**Chapter 6**

**Data Collection**

In order to ensure that all of the comments found in the source code are properly pulled for analysis a powerful language parsing tool, srcML, is used. At its core srcML is a tool designed to take source code and represent it in autogenerated XML. This of course is an oversimplification of just what this tool does. srcML processes source code independent of the preprocessor, which is great for the purposes of this project because it means that when comments are being extracted from the source code we do not have to worry about things such as missing libraries needed to actually run the source code. Further, because srcML does not need to compile the code in order to analyze and extract information it is able to run extremely quickly, which is great for the purpose of this project due to the large number of files that are being analyzed. Another reason that srcML is selected as the extraction tool for this project is because of the tools ability to leave the original structure of the source code entirely intact, meaning that whitespace, comments, and all preprocessing comments are left untouched. Once source code has been converted to XML using srcML the user is able to write XPath queries to pull any specific information needed from the original source code quickly and easily. This is great for the purpose of this project as the actual code from the source code can be ignored and just the comments can be extracted. The original path of the files is preserved and in the case of scanning whole directories XSLT may be used in conjunction to create an archive of these queries. Currently, the greatest limitation of srcML is that it can only parse C, C++, C#, and Java, though for the purposes of this research this is not an issue.

To ensure that the quality of the base source code that is being used in this project is maintained and written by programmers with more experience it is decided that the best approach is to pull highly trafficked projects from GitHub. The reason for this is two-fold, first, projects that have higher rates of traffic are likely to be better maintained as there is greater scrutiny on the projects, and second, these projects are more likely to be written by programmers with greater experience and better represent the general population of programmers. Based on this, the 20 topmost trafficked C, C++, C# and Java projects have been selected and pulled for the use in building the data artifact used in this project. The reason for choosing C, C++, C# and Java over other languages is that they represent such a large chunk of the source code currently maintained today and additionally the tool that is used for extracting the comments from the source code, srcML, supports all of these languages.

To do this the first step is to convert the entire series of projects into one large archive XML file of all of the code present in all of the source code of each of the 20 projects. This archive can be simultaneously broken down into just the comments from these projects by including an XPATH query that looks for just the comments in the source code. In the case of this research, this is the appropriate step to take as the rest of the source code is not needed. Since the long term goal of this research is to use automated verification of comments through machine learning it is important to ensure that the initial variables being given to the machine learning algorithm are as accurate as possible, to this end a manual verification approach was decided by reviewing thousands of lines of comments term by term.

The entire process of manual verification covered a spread of 2935 lines of comments from amongst the 20 different projects and covers a mix of all four languages selected for this project. We have decided that it is best to verify all comments on a line by line basis, this is to include block comments on a line to line basis, the reason for reviewing even block comments in this manner is that it is very possible to have a block comment that is a mix of both commented out code and standard English prose. The manual verification process took a total of 185 hours both of initial review and double verification over the course of two months. The results of this manual verification have been stored inside a data artifact in the form of a 7-column csv file for ease of use and the sake of future research regarding this topic.

Each of the 7 columns of the csv file represent what we feel are the most important notes on each comment though only two of the columns will actually be used for the machine learning process, namely the comment itself and the column that specifies whether or not a line is code. The first of these columns contains the comments themselves, in the case of block comments, each line is stored independently in the csv, as described in the previous paragraph. In the interest of maintaining the integrity of the data, all of the blank lines within block comments have been kept as well and are stored on their own lines. To maintain comments of all different types the markers for the comments are also maintained in these lines. Some examples of this include ‘//’, ‘/\*’, ‘\*’, ‘///’ and in the case of C++ and C style block comments potentially no marker at all. The purpose of this was to determine if certain types of comments were more likely to generate false positives in the machine learning algorithm and, if this was the case, to ensure that we manipulate the comments by removing these markers before feeding them into the machine learning algorithm. The second through fourth columns are used primarily for bookkeeping purposes but do provide important information especially towards future research.

The second column is the name of the source-code file from which the comment has been pulled from. This file name is extracted from the path information provided by srcML in the XML archive used in the production of this data artifact. The third column of the csv file is labeled block comment, and there are two different ways that this is marked down. If this column is marked with a n then the line is not part of a block comment. If the line is given a range of numbers then those numbers represent the range of lines that are a block comment that the line is a part of, not here this number applies only to the csv document and not to the source code itself. The fourth column is labeled as language and represent the coding language that the source code was written in. We decided to add this column for the purpose of both future research and to ensure that anyone viewing the data artifact will know what language the comment was written in regardless of whether or not they are familiar with all of the different file endings attributed to a language. The language column is followed by two different column’s that are related to one another, the first is the contains code column and the second is the code column.

