MSIS 5600

Special Project in Business Information

Project Report

Analysis of Reported Issues in Software Development for C# and Java?

Under the guidance of
Dr. Bryan Hammer

Name: Hitesh Gaur

CWID: A20054244

Term: Spring 2018

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Software development is a process and programming are an art. Like every other type of art, development of software has a lot of challenges. Over the years, several methodologies have evolved to reduce number of obstacles faced by 'Software Artists'. Mostly, all of these Software Development Methodologies have common phases like Requirement Collection, Feasibility Study, Design and Planning, Development, Testing and Debugging, Implementation and Maintenance. The big picture is more than one Software Artists are involved with separate phases of software development process. Mostly, all of these, phases have to deal with several types of issues, during software development. While the processes, tools or programming language for software development varies with context, domain and application of a software, the fact, that issues will surface, is invariable.

According to Google, an issue is "an important topic or problem up for a discussion". Usually, a forum or a tool is adopted by the stockholders to keep an eye on problems related to a project. With ubiquitous presence of software application in minuscule activities at every juncture around lives of people, the issues in software development process are continuously amplifying. Considering the time constraints, in this rapidly growing world, these issues needs to be prioritized. For example, issues related to security of a system, are most critical issues. With attackers waiting around every corner, to exploit vulnerabilities in software applications, the problems that are related to security topic, should be attended with highest priority, certainly as soon as it gets reported.

There are primarily 12 distinct categories of possible errors & exceptions, towards which an issue can point. MSDN web site has identified these categories as 'Mobility', 'Cryptographic', 'Portability', 'Maintainability', 'Reliability', 'Globalization', 'Interoperability', 'Performance', 'Naming', 'Usage', 'Security’, and 'Design' [6]. It has created description of some specific rules which can be identified as possible flaws and issues in Software Development.

Business Problem

In software development, issues are being reported through various issue tracking systems. Then issues are attended, discussed and resolved. Most of the time entire process is manual. If an issue can be automatically categorized, it can speed up its process.

Recommended solution

Github is a web-based repository service, widely popular and used for software projects. It has an issue tracking and reporting system. Over the years, numerous issues have been accumulated in this system. With linguistic analysis of these reported issues, it can be possible to identify primary category for the reported issue. Term frequencies of these issues can be inspected against the term frequencies of rules written under distinct categories at MSDN website [12].

Historical issues reported over Github can be utilized to identify best possible categories. Github archive, has more than 5 million records, which can be narrowed down for two of most prevalent programming languages in Enterprise software application development. These are C# and Java.

Statement of Scope

Topic mining based on rules described at MSDN, for issues related to repositories having source code in C# or Java, gives opportunity to automatically categorize issues with reduced data. Identification of programming language for an issue over Github, is to be done by extracting content over the internet. For 5 million records, it becomes a time taking process.

Project Summary

The study has been done using nearly one-third records out of available 5 million issues. it took around 75 days, to identify programming language for these issues and 35 days to clean and transform data. The identification of programming language is performed by using its URL [figure-1 & 2].

