**Code Readability Comparison between Java and Python using Apriori Algorithm**

|  |  |  |
| --- | --- | --- |
| **Reinon Vien Irvinz M. Jayag** | **Lincoln D. Morales** | **Jerizsa Hannah Reyes** |

**Abstract**

The purpose of this research is to be able to determine which among two widely used programming languages are easier to work with. These two languages include Java and Python. To provide a little bit of insight, Java is an object- oriented programming language invented by James Gosling. It is mostly used for Android devices and edge devices. There are currently three versions of Java which are: Java SE (standard edition), Java EE (enterprise edition) and Java ME (micro edition). Python, invented by Guido Van Rossum, is also an object-oriented programming language although it is dynamically typed as opposed to Java being statically typed. Popular websites such as Reddit, Spotify, YouTube, Instagram and Google were all developed using Python. These two will be tested using the apriori algorithm, an algorithm used for item set mining and association learning, in order to verify as to which is more code readable or in other words, better to maintain and understand.

**Chapter 1 - Introduction**

Code readability is an important factor when creating competent software and programs. It determines the ease of knowing why the software is not working properly or how to better improve upon the software. Some programmers do not bother making their code cleaner and easier to understand as long as the program created fulfills its predetermined function. With this in mind, the researchers created this study to determine which programming language among Java and Python is easier to read for programmers. The focus of the study is readability between Java and Python, mainly in the field of semantics, syntax and how lines of code should behave in each language.

The scope of this research is the two programming languages Java and Python, comments in the particular languages (syntaxes that does not affect the code), readability in terms of semantics and syntaxes, and last but not the least the apriori algorithm which will be used to determine the readability of the aforementioned languages. The limitation of this research is the quarantine in effect because of COVID-19, forcing the researchers to be restricted to their own homes as well as the insufficient knowledge of the apriori algorithm. Due to time constraints and the nature of the research, interviews and surveys from programmers will not be taken into account in the results. Results will also be limited as the researchers do not have access to machines capable of doing intensive calculations which the apriori algorithm may require.

The paper is separated into different sections. Section 1 contains the introduction. Section 2 contains the related literature including apriori Algorithm and Code Readability Metrics used in this research. Section 3 includes Experimental Results. Lastly, Section 4 is Conclusion and Recommendations.

**Chapter 2 - Related Literature**

**2.1 Apriori Algorithm - Association Rules**

The apriori algorithm is an algorithm that uses association rules commonly used in frequency data mining. It was originally presented by Agrawal and Srikant in 1994 as a way to generate significant association between databases of customer transactions (Agrawal, R, & Srikant, R., 1994). It contains two key concepts: Support and Confidence. Support is defined as the number of times an item has appeared in an itemset. This can determine the association rules of the items chosen. Meanwhile, the confidence is how likely an item is connected in relation to another item. These concepts help determine the associations and the connections between data. There are also two steps to take in order to apply the algorithm (Borgelt, C. & Kruse, R., 2002) . The first step involves finding frequent itemsets. This is the support and a minimum number of support or support threshold can be used to ignore items with minimal data in order to increase the algorithm's efficiency. The second step involves generating the association rules between items. This involves finding the confidence percentage of each item in relation to each other. Items to be used can also be subjected to the minimum support allowing for reduced computations.

|  |  |
| --- | --- |
| Transactions | Items |
| 1 | {Book, Paper, Pencil} |
| 2 | {Book, Eraser} |
| 3 | {Book, Pencil, Ruler} |
| 4 | {Book, Paper, Pencil, Eraser} |
| 5 | {Eraser, Paper, Pencil} |

**Figure 1.** Example of a transactional database using apriori algorithm

**2.1.1 Support - Frequent Itemsets**

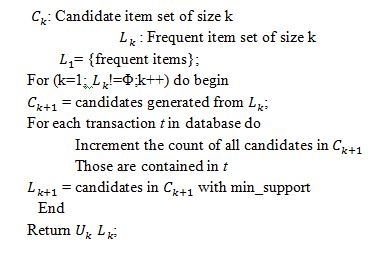
Each item in an itemset has its own support percentage which is used to determine its significance within the whole itemset (Ginting, D. S., Mawengkang, H., & Efendi, S. 2018). In the table shown in Figure 1, the support of {Book} is equal to the number of transactions that contains {Book} over the total number of transactions in the itemset . Using this formula, the support for book is (⅘) or 80%. A minimum support threshold (K+1) can also be specified as a way to reduce the amount of data needed to be processed (Yuan, X., 2017). Looking at Figure 1 again, if K=2, then {Ruler} will not be included in the final results as it only has a support of (⅕) or 20% which is less than the threshold support of () 60%.

