{s,t}.$$

Several specializations of this kernel function lead to different feature spaces and consequently have very different interpretations. If, for example, we set $[\gamma]_{i,j} = \theta^i \theta^j$, where $\theta \in \mathbb{R}^+$ is a parameter, we obtain the kernel (2.3)

$$\widehat{k}_n(G, G') = \langle L \left(\sum_{i=0}^n \theta^i A^i \right) L^T, L' \left(\sum_{j=0}^n \theta^j A'^j \right) L'^T \rangle_F,$$

which corresponds to the inner product between the feature vectors of the G and G' , where features $\phi_{s,t}(G)$ and $\phi_{s,t}(G')$ are the weighted count of walks of length up to n from the vertices labelled l_s to the vertices labelled l_t in each of the graphs. On the other hand, if we set in (2.2) $[\gamma]_{i,j} = \theta^i$ when $i = j$ and zero otherwise, we can obtain the kernel corresponding to the inner product with features $\phi_{i,s,t}(G)$ and $\phi_{i,s,t}(G')$ that can be interpreted as θ^i times the count of walks of length i from the vertices labelled l_s to the vertices labelled l_t in each of the graphs.

2.2 Semantic Graph Kernels With semantic graph kernels defined below, we aim to take into account the co-occurrence information of the features (random walks) contained in the whole set of the graphs used for training of the algorithm. This is in contrast to the previously described geometric graph kernels that measure similarity between pairs of graphs without taking any additional (*contextual*) information into account. Furthermore, the feature space constructed using a semantic graph kernel can contain walks between sparsely connected parts of the graph if such connections are present

in the other training examples. This can make semantic graph kernels applicable to situations where geometric graph kernels might not lead to satisfactory results. Our approach is related to latent semantic indexing (LSI) [6] formulated within the framework of kernel methods [5].

Let us consider the graph-label matrix $H \in \mathcal{M}_{|\mathcal{L}|^2 \times m}(\mathbb{R})$ having, for example, the following form $H_{:,j} = \text{vec}(L(\sum_{i=0}^n \theta^i A^i)L^T) = \phi(G)$. By performing a singular value decomposition (SVD) of the matrix H we can project graphs onto the subspace spanned by the first p singular vectors to create a new dimensionally reduced feature space. Also, one can vary the dimensionality of the feature space by making a particular choice of p . We have $H = U\Sigma V^T$ and assume that the columns of U are the singular vectors of the feature space in order of decreasing singular value. Then, the projection operator on the first p principal components is $I_p U^T$, where I_p is the identity matrix with only the first p nonzero diagonal elements. Now we can use the dimensionality reduced feature space to calculate the similarity between two graphs

$$(2.4) \quad k_{\text{sg}}(G, G') = (I_p U^T \phi(G))^T (I_p U^T \phi(G')) = \phi(G)^T U I_p U^T \phi(G').$$

This kernel function uses a particular mapping to identify highly correlated features in two graphs. Unlike previously proposed graph kernels that take into account only features contained in graphs G and G' , here the information about features (random walks) that are most important in the complete training dataset is implicitly present. For example, the random walks that co-occur very often in the same graph of the training set are creating new single dimension of the new feature space.

Consider a kernel matrix having the form $K = H^T H$ that can be computed using the kernel described in equation (2.3). The kernel matrix constructed with the semantic graph kernel (2.4) is $K_{\text{sg}} = V \Lambda_p V^T$, where V are the eigenvectors of K and Λ_p is a diagonal matrix with only the first p nonzero eigenvalues. Thus, to compute the kernel matrix K_{sg} we do not need to construct H explicitly. The same approach can be used to construct a semantic kernel matrix corresponding to the graph kernel (2.1). To make a prediction we consider a column vector $\mathbf{k} = H^T \phi(G')$ that can be computed using the appropriate graph kernel function, for example, (2.3) and calculate

$$f(G') = \sum_{i=1}^m a_i k_{\text{sg}}(G_i, G') = \mathbf{a}^T V I_p V^T \mathbf{k}.$$

When constructing the kernel matrix or making a prediction we avoid dealing with the feature vectors of the graphs explicitly. However, in some circumstances, examining new features created by our kernel could pro-

vide additional insights about the problem domain. For this purpose we employ a simple stochastic search algorithm that aims to reconstruct input space representation of the projected feature vector of the graph. We note that this “feature discovery” task is related to the graph pre-image finding problem described in [2]. However, our task is simpler because the set of vertices is fixed and we only need to discover edges of the graph corresponding to the new features created by our kernel.

