MULTI-OUTPUT RANKING FOR AUTOMATED REASONING

Daniel Kühlwein, Josef Urban, Evgeni Tsivtsivadze, Herman Geuvers and Tom Heskes
Institute for Computing and Information Sciences, Radboud University Nijmegen, Nijmegen, The Netherlands

Keywords: Ranking, Automated theorem proving, Premise selection, Machine learning.

Abstract: Premise selection and ranking is a pressing problem for applications of automated reasoning to large formal theories and knowledge bases. Smart selection of premises has a significant impact on the efficiency of automated proof assistant systems in large theories. Despite this, machine-learning methods for this domain are underdeveloped. In this paper we propose a general learning algorithm to address the premise selection problem. Our approach consists of simultaneous training of multiple predictors that learn to rank a set of premises in order to estimate their expected usefulness when proving a new conjecture. The proposed algorithm efficiently constructs prediction functions and can take correlations among multiple tasks into account. The experiments demonstrate that the proposed method significantly outperforms algorithms previously applied to the task.

1 INTRODUCTION

Over the last two decades, the body of formally expressed knowledge has grown substantially. Formal mathematics is becoming increasingly well-known, used, and experimented with (Hales, 2008). Projects like the formal proof of the Kepler Conjecture (FlySpeck) (Hales, 2005), the formal proof of the Four Color Theorem (Gonthier, 2008), verification of tiny (but real) operating systems (Klein et al., 2009), and the increased use of proof assistants for software and hardware verification (D'Silva et al., 2008) are stimulating the development of interactive verification tools and interactive theorem provers (ITPs), and the growths of the libraries of formal proofs, definitions, and theorems. Linked to this is the development of strong automated theorem provers (ATPs), which are used either independently to solve hard problems in suitable domains (McCune, 1997; Phillips and Stanovsky, 2008), or assisting the interactive tools (Urban, 2006; Meng and Paulson, 2008; Hurd, 2003; Urban and Sutcliffe, 2008). In the usual setting, the ATP is given a set of premises and a conjecture. The ATP then has to prove that the conjecture is a logical implication of the premises; or show that this isn't the case. In general, the more premises a problem has, the bigger the search space for the ATP and the harder it is to find a proof.

With the continuing growth of formal knowledge bases, the selection of relevant knowledge, when one is presented with a new conjecture that needs to be proven, becomes a concrete and pressing task. Providing good solutions to this problem is important both for mathematicians, and for existing ATPs which typically cannot be successfully used directly with hundreds or thousands of axioms. Experiments with large theory benchmarks like the MPTP Challenge[1], or the LTB (Large Theory Batch) division of the CASC competition(Sutcliffe and Suttner, 2006), showed that smart selection of relevant knowledge can significantly boost the performance of ATPs in large domains (Urban et al., 2008; Urban et al., 2010).

Premise selection for automated theorem proving can be seen as a ranking problem: Given a ranking for a large set of premises, the ATP can try to prove the conjecture by using only the highest ranked premises. Several attempts to solving this problem have been made (e.g. (Urban et al., 2010; Roederer et al., 2009)). However, no state-of-the-art machine learning techniques have been used.

In this paper, we develop new learning methods for the selection of premises of new conjectures in large formal theories. We apply our methods to available large libraries of formal proofs and compare them with already existing algorithms.

[1] http://www.tptp.org/MPTPChallenge

Kühlwein D., Urban J., Tsivtsivadze E., Geuvers H. and Heskes T..
MULTI-OUTPUT RANKING FOR AUTOMATED REASONING.
DOI: 10.5220/0003650400420051
In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval* (KDIR-2011), pages 42-51
ISBN: 978-989-8425-79-9

1.1 Formulation of the Problem as a Machine-learning Task

Informally, we have to solve the following problem: Given a large knowledge base $\mathcal{P}$ of hundreds of premises and a conjecture x, find the premises $\mathcal{P}_x$ that are most relevant for proving x. Note that this can be seen as a multi-label classification problem where conjectures would correspond to examples and premises to labels (Tsoumakas et al., 2010). We present two different ways to approach this problem:

Let Γ be the set of all first order formulas over a fixed countable alphabet, $\mathcal{X} = \{x_i \mid 1 \leq i \leq n\} \subset \Gamma$ be the set of conjectures, $\mathcal{P} = \{p_j \mid 1 \leq j \leq m\} \subset \Gamma$ be the set of premises, and $\mathcal{Y} : \mathcal{X} \times \mathcal{P} \to \{0,1\}$ be the indicator function such that $y_{x_i,p_j} = 1$ if p_j is used to prove x_i and $y_{x_i,p_j} = 0$ if p_j is not used to prove x_i.

