Jonathan Alexander Gibson
jagibson44
CSC 5220-001
Project Update 1 Report
Dr. Akond Rahman

The TF-IDF for a set of scripts ('s') and a set of tokens ('t') is relatively easy to find, is a good metric to compare similarities between documents, and gives a basic understanding about the most important tokens in a single document. However, TF-IDF is not a very sophisticated model. It is based on the BoW (Bag of Words) model and does not include other important metrics like token positions in a script or token semantics.

For our project the problems associated with TF-IDF are most commonly seen in the quantity of tokens that hold a score of "1.0". Although TF-IDF is supposed to normalize the token scores within a particular document, running TF-IDF on our dataset still skews the token values towards "1.0" for both the "INSECURE" and the "NEUTRAL" groups. For these reasons, I do not believe that TF-IDF is the best text mining approach for our dataset. All of the blame should not be placed on the approach, however. Even though our dataset is very large, only roughly 20% (1.55x10^6/7.6x10^6) of the dataset rows were left to accommodate the requirement for valid token names from the "MODIFIED_FILE" column. This is the first reason why the state of the dataset hindered the potential for better, more insightful results.

Secondly, TF-TDF is typically used on scripts that are "actual documents". These documents could be research papers, books, magazine articles, etc. and the tokens are typically "normal English words". For our TF-IDF implementation we had to use Git commit hashes as the scripts and modified files (associated with each hash) as the token values. This caused an automatic imbalance between the quantity of modified files per commit hash.

We examined the top 1000 token feature scores for the "INSECURE" and the "NEUTRAL" groups. While some commit hashes had a large number of modified files associated with them, the majority of the commit hashes had very few (1~5) modified files associated with them. In the case that a token feature got a value of "1.0", it was very common that the particular token appeared in lists where there were few modified files associated with the particular hash. This may tell the TF-IDF model that the token is very important to a collection of hashes with small modified file token lists, while tokens that may appear often in large modified file token lists could suffer and receive much lower scores.

Some modified file tokens even occur more than once, at least once for multiple hashes. Or for the token "emon.py", more than once for one particular commit hash, which is not supposed to occur, given the format of Git commits (for hashes "*de34089011627304e8e7588def5f6848311a9843*", "*4c5455bc957b4de90eac9df0c29e9874ec6bd143*", & *242d3272391baeb95eb0a5e4e51627c2d54e7bc6*"). Further, the token "mple.py" is ranked with a score of "1.0", like many others in the top 1000 insecure scores, but only appears once, as the only token in the list associated with the hash "*936cc3a6c596d2c5c5542553f32d5a7b5b972266*".

After examination of both groups, 52% (520) of the tokens (from the top 1000), from the "INSECURE" group, got a score of "1.0". Additionally, 100% (1000) of the tokens (from the top 1000), from the "NEUTRAL" group, got a score of "1.0". Although this may not seem surprising on the surface, let me give you some perspective. Of the 1,557,292 rows that remained after the filtering of the dataset, 1,539,025 of those rows belonged to the "NEUTRAL" group, and only 18,267 of those rows belonged to the "INSECURE" group. This means that 2.8% of the "INSECURE" group's tokens were given a score of "1.0". If a similar ratio held true for the "NEUTRAL" group's tokens, 43,604 of its tokens would also be given a score of "1.0".

Overall, I think the huge size of the dataset, the large imbalance between the quantity of rows in each group, the irregularity of the data used for the TF-IDF model, the invalidity of ≈80% of the original dataset, and the inconsistencies within the token-hash pairs are all reasons why this text mining approach was not as effective as it could have been with different parameters. Even though we can compare the relative importance, of certain tokens within their respective groups, the results from this text mining task don't seem to provide much useful information.