2.3 Kernel-Based Learning Algorithm We use kernel functions, proposed in Section 2, together with regularized learning algorithm that selects hypothesis from the reproducing kernel Hilbert space \mathcal{H} , determined by the input space \mathcal{X} and the positive definite kernel function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ (for details see [18]). We aim to minimize following objective function

$$(2.5) \quad \min_{f \in \mathcal{H}} J(f) = c(f, \mathcal{D}) + \lambda \|f\|_{\mathcal{H}}^2$$

where $c(\cdot, \cdot)$ is the loss measuring the error of the prediction function f on the training set \mathcal{D} , $\|\cdot\|_{\mathcal{H}}$ denotes the norm in \mathcal{H} , and $\lambda \in \mathbb{R}^+$ is a regularization parameter controlling the tradeoff between the error on the training set and the complexity of the hypothesis. Note that by specializing the loss in the above formulation we can obtain support vector machines [25] (by choosing $c(f, \mathcal{D}) = \sum_{i=1}^m \max(1 - y_i f(\mathbf{x}_i), 0)$) or regularized least-squares (RLS) [16] (by choosing $c(f, \mathcal{D}) = \sum_{i=1}^m (y_i - f(\mathbf{x}_i))^2$). The RLS algorithm with slight modifications (e.g. including the bias term) is also known as least-squares support vector machines [22], proximal vector machines [8], kernel ridge regression [17], and is closely related to many other methods. It has been shown that the RLS algorithm have a classification performance similar to the regular SVMs (see e.g. [11, 28]). Because of its simple closed form solution, yet competitive performance, the RLS algorithm is our choice for conducting experiments.

3 Application to Automated Reasoning

In the previous sections we have described and proposed kernels for structured representations that could be useful when learning from mathematical data. In this section, we give a concrete example of an application of the presented kernels to the task of automated reasoning and demonstrate notable improvement in performance when taking into account graph-based structure of the formulas.

Datasets We test the performance of our method on two subsets of the formal Mizar mathematical library⁵.

The first subset contains ten datasets each consisting of 530 examples⁶. The second subset contains ten datasets each consisting of 347 examples⁷. A single dataset corresponds to a binary classification task where it is necessary to predict whether a particular formula is useful in proving some theorem (see Section 1). We note that several datasets were recently used to evaluate performance of the ATP systems⁸ in the domain of automated reasoning and it has been demonstrated that heuristics for selecting relevant knowledge can significantly boost the performance of existing ATP techniques in large domains [24]. Below we briefly discuss existing feature representations that have been previously used in automated reasoning systems and their relation to several kernel functions.

Existing Feature Representations Depending on the structure of the particular mathematical domain, there are several possible kinds of features that can be more or less suitable for characterizing the formulas.

Symbols (functors, predicates, etc.).

This is the most obvious and probably most commonly used characterization of mathematical formulas. The formula is simply characterized as a set or list or multiset of the mathematical symbols that it uses. Mathematical punctuation (brackets), logical connectives and quantifiers, and also variables are typically omitted from these characteristics. Thus, for example, the formula: `forall n:Nat (n < n+1)` will be characterized by predicate symbols `<`, functor symbols `+`, `1`, and possibly a type symbol `Nat`. Note that for pure first-order automated reasoning, type symbols are not part of the basic logic, so they are typically translated to predicates. The feature representation of the above formula will thus be as follows: `[<,Nat,+,1]`. This representation was used e.g. for the first version of the MaLAREa system [24]. This feature representation is also used with linear kernel in our experiments.

Symbols (functors, predicates, etc.) via latent semantics.