Binary Classification. For each premise $p \in \mathcal{P}$ we can construct a dataset $\mathcal{D}_p = \{(x, y_{x,p}) \mid x \in \mathcal{X}\}$. Based on $\mathcal{D}_p$, a suitable algorithm can *learn* a classifier $C_p(\cdot) : \Gamma \to \mathbb{R}$ which, given a formula x as input, can *predict* whether the premise p is relevant for proving x. Typically, classifiers give a graded output. Having learned classifiers for all premises $p \in \mathcal{P}$, the classifier predictions $C_p(x)$ can be ranked: the premises that are predicted to be most relevant will have the highest output $C_p(x)$. This approach to premise selection/ranking is proposed in (Tsivtsivadze et al., 2011).

Multi-output Ranking. We can also consider the above problem as a label ranking task (see e.g. (Fuernkranz and Huellermeier, 2011)): We assume that for every conjecture x, there is a transitive, asymmetric preference relation $\leq_x \subseteq \mathcal{P} \times \mathcal{P}$ such that for all $p, q \in \mathcal{P}$ $p \leq_x q$ if and only if q is more relevant for proving x than p. The goal is to predict this relation. For each conjecture, we are given the output vector $\mathbf{y}_x = (y_{x,p_1}, \ldots, y_{x,p_m})$. Note that due to the nature of our data we are concerned with bipartite ranking (Agarwal, 2005), but the formulation stands for real-valued $y_{x,p}$ as well. If we consider a matrix of dimension $n \times m$ constructed with the output vectors of all conjectures in the dataset, predicting each column corresponds to a single classification task as described above.

Finally, the training set can be written as $\mathcal{D} = \{(x, \mathbf{y}_x) \mid x \in \mathcal{X}\}$. Given the training set, our task is to find a *ranking function* $f : \Gamma \to \mathbb{R}^m$ such that for each conjecture $x \in \mathcal{X}$ the ranking $\leq_{f(x)} \subseteq \mathcal{P} \times \mathcal{P}$ induced by the function f is a good "prediction" of the true preference relation $\leq_x \subseteq \mathcal{P} \times \mathcal{P}$.

Differences to the Standard Ranking Setting. It is interesting to note that the ranking problem described above is somewhat different to standard ranking problems that are frequently encountered in information retrieval, bioinformatics, natural language processing, and many other domains. For example, a common task in information retrieval is document ranking task. Given a query we are interested in predicting the ranking of some set of retrieved documents for the particular query. Drawing a parallel with the premise selection task, given a conjecture we are interested in determining a ranking of the premises so that the premises that are most useful are ranked on top. However, there is a crucial difference: The datasets for document ranking tasks (e.g. LETOR dataset[2]) contain feature representations of the query and the documents as well as their rankings, while in our dataset only feature representations of the conjectures and the appropriate rankings of the premises (but not their feature representations) are available[3].

Therefore, the approach we take in the next section can be informally summarized as follows. We aim to determine the ranking of the set of premises for a particular conjecture. However, because no feature representation of the premises is available for learning we instead solve the problem by determining how useful a particular premise is to prove a conjecture. Once that score (rank) is determined we can induce a total order over the set of premises that can be used to prove the conjecture.

1.2 Organization

The remainder of the paper is organized as follows: Section 2 describes the developed kernel algorithm and the algorithms that have already been used in this setting. An evaluation of the different methods can be found in section 3. We discuss the results of the experiments in section 4. In section 5, we argue about the significance and the impact of this work. Finally, a conclusion is presented in section 6.

2 ALGORITHMS

In this section we present our framework for multi-output ranking for premise selection as well as two algorithms that have already been used for this problem. We will compare all introduced algorithms in the experiments.

[2]http://research.microsoft.com/users/tyliu/LETOR/

[3]One example for such a problem is the cross-verification of a single proof within a large library where we do not have access to the imported theorems.

2.1 Multi-output Ranking

We propose a method for multi-output ranking (MOR) that is a relatively straightforward extension of the preference learning algorithm described in (Tsivtsivadze et al., 2010). We also note that the algorithm is a generalization of the multi-output regularized least-squares method and is not specifically tied to the automated reasoning domain. It can be applied to various problems in bioinformatics (e.g. protein ranking), information retrieval (e.g. collaborative filtering), natural language processing (e.g. parse ranking), etc.

Our method is based on the regularized least-squares (RLS) (Rifkin et al., 2003) algorithm. This choice is motivated by the fact that RLS has been shown to perform comparably to state-of-the-art supervised learning algorithms (e.g. SVM) (van Gestel et al., 2004; Zhang and Peng, 2004) and has several computational advantages, e.g. the fact that it can be efficiently extended to handle multiple output prediction problems. The standard version of the RLS algorithm can be considered as as special case of the following regularization problem known as Tikhonov regularization (for a more comprehensive description, see e.g., (Poggio and Smale, 2003)):

$$\min_f \sum_{i=1}^{n} l(f(x_i), y_i) + \lambda \|f\|_k^2, \tag{1}$$

where l is the loss function used by the algorithm, $f : \mathcal{X} \to \mathbb{R}$ is a prediction function, $\lambda \in \mathbb{R}_+$ is a regularization parameter, and $\|\cdot\|_k$ is a norm in a Reproducing Kernel Hilbert Space (RKHS) (Schoelkopf et al., 2001) defined by a positive definite kernel function k. The loss function used with RLS for regression problems is defined as $l(f(x), y) = (y - f(x))^2$. When we restrict the prediction function f to be an element of