**2.1.2 Confidence - Association Rules**

Similarly, confidence uses a formula that takes two items X and Y and shows how likely these two items are associated with one another (Ginting, D. S., Mawengkang, H., & Efendi, S. 2018). In Figure 1, if the confidence percentage between {Book} and {Pencil} is to be determined, the support for both needs to be taken into account. The support for {Book,Pencil} is (⅗) or 60%. Meanwhile the support for {Book} is (⅘) or 80%. Using the formula given, the confidence of {Book -> Pencil} is (3/4) or 75%.

**2.1.3 Rules and Examples**

**Pseudocode:**

****

|  |  |
| --- | --- |
| TID | ITEMSET |
| 10 | {RED, BLACK, PINK} |
| 11 | {BLUE, BLACK, CYAN} |
| 12 | {RED, BLUE, BLACK , CYAN} |
| 13 | {BLUE, CYAN} |
| 14 | {GREEN, BLUE, BLACK. CYAN} |

L1

|  |  |
| --- | --- |
| 1-ITEMSET CANDIDATES | SUPPORT |
| {RED} | 2 |
| {BLUE} | 4 |
| {BLACK} | 4 |
| {PINK} | 1 |
| {CYAN} | 4 |

Generated Candidates form L1

|  |  |
| --- | --- |
| 1-ITEMSET | SUPPORT |
| {RED} | 2 |
| {BLUE} | 4 |
| {BLACK} | 4 |
| {CYAN} | 4 |

L2

|  |
| --- |
| 2-ITEMSET CANDIDATES |
| {RED, BLUE} |
| {RED, BLACK} |
| {RED, CYAN} |
| {BLUE, BLACK} |
| {BLUE, CYAN} |
| {BLACK, CYAN} |

Generated Candidates from L2

|  |  |
| --- | --- |
| 2-ITEMSET CANDIDATES | SUPPORT |
| {RED, BLUE} | 1 |
| {RED, BLACK} | 2 |
| {RED, CYAN} | 1 |
| {BLUE, BLACK} | 3 |
| {BLUE, CYAN} | 4 |
| {BLACK, CYAN} | 3 |

|  |  |
| --- | --- |
| 2-ITEMSET | SUPPORT |
| {BLUE, BLACK} | 3 |
| {BLUE, CYAN} | 4 |
| {BLACK, CYAN} | 5 |

L3

|  |
| --- |
| 3-ITEMSET CANDIDATES |
| {BLUE, BLACK, CYAN} |

Generated Candidates from L3

|  |  |
| --- | --- |
| 3-ITEMSET CANDIDATES | SUPPORT |
| {BLUE, BLACK, CYAN} | 3 |

Final frequent item set results

**2.2 Code Readability Metrics**

The paper will use Dorn’s model, Scalabrino’s model, and Dhabhai’s model for code readability. In Dorn’s model, code readability is defined by four defining categories which are: Visual features, Spatial features, Alignment features, and Natural- language features (Scalabrino, S., Linares-Vásquez, M., Oliveto, R., & Poshyvanyk, D., 2018). The model used in Scalabrino research on the other hand follows a procedure in order to determine the readability of a source code. The procedure involves removing non-textual tokens from the corpora operators, splitting the remaining tokens into separate words by using the underscore or camel case separators, removing words that belong to a stop-word list and extracting stems from words by using the Porter algorithm. (Scalabrino, S., Linares-Vásquez, M., Oliveto, R., & Poshyvanyk, D., 2018). Lastly, Dhabai’s model relies on seven rules for calculating readability. These rules are: total lines of code, line length, number of comment lines, number of blank lines, number of lines after semicolon, number of space after directive statements, and number of methods. (Dhabhai, D., Dua, A.K., Saroliya, A., Purohit, R., 2015)