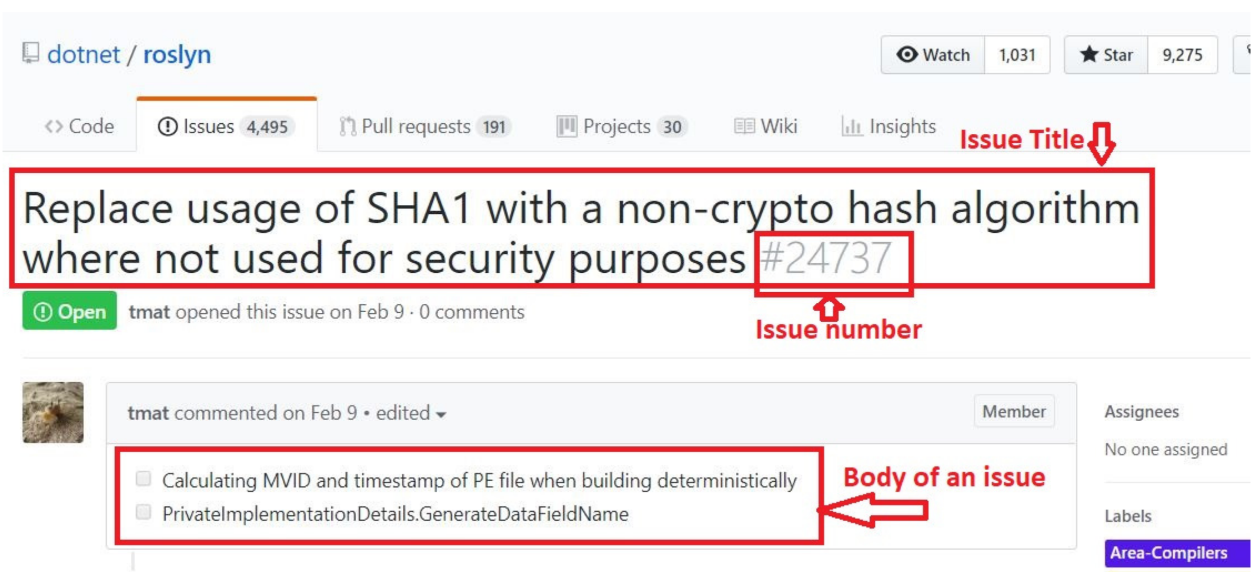

Figure 1: An issue on Github Platform

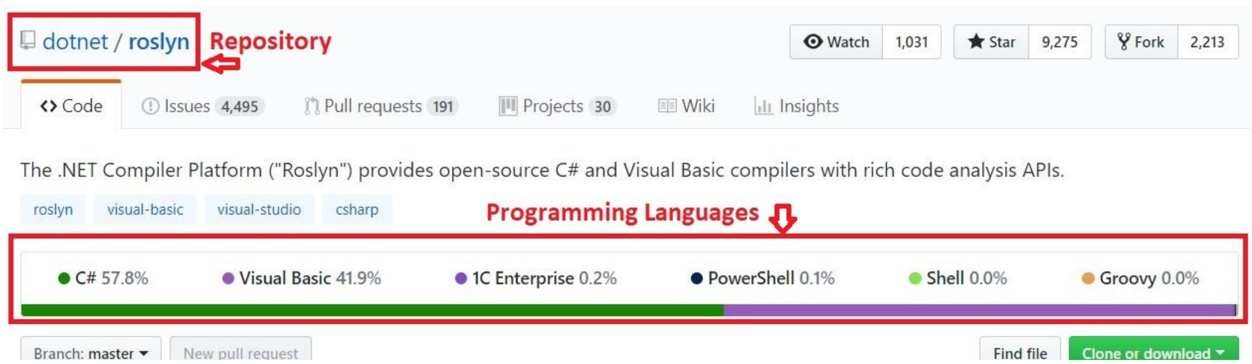

Figure 2:Programming Language of Source Code in a repository

After processing data, terms-frequency and inverse document frequencies were used to associate issue with a category, using a similarity index score. The issues were successfully identified with multiple categories. The statistics for C# and Java were nearly same.

The algorithm can be improved with more data collection and Language Specific Rules.

Project Schedule

Table 1: Time table of Tasks Performed

Analysis of Reported Issues Over Github			
TASKS	START DATE	END DATE	DURATION (days)
Project Objective	1/26/18	3/23/18	57
Data Collection	1/27/18	4/14/18	77
Data Splitting	2/2/18	2/5/18	3
Data Cleaning	2/4/18	3/11/18	37
Data Transformation	2/4/18	3/11/18	37