$$\begin{aligned}\mathcal{F} = \{ f \in \mathbb{R}^{\Gamma} \mid f(x) = \sum_{i=1}^{\infty} \beta_i k(x, z_i), \beta_i \in \mathbb{R}, \\ z_i \in \Gamma, \|f\|_k < \infty \}.\end{aligned} \tag{2}$$

then, by the Representer Theorem (see e.g., (Schoelkopf et al., 2001)), the minimizer of equation (1) has the following form:

$$f(x) = \sum_{i=1}^{n} a_i k(x, x_i), \tag{3}$$

where $a_i \in \mathbb{R}$ for $1 \le i \le n$ and k is the kernel function associated with the RKHS mentioned above.

We can also consider a loss function that considers pairs.

This can be further specified by defining a relevance matrix $W \in \mathbb{R}^{n \times n}$ for the data points. For example, we could have that $[W]_{i,j} = 1$ for the difference between the predictions for x_i, x_j if they are relevant for the learning task and 0 otherwise (Tsivtsivadze et al., 2010). We get the following problem:

$$\min_{f \in \mathcal{F}} \sum_{i,j=1}^{n} [W]_{ij}((y_i - y_j) - (f(x_i) - f(x_j))^2 + \lambda \|f\|_{\mathcal{H}}^2 \tag{4}$$

Instead of simply regressing the scores, our extension of the algorithm predicts pairwise preferences among the output. We also show how to take information about relevant data points that is shared across multiple rankings into account. For all $p \in \mathcal{P}$ let $f_p \in \mathcal{F}$ be a prediction function and let W_p be a relevance matrix. We write the minimization problem as

$$\begin{aligned}\min_{f_{p_1}, \ldots, f_{p_m}} \sum_{i=1}^{m} \sum_{k,j=1}^{n} [W_{p_i}]_{kj}((y_{x_k, p_i} - y_{x_j, p_i}) \\ -(f_{p_i}(x_k) - f_{p_i}(x_j)))^2 + \lambda \|f_{p_i}\|_{\mathcal{H}}^2.\end{aligned} \tag{5}$$

Since we consider all conjecture pairs as relevant, we have that $[W_{p_i}]_{kj} = 1$ for all $p_i \in \mathcal{P}, 1 \le j, k \le n$. Using the Representer Theorem we know that each prediction function f_p can be written as $f_p(x) = \sum_{i=1}^{n} a_{i,p} k(x, x_i)$. Let $A = (a_{i,p})_{i,p}$ with $1 \le i \le n, p \in \mathcal{P}$, i.e. A is the matrix where each column contains the parameters of one premise classifier , $K = (k(x_i, x_j))_{i,j}, 1 \le i, j \le n$ be the kernel matrix and let $Y = (y_{x,p})_{x,p}, x \in \mathcal{X}, p \in \mathcal{P}$. Similar to (Tsivtsivadze et al., 2010), we can rewrite the minimization problem in matrix notation as

$$\min_A tr\left((Y - KA)^{\mathrm{t}} L (Y - KA) + \lambda A^{\mathrm{t}} K A\right), \tag{6}$$

where L is the Laplacian matrix of the graph defined by the relevance matrix W. To minimize (5), we take the derivative with respect to A:

$$\begin{aligned}&\frac{\partial}{\partial A} tr\left((Y - KA)^{\mathrm{t}} L (Y - KA) + \lambda A^{\mathrm{t}} K A\right) \\ &= -2KL(Y - KA) + 2\lambda KA \\ &= -2KLY + (2KLK + 2\lambda K)A\end{aligned} \tag{7}$$

We set the derivative to zero and solve with respect to A:

$$A = (KLK + \lambda K)^{-1} KLY \tag{8}$$
$$= (LK + \lambda I)^{-1} LY \tag{9}$$

The last equality follows from the strict positive definiteness of $K \in \mathbb{R}^{n \times n}$.

It is interesting to note that a special case of our algorithm corresponds to multi-output regression, that is when $L = I$. In later sections we refer to the learning algorithm where L is constructed from W as MOR-ALLPAIRS and use the naming MOR-REGRESSION when $L = I$.

Using a square loss function leads to an efficient multi-output ranking solution, namely we obtain predictions for each output by inverting the kernel matrix only once and therefore the complexity of the algorithm is hardly increased compared to a standard single output problem.

2.2 SNoW

SNoW (Sparse Network of Winnows) is a machine learning toolkit (Carlson et al., 1999) which is used in the MaLARea (Urban et al., 2008) system. MaLARea using SNoW in naïve Bayes mode is currently 1.3 times stronger on the large-theory division of the MPTP Challenge benchmark[4] than other automated reasoning systems and meta systems.

