SMT Algorithm Selection with High-Level Natural-Language Descriptions

Zhengyang Lu[1], Paul Sarnighausen-Cahn[2], Jiahao Chen[3], Arie Gurfinkel[1], Florin Manea[2], and Vijay Ganesh[3]

[1] University of Waterloo, Canada
[2] University of Göttingen and CIDAS, Germany
[3] Georgia Institute of Technology, USA

Abstract. Machine-learning-based algorithm selection has been shown to improve SMT solver performance by exploiting solver complementarity across diverse problem families. Existing SMT algorithm selectors primarily rely on handcrafted syntactic features to represent problem instances for learning models. In this work, we explore whether high-level natural-language descriptions of SMT instances can provide additional useful information. We present `SMT-Select`, an SMT algorithm selection framework that combines traditional syntactic features with semantic embeddings derived from natural-language descriptions using pretrained transformer models, and incorporates an improved implementation of the algorithm selection pipeline compared to the state-of-the-art selector `MachSMT`. Evaluated on SMT-COMP 2024 benchmarks across 9 logics, `SMT-Select` achieves improved performance in 5 logics when description-based features are added, while performance noticeably degrades in one logic, suggesting that effectiveness depends on the underlying logic and the quality of available descriptions. We further outline future work on using LLM-based agents to generate higher-quality SMT instance descriptions, supported by preliminary demos.

Keywords: Algorithm Selection · SMT · Representation Learning.

1 Introduction

Satisfiability Modulo Theories (SMT) solvers decide the satisfiability of first-order logic formulas and are a core reasoning tool in formal verification and program analysis [2]. Due to the inherent difficulty and diversity of SMT problems, no single algorithm performs best across all SMT instances; instead, different solvers exhibit complementary strengths across different problem families. This motivates algorithm selection, which seeks to exploit solver complementarity by selecting the most suitable solver for each problem instance. Machine-learning (ML) based methods have emerged as an effective data-driven approach for this problem: `SATzilla` [11] pioneered ML-based algorithm selection in the SAT domain, and `MachSMT` [7] later extended the idea to SMT solving.

As in many ML applications, the effectiveness of algorithm selection critically depends on how problem instances are represented. Most algorithm selectors rely

on handcrafted, domain-specific features intended to capture properties of problem instances relevant to solver performance. For example, `MachSMT` used a set of 189 syntactic features, primarily consisting of symbol counts, to characterize SMT instances. Beyond handcrafted features, there has been a growing interest in leveraging advances in ML-based feature learning. Loreggia et al. [4] proposed learning instance representations by converting instance texts into grayscale images and applying convolutional neural networks, while `GraSS` [12] leveraged graph neural networks over abstract syntax trees of problem instances.

More recently, advances in transformer-based models for text representation have enabled feature learning directly from textual inputs [10]. In this direction, Pellegrino et al. [6] proposed using a transformer encoder to learn features for algorithm selection from high-level problem descriptions written in ESSENCE[3], a domain-specific language for modeling combinatorial optimization problems.

For SMT, raw SMT-LIB instances are typically too long to be used directly as transformer inputs. We therefore focus on high-level natural-language descriptions associated with SMT instances. These descriptions, often provided by instance creators, capture information such as instance origin, intended semantics, key constraints, and application context, but are inherently coarse and may omit instance-specific details. We therefore explore using such descriptions in conjunction with syntactic features that capture instance-specific information.

Our Contributions

1. We develop `SMT-Select`, an SMT algorithm selection framework that augments syntactic features with embeddings learned from natural-language instance descriptions. Across 9 tested SMT logics, this augmentation improves performance in 5 cases compared to using syntactic features alone.
2. `SMT-Select` implements an improved algorithm selection pipeline over the state-of-the-art `MachSMT`, yielding better performance with syntactic features alone in 8 of the 9 evaluated logics.
3. We outline a possible future direction involving LLM-based agents for generating and refining natural-language descriptions of SMT instances, and provide illustrative demos to demonstrate feasibility.