Data Reduction	2/23/18	4/22/18	59
Data Dictionary	3/19/18	3/20/18	1
Data Consolidation	3/20/18	4/22/18	32
Data Description	3/31/18	4/23/18	23
Data Analysis	4/7/18	5/5/18	28
Model Selection	4/13/18	4/20/18	7
Model Building	4/15/18	5/7/18	22
Model Assessment	4/22/18	5/7/18	15
Final Report	5/5/18	5/10/18	5

Gantt Chart

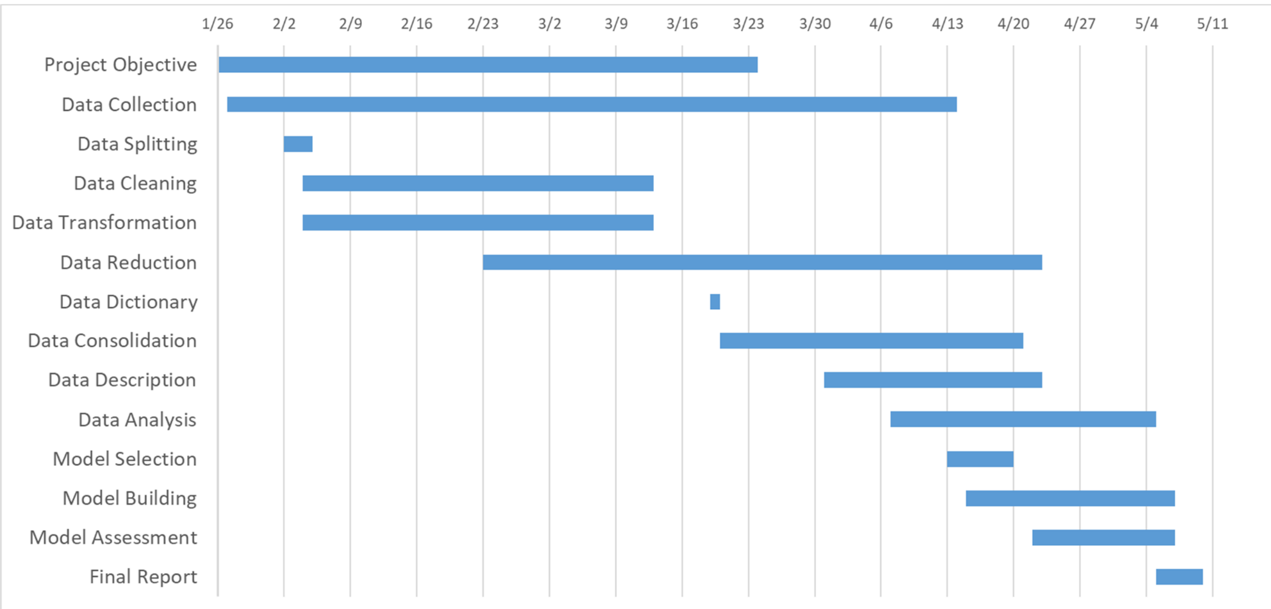

Figure 3: Gantt chart for the project tasks

Data Access and Collection

Software programmers, from around the globe, use millions of repositories from GitHub. GitHub provides service to report and track issues related to it. While more than 5 million issues are accessible over internet to be downloaded as one single document, statistics of programming languages for a single repository needed to be extracted for each issue and its corresponding repository. The expected time to extract such information for complete set of data is 10 months.

To meet the timelines and scope of this project statistics related to programming languages has been extracted for more than one million and four hundred thousand records in 77 days.

Data Access

The GH Archive is the actual provider of first set of data, from where it has been tailored to populate more than 5 million issues.

According to GH Archive website, "GitHub Archive" is a project to catalog timeline of publicly available GitHub repositories [1]. It archives the timeline and repository documents to preserve information so that, it is available for research and analysis in future. The dataset of issues is available in a comma separated (csv) file, of size greater than 2.8 GB at Kaggle (https://www.kaggle.com/davidshinn/github-issues) [2].

