CS 6704: CTTC 9

Sourabh Shetty

code2vec: Learning Distributed Representations of Code

I'm not very familiar with deep learning, but this was an immensely interesting paper to read. It really expanded what I imagined code was capable of. After reading the paper I visited the website (https://code2vec.org/) and saw the examples they provided and it seemed incredibly advanced.

I thought the authors did a good job training on 14 million methods, which would cover a wide variety of use cases. I wish the paper had delved more into the potential applications of code2vec beyond just explaining it. I think a potential use case could be if a user wrote a function but there was a minor logical mistake that the user was unable to figure out, code2vec could be run on it and it could find the closest matching AST, and generate code that the user could use to refer to in order to rewrite their function.

As a future research idea, I think there could be immense potential to use a similar methodology to categorize conversation transcripts. This could have use in national security purposes since a lot of manpower is required to listen and determine whether a conversation is benign or suspicious, so if trained well, a similar application could describe conversations and help determine whether a phone call or online conversation required further scrutiny.

Deep Learning Similarities from Different Representations of Source Code

I like the idea the authors had for this paper. This was especially true after reading the paper, because it had made me wonder what representation of code would be best to check for similarity. I had originally been thinking only in terms of AST and identifiers, so I liked that the authors even evaluated Bytecode.

The research questions were also quite thorough and extensive and covered a decent breadth of possible results. I would have liked to see a small section comparing the speed of processing for each of the different code representations.

A potential future research idea would be to evaluate even more code representation formats and compare them across multiple platforms and languages, since this one only looks at Java code.

An Empirical Investigation into Learning Bug-Fixing Patches in the Wild via Neural Machine Translation

I enjoyed reading this paper because last semester, my project for the 5000-level Software Engineering course involved detection of bugs from code and had a proposed outcome of automatically suggesting fixes to common bugs.

The scope of our project was relatively limited so we hadn't even considered using deep learning, and used a simple database along with gumtree to look at edit actions. Due to this, I can vouch for gumtree and know that this methodology does indeed work.

As a further research proposal, I would recommend actually doing what we had proposed in my project proposal last semester, and find a way to automatically fix common bugs by generating the AST and replacing identifiers wherever necessary.