SNoW is used mainly for natural language processing tasks, and is designed to work efficiently in domains where the number of features and targets is very large, which is useful for large theories with their large numbers of symbols and premises. SNoW implements several learning algorithms (Winnow, Perceptron, Naïve Bayes), and also comes with a preprocessor for efficient emulation of some first-order learning methods. An earlier preliminary evaluation of the machine learning methods available in the SNoW toolkit suggested using naïve Bayesian learning.

The setup for the training of SNoW is to select some suitable features characterizing the conjectures, and to try to learn the association of such features with the premises occurring in their proofs. The output features are the premises used in the proof of the conjecture, in particular, their ranking given by the activation weights.

2.3 APRILS

APRILS is a ranking method which was used in the Divvy system (Roederer et al., 2009). APRILS is based on Latent Semantic Analysis (LSA), a technique for analyzing the relationships between documents, using the terms they contain (Deerwester et al., 1990).

LSA is used to compute the relevance of premises to a conjecture by treating the formulas as documents, and the predicate and function symbols as the terms they contain. The computation of premise relevance using LSA is a three step process. First, a relationship strength between every pair of symbols is computed. An initial relationship strength is computed based on the co-occurrences of the symbols in the formulas, and the total number of formulas containing the symbols. The final relationship strength is computed by repeatedly combining the existing relationship strength with the relationship strengths between each of the two symbols and each other symbol, i.e., taking into account transitive relationships between symbols. Second, a relationship strength vector is computed for each formula. The vector has an entry for each symbol. A symbol's entry is the sum, across all other symbols, of the product of the relationship strength between the two symbols, and the number of occurrences of the other symbol in the formula (so that other symbols that do not occur in the formula make no contribution to the vector entry). Finally, the relevance of each premise to the conjecture is computed as the dot product of their symbol relationship strength vectors.

[4]http://www.tptp.org/MPTPChallenge/

3 EXPERIMENTS

In this section, we compare the rankings obtained from MOR-REGRESSION, MOR-ALLPAIRS, SNoW and APRILS. Our experiments are conducted on three subsets of the Mizar mathematical library (MML)[5]. We note that the MML was recently used to evaluate the performance of ATP systems[6]. An implementation of the MOR algorithm in Python, and all datasets are freely available at the authors website.

Each subset consists of several examples (conjectures). For each conjecture we are given a bipartite ranking of all premises, where the premises which were used in the proof of the conjecture have rank 1, and the premises that were not used in the proof have rank 0. We randomize the complete dataset and use 90% for training and reserve 10% for testing purposes. The goal is to learn to predict the ranking of the premises for the unseen examples (conjectures).

In our experiments we use the following settings. For simplicity and comparability of the results we

[5]The MML contains today over 1 100 formal mathematical articles, covering a substantial part of standard undergraduate mathematical knowledge. The library has about 50 000 theorems, proved with about 2.5 million lines of mathematical proofs. Such proofs often contain nontrivial mathematical ideas, sometimes evolved over decades and centuries of development of mathematics and abstract formal thinking. It is available at http://www.mizar.org

[6]http://www.tptp.org

choose the “bag of symbols” feature representation[7] for SNoW and the MOR algorithms. The features of a conjecture are the number of occurrences of the symbols of the conjecture. I.e. the feature map ϕ is defined as $\phi : \Gamma \rightarrow \mathbb{R}^n$ where n is the number of different symbols in the whole data set (the cardinality of the signature of Γ). Since APRILS only takes TPTP[8] input, we use this representation for the APRILS experiments.

As the kernel function, we use the so-called *Gaussian kernel* (Shawe-Taylor and Cristianini, 2004) which is parameterized by a single parameter σ. It is defined as

$$\begin{aligned} k(x,x') = &\exp\big[-\frac{1}{2\sigma^2}(\langle\phi(x),\phi(x)\rangle \\ &-2\langle\phi(x),\phi(x')\rangle + \langle\phi(x'),\phi(x')\rangle)\big] \end{aligned} \tag{10}$$

with $\langle\cdot,\cdot\rangle$ being the normal dot product on $\mathbb{R}^n$.

The kernel algorithm uses a 10-fold cross-validation on the training set to select the optimal parameters λ, σ. Once the parameters are estimated we train the algorithm on the complete training set and evaluate the performance on the test set. SNoW is used in naïve Bayes mode. APRILS has no parameters.

Finally, to compute the performance of an algorithm we create 100 randomized copies of each dataset. Each copy is split into a test (90%) and a training (10%) part. The AUC performance of the algorithm on a copy is the average of the AUC of the conjectures in the test set. The final performance is the average AUC performance over all 100 copies.

3.1 Performance Metrics

We present the performance measures used in the experiments.

3.1.1 AUC

Our basic performance measure is the AUC (see e.g. (Cortez and Mori, 2004)) - the area under the ROC curve. It can be interpreted as the probability that, given a randomly drawn positive example and a randomly drawn negative example, the decision function assigns a higher value to the positive example. Values closer to 1 show better performance. The AUC measure is appropriate for evaluating the performance of bipartite rankings.