This is a modification of the above method used experimentally e.g. in the SRASS system [21] for formula ordering. The symbols are treated as words and the formulas are treated as documents, and latent semantic analysis techniques are used to suitably transform the set of original symbols into a derived set of abstract “similar symbol” classes and analogously the same for formulas. Latent semantic kernel allows us to construct

⁵Publicly available at <http://www.mizar.org>

⁶Consisting of the formulas involving *sin/cos* expressions

⁷No restriction on the content of the formulas

⁸<http://www.tptp.org>

similar feature space.

Subterms and subformulas.

While some theories introduce and use concepts and notation a lot, some other theories use just a basic set of predicates and functors. This is predominantly of mathematical style and conventions. For example, all of the Mizar mathematical library can be today expanded just to the basic set-theoretical notation, using only the membership and equality predicates. If a theory is expressed in such a concept-economical way, the potential of the symbol-based characterization is very poor. On the other hand, there will be many typical repeated patterns in such formulas.

Term and formula patterns.

Not only variables in terms and formulas can be normalized, but also the functor and predicate symbols can be treated as just a structural phenomenon, and their name (label) abstracted away. This way of pattern extraction was experimentally implemented by Stephan Schulz in the E prover [20]. Such techniques are generally useful for drawing analogies between different theories (using different symbol naming).

Models of formula.

A feature characterizing a formula in a much more “semantic” way rather than just a symbolic way can be chosen from large representative set of models of the formula. This is again a very simple “bag of models” characterization of the formula. This was also quite successfully used in the latest version of MaLARea [24].

Proposed Feature Representation Designing and extracting previously described features from the Mizar mathematical library can be a tedious task, requiring significant amount of preprocessing, and the structure of the formula might not be retained. Alternatively, one can design a feature space and take into account natural representation of the formula with graph kernels, as described in Section 2. Let us consider an example depicted in Figure 1.

Random walk features.

The adjacency matrix of the graph allows us in addition to simple unigram features (labels of the vertices e.g. **m**, **n**, **prime**, etc.) to obtain feature bigrams corresponding to the edges of the graph e.g. **m-prime**, **m-power**, **and-implies**, etc. If we consider the second power of the adjacency matrix in the presented example, that is, the walks of length 2 in the graph representation of the formula, we obtain a walk between two vertices if they are connected with an edge with some other vertex in

the formula e.g. **m-and**, **prime-implies**, **a-and**, etc.

These bigram features allow the algorithm to identify pairs of predicates, functors, and other combinations that are commonly linked. Further, if erroneous formulas are provided in training, the algorithm also has the opportunity to learn to avoid links between predicates/functors that should not be linked. If we consider the higher powers of the adjacency matrices, we obtain new features, e.g. **m-implies**, **a-implies** pairs in the third power. How these features are weighted, combined, etc., depends on particular formulation of the graph kernel.

Learning Algorithm and Parameters We conduct the experiments using the regularized least-squares (RLS) algorithm [16]. We evaluate the classification performance of the algorithm using five kernels: linear, Gaussian [19], latent semantic [5], geometric graph (2.3), and semantic graph (2.4) denoted as k_{LIN} , k_{GAUSS} , k_{SEM} , k_{G} , and k_{SG} respectively. The RLS has one regularization parameter controlling the tradeoff between the error on the training set and the complexity of the hypothesis. In addition when using k_{GAUSS} the width, for k_{SEM} , k_{SG} the number of principal components, and for k_{G} , k_{SG} the decay coefficient and the length of the walk has to be estimated. To find optimal regularization and kernel parameters we use a 10-fold cross-validation procedure on the training set.

Experimental Setup We treat the problem as a binary classification task, where the formula useful in proving a theorem belongs to the positive class and it belongs to the negative class otherwise. For each of the ten datasets two thirds of the examples are used for training, while one third is reserved for testing. On the training set we use 10-fold cross-validation to estimate optimal parameters. We select parameters that on average lead to the best performance over 10-folds. With these parameters fixed, we retrain the algorithm on the training set (two thirds) and test it on the test set (one third). Finally, we repeat the complete procedure on a 10 times re-randomized dataset and average the results. We evaluate performance of the classifier using AUC (see e.g. [4]) - the area under the ROC curve that is equal to the probability that the decision function assigns a higher value to a randomly drawn positive example rather than randomly drawn negative example. The obtained results for the datasets are reported in the Table 1 and 3.