2 Algorithm Selection with Natural-Language Descriptions

Given an SMT instance, an algorithm selector selects the solver expected to perform best from a portfolio of candidate solvers. Figure 1 illustrates the inference-time workflow of `SMT-Select`. The selector first extracts feature representations from two sources: syntactic features are obtained using a dedicated feature extractor, while description-based features are derived from the corresponding natural-language description of the instance. These features are then combined into a unified feature representation of the SMT instance, which is used by the trained machine-learning model to predict the most suitable solver. The selected solver is subsequently executed on the instance.

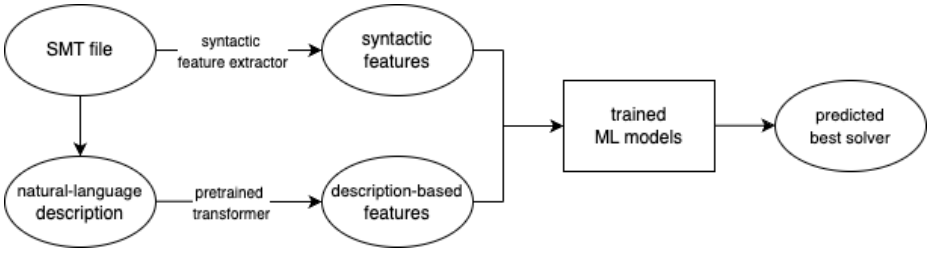

Fig. 1. Workflow of `SMT-Select` during inference for algorithm selection.

Description-based Feature Learning Transformer-based language models enable learning semantic representations from natural-language text by encoding inputs into fixed-dimensional vectors [10]. Recent pretrained models produce high-quality text embeddings without task-specific training and transfer effectively across downstream tasks. In this work, we use the pretrained sentence embedding model `all-mpnet-base-v2` [8] from the Sentence-Transformers library, which is based on the MPNet architecture [9]. The model encodes each natural-language description into a fixed-length 768-dimensional embedding, which is then used as input features for the algorithm selection model; descriptions that tokenize to more than 384 subword tokens are truncated by default. We do not fine-tune the language model for our task, leaving task-specific adaptation to future work.

Algorithm Selection Paradigm Given a feature representation of an SMT instance, `SMT-Select` adopts a cost-sensitive pairwise classification (PWC) approach [11] for algorithm selection. The selector is trained on performance data from all solvers over a set of training benchmarks, constructing a binary classifier for each solver pair to predict which solver performs better on a given instance. Training is cost-sensitive, assigning higher weights to instances with larger performance differences. At inference time, the pairwise classifiers vote for their preferred solvers, and the solver with the most votes is selected and executed.

3 Experiments

3.1 Experimental Setup

Following standard practice in ML-based algorithm selection, we train algorithm selectors using solver performance data on a training benchmark set, and evaluate them on a held-out test benchmark set. Our experiments are based on the SMT-COMP 2024 [1] non-incremental benchmarks and performance data, focusing on 9 selected SMT-COMP logics[4], with the goal of learning selectors

[4] Selected from 62 competition logics based on benchmark size, algorithm-selection potential, and availability of natural-language descriptions.

Table 1. PAR-2 VBS-SBS gap closed (%) by `MachSMT` and `SMT-Select` with different feature representations on the test benchmark set (for each logic, the result of the leading setting is highlighted in bold, while the second-best is underscored).

Logic	MachSMT (Syntactic)	SMT-Select		
		Syntactic	Desc.	Syntactic+Desc.
ABV	11.1	**92.4**	-3.7	<u>88.6</u>
ALIA	<u>56.7</u>	48.8	24.2	**61.1**
BV	-66.4	12.0	**57.0**	<u>32.1</u>
QF_IDL	-2.6	<u>62.9</u>	33.1	**69.1**
QF_LIA	59.5	<u>92.4</u>	**92.7**	92.3
QF_NRA	6.1	55.5	<u>64.5</u>	**79.5**
QF_SLIA	73.9	<u>94.0</u>	87.7	**97.3**
UFLIA	-138.0	**26.1**	-61.7	<u>-4.2</u>
UFNIA	-173.8	**39.1**	0.0	<u>37.7</u>

that choose among competition solver participants. For each logic, benchmarks are randomly split into 80% for training and 20% for testing, using performance data from all participating solvers for training and evaluation.

