FLOC 2018:
FEDERATED LOGIC CONFERENCE 2018

MLP ON THURSDAY, JULY 19TH

Days: previous day all days

View: session overview talk overview side by side with other conferences

09:00-10:30 Session 131C

CHAIR: *Viktor Kuncak*

LOCATION: Blavatnik LT1

09:00 *Prem Devambu*

Why is Software Natural? and how can Naturalness be exploited?

ABSTRACT. Sometime during the Summer of 2011, several of us at UC Davis were quite puzzled and shocked to discover that software is "natural", viz., just as repetitive and predictable as natural language corpora; In fact much more so! By now, this early experiment has been replicated many times, in many ways, and various applications of naturalness have been developed. But why is this? Is it just because of programming language syntax? or is it due to something else, like conscious programmer choice? How can we study this question? Are there are other "natural" corpora (other than software) that are similar to software?

09:45 *Swarat Chaudhuri*

Program Synthesis as High-Level Machine Learning

ABSTRACT. The area of deep learning has had

many remarkable successes in the recent past. However, deep neural networks have some well-documented weaknesses: they are data-hungry, brittle, opaque to users, hard to analyze, and not easily constrained with axiomatic knowledge about the world. In this talk, I will argue that ideas from programming languages and program synthesis can offer a way of overcoming these weaknesses. Specifically, one can use higher-level programming languages, possibly with access to neural "subroutines", to describe statistical models normally described neurally. Such programmatic models are more easily understood by humans, can more easily accommodate axiomatic knowledge about the world, and are easier to analyze using symbolic techniques. The use of "neurosymbolic", as opposed to purely symbolic, models can lead to lower complexity of learning. The compositionality inherent in modern languages can allow transfer of knowledge across learning tasks. Discovering such programmatic models from data is a form of program synthesis, and can perhaps also be described as "high-level machine learning". Early experience with the problem suggests that the literature on language-integrated program synthesis can offer powerful tools for solving this problem. At the same time, the problem is different from traditional synthesis in key ways, and opens up many new technical challenges.

10:30-11:00 Coffee Break

11:00-12:30 Session 133C

Deep Learning for Code

CHAIR: *Viktor Kuncak*

LOCATION: Blavatnik LT1

11:00 *Eran Yahav*

code2vec: Learning Distributed Representations of Code

11:45 *Miltos Allamanis*

Understanding & Generating Source Code with Graph Neural Networks

ABSTRACT. The rich structure of source code presents an interesting challenge for machine learning methods. Recently, Graph Neural Networks (GNN) have shown promising results in

code understanding and code generation tasks. In this talk, I will briefly discuss two neural models that employ GNNs: one of them for catching variable misuse bugs and the other for generating code expressions. Finally, I will discuss some of the open challenges that GNNs face on many source code-related tasks.

12:30-14:00 Lunch Break

14:00-15:30 Session 135C

CHAIR: *Viktor Kuncak*

LOCATION: Blavatnik LT1

14:00 *Danny Tarlow*

Neural Network Models of Code Edits

ABSTRACT. I'll discuss some of our recent efforts towards building neural network models of edit sequences, with an eye towards casting source code autocompletion as learning to edit code. I'll frame the problem and explain why it's a bit different from other related formulations and then describe a new attention-based model for the problem. I'll show results on carefully designed synthetic data and a large dataset of fine-grained edit sequences gathered from thousands of professional software developers writing code.

14:45 *Pushmeet Kohli*

Program synthesis and its connections to AGI

ABSTRACT. In this talk, I will address the questions of

1. how we specify arbitrary tasks to a learning system
2. how we interpret its behaviour, and finally
3. how do we verify or debug it to ensure that its behaviour is consistent with the task specification.

I will also describe my initial attempts to make progress on these questions through program synthesis and verification.

15:30-16:00 Coffee Break

16:00-18:00 Session 137C

CHAIR: *Viktor Kuncak*

LOCATION: Blavatnik LT1

16:00 *Mukund Raghothaman*, *Sulekha Kulkarni*, *Richard Zhang*, *Xujie Si*, *Kihong Heo*, *Woosuk Lee* and *Mayur Naik*

Difflog: Beyond Deductive Methods in Program Analysis

SPEAKER: *Mukund Raghothaman*

ABSTRACT. Building effective program analysis tools is a challenging endeavor: analysis designers must balance multiple competing

objectives, including scalability, fraction of false alarms, and the possibility of missed bugs. Not all of these design decisions are optimal when the analysis is applied to a new program with different coding idioms, environment assumptions, and quality requirements. Furthermore, the alarms produced are typically accompanied by limited information such as their location and abstract counter-examples. We present a framework Difflog that fundamentally extends the deductive reasoning rules that underlie program analyses with numerical weights. Each alarm is now naturally accompanied by a score, indicating quantities such as the confidence that the alarm is a real bug, the anticipated severity, or expected relevance of the alarm to the programmer. To the analysis user, these techniques offer a lens by which to focus their attention on the most important alarms and a uniform method for the tool to interactively generalize from human feedback. To the analysis designer, these techniques offer novel ways to automatically synthesize analysis rules in a data-driven style. Difflog shows large reductions in false alarm rates and missed bugs in large, complex programs, and it advances the state-of-the-art in synthesizing non-trivial analyses.

16:30 *Milan Cvitkovic*, *Badal Singh* and *Anima Anandkumar*

Deep Learning On Code with an Unbounded Vocabulary

SPEAKER: *Milan Cvitkovic*

ABSTRACT. A major challenge when using techniques from Natural Language Processing for supervised learning on computer program source code is that many words in code are neologisms. Reasoning over such an unbounded vocabulary is not something NLP methods are typically suited for. We introduce a deep model that contends with an unbounded vocabulary (at training or test time) by embedding new words as nodes in a graph as they are encountered and processing the graph with a Graph Neural Network.

17:00 *Vadim Markovtsev*, *Waren Long*, *Egor Bulychev*, *Romain Keramitas*, *Konstantin Slavnov* and *Gabor Markowski*

Splitting source code identifiers using Bidirectional LSTM Recurrent Neural Network

SPEAKER: *Vadim Markovtsev*

ABSTRACT. Programmers make rich use of natural language in the source code they write through identifiers and comments. Source code identifiers are selected from a pool of tokens which are strongly related to the meaning, naming conventions, and context. These tokens are often combined to produce more precise and obvious

designations. Such multi-part identifiers count for 97% of all naming tokens in the Public Git Archive - the largest dataset of Git repositories to date. We introduce a bidirectional LSTM recurrent neural network to detect subtokens in source code identifiers. We trained that network on 41.7 million distinct splittable identifiers collected from 182,014 open source projects in Public Git Archive, and show that it outperforms several other machine learning models. The proposed network can be used to improve the upstream models which are based on source code identifiers, as well as improving developer experience allowing writing code without switching the keyboard case.

17:30 *Bruno Marnette*

Open Business Meeting for Future of Machine Learning for Programming

---