To Identify relationship of a reported issue with type of vulnerability, rules are being provided at Microsoft's MSDN website [6]. It describes several rules for, twelve distinct categories, which are 'Mobility', 'Cryptographic', 'Portability', 'Maintainability', 'Reliability', 'Globalization', 'Interoperability', 'Performance', 'Naming', 'Usage', 'Security', and 'Design'. The rules are manually copied into twelve separate csv files with two columns. First column is Rule Identifier and second column is description for a rule.

Data Collection

To gather data pertaining to programming languages, for the millions of repositories available in first set of data, web scrapping tools have been utilized. For each repositories Github, shows language statistics on home page of a repository. On performing a closer inspection of web page, the html elements containing information of programming languages are as shown in figure-4.

Figure 4: HTML elements to extract programming language and its share in a repository

NodeJS framework has been utilized to create a request for a repository and collect data from HTML form received as response (figure-5).

```
$langugesElement.each(function (index, element) {
    var language_index = index ;
    var language_name = $(this).find('span.lang').text();
    var weightage = $(this).find('span.percent').text();

    writer.write({
        'row_number': url_row_index, 'repo_owner': repo_owner, 'repo_name': repo_name,
        'language': language_name, 'weightage': weightage, 'language_index': language_index

    });
});
```

Figure 5: NodeJS code to extract language statistics of a repository

If continuously, requests were to be made to Github server, it would have overloaded server. To avoid such a strain on server, requests were made with 5 seconds of delay using request-promise library package. It uses deferred actions, which acts as asynchronous functionality.

```
In [53]: df = pd.read_csv('languageStats\\repoLanguages_9.csv', header=None,
names = ['row_number', 'repo_owner', 'repo_name', 'language', 'weightage'])
    ...:
    ...: df.head(5)
Out[53]:
   row_number     repo_owner  repo_name    language weightage
0      450007  eclipsesource  tabris-js  JavaScript     92.3%
1      450007  eclipsesource  tabris-js  TypeScript      7.6%
2      450007  eclipsesource  tabris-js        HTML      0.1%
3      450001      Jumpscale  jscockpit  JavaScript     91.3%
4      450005        codenvy    codenvy        Java     48.1%
```

Figure 6: Extracted Programming languages for a repository by web scrapping

Data Splitting

The data in 2.8 GB of csv file proved too much for the computer to process and load in memory. Since it had more than 5 million records, the data was divided in smaller csv files with exactly

50000 rows (figure-7). As a result, 106 files were being created in total, of which 105 files contains 50000 rows and one file has 32154 records before cleaning (figure-8).

```
for i,chunk in enumerate(pd.read_csv('github_issues.csv', chunksize=50000)):
    print(i)
    chunk.to_csv('multiplefiles\\subset_{}.csv'.format(i))
    print("done")
```

Figure 7: Python Code to split huge csv file into smaller files with 50000 rows.

```
In [11]: df = pd.read_csv('multiplefiles\\subset_106.csv', encoding = "ISO-8859-1")
    ...: len(df)
    ...:
Out[11]: 32154
```

Figure 8: Getting number of records from the last chunk of records.

It helps in calculating number of aggregate records, as 5332204 issues in total are found in the dataset [figure-9].

```
In [22]: total_records_count = 0
    ...: for x in range(0, 107):
    ...:     tempdf = pd.read_csv("multiplefiles\\subset_{}.csv".format(x), encoding = "ISO-8859-1")
    ...:     total_records_count += len(tempdf)
    ...:
    ...:
    ...: print(total_records_count)
5332204
```

Figure 9: Counting total number of records for issues.