Formally, let c be a classifier, $x_1,..,x_n$ be the output of c on the positive examples and $y_1,..,y_m$ be the output on the negative examples. Then, the AUC of c is

$$\text{AUC}(c) = \frac{\sum_i^n \sum_j^m 1_{x_i>y_j}}{mn} \tag{11}$$

where $1_\varphi = 1$ iff φ is true and zero otherwise.

3.1.2 Wilcoxon Signed-ranks

We use the Wilcoxon signed-ranks test (Wilcoxon, 1945) to test whether the AUC difference between two algorithms is statistically significant. The Wilcoxon signed-ranks test is a non-parametric statistical hypothesis test which ranks the differences in performance of two classifiers for each data set, ignoring the signs, and comparing the ranks for the positive and negative differences. The exact definition is as follows:

Let c_i^1 and c_i^2 denote the performance scores of the two classifiers on the $i-th$ of N data sets, and let $d_i = c_i^1 - c_i^2$ denote the difference. We rank the differences according to their absolute values. In case of a tie, average ranks are assigned. Let

$$R^+ = \sum_{d_i>0}\text{rank}(d_i) + \frac{1}{2}\sum_{d_i=0}\text{rank}(d_i) \tag{12}$$

$$R^- = \sum_{d_i<0}\text{rank}(d_i) + \frac{1}{2}\sum_{d_i=0}\text{rank}(d_i) \tag{13}$$

and

$$T = \min(R^+, R^-) \tag{14}$$

For large values of N, the statistics

$$z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} \tag{15}$$

is distributed normally. With $\alpha = 0.05$ we can assume that the ranking differences are not coincidental if $z < -1.96$ (Demšar, 2006).

3.2 Datasets

The evaluation is done on three datasets extracted from the large Mizar Mathematical Library using the MPTP system.

3.2.1 The MPTP Challenge Dataset

The MPTP Challenge is a MPTP-based benchmark established for large-theory automated reasoning in

[7] It has been demonstrated that suitable feature representation can significantly boost the performance of existing ATP techniques in large domains (Urban et al., 2008). Constructing an appropriate feature space for the learning algorithm is an important and relevant task, which is outside of the scope of this paper. We refer to (Tsivtsivadze et al., 2011) for the discussion on feature representations that have been previously used in automated reasoning systems.

[8] http://www.cs.miami.edu/~tptp/TPTP/SyntaxBNF.html

2006. This is a small subset (subtree) of the Mizar library leading to the general topological proof of the Bolzano-Weiestrass theorem (Bancerek et al., 2001) (formulated generally in terms of *nets* instead of countable sequences), and can be seen as the basic benchmark for premise selections algorithms. Earlier experiments have shown that learning over this dataset significantly helps with premise selection for ATP system (Urban et al., 2008).

The MPTP Challenge dataset consists of 211 conjectures. 341 different premises are used. The total number of used premises is 1227 which gives an average of 5.82 used premises per conjecture.

Table 1: AUC Results for the MPTP Dataset.

Algorithm	Av. AUC	Std. Deviation
MOR-REGRESSION	0.85588	0.03457
MOR-ALLPAIRS	0.81466	0.03680
SNoW	0.78685	0.04663
APRILS	0.85478	0.02629

Table 1 shows the average AUC and the standard deviation over 100 randomizations of all four algorithms on the MPTP dataset. In figure 1, we see the statistical significance of the results. Two algorithms are connected by a bold line iff the difference in their AUCs is insignificant. In this example the AUC difference between APRILS and the MOR-REGRESSION is insignificant.

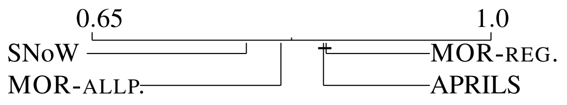

Figure 1: Comparison of the AUCs on the MPTP Dataset.

Figure 2 shows the detailed performance of the algorithms on the MPTP dataset. The diagonal plots show the histogram of the AUC performance on the 100 randomizations. The scatter plots show a pairwise comparison of the algorithms.

3.2.2 The Dataset of 370 Most used Mizar Premises

This dataset was created in order to have the most balanced positive/negative set of examples from MML. Use of premises is generally quite sparse in mathematics, leading generally to very unbalanced training examples. This small dataset contains the premises with the highest number of positive occurrences in the MML, allowing experiments with methods that require reasonably balanced data.

This dataset consists of 351 conjectures with 525 different premises. The total number of used premises is 2280, i.e. on average 6.50 used premises per conjecture.

Table 2: AUC Results for the 370 Dataset.

Algorithm	Av. AUC	Std. Deviation
MOR-REGRESSION	0.88868	0.02221
MOR-ALLPAIRS	0.84890	0.02809
SNoW	0.82823	0.03545
APRILS	0.85301	0.02194

Table 2 shows the average AUC and the standard deviation over 100 randomizations of all four algorithms on this dataset. In figure 3, we see the statistical significance of the results. In this dataset, the AUC difference between the MOR-ALLPAIRS and APRILS is statistically insignificant.

