TITLE: “Natural Software Revisited”
WHO: Musfiqur Rahman and thesis advisor (Peter C. Rigby)
Please see Chapter 3 only of Rahman’s thesis:
https://spectrum.library.concordia.ca/983846/1/Rahman_MComp_F2018.pdf

WHAT was replicated: A. Hindle, E. T. Barr, Z. Su, M. Gabel, and P. Devanbu. On the naturalness of software. In Proceedings of the 34th International Conference on Software Engineering, ICSE ’12, pages 837–847, Piscataway, NJ, USA, 2012. IEEE Press

WHY: Hindle et al concluded that software is more repetitive and predictable, i.e. more natural, than English texts. We find that much of the apparent “naturalness” is artificial and is the result of language specific tokens (syntax). Since code is usually represented as an AST, we also study non-sequential, graph representations of code to understand the impact of representation on repetitiveness.

HOW: We downloaded a representative sample of project for seven programming languages including Java. We replicated Hindle et al’s result using the same measure of self-cross entropy. We also examine the degree of repetition in technical English on StackOverflow.

WHERE: We replicated all parts of Hindle et al. We also add three new questions. How repetitive is code without syntax tokens, how repetitive are API calls, and how repetitive are non-sequential graph representations of code.

DISCUSSION: Recent works have concluded that software is more repetitive and predictable, i.e. more natural, than English texts. We find that much of the apparent “naturalness” is artificial and is the result of language specific tokens. For example, the syntax of a language, especially the separators e.g., semi-colons and brackets, make up for 59% of all uses of Java tokens in our corpus. Furthermore, 40% of all 2-grams end in a separator, implying that a model for autocompleting the next token, would have a trivial separator as top suggestion 40% of the time. By using the standard NLP practice of eliminating punctuation (e.g., separators) and stopwords (e.g., keywords) we find that code is less repetitive and predictable than was suggested by previous work. We replicate this result across 7 programming languages.

Unlike the code written for a particular project, API code usage is similar across projects. For example a file is opened and closed in the same manner irrespective of domain. When we restrict our n-grams to those contained in the Java API we find that the entropy for 2-grams is significantly lower than the English corpus. This repetition perhaps explains the successful literature on API usage suggestion and autocompletion.

We study the impact of the representation of code on repetition. The n-gram model assumes that the current token can be predicted by the sequence of n previous tokens. When we extract program graphs of size 2, 3, and 4 nodes we see that the abstract graph representation is much more concise and repetitive than the ngram representations of the same code. This suggests that future work should focus on graphs that include control and data flow dependencies and not linear sequences of tokens.