It can be observed that using graph kernel functions consistently leads to better classification performance compared to the kernels that do not take structural representation of the examples into account. The results

Table 1: Classification performance in AUC of the RLS algorithm with the k_G , k_{SG} , k_{LIN} , k_{SEM} , and k_{GAUSS} kernel functions on a subset from the Mizar library consisting of 10 datasets, each containing 530 examples. The datasets consist of the formulas involving \sin/\cos expressions. Bold numbers indicate the method with the best score on the particular dataset.

DATASET	k_G	k_{SG}	k_{LIN}	k_{SEM}	k_{GAUSS}
\mathcal{D}_{t_1}	0.751	0.759	0.675	0.747	0.734
\mathcal{D}_{t_2}	0.993	0.991	0.957	0.976	0.981
\mathcal{D}_{t_3}	0.673	0.651	0.652	0.656	0.658
\mathcal{D}_{t_4}	1.000	1.000	0.992	1.000	0.995
\mathcal{D}_{t_5}	0.942	0.945	0.751	0.827	0.856
\mathcal{D}_{t_6}	0.712	0.706	0.705	0.693	0.685
\mathcal{D}_{t_7}	0.589	0.589	0.539	0.558	0.567
\mathcal{D}_{t_8}	0.718	0.733	0.593	0.685	0.699
\mathcal{D}_{t_9}	0.594	0.589	0.581	0.571	0.582
$\mathcal{D}_{t_{10}}$	0.991	0.991	0.955	0.977	0.963

Table 2: Classification performance in AUC of the RLS algorithm with the k_G , k_{SG} , k_{LIN} , k_{SEM} , and k_{GAUSS} kernel functions on a subset of Mizar library consisting of 10 datasets, each containing 530 examples. The datasets consist of the formulas involving \sin/\cos expressions. In this experiment 30% of the functors have different syntactical representation but the same meaning. Bold numbers indicate the method with the best score on the particular dataset.

DATASET	k_G	k_{SG}	k_{LIN}	k_{SEM}	k_{GAUSS}
\mathcal{D}_{t_1}	0.674	0.699	0.677	0.676	0.676
\mathcal{D}_{t_2}	0.938	0.958	0.898	0.898	0.903
\mathcal{D}_{t_3}	0.573	0.621	0.559	0.560	0.583
\mathcal{D}_{t_4}	0.981	0.981	0.975	0.973	0.980
\mathcal{D}_{t_5}	0.557	0.583	0.555	0.559	0.562
\mathcal{D}_{t_6}	0.574	0.591	0.543	0.543	0.562
\mathcal{D}_{t_7}	0.501	0.529	0.505	0.509	0.512
\mathcal{D}_{t_8}	0.636	0.641	0.573	0.573	0.585
\mathcal{D}_{t_9}	0.523	0.556	0.511	0.521	0.534
$\mathcal{D}_{t_{10}}$	0.948	0.962	0.931	0.927	0.947

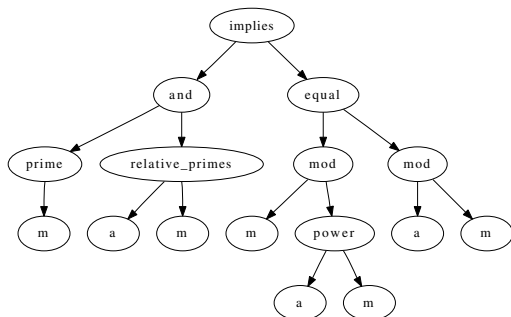


Figure 1: Example of the formula for Fermat’s Little Theorem: $prime(m)$ and $relative_primes(a,m)$ implies $power(a,m) \bmod m = a \bmod m$.

Table 3: Classification performance in AUC of the RLS algorithm with the k_G , k_{SG} , k_{LIN} , k_{SEM} , and k_{GAUSS} kernel functions on a subset from the Mizar library consisting of 10 datasets, each containing 347 examples. There is no restriction on the content of the formulas in the datasets. Bold numbers indicate the method with the best score on the particular dataset.