3.2.3 The Trigonometric Dataset

This is a dataset suitable for testing methods working with structural input features. The examples are created from a number of Mizar theorems about trigonometric functions. These theorems very often contain the same set of symbols, for example $\{sin, cos, tan, =, \pi, +, *\}$, and they thus mainly differ in the term and formula structure.

The trigonometric challenge dataset consists of 530 conjectures with a total of 1020 different premises. There are 5705 used premises altogether, which gives an average of 10.76 used premises per conjecture. Due to the formula structure and the number of premises, this dataset can be seen as the 'hardest' of the three.

Table 3: AUC Results for the Trigonometric Dataset.

Algorithm	Av. AUC	Std. Deviation
MOR-REGRESSION	0.93977	0.01181
MOR-ALLPAIRS	0.92964	0.01674
SNoW	0.77078	0.02166
APRILS	0.70676	0.02288

Table 3 shows the average AUC and the standard deviation over 100 randomizations of all four algorithms on the trigonometric dataset. In figure 4, we see the statistical significance of the results. Here, all AUC differences are statistically significant.

Figure 5 shows the detailed performance of the algorithms on the trigonometric dataset. The diagonal plots show the histogram of the AUC performance on

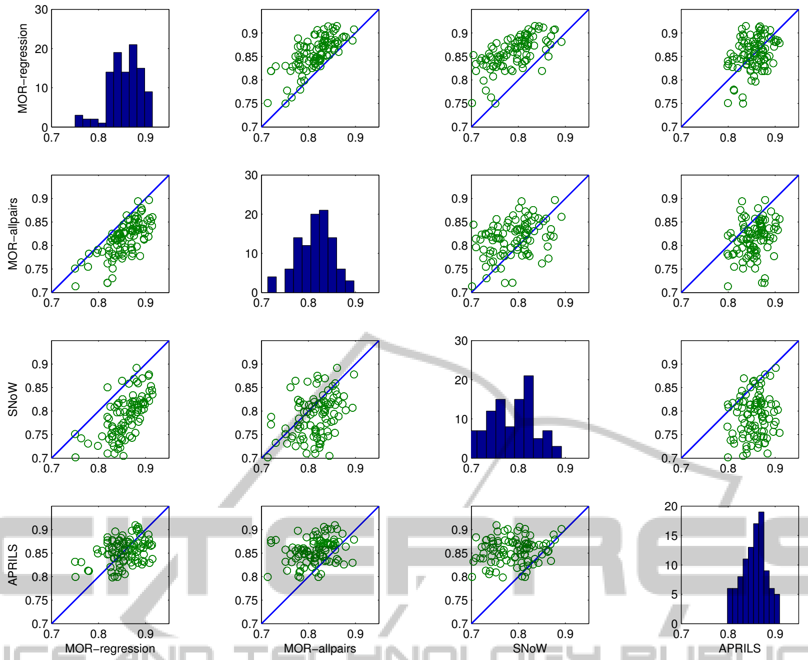

Figure 2: The results of the MPTP experiments.

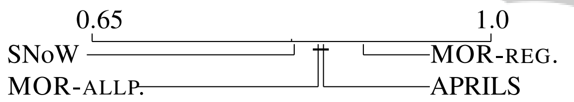

Figure 3: Comparison of the AUCs on the 370 Dataset.

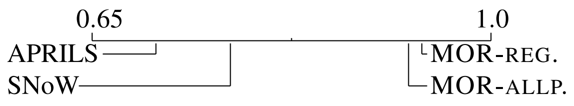

Figure 4: Comparison of the AUCs on the trigonometric Dataset.

the 100 randomizations. The scatter plots show a pairwise comparison of the algorithms.

4 DISCUSSION

On all three datasets, the MOR-REGRESSION algorithm outperform previously used methods for premise selection. While the performance of APRILS is not satisfactory on the largest dataset, both MOR-ALLPAIRS and MOR-REGRESSION perform better the more training data is available. It can be observed that SNoW performs quite well on the 370 dataset, but its average AUC score decreases again in the trigonometric dataset. We think that one reason for the good performance of the MOR-REGRESSION algorithm is the suitable formulation and use of a non-linear kernel function, namely a Gaussian kernel. We also note that the MOR-ALLPAIRS approach is in many cases inferior to MOR-REGRESSION. Apparently, ranking premises on conjectures does not lead to better performance when the final aim is to rank premises on conjectures.

Furthermore, for estimating the ranking of all premises the MOR algorithm requires time comparable to training a single binary classifier. Formally, given p binary classification tasks usually the training time of the algorithm is multiplied by the number of problems to be solved. For $p < n$, the by far most computational demanding operation is the inversion of the kernel matrix. However, training of the MOR algorithm requires only $O(n^3)$ for $p < n$. This corresponds to the time necessary to train a single RLS classification algorithm.