**3. Experimental Results**

**3.1.1 Test Parameters**

The apriori module used in this research contains five key parameters (min\_support, min\_confidence, lift, length, and number of transactions). Six different experiments with differing parameters were used in determining the apriori results. The changes were mainly done to min\_support and the number of transactions as these have the most effect on the number of results shown in the given dataset.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test | Min\_sup | Min\_conf | Lift | Len | No. Trans |
| 1 | 10% | 50% | 5 | 5 | 5000 |
| 2 | 5% | 50% | 5 | 5 | 5000 |
| 3 | 10% | 50% | 5 | 3 | 5000 |
| 4 | 10% | 50% | 5 | 5 | 2500 |
| 5 | 5% | 50% | 5 | 5 | 2500 |
| 6 | 10% | 25% | 5 | 5 | 2500 |

**3.1.2 Source Code Dataset**

The dataset from this research was taken from Kaggle. The source code are snippets which come from a set of 35,000 lines of code and further divided into 6 item sets of 5,000 lines each. The source code was modified to remove all spacing and indentations to better reflect code readability measures for both Python and Java.

**3.2 Results**

The results in the research show that the overall percentage of association found within the source code is higher in Java. When testing, it showed that Java source codes had more results per test and fewer outliers than in Python. This could be the result of a number of factors such as the inclusion of semicolons and curly braces as part of the source code when testing using the apriori algorithm since Python does not often use either symbols.

**Test 1**

|  |  |  |
| --- | --- | --- |
| Item set | Java | Python |
| 1 | 138 | 152 |
| 2 | 132 | 151 |
| 3 | 108 | 46 |
| 4 | 84 | 58 |
| 5 | 212 | 55 |

**Test 2**

|  |  |  |
| --- | --- | --- |
| Item set | Java | Python |
| 1 | 398 | 808 |
| 2 | 434 | 234 |
| 3 | 392 | 236 |
| 4 | 8548 | 2566 |
| 5 | 780 | 410 |

**Test 3**

|  |  |  |
| --- | --- | --- |
| Item set | Java | Python |
| 1 | 138 | 152 |
| 2 | 132 | 152 |
| 3 | 116 | 48 |
| 4 | 84 | 58 |
| 5 | 212 | 55 |

**Test 4**

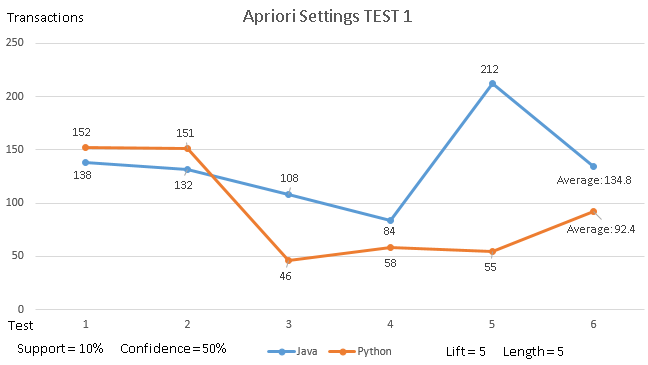
|  |  |  |
| --- | --- | --- |
| Item set | Java | Python |
| 1 | 84 | 620 |
| 2 | 124 | 52 |
| 3 | 56 | 79 |
| 4 | 256 | 55 |
| 5 | 190 | 316 |

