Detecting License Inconsistencies Based on Token Size in Open Source Software

Daniel Kim
University of California, Berkeley
Software Engineering Laboratory, Professor Katsuro Inoue
Graduate School of Information Science and Technology

Abstract

Free open source software (FOSS) is an integral part of modern day software engineering. It allows for the reuse of source code in other pieces of software. To facilitate and stipulate the conditions of reuse of the source code, such code usually have one or more software licenses, found in a license file or in the header of the source file. During software reuse, developers and authors of the source code may add, alter, or remove licenses from the source code, creating inconsistencies in licenses between new and old versions of source code. The purpose of this research is to validate previous research conducted by the Software Engineering Laboratory and to explore the relationships between minimum token size for code clone detection, detection of license inconsistencies, and types of license inconsistencies. The main target of the research is the source files found in Debian 8.2. Python 2.7 was used to run scripts that analyzed the source code using CCFX, a code clone detector, Ninka, a license detector, and SQLite3, a lightweight database management system. In addition, manual inspection was used to verify if detected code clones are actual clones instead of chance resemblances. Results were largely predictable, that is, that as minimum token size increased, detected license inconsistencies decreased. However, results also showed that the average number of snippets found in a given clonesets decreased very slightly as token size increased heavily, implying that a large number of detected clones are chance resemblances, further reinforced by manual inspection. The most common type of license inconsistency, license change, also drops rapidly as token size increases. In conclusion, token size does influence the detection of license inconsistencies and type of license inconsistency.

1 Introduction

Free open source software (hereon on referred to as FOSS) has become an essential component of software development. It allows for the free reuse and alteration of software components and in order to regulate its usage based on the original developer's wishes, licenses naturally followed. Throughout the lifetime of such software, these licenses may change due to either the developer or the author's wishes, creating license inconsistencies. That is, source files from the same or different packages that contain the same code but different licenses.

These license inconsistencies, if legally invalid, could pose as license violations and should be changed promptly. Due to the prevalence of software reuse and difficulties of maintaining different versions of software, there are likely many license inconsistencies and possibly license violations.

2 Related Work

Work exploring the existence and classification of license inconsistencies has been conducted in large part by the Software Engineering Laboratory at Osaka University. [1] Their research has shown the license inconsistencies found in the source files for Debian 7.5 and several large projects written in Java on Github.

Motivation for this research comes in part from their research in part to validate previous research as well as to explore the relationship between token size and both the amount and types of license inconsistencies found.

3 Methodology

The target of this research is the code base for Debian 8.2. Unfortunately, due to time constraints, a full-scale analysis of the entire code base has proved infeasible and instead this research will focus on a small subset of the code base. The files in question are written in C/C++ and Java.

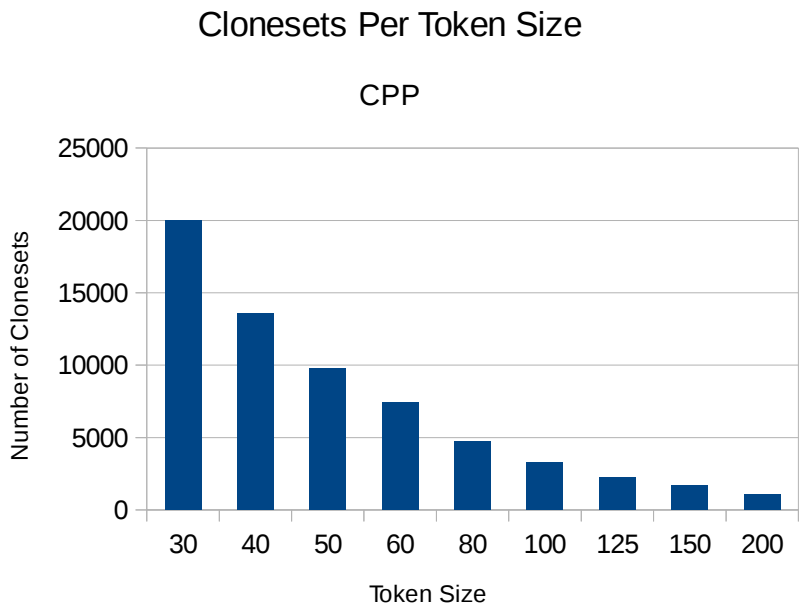
In order to detect source code where reuse has occurred, this research will be using a tool called CCFinderX (CCFX). This tool detects both Type 1 and Type 2 code clones and analyzed the target. The output of CCFX was then analyzed with a script written in the Python language and the appropriate data stored in a SQLite3 database.

During this procedure, another tool developed by the Software Engineering Laboratory at Osaka University, called Ninka, was used to identify the license of a source code file. Lastly, all files that were determined to be code clones according to CCFX but with different licenses were compiled. A few code clones were also randomly chosen for manual inspection. The results are as follows.