The AUC together with 100 randomizations and the Wilcoxon Signed-Ranks test seems like a very reasonable measure to compare different ranking algorithms, but it would also be interesting to test whether high AUC scores actually translate to better ATP performance. First experiments show that this is indeed the case. Experimentally, we plugged the MOR-REGRESSION algorithm into the MaLARea (Urban et al., 2008) system, and compare its speed and precision on the MPTP Challenge benchmark with MaLARea running with naive Bayes (SNoW) as a

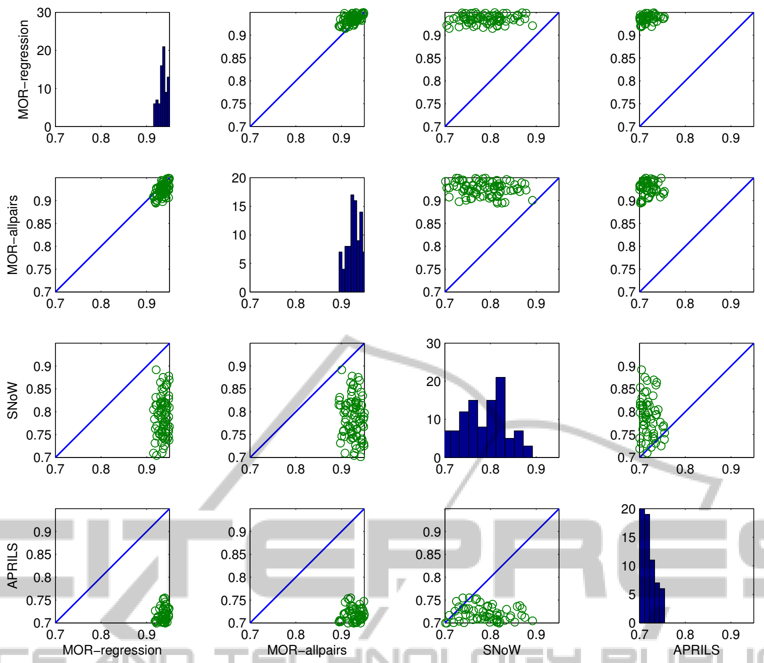

Figure 5: The results of the trigonometric experiments.

learning algorithm. In this first test, the MOR-REGRESSION algorithm solved more problems than SNoW while needing approximately the same amount of time. Furthermore, the number of problems solved by the system after the sixth fast one-second run using the MOR-REGRESSION-based premise selection outperforms any other non-learning (non-MaLARea-based) system run in 21 hours on the MPTP Challenge problems.

5 SIGNIFICANCE AND IMPACT

Trained mathematicians know a large number of theorems, solved problems, and tricks spanning many mathematical areas. They have also developed an intuition about the relevance of various parts of their knowledge for various new problems. Existing automated deductive tools typically attack problems by trying several human-programmed deductive strategies, typically in an exhaustive and (theoretically) complete way. This approach can be successful in limited domains and/or for reasonably easy tasks, it however does not scale to large complicated domains, where the search space grows enormously.

Our approach in this situation is to complement existing "theory-driven" implementations by developing suitable "data-driven" approaches for automated reasoning. Once large numbers of formally expressed theorems exist, they indeed complicate the exhaustive search methods of existing deductive tools. On the other hand, proofs of the large number of theorems provide a means to remedy the problem by learning and re-using previously successful ideas for steering the automated proof search methods. Suitable pre-selection of premises – and in particular their ranking according to their expected proof relevance – provides a way to efficiently combine existing ATP systems with external "intuitive" advice. This can lead to interesting combinations and feedback loops between intuition-assisted deductive finding of facts, and learning new intuitions from them. In this sense, large formal theories provide an opportunity for creating new smart AI methods and systems. Suitable learning methods for capturing the mathematical intuitions – as developed in this work – are a necessary prerequisite for building such smart mathematical assistants.

6 CONCLUSIONS AND FUTURE WORK

The contributions of this paper are threefold. First, we present premise selection for automated theorem proving as an interesting and challenging domain for machine learning. Second, we propose a framework for kernel based multi-output-ranking (MOR) and make it, and the datasets that we use, publicly available. Our MOR framework is much more computationally efficient compared to a binary classification approach. Note that although in this study we are primarily concerned with the automated reasoning domain, our method is general enough to be applicable to ranking tasks in bioinformatics, natural language processing, information retrial, etc. Third, we compare our framework with existing premise selection algorithms on three different datasets. The experiments show that our method significantly outperforms the existing algorithms, in particular on the harder problems.

In the future, we will first extract and then utilize feature representations of the premises in order to improve the ranking performance of the proposed algorithm. So far, we were only concerned with relatively small problems. Our biggest dataset had only 1020 distinct premises. Eventually, we would like to use our algorithm efficiently over datasets with tens or even hundreds of thousands of premises. Our final goal is to incorporate the developed algorithm into open source ATP systems which hopefully leads to notable benefits both in terms of accuracy and efficiency.

ACKNOWLEDGEMENTS

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

REFERENCES

Agarwal, S. (2005). *A study of the bipartite ranking problem in machine learning*. PhD thesis, University of Illinois at Urbana-Champaign, Champaign, IL, USA. AAI3182211.

Bancerek, G., Endou, N., and Sakai, Y. (2001). On the characterizations of compactness. *Journal of Formalized Mathematics*, 13(4):733–738.