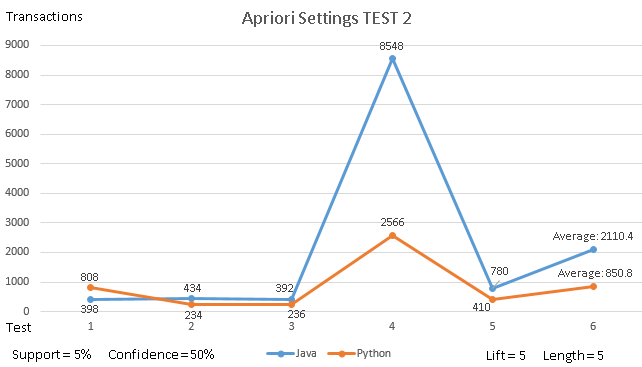
**Test 5**

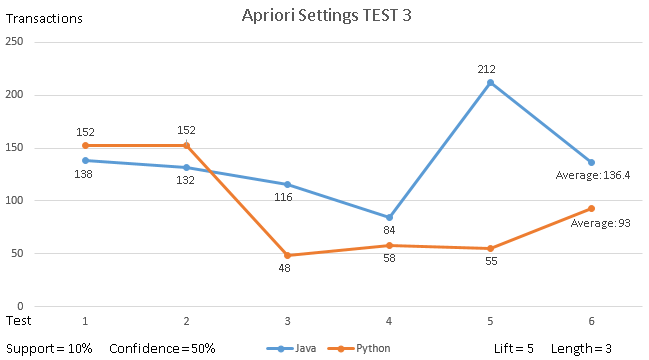
|  |  |  |
| --- | --- | --- |
| Item set | Java | Python |
| 1 | 952 | 2926 |
| 2 | 472 | 156 |
| 3 | 288 | 290 |
| 4 | 3574 | 510 |
| 5 | 2926 | 1311 |

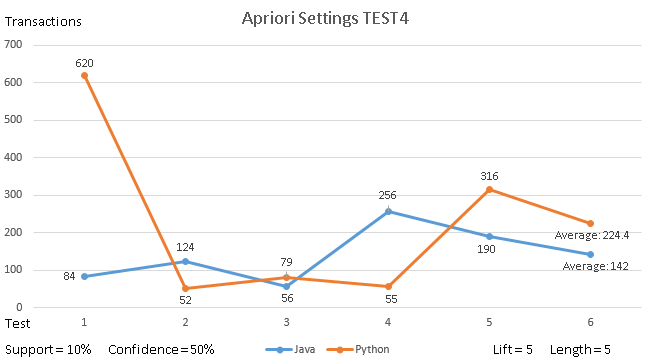
**Test 6**

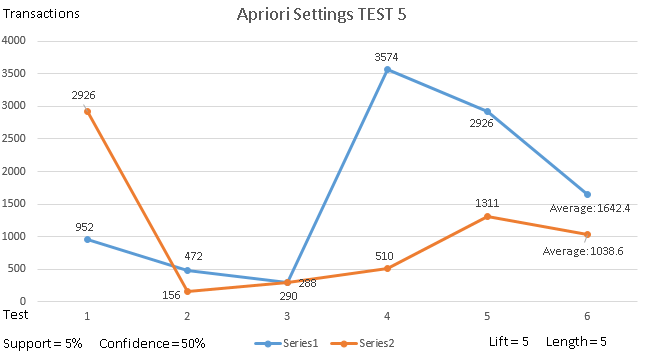
|  |  |  |
| --- | --- | --- |
| Item set | Java | Python |
| 1 | 142 | 160 |
| 2 | 140 | 152 |
| 3 | 120 | 56 |
| 4 | 88 | 64 |
| 5 | 236 | 63 |

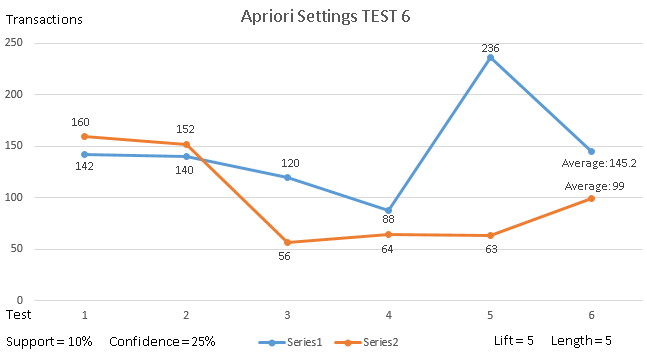
****











**4. Conclusion and Recommendation**

The apriori algorithm was used to determine code readability through the use of association rules using source codes of sample programs in Java and Python. In 6 different tests, it was found that Java on average returned more results given the parameters used.