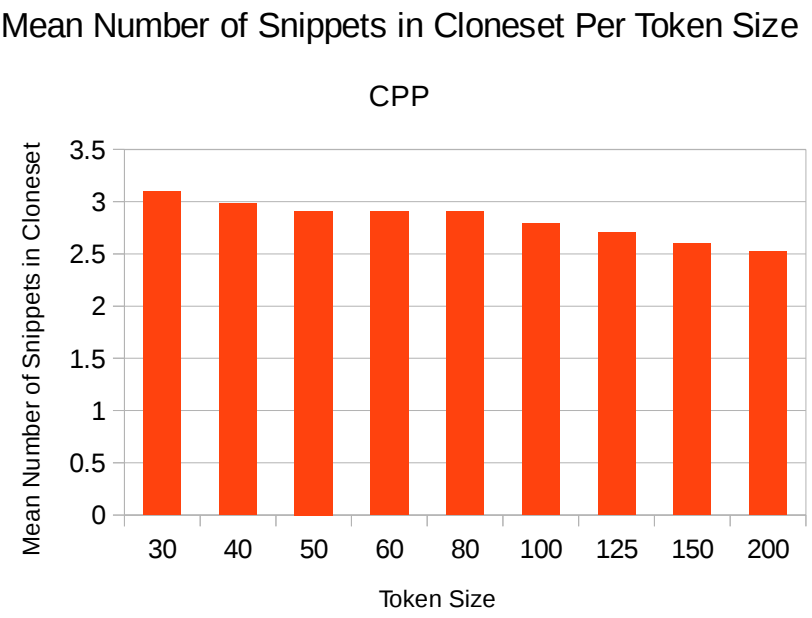
4 Results

Type of File	Number of Files
C/C++	6109
Java	439

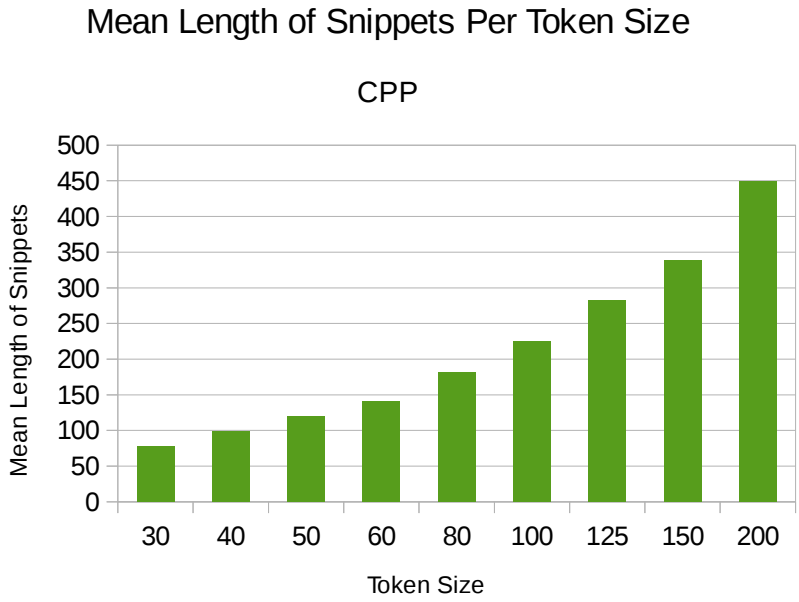
The above table describes the number of files found in the target based on file type. The following graphs illustrate the resulting data based on token size. Nothing is shown for Java files as there were no license inconsistencies found.



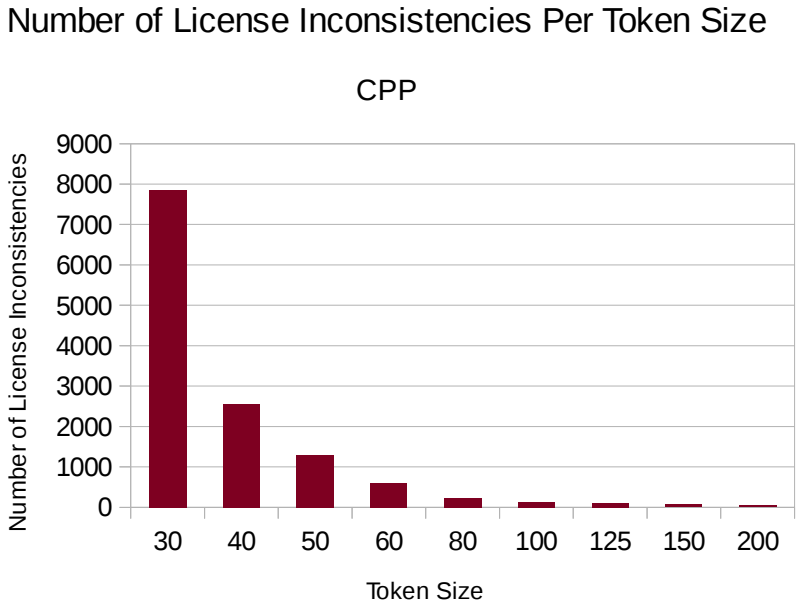
As seen in the first graph, as token size increases, the number of clonesets, that is, the number of sets of code clones as determined by CCFX decreases. This is expected, as the threshold for what is considered to be a code clone is increased.