Carlson, A., Cumby, C., Rizzolo, N., and Rosen, J. (1999). SNoW user manual. http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:SNoW+User+Manual#1.

Cortez, C. and Mori, M. (2004). AUC optimization vs. error rate minimization. *Advances in Neural Information Processing Systems*, 16:313.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30.

D'Silva, V., Kroening, D., and Weissenbacher, G. (2008). A survey of automated techniques for formal software verification. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(7):1165–1178.

Fuernkranz, J. and Huellermeier, E. (2011). *Preference Learning*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Gonthier, G. (2008). Formal Proof – The Four-Color Theorem. *Notices of the American Mathematical Society*, 55(11):1382–1393.

Hales, T. (2005). A proof of the Kepler conjecture. *Annals of Mathematics*, 162(3):1065–1185.

Hales, T. (2008). A Special Issue on Formal Proof. *Notices of the American Mathematical Society*, 55(11).

Hurd, J. (2003). First-order proof tactics in higher-order logic theorem provers. Technical report, Design and Application of Strategies/Tactics in Higher Order Logics, number NASA/CP-2003-212448 in NASA Technical Reports.

Klein, G., Elphinstone, K., Heiser, G., Andronick, J., Cock, D., Derrin, P., Elkaduwe, D., Engelhardt, K., Kolanski, R., Norrish, M., Sewell, T., Tuch, H., and Winwood, S. (2009). seL4: Formal Verification of an OS Kernel. In Anderson, T., editor, *Proceedings of the CICM Workshop on Empirically Successful Automated Reasoning in Mathematics*, pages 207–220, Big Sky, USA. ACM Press.

McCune, W. (1997). Solution of the Robbins Problem. *Journal of Automated Reasoning*, 19(3):263–276.

Meng, J. and Paulson, L. C. (2008). Translating Higher-order Problems to First-order Clauses. *Journal of Automated Reasoning*, 40(1):35–60.

Phillips, J. and Stanovsky, D. (2008). Automated Theorem Proving in Loop Theory. In Sutcliffe, G., Colton, S., and Schulz, S., editors, *Proceedings of the CICM Workshop on Empirically Successful Automated Reasoning in Mathematics*, pages 42–54, Birmingham, United Kingdom.

Poggio, T. and Smale, S. (2003). The Mathematics of Learning : Dealing with Data. *Notices of the American Mathematical Society*, 50(5):537–544.

Rifkin, R., Yeo, G., Poggio, T., Rifkin, R., Yeo, G., and Poggio, T. (2003). *Regularized Least-Squares Classification*, chapter 131-154. IOS Press.

Roederer, A., Puzis, Y., and Sutcliffe, G. (2009). Divvy: An ATP Meta-system Based on Axiom Relevance Ordering. *Automated Deduction - CADE-22*, pages 157–162.

Schoelkopf, B., Herbrich, R., Williamson, R., and Smola, A. J. (2001). A Generalized Representer Theorem. In Helmbold, D. and Williamson, R., editors, *Proceedings of the 14th Annual Conference on Computational Learning Theory*, pages 416–426, Berlin, Germany.

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*, volume 47. Cambridge University Press.

Sutcliffe, G. and Suttner, C. (2006). The state of CASC. *AI Communications*, 19:35–48.

Tsivtsivadze, E., Pahikkala, T., Boberg, J., Salakoski, T., and Heskes, T. (2010). *Co-Regularized Least-Squares for Label Ranking*, chapter 5, pages 107–123. Springer.

Tsivtsivadze, E., Urban, J., Geuvers, H., and Heskes, T. (2011). Semantic Graph Kernels for Automated Reasoning. In *SIAM Conference on Data Mining*.

Tsoumakas, G., Katakis, I., and Vlahavas, I. P. (2010). Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*, pages 667–685. editor.

Urban, J. (2006). MPTP 0.2: Design, Implementation, and Initial Experiments. *Journal of Automated Reasoning*, 37(1-2):21–43.

Urban, J., Hoder, K., and Voronkov, A. (2010). Evaluation of Automated Theorem Proving on the Mizar Mathematical Library. *Mathematical Software - ICMS 2010*, pages 155–166.

Urban, J. and Sutcliffe, G. (2008). ATP-based Cross-Verification of Mizar Proofs: Method, Systems, and First Experiments. *Mathematics in Computer Science*, 2(2):231–251.

Urban, J., Sutcliffe, G., and Pudlák, P. (2008). MaLARea SG1-Machine Learner for Automated Reasoning with Semantic Guidance. *Automated Reasoning*, pages 441–456.

van Gestel, T., a.K. Suykens, J., Baesens, B., Viaene, S., Vanthienen, J., Dedene, G., de Moor, B., and Vandewalle, J. (2004). Benchmarking Least Squares Support Vector Machine Classifiers. *Machine Learning*, 54(1):5–32.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.

Zhang, P. and Peng, J. (2004). SVM vs Regularized Least Squares Classification. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 1, pages 176–179. IEEE.