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

Software and its engineering [**Software creation and management**]: [Designing software]; Computing methodologies [**Machine learning**]: Machine learning approaches—*Logical and relational learning*.

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 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 differs from traditional ML ap-

ICSE '16 May 14-22, 2016, Austin, TX, USA

ACM ISBN 978-1-4503-4205-6/16/05.
DOI: http://dx.doi.org/10.1145/2889160.2891050

proaches in that (1) data, problem and generated hypothesis are all represented in a declarative way, thus providing means for interpretable computation and making it easier to inspect and modify; (2) they supports integration of human expertise when determining the scope of the solution space; (3) do not suffer from the problem of overfitting when handling 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 applications of logic-based learning in different SE tasks, ranging from diagnosis and repair of requirements specifications, generation of assumptions about the environment, adaptation to users' behaviour and forensic readiness. We will highlight, in particular, how to apply a logic-based learning system 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, practitioners 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 Computing, 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, diagnosis and correction of behavioural specifications and inductive logic programming, and their application to the development 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 Engineering research group. She has pioneered logic-based learning algorithms and systems and gained a recognised track record on their application to policy-based management systems, security and software engineering. She is editor-in-Chief of IET Software journal, associate editor of ACM Computing Survey, and PC member of ESEC/FSE, FSE, KR, IJCAI, ICLP, AAAI. General Chair of ILP 2016.

Jeff Kramer: Professor at Imperial College London. His research work is primarily concerned with software architecture, behaviour analysis, models in requirements engineering and architectural approaches to adaptive software systems. Jeff was Program Co-chair of ICSE99, ICSE Steering Committee Chair (2000-2002), and General Co-chair of ICSE 2010. Editor in Chief of IEEE TSE (2006-2009). Jeff was awarded the 2005 ACM SIGSOFT Outstanding Research Award and the 2011 ACM SIGSOFT Distinguished Service Award. He is a Fellow of the RAE and of the ACM.

Sebastian Uchitel: Professor at University of Buenos Aires, and Reader at Imperial College London. His research interests are in behaviour modelling, analysis and synthesis applied to requirements engineering, software architecture and design and adaptive systems. Dr Uchitel was associate editor of the TSE Journal, and currently associate editor of the RE Journal and the Science of Computer Programming Journal. Program co-chair of ASE06 and ICSE10, and General Chair of ICSE17. Dr Uchitel has been distinguished with the Philip Leverhulme Prize, ERC StG, the Konex Foundation Prize and the Houssay Prize.

6. REFERENCES

[1] V. Challagulla et al. Empirical assessment of machine learning based software defect prediction techniques. 17:389–400, 2008.

[2] J. Clark and et al. Reformulating software engineering as a search problem. *IEE Proceedings - Software*, 150:161–175, 2003.

[3] N. Fenton and et al. On the effectiveness of early life cycle defect prediction with bayesian nets. *Empirical Software Engineering*, 13:499–537, 2008.

[4] S. Gulwani et al. Inductive programming meets the real world. *Commun. ACM*, 58(11):90–99, 2015.

[5] M. Harman. The role of artificial intelligence in software engineering. In *Proc. of the 1st International Workshop on Realizing AI Synergies in Software Engineering*, 2012.

[6] E. Horvitz and et al. The lumiere project: Bayesian 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.

[7] C. Le Goues et al. A systematic study of automated program repair: Fixing 55 out of 105 bugs for $8 each. In *Proc. of the 34th International Conference on Software Engineering*, pages 3–13, 2012.

[8] A. Markitanis et al. Learning user behaviours in real mobile domains. In *Latest Advances in Inductive Logic Programming*, pages 43–51. Imperial College Press, 2015.

[9] T. Menzies. Practical machine learning for software engineering and knowledge engineering. In *Handbook of Software Engineering and Knowledge Engineering*. 2001.

[10] A. van Lamsweerde. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, 2009.