Similarly, in approximately 75 days, more than two million (2277807) records were extracted over internet. These set of CSV files contains statistics about programming languages for repositories mentioned in, first 37 subset files of Github issues dataset, which targets approximately one million three hundred fifty-three thousand issues [figure-10].

```
In [55]: total_records_count = 0
    ...: for x in range(0, 37):
    ...:     tempdf = pd.read_csv('languageStats\\repoLanguages_{}.csv'.format(x), header=None, names =
['row_number', 'repo_owner', 'repo_name', 'language', 'weightage'])
    ...:
    ...:     total_records_count += len(tempdf)
    ...:
    ...:
    ...: print(total_records_count)
    ...:
2277807
```

Figure 10: Counting total records extracted over 75 days from internet, about programming languages used with Github repositories

Data Cleaning

When trying to read data and preprocess it, encoding of data must be supplied ('ISO-8859-1' in figure-8), because the text in these files does not have default or utf-8 encoding. All of the CSV files contains URL of issue, title of issue and details from body of issue reported (figure-11). The first column is the index of row generated by data-frame generated for splitting files using pandas library. The last 5 records, from 106th subset of data shows that values in all columns are in the form of text (figure-12).

```
In [12]: df.dtypes
Out[12]:
Unnamed: 0     object
issue_url      object
issue_title    object
body           object
dtype: object
```

Figure 11: Columns in CSV files before cleaning.

```
In [13]: df.tail(5)
Out[13]:
       Unnamed: 0                                          issue_url  \
32149     5332148  "https://github.com/bayborodin/ror-full-3/issu...
32150     5332149  "https://github.com/eclipse/paho.mqtt.java/iss...
32151     5332150  "https://github.com/rzwitserloot/lombok/issues...
32152     5332151  "https://github.com/Gizra/productivity/issues/...
32153     5332152  "https://github.com/jacobmischka/coyote-grill/...

                                              issue_title  \
32149  ??????? ?????? instancecounter, ?????????? ???...
32150  at org.eclipse.paho.client.mqttv3.internal.cli...
32151  java.lang.linkageerror: loader constraint viol...
32152  node : pdoexception: sqlstate 40001 : serializ...
32153  uncaught error: { error :{ errors : { domain :...

                                                     body
32149  ?????? ??????: - instances, ??????? ??????????...
32150  - bug exists release version 1.1.0 master bran...
32151  java.lang.linkageerror: loader constraint viol...
32152  view details in rollbar: https://rollbar.com/b...
32153  view details in rollbar: https://rollbar.com/j...
```

Figure 12: peeking into tail of data - LAST 5 rows.

The textual data in body and title contains a lot of characters, that are not alpha-numeric. There are a lot of punctuation marks and URLs in text of body. Using translation table from Python, punctuation marks have been removed for [figure-12(a)]. Regular expression has been used to identify and delete any URLs [figure-12(b)].

```
table = str.maketrans('', '', string.punctuation)
```

Figure 6(a): to remove punctuation

```
#removing urls
body_text_processed = re.sub(r"http\S+", "", row["body"], flags=re.MULTILINE)
#clean html
body_text_processed = cleanHTML(body_text_processed)
#removing punctuations
body_text_processed = body_text_processed.translate(table).lower()
# split into sentences
print(body_text_processed)
df.loc[row_index, 'body'] = body_text_processed
```

Figure 12(b): Cleaning process for each row

On deeper inspection, a lot of records were found with html data and heml tags appended in body. Beautiful Soup library was used to clean tags and get content using html parser [figure-12(c)]. Cleaning was done for entire data set by using all 106 subset CSV files, hence it took a lot of time, more than 35 days, to process more than 5 million records.

```
def cleanHTML(html):
    soup = BeautifulSoup(html, "html.parser") # create a new bs4 object from the html data loaded
    for script in soup(["script", "style"]): # remove all javascript and stylesheet code
        script.extract()
    # get text
    text = soup.get_text()
    # break into lines and remove leading and trailing space on each
    lines = (line.strip() for line in text.splitlines())
    # break multi-headlines into a line each
    chunks = (phrase.strip() for line in lines for phrase in line.split("  "))
    # drop blank lines
    text = '\n'.join(chunk for chunk in chunks if chunk)
    return text
```