DATASET	k_G	k_{SG}	k_{LIN}	k_{SEM}	k_{GAUSS}
\mathcal{D}'_{t_1}	0.620	0.620	0.577	0.577	0.552
\mathcal{D}'_{t_2}	0.867	0.906	0.842	0.845	0.834
\mathcal{D}'_{t_3}	0.729	0.718	0.696	0.696	0.691
\mathcal{D}'_{t_4}	0.857	0.844	0.820	0.824	0.830
\mathcal{D}'_{t_5}	0.656	0.641	0.638	0.638	0.639
\mathcal{D}'_{t_6}	0.708	0.689	0.647	0.658	0.667
\mathcal{D}'_{t_7}	0.754	0.766	0.671	0.673	0.691
\mathcal{D}'_{t_8}	0.990	0.982	0.987	0.987	0.981
\mathcal{D}'_{t_9}	0.569	0.567	0.561	0.561	0.561
$\mathcal{D}'_{t_{10}}$	0.880	0.871	0.801	0.801	0.824

Table 4: Classification performance in AUC of the RLS algorithm with the k_G , k_{SG} , k_{LIN} , k_{SEM} , and k_{GAUSS} kernel functions on a subset from the Mizar library consisting of 10 datasets, each containing 347 examples. There is no restriction on the content of the formulas in the datasets. In this experiment 30% of the functors have different syntactical representation but the same meaning. Bold numbers indicate the method with the best score on the particular dataset.

DATASET	k_G	k_{SG}	k_{LIN}	k_{SEM}	k_{GAUSS}
\mathcal{D}'_{t_1}	0.529	0.525	0.502	0.505	0.510
\mathcal{D}'_{t_2}	0.579	0.607	0.536	0.536	0.522
\mathcal{D}'_{t_3}	0.584	0.605	0.580	0.600	0.575
\mathcal{D}'_{t_4}	0.589	0.612	0.587	0.594	0.603
\mathcal{D}'_{t_5}	0.563	0.602	0.559	0.585	0.568
\mathcal{D}'_{t_6}	0.559	0.574	0.554	0.558	0.559
\mathcal{D}'_{t_7}	0.538	0.589	0.571	0.569	0.576
\mathcal{D}'_{t_8}	0.812	0.804	0.673	0.668	0.752
\mathcal{D}'_{t_9}	0.562	0.584	0.545	0.545	0.557
$\mathcal{D}'_{t_{10}}$	0.840	0.868	0.848	0.858	0.843

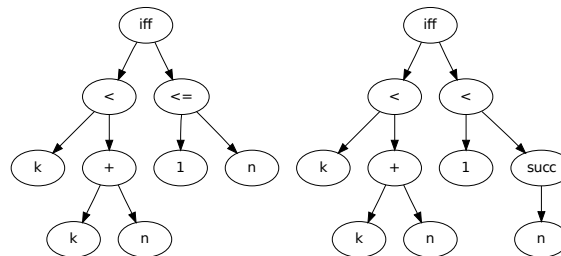


Figure 2: Example of different phrasing of the same mathematical fact, and the corresponding parts of formulas. The first formula $k < k + n \text{ iff } 1 \leq n$ uses the non-strict version of integer ordering, while the second uses the strict version, together with the successor function.

on the datasets in case when some of the functors are syntactically different but semantically represent the same concept are reported in Table 2 and 4. In these datasets, examples similar to the ones depicted in Figure 2 are frequent. In several cases some of the edges/vertices of the formula graphs are randomly removed, simulating the setting of learning from the dataset containing errors⁹. It can be observed that proposed kernel k_{SG} usually performs better compared to the standard geometric graph kernel k_{G} , indicating that taking into account the semantic similarity of the graphs can improve classification performance. We use the Wilcoxon signed-rank test [7] on the results obtained from 10 independent runs of the algorithm to estimate whether differences are significant. In all cases except for the datasets $\mathcal{D}_{\text{t}_2}, \mathcal{D}_{\text{t}_5}$ reported in Table 1 and $\mathcal{D}'_{\text{t}_8}, \mathcal{D}'_{\text{t}_9}$ reported in Table 3 the differences for the best performing method on a particular dataset are statistically significant.