The first of these two columns, the contains code column, is the fifth column of the csv file and was determined to be extremely important when verifying false positives when catching commented out code with the machine learning algorithm. The primary thing that we check for when determining whether or not to mark this comment with a yes are function names and equations. While equations seem to be less common function names may be included in order to aid in the description of what a section of source code does or to mark what functions need to be called within an area of the source code. The sixth column, which is the column labeled is code, is the second column directly important to the machine learning algorithm. This column is very straight forward and is marked with either a y or n depending on whether or not it is determined that a comment line is commented out code. However, it is important to note that this has nothing to do with the actual source code itself, rather, we decided to mark anything that if uncommented a compiler would attempt to compile it as code. The last columns purpose is purely for future research and the possibility of additional checks that can be made in a multilayered machine learning approach.

The seventh column, which is labeled contains standard terms, is only ever filled when a comment line is commented out code. The primary purpose of this column is to provide a list of terms that could be used as a bag of words when identifying lines of commented out code. for example, in C++ *#include*, return, void, int, string, virtual, float, and double are all fairly common within code and are terms that could be used to identify commented out code. We are also marking things such as if, else, else if etc. though these are less likely to be helpful due to the fact that they are common English words. In Fig ## a sample section of the csv is shown to help visualize everything that has been discussed in this chapter.

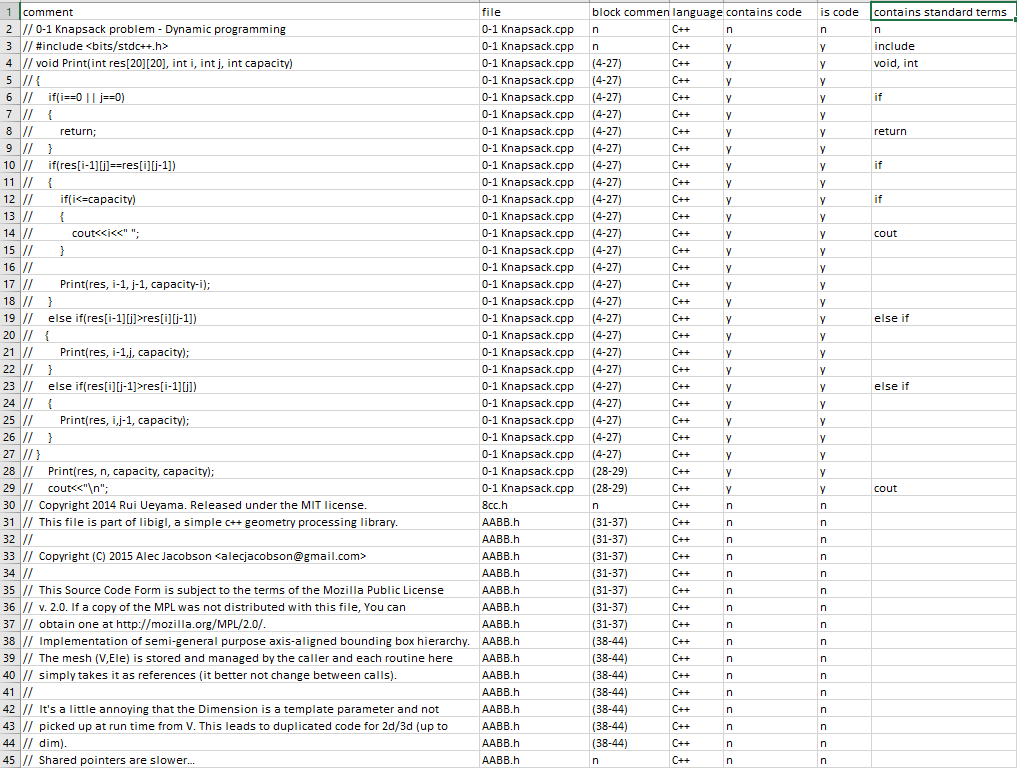


Fig ## Sample of CSV File