Figure 12(c): function to clean HTML tags from body of issue

A sample result after cleaning a row is shown in figure-10.

```
the urls we are currently obtaining and using are undocumented and for some reason they dont work on
mobile the correct way to handle this is by using twitter apis to retrieve the url to the image then
setting it to a new field on the page column the new field avatar url should take precedence over
facebooks and twitters urls when existing the main problem with this is that we may need the image in
two formats thumbnail and normal  thus we may need two urls the alternative is attaching the image to
the user using paperclip this is maybe much more flexible as it allows any image to be used but this
requires setting up uploads
```

Figure 7: Sample result after cleaning data

Data Transformation

The original data contains URLs of issues, formed using three attributes: a repository owner, name of repository and issue identification number [figure-14].

```
In [29]: df.loc[0:10, 'issue_url']
Out[29]:
0     "https://github.com/MicrosoftDX/Vorlonjs/issue...
1     "https://github.com/kozec/dumbxinputemu/issues/7"
2     "https://github.com/nextcloud/server/issues/5208"
3     "https://github.com/Remillard/VHDL-Mode/issues/6"
4     "https://github.com/Microsoft/TypeScript/issue...
5     "https://github.com/resin-io-playground/staged...
6            "https://github.com/w3c/preload/issues/94"
7        "https://github.com/nltk/nltk_data/issues/106"
8       "https://github.com/turbobytes/pulse/issues/42"
9     "https://github.com/alwx/react-native-photo-vi...
10      "https://github.com/fchollet/keras/issues/8629"
Name: issue_url, dtype: object
```

Figure 8: Identifying pattern in URL to transform data

The pattern in URL to use these three pieces are as following:

"https://github.com/{REPOSITORY-OWNER}/{REPOSITORY-NAME}/issues/{ISSUE-ID-NUMBER}"

Using this information, three calculated columns have been created to separate the attributes [figure-15].

```
default_url = "https://github.com/"
default_url_length = len(default_url)
def processUrl(issue_url):
    print(issue_url)
    issue_url = issue_url[default_url_length+1:-1]
    print(issue_url)
    slash_index = issue_url.find('/')
    print(slash_index)
    repo_owner = issue_url[0:slash_index]
    issue_url = issue_url[slash_index+1:]
    print(issue_url)
    slash_index = issue_url.find('/')
    print(slash_index)
    repo_name = issue_url[0:slash_index]
    issue_number = issue_url[(issue_url.rfind('/') + 1):]
    print(repo_owner)
    print(repo_name)
    print(issue_number)
    return repo_owner, repo_name, issue_number
```

Figure 9:function to transform a URL into its components

These components of URL have been added as new columns to all the subset files for each record. The transformation task was carried out in conjunction with operations to clean data for all 5332204 records, for approximately 35 days.

Data Reduction

Data reduction process was carried out in two parts. First, reduction was done for all of issues from all subset files based on the text of body and title. In some of the observations, it was found that natural language of the textual data in body is not always English and it contains text in the form of other spoken languages [figure-16].

```
                                                   body
49995  does this container accept a variable for user...
49996  merhaba github ile giri? yap?nca benim gibi is...
49997  outcome ! screen shot 2017-05-18 at 10 12 54 h...
49998  your webfaction username/folder bvodola is har...
49999  elke account die nu word gecreert in de regist...
```

Figure 10: English is not the only natural language, used for reporting issues

A library named 'langdetect', in python is used to identify the language of both title and body [4]. The language was detected on the original data, during the process of cleaning and transformation of data. If the text does not contain any language, then an error is being reported as 'No feature detected' [figure-17].

```python
from langdetect import detect
detect("War doesn't show who's right, just who's left.")# 'en'
detect("Ein, zwei, drei, vier") #'de' `
detect(" 0 ?? *#$&@$*") #An error is reported as - No feature detected for any language
```