4 Significance and Impact

Encoding of large parts of mathematics in computer-understandable form is becoming more common and used by mathematicians. Large computer-assisted proofs have appeared, for which human checking and reviewing do not scale. Well-known examples are the proof of the Kepler conjecture [12], and proof of the Four Color theorem [1]. Formal verification of such advanced theorems is the only way how to make sure that their proofs are correct. Such verification projects give rise to large computer-understandable mathematical libraries of theorems and definitions for which development of tailored search methods and automated reasoning advice are necessary.

The existing automated theorem proving technology has been so far researched only in much smaller settings. Standard methods like resolution proving suffer from very large search space when used in large libraries, which renders existing automated reasoning systems practically unusable in large libraries without suitable additional algorithms for heuristic premise selection. For example, in the recent evaluation [23] done over the whole Mizar library, the performance of state-of-the-art automated reasoning systems like E prover is negligible (2% success in proving of the Mizar theorems) when used on the whole library of 50000 theorems without any premise-selection method.

Symbol-based selection methods have been developed to provide initial solutions over medium-size problems [13], and heuristics like SiNE [23] already perform

reasonably well over large ontologies consisting mainly of definitions, like SUMO [14] and Cyc [15]. However, such methods are still relatively weak (15% success in proving of the Mizar theorems, see [23]) in comparison with human-based premise selection when used on very large libraries with complicated proof structure and many complicated theorems. Existing simple symbol-based methods provide no information about the relative importance of theorems, based on their use for solving related previous problems. Structured and complicated large mathematical libraries thus provide an interesting challenge and application field for development of machine learning methods that are aware of the libraries' contents, semantics, and proof structure. The work presented here is a first serious attempt at developing kernel methods for this application field.

5 Conclusions

The contributions of this paper are twofold. First, we point to automated reasoning as an interesting and challenging domain for machine learning and in particular machine learning techniques applicable to structured data. Second, we introduce semantic graph kernels that unlike standard geometric graph kernels can also be used to learn from sparse graphs. Although in this study we are primarily concerned with the automated reasoning domain, semantic graph kernels can be applied to various tasks in computational biology, natural language processing, social computing, etc., where data have a natural graph-based representation. In our empirical evaluation we demonstrate that graph kernels in general and our semantic graph kernel in particular lead to improvement in classification performance compared to approaches that do not take the structured representation of the data into account. In the future we plan to address the task by considering large scale multi-class classification setting. Our final goal is incorporation of the proposed method into open source ATP systems which can lead to notable benefits both in terms of accuracy and efficiency when applied to the automated reasoning domain.

Acknowledgments

We acknowledge support from the Netherlands Organization for Scientific Research (NWO), in particular **Learning2Reason**, Vici grant (639.023.604), and CLS grant (635.100.020).

References

- [1] K. Appel, W. Haken, and J. Koch. Every planar map is four colorable. *Illinois Journal of Mathematics*, 21:429–567, 1977.

⁹Errors can occur, for example, in datasets obtained from old mathematical journals by digitization.