Further recommendations for this paper includes the removal of semicolons, curly braces, and other symbols which in order to improve code readability accuracy. The researchers also suggest using a higher min\_support and more item sets to increase the amount of information gathered and find a more accurate result. Another suggestion is to compare programming languages with similar syntax, like Python and JavaScript or Java and C++.

**References**

Börstler, J., & Paech, B. (2016). The role of method chains and comments in software readability and comprehension—an experiment. IEEE Transactions on Software Engineering, 42(9), 886-898. DOI: [10.1109/TSE.2016.2527791](https://doi.org/10.1109/TSE.2016.2527791)

Chen, H., Huang, Y., Liu, Z., Chen, X., Zhou, F., & Luo, X. (2019). Automatically detecting the scopes of source code comments. Journal of Systems and Software, 153, 45-63. DOI: <https://doi.org/10.1016/j.jss.2019.03.010>

Dhabhai, D., Dua, A.K., Saroliya, Anil, & Purohit, R. (2015). Evaluation of Quality of Source Code By Code Readability. Retrieved from <http://ijarcsse.com/Before_August_2017/docs/papers/Volume_5/10_October2015/V5I10-0110.pdf>. ISSN: 2277 128X

dos Santos, R. M., & Gerosa, M. A. (2018, May). Impacts of coding practices on readability. In Proceedings of the 26th Conference on Program Comprehension (pp. 277-285). DOI: <https://doi.org/10.1145/3196321.3196342>

Donges, N. (2019, June 16). A complete guide to the random forest algorithm. Retrieved March 1, 2020, from https://builtin.com/data-science/random-forest-algorithm#real

Fakhoury, S., Roy, D., Hassan, A., & Arnaoudova, V. (2019, May). Improving source code readability: theory and practice. In 2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC) (pp. 2-12). IEEE. DOI: [10.1109/ICPC.2019.00014](https://doi.org/10.1109/ICPC.2019.00014)

Johnson, J., Lubo, S., Yedla, N., Aponte, J., & Sharif, B. An Empirical Study Assessing Source Code Readability in Comprehension. In 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 513-523). IEEE. DOI: [10.1109/ICSME.2019.00085](https://doi.org/10.1109/ICSME.2019.00085)

Scalabrino, S., Linares-Vásquez, M., Oliveto, R., & Poshyvanyk, D. (2018). A comprehensive model for code readability. Journal of Software: Evolution and Process, 30(6). doi: 10.1002/smr.1958

Sedano, T. (2016, April). Code readability testing, an empirical study. In 2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET) (pp. 111-117). IEEE. DOI: [10.1109/CSEET.2016.36](https://doi.org/10.1109/CSEET.2016.36)

Song, X., Sun, H., Wang, X., & Yan, J. (2019). A survey of automatic generation of source code comments: Algorithms and techniques. IEEE Access, 7, 111411-111428. DOI: [10.1109/ACCESS.2019.2931579](https://doi.org/10.1109/ACCESS.2019.2931579)  
Yuan, X. (2017, March). An improved Apriori algorithm for mining association rules. In AIP

conference proceedings (Vol. 1820, No. 1, p. 080005). AIP Publishing LLC.

Borgelt, C., & Kruse, R. (2002). Induction of association rules: Apriori implementation. In Compstat (pp. 395-400). Physica, Heidelberg.

Agrawal, R., & Srikant, R. (1994, September). Fast algorithms for mining association rules. In Proc. 20th int. conf. very large data bases, VLDB (Vol. 1215, pp. 487-499).

Ginting, D. S., Mawengkang, H., & Efendi, S. (2018, September). Modification of A priori Algorithm focused on confidence value to association rules. In *IOP Conference Series: Materials Science and Engineering* (Vol. 420, No. 1, p. 012125). IOP Publishing.

Ye, Y., & Chiang, C.-C. (2006). A Parallel Apriori Algorithm for Frequent Itemsets Mining. *Fourth International Conference on Software Engineering Research, Management and Applications (SERA06)*. doi: 10.1109/sera.2006.6