Figure 11: langdetect library in python

A new boolean column was appended "IsLanguageEnglish" for all files. Only if the language is found to be English, the row will have value TRUE for this column. From each subset file a new csv file is generated and stored, filtering out the issues which were not reported in English language [figure-18].

```
## Iterating through each record
for row_index, row in df.iterrows():
    if(row_index < start_index):
        continue;
    if(end_index > -1 and row_index > end_index):
        break;
    print('\n#################################################################################')
    #logging row index, needed to resume manually
    print(row_index)
    try:
        titleLang = detect(row["issue_title"])
        print(titleLang)
        if(titleLang == "en"):
            bodyLang = detect(row["body"])
            print(bodyLang)
            if(bodyLang == "en"):
                df.loc[row_index, 'IsLanguageEnglish'] = True
                repo_owner, repo_name, issue_number = processUrl(str(row["issue_url"]))
                df.loc[row_index, 'Owner'] = repo_owner
                df.loc[row_index, 'RepoName'] = repo_name
                df.loc[row_index, 'IssueNumber'] = issue_number
                #removing urls
                body_text_processed = re.sub(r"http\S+", "", row["body"], flags=re.MULTILINE)
                #clean html
                body_text_processed = cleanHTML(body_text_processed)
                #removing punctuations
                body_text_processed = body_text_processed.translate(table).lower()
                # split into sentences
                print(body_text_processed)
                df.loc[row_index, 'body'] = body_text_processed
    except Exception as e:
        print(e)
    print(datetime.datetime.now())
df = df.drop(['issue_url'], axis=1)
df = df.loc[df['IsLanguageEnglish'] == True]
try:
    df.to_csv("multiplefilesWithLang\\subset_{}.csv".format(x), index=False, encoding = "utf-8")
except Exception as ex:
```

Figure 12:Detection of Language with cleaning and transformation steps to store data

It resulted in reducing number of records to less than four million records, from more than five million records [3983025].

```
In [31]: total_records_count = 0
    ...: for x in range(0, 107):
    ...:     tempdf = pd.read_csv("multiplefilesWithLang\\subset_{}.csv".format(x))
    ...:     total_records_count += len(tempdf)
    ...:
    ...:
    ...: print(total_records_count)
    ...:
3983025
```

Figure 13: number of issues, reported in English

Second part of data reduction task was carried out in conjunction with Data Collection operations. The programming languages were fetched only for issues which were reported in English. Github platform is available for all programming languages [figure-20]. For the collected dataset, 263 different programming languages were found.

```
In [59]: unique_languages = df.language.unique()
    ...: print(len(unique_languages))
    ...: print(unique_languages)
    ...:
263
['JavaScript' 'TypeScript' 'HTML' 'Java' 'Go' 'Objective-C' 'CoffeeScript'
 'CSS' 'Python' 'Assembly' 'Other' 'C' 'Shell' 'Perl' 'C#' 'Haskell'
 'PowerShell' 'Ruby' 'Prolog' 'C++' 'PHP' 'Makefile' 'CMake' 'GLSL' 'OCaml'
 'Batchfile' 'Smarty' 'Fortran' 'ColdFusion' 'Roff' 'Emacs Lisp' 'TeX'
 'GAP' 'Gherkin' 'Vue' 'Lua' 'M4' 'Scala' 'XSLT' 'wdl'
 'Common Workflow Language' 'RobotFramework' 'Tcl' 'Groovy' 'R' 'NSIS'
 'HCL' 'Vim script' 'Matlab' 'Elm' 'sed' 'Cuda' 'NCL' 'IDL' 'ANTLR' 'Mask'
 'ShaderLab' 'Jupyter Notebook' 'LSL' 'Scheme' 'Visual Basic' 'QMake'
 'Inno Setup' 'Rust' 'Swift' 'Kotlin' "Cap'n Proto" 'QML' 'Dart' 'Coq'
 'Liquid' 'F#' 'PLSQL' 'SuperCollider' 'Processing' 'Solidity'
 'Standard ML' 'Modelica' 'PLpgSQL' 'SQLPL' 'FreeMarker' 'Protocol Buffer'
 'OpenEdge ABL' 'Elixir' 'XQuery' 'WebIDL' 'Julia' 'SaltStack' 'Ballerina'
 'Vala' 'PowerBuilder' 'Brightscript' 'DM' 'Gnuplot' 'VHDL' 'Ada' 'Clojure'
 'Awk' 'Squirrel' 'D' 'Erlang' 'Isabelle' 'Smalltalk' 'Mako' 'ASP' 'SQF'
 'Nix' 'Yacc' 'Nemerle' '1C Enterprise' 'Objective-C++' 'Xtend'
 'AutoHotkey' 'Hack' 'LLVM' 'nesC' 'PostScript' 'AppleScript' 'HLSL'
 'Logos' 'Red' 'Rebol' 'API Blueprint' 'Meson' 'WebAssembly' 'Pascal' 'Lex'
 'Nim' 'PureBasic' 'Forth' 'BitBake' 'Haxe' 'Scilab' 'Nginx' 'SMT' 'XProc'
 'Perl 6' 'Ceylon' 'Cool' 'SourcePawn' 'Mathematica' 'ApacheConf' 'VCL'
```