- [2] Gökhan H. Bakir, Alexander Zien, and Koji Tsuda. Learning to find graph pre-images. In Carl Edward Rasmussen, Heinrich H. Bühlhoff, Bernhard Schölkopf, and Martin A. Giese, editors, *Pattern Recognition, 26th DAGM Symposium*, volume 3175 of *Lecture Notes in Computer Science*, pages 253–261. Springer, 2004.
- [3] Karsten M. Borgwardt, Hans-Peter Kriegel, S. V. N. Vishwanathan, and Nicol N. Schraudolph. Graph kernels for disease outcome prediction from protein-protein interaction networks. In *Pacific Symposium on Biocomputing (PSB)*, volume 12, pages 4–15, Maui, Hawaii, 2007. World Scientific.
- [4] Corinna Cortes and Mehryar Mohri. AUC optimization vs. error rate minimization. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004. MIT Press.
- [5] Nello Cristianini, John Shawe-Taylor, and Huma Lodhi. Latent semantic kernels. *Journal of Intelligent Information Systems*, 18(2-3):127–152, 2002.
- [6] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.
- [7] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.
- [8] Glenn Fung and Olvi L. Mangasarian. Proximal support vector machine classifiers. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 77–86, New York, NY, USA, 2001. ACM.
- [9] Thomas Gärtner. A survey of kernels for structured data. *SIGKDD Explorations*, 2003.
- [10] Thomas Gärtner, Peter A. Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In Bernhard Schölkopf and Manfred K. Warmuth, editors, *Sixteenth Annual Conference on Computational Learning Theory*, volume 2777 of *Lecture Notes in Computer Science*, pages 129–143, 2003.
- [11] Tony Van Gestel, Johan A. K. Suykens, Bart Baesens, Stijn Viaene, Jan Vanthienen, Guido Dedene, Bart De Moor, and Joos Vandewalle. Benchmarking least squares support vector machine classifiers. *Machine Learning*, 54(1):5–32, 2004.
- [12] Thomas C. Hales and Samuel P. Ferguson. A formulation of the kepler conjecture. *Discrete & Computational Geometry*, 36(1):21–69, 2006.
- [13] Jia Meng and Lawrence C. Paulson. Lightweight relevance filtering for machine-generated resolution problems. *Journal of Applied Logic*, 7(1):41–57, 2009.
- [14] I. Niles and A. Pease. Towards A Standard Upper Ontology. In C. Welty and B. Smith, editors, *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems*, pages 2–9, 2001.
- [15] K. Panton, C. Matuszek, D. Lenat, D. Schneider, M. Witbrock, N. Siegel, and B. Shepard. Common Sense Reasoning - From Cyc to Intelligent Assistant. In Y. Cai and J. Abascal, editors, *Ambient Intelligence in Everyday Life*, number 3864 in *Lecture Notes in Artificial Intelligence*, pages 1–31. Springer-Verlag, 2006.
- [16] Ryan Rifkin, Gene Yeo, and Tomaso Poggio. Regularized least-squares classification. In J.A.K. Suykens, G. Horvath, S. Basu, C. Micchelli, and J. Vandewalle, editors, *Advances in Learning Theory: Methods, Model and Applications*, pages 131–154, Amsterdam, 2003. IOS Press.
- [17] Craig Saunders, Alexander Gammerman, and Volodya Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine Learning*, pages 515–521, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.
- [18] Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In David P. Helmbold and Bob Williamson, editors, *Proceedings of the 14th Annual Conference on Computational Learning Theory*, pages 416–426, London, 2001. Springer.
- [19] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.
- [20] S. Schulz. E – a brainiac theorem prover. *Journal of AI Communications*, 15(2-3):111–126, 2002.
- [21] Geoff Sutcliffe and Yury Puzis. S-rass - a semantic relevance axiom selection system. In Frank Pfenning, editor, *CADE*, volume 4603 of *Lecture Notes in Computer Science*, pages 295–310. Springer, 2007.
- [22] J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.
- [23] Josef Urban, Krystof Hoder, and Andrei Voronkov. Evaluation of automated theorem proving on the mizar mathematical library. In *ICMS*, pages 155–166, 2010.
- [24] Josef Urban, Geoff Sutcliffe, Petr Pudlák, and Jiri Vyskocil. MaLAREa sg1- machine learner for automated reasoning with semantic guidance. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *IJCAR*, volume 5195 of *Lecture Notes in Computer Science*, pages 441–456. Springer, 2008.
- [25] Vladimir Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.
- [26] S.V.N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11:12011242, 2010.
- [27] Freek Wiedijk, editor. *The Seventeen Provers of the World, Foreword by Dana S. Scott*, volume 3600 of *Lecture Notes in Computer Science*. Springer, 2006.
- [28] Peng Zhang and Jing Peng. SVM vs regularized least squares classification. In Josef Kittler, Maria Petrou, and Mark Nixon, editors, *Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04) Volume 1*, pages 176–179, Washington, DC, USA, 2004. IEEE Computer Society.