In our experiments, we evaluate `SMT-Select` under three feature configurations: syntactic features alone, description-based features alone, and their combination. Syntactic features consist of logic-specific symbol counts, while description-based features are embeddings of natural-language descriptions generated using the pretrained `all-mpnet-base-v2` model.

The implementation of our algorithm selection framework is adapted from Btor2-Select [5]. In contrast to `MachSMT`, which employs a regression-based empirical-hardness-model approach, our approach uses PWC instead, relying on simpler SVM models rather than AdaBoost. Our implementation achieves improved performance over `MachSMT` when using syntactic features alone, as demonstrated by our experimental results in Table 1.

3.2 Experimental Results

Table 1 reports test results for each logic, measured as the fraction of the PAR-2 performance gap between the virtual best solver (VBS) and the single best solver (SBS) closed by the algorithm selector. The VBS represents an ideal oracle that always selects the best solver per instance, while the SBS denotes the best individual solver. A larger fraction of the SBS–VBS gap closed indicates better performance, with 100% matching the VBS and negative values performing worse than the SBS.

Across the 9 evaluated logics, `SMT-Select` benefits from augmenting syntactic features with description-based features in 5 cases, where the combined representation outperforms syntactic features alone. In particular, for the BV and QF_NRA logics, description-based features by themselves substantially out-

perform purely syntactic features, suggesting that natural-language descriptions capture useful high-level information for these logics.

However, this effect is not uniform across all logics. For ABV, UFLIA, and UFNIA, selectors relying solely on description-based features perform markedly worse than those using syntactic features, which in turn degrades the performance of the combined feature representation. These results indicate that while natural-language descriptions can provide complementary information for algorithm selection, their effectiveness depends strongly on the logic and the quality of the available descriptions.

4 Conclusions and Next Steps

We investigate the use of natural-language description features for SMT algorithm selection and observe that their impact varies across logics. Looking forward, we plan to investigate the use of LLM-based agents to improve the quality of existing descriptions and to automatically generate descriptions for instances that lack them. We have built a prototype of such an agent and observed promising preliminary results. Interested readers may refer to an example of a generated description (example) and the agent implementation (repository).

References

1. Bromberger, M., Bobot, F., Jonáš, M.: SMT-COMP 2024. https://smt-comp.github.io/2024/ (2024), accessed: 2025-12-22
2. De Moura, L., Bjørner, N.: Satisfiability modulo theories: introduction and applications. Communications of the ACM (2011)
3. Frisch, A.M., Harvey, W., Jefferson, C., Hernández, B.M., Miguel, I.: Essence : A constraint language for specifying combinatorial problems. Constraints (2008)
4. Loreggia, A., Malitsky, Y., Samulowitz, H., Saraswat, V.A.: Deep learning for algorithm portfolios. In: AAAI (2016)
5. Lu, Z., Chien, P., Lee, N., Gurfinkel, A., Ganesh, V.: Btor2-select: Machine learning based algorithm selection for hardware model checking. In: CAV (2025)
6. Pellegrino, A., Akgün, Ö., Dang, N., Kiziltan, Z., Miguel, I.: Transformer-based feature learning for algorithm selection in combinatorial optimisation. In: CP (2025)
7. Scott, J., Niemetz, A., Preiner, M., Nejati, S., Ganesh, V.: Machsmt: A machine learning-based algorithm selector for SMT solvers. In: TACAS (2021)
8. Sentence-Transformers: all-mpnet-base-v2. `https://huggingface.co/sentence-transformers/all-mpnet-base-v2` (2021)
9. Song, K., Tan, X., Qin, T., Lu, J., Liu, T.: Mpnet: Masked and permuted pre-training for language understanding. In: NeurIPS 2020 (2020)
10. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NIPS (2017)
11. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Evaluating component solver contributions to portfolio-based algorithm selectors. In: SAT (2012)
12. Zhang, Z., Chételat, D., Cotnareanu, J., Ghose, A., Xiao, W., Zhen, H., Zhang, Y., Hao, J., Coates, M., Yuan, M.: GraSS: Combining graph neural networks with expert knowledge for SAT solver selection. In: KDD (2024)