On the Naturalness of Software

Summary

The authors of the paper looked in to the conjecture that most software is natural, where natural means a naturual product of human. The goal of looking at this problem is to examine whether methods used for NLP could also apply to code in software. The authors used the n-gram language models on some open source Java programs and looked at the result which showed that software is more regular as predictable compared to english. Some more digging into the problem, the author noticed that the n-gram model's suggestions are project-specific about half the time. With the n-gram model, the authors went on and build a tool for eclipse for code completion, and by empirical investigation, the new tool out performs the build in completion tool by a large margin.

Things I would like to see discussed

- The research is conducted on fairly high level languages which I would argue part of the goal is for human readable code. If it's human readable it sort of represents a real language. I wonder what would the results be if it's conducted on a different style of language such as assembly or functional.

- I'm not very familiar with NLP, but to my understanding the ultimate goal of NLP is to have the computer really understand natural language. If the conjecture is true that software is also a "natural" language, could it imply that if we were to have one day such that machines could understand natural language, it could also understand code? (Personally, I think code will be first understood by machine first since machine could read code)

- NLP is fast developing. Now Google can even auto-complete sentences. Why aren't there tools for autocompleting code segments? Is it because code are much more harder to understand and context specific?