In the second graph, as token size increases, the mean number of snippets in cloneset decreases. Interestingly, the decrease is relatively small, given that the average number of snippets per clonesets decreases from about 3 to about 2.5.

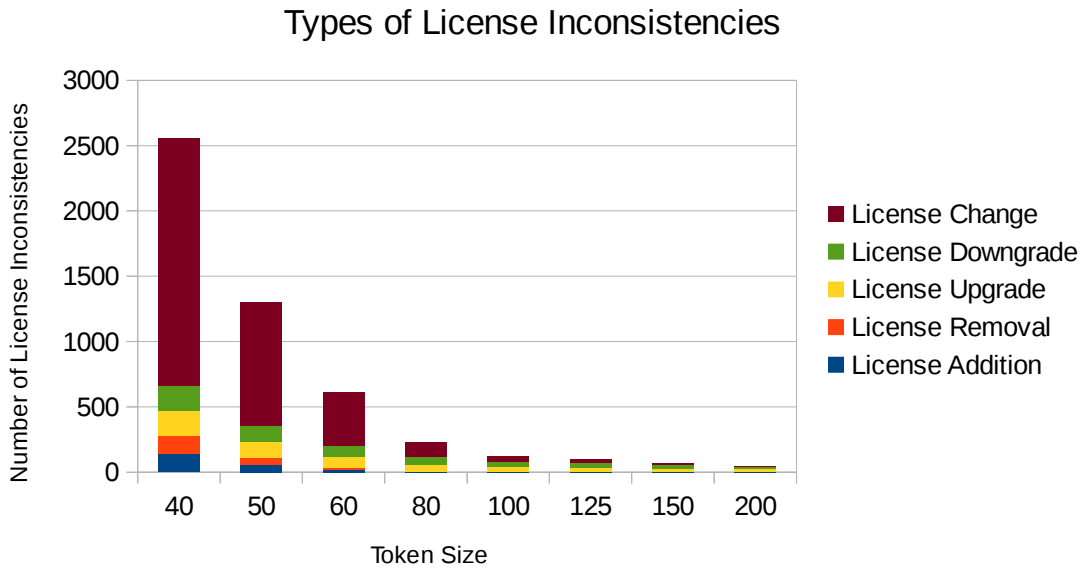


In the third graph, as token size increases, the mean length of snippets increases as well. This is expected, as the minimum threshold for what can be considered a code clone increases and thus the length of each snippet increases.



As seen in the fourth graph, the number of license inconsistencies decreases dramatically as token size increases. This result is surprising, as this implies that a large amount of license inconsistencies are found as token size decreases. To further elaborate on these detected license inconsistencies, they were classified by previous research [1] as follows.

1. License Addition: The source file was without a license, and a license is added in at a later time.
2. License Removal: The source file was under a certain license, and the license is removed at a later time.
3. License Upgrade: The source file was under a certain version of the GPL license, a license that allows an upgrade, and it is upgraded to a newer version of the license.
4. License Downgrade: The source file was under a certain version of a license and has been downgraded to an older version of the same license.
5. License Change: The source file was under a certain license, and it is changed to another license.



In the fifth and last graph, we see that a large portion of these license inconsistencies are license changes for smaller token sizes. However, they dramatically decrease in number as token size increases. Also interestingly, the number of instances of license removal and addition also appear to reduce to a very small percentage of total license inconsistencies as token size increases.

5 Conclusion

In conclusion, it appears that smaller token sizes seem to include a large portion of false positives. Upon manual inspection of several snippets of shorter lengths, they were appeared to be clones that were similar by chance instead of intentional reuse of software. Based on the results as seen in the relatively stable mean number of snippets per cloneset and the dramatic decrease in detected license inconsistencies as token size increases, it can be safely concluded that a token size of 30 does not adequately detect license inconsistencies.

However, specifically which token size is optimal for detecting license inconsistencies was unable to be determined from this research. Statistical analysis of each token size would be required to see clearly which token size excludes the most false positives while including the most true positives.

6 Further Research

There is room for much future research. Given the time constraints of this research term, a future full-scale analysis of the entire code base of Debian 8.2 would prove useful. In addition, analyzing different versions of Debian and tracking changes in the code base over time may provide interesting results.

There also can be analysis of which sorts of licenses experience which type of license inconsistency. Further research can analyze the relationship between token size and the prevalence of different types of license inconsistencies for specific types of licenses.

Lastly, there could be further research conducted on determining the optimal token size for detecting license inconsistencies.

Acknowledgement

This work is supported by the Graduate School of Information Science and Technology at Osaka University.

References

[1] Y. Wu, Y. Manabe, T. Kanda, D. German, K. Inoue. Analysis of License Inconsistency in Large Collections of Open Source Projects. 2015. Retrieved June 21, 2017 from http://sel.ist.osaka-u.ac.jp/lab-db/betuzuri/archive/1059/1059.pdf.