Logic-based Learning in Software Engineering

Dalal Alrajeh*, Alessandra Russo†, Sebastian Uchitel†, Jeff Kramer*

*Imperial College London, UK
{dalal.alrajeh, a.russo, s.uchitel, j.kramer}@ic.ac.uk
† Departamento de Computación, Universidad de Buenos Aires and CONICET, Argentina

ABSTRACT

In recent years, research efforts have been directed towards the use of Machine Learning (ML) techniques to support and automate activities such as program repair, specification mining and risk assessment. The focus has largely been on techniques for classification, clustering and regression. Although beneficial, these do not produce a declarative, interpretable representation of the learned information. Hence, they cannot readily be used to inform, revise and elaborate software models. On the other hand, recent advances in ML have witnessed the emergence of new logic-based learning approaches that differ from traditional ML in that their output is represented in a declarative, rule-based manner, making them well-suited for many software engineering tasks.

In this technical briefing, we will introduce the audience to the latest advances in logic-based learning, give an overview of how logic-based learning systems can successfully provide automated support to a variety of software engineering tasks, demonstrate the application to two real case studies from the domain of requirements engineering and software design and highlight future challenges and directions.

Categories and Subject Descriptors


1. INTRODUCTION

Machine Learning (ML) has been shown to provide a promising approach to support and automate various software engineering (SE) activities such as program repair, specification mining and risk assessment. It has the potential to reduce human effort and potential errors. Furthermore, ML techniques have been applied to enable the design of software systems that are capable of exhibiting some form of intelligent behaviour, such as the understanding of the contextual environment in which they operate and the ability to adapt at run-time so as to maximize the achievement of overall system goals. Bayesian probabilistic reasoning has, for instance, been used to model software reliability [3] as well as users’ needs [6]. Computational search algorithms have been used to tackle a variety of software engineering problems, ranging from requirements and design to maintenance and testing [2] by reformulating software engineering problem as optimisation problems. Numerous traditional ML techniques have been used for modelling and predicting software costs predicting software defects [1], performing program repair [7] and mining quantitative knowledge from data related to past software engineering projects [9]. However, as noted in [5], most of these applications of ML can be seen as tasks of optimisation of processes or software products. The synergy between ML techniques and SE has the potential to go beyond this.

2. RELEVANCE TO THE SE COMMUNITY

Software engineering activities are predominately knowledge-intensive. Some of this knowledge is explicit at design time whilst some becomes apparent only after the deployment of the software within real environments. For example, in requirements engineering, knowledge about the domain is key to the development of correct specifications with respect to given system goals, and its absence may lead to significant system failures [10]. Domain knowledge is also relevant at run-time. Complex software systems are increasingly required to be context-aware and self-adaptive. In other words, they must be sufficiently intelligent, i.e., to know when to evolve and how to react to changes in the environment. To demonstrate intelligent behaviour, software need to be able to learn new knowledge which may be hidden in the effects of its interaction to its environment, in order to improve its behaviour over time and with experience. At the same time, adaptation must also be sensitive to human intervention and interpretable by humans. For example, pervasive software systems for mobile devices need to be able to continuously adapt to user’s preferences and behaviour with little or no intervention by the user, yet also facilitate user validation [8]. How can relevant knowledge be automatically extracted, integrated into software specifications at design time, and into software behaviour at run-time whilst also expressed in way that is accessible to users if and when required?

Recent advances in Artificial Intelligence have witnessed the development of new ML approaches, called logic-based learning methods [4]. They differ from traditional ML ap-
proaches in that (1) data, problem and generated hypothe-
sis are all represented in a declarative way, thus providing
means for interpretable computation and making it easier to
inspect and modify; (2) they support integration of human
expertise when determining the scope of the solution space;
(3) do not suffer from the problem of overfitting when han-
dling small examples; and (4) produce alternative solutions
if any exist.

3. CONTENT

The aim of this technical briefing is to introduce researchers,
practitioners and educators in software engineering to the
latest advances in logic-based learning and to show how
these new approaches can be used to support a variety of
knowledge-intensive SE tasks. The technical briefing will be
composed of three parts.

In the first part, we will present the general principles of
logic-based learning, focusing on key features of state-of-the-
art learning approaches that make them particularly suited
to SE. We will answer the main question How logic-based
learning works. We will also include presentation of learning
systems and demonstrate their learning capabilities.

The second part will provide an overview of successful ap-
plications of logic-based learning in different SE tasks, rang-
ing from diagnosis and repair of requirements specifications,
generation of assumptions about the environment, adapta-
tion to users’ behaviour and forensic readiness. We will high-
light, in particular, how to apply a logic-based learning sys-
tem to a SE task, and discussing advantages and limitations.
In the third party, we will present two real case studies from
the domain of requirements elaboration and software design
that, although different in nature, they have been addressed
using the same underlying logic-based learning system.

4. TARGET AUDIENCE

The technical briefing is suitable for researchers, practi-
tioners and educators. No prior knowledge of logic-based
learning or ML is required. While some technical aspects of
logic-based learning will be covered, the overall technical
level of the presentations will be accessible to people not
knowledgeable in the area. The audience will be referred
to further readings, recommended for those who want to
acquire in depth details about these approaches.

5. ABOUT THE ORGANISERS

Dalal Alrajeh: Research Fellow, Department of Comput-
ing, Imperial College London, and a visiting lecturer at the
Department of Security and Crime Science at UCL. Her
main research interests are in requirements engineering, di-
agnosis and correction of behavioural specifications and in-
ductive logic programming, and their application to the de-
velopment of crime intelligence systems. She is the Deputy
Editor-in-Chief of the IET Software Journal, and served as
PC member of ICSE, RE, ILP, KR and SEFM.

Alessandra Russo: Reader in Applied Computational Logic,
Department of Computing, Imperial College London, and
head of the Structured and Probabilistic Knowledge En-
gineering research group. She has pioneered logic-based
learning algorithms and systems and gained a recognised
track record on their application to policy-based manage-
ment systems, security and software engineering. She is
currently Chair of ICSE 2017. Dr Russo has been distin-
guished with the Suresearch Prize, ERC StG, the Konex Foundation
Prize and the Houssay Prize.

6. REFERENCES

learning based software defect prediction techniques.

as a search problem. IEEE Proceedings - Software,

[3] M. Harman. The role of artificial intelligence in
software engineering. In Proc. of the 1st International
Workshop on Realizing AI Synergies in Software

user modeling for inferring the goals and needs of
software users. In Proc. of the 14th Conference on
Uncertainty in Artificial Intelligence, pages 256–265,
1998.

program repair: Fixing 55 out of 105 bugs for $8 each.
In Proc. of the 34th International Conference on

mobile domains. In Latest Advances in Inductive Logic

engineering and knowledge engineering. In Handbook
of Software Engineering and Knowledge Engineering.

System Goals to UML Models to Software