Figure 14: Programming languages on Github for scrapped data in nearly 75 days.

To narrow down and meet scope of this project, repositories with, two of the most prevalent and trending programming languages, for enterprise application development for last 10 years, are the focus of this study. These languages are 'C#' and 'Java'. The records with these programming languages are only selected, others were discarded. It is clear from the bar-chart in figure-21 that these two languages are among top 10.

```
In [87]: import matplotlib.pyplot as plt
    ...:
    ...: #function to get weighted share in dataframe
    ...: def weighted_sum(group, w, length):
    ...:     d = group[w]
    ...:     return d.sum()/length
    ...:
    ...:
    ...: #For each subset getting weighted sum and get average for all csv files
    ...: list_languages = df.groupby("language").apply(weighted_sum, "weightage", len(df)).sort_values(ascending=False)
    ...: type(list_languages)
    ...:
    ...: counts = list_languages.value_counts()
    ...: ax = list_languages.iloc[:10].plot(kind="barh")
    ...: ax.invert_yaxis()
    ...: ax.get_xaxis().set_visible(False)
```

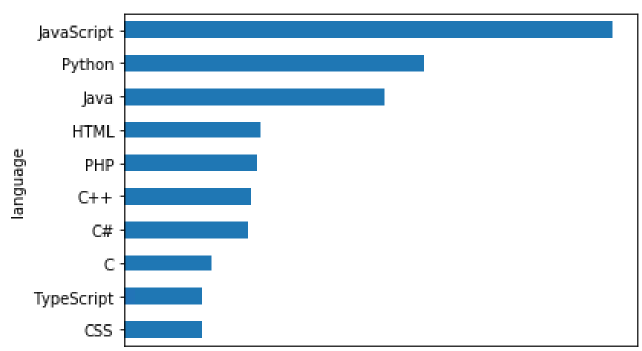

Figure 15: Comparing weighted sum for all programming languages for all files

After selecting records based on 'C#' and 'Java' language, there are around one hundred sixty-one thousand repositories that have programs written in either of these programming languages.

```
In [97]: total_c_sharp_repositories = 0
    ...: total_java_repositories = 0
    ...: for x in range(0, 37):
    ...:     df = pd.read_csv('languageStats\\repoLanguages_{}.csv'.format(x), header=None, names =
['row_number', 'repo_owner', 'repo_name', 'language', 'weightage'])
    ...:     grouped_data = df.groupby(['language'])['row_number'].count()
    ...:     total_c_sharp_repositories += grouped_data['C#']
    ...:     total_java_repositories += grouped_data['Java']
    ...:
    ...: print(total_c_sharp_repositories)
    ...: print(total_java_repositories)
49702
111368
```

Figure 16: Number of repositories for 'C#' and 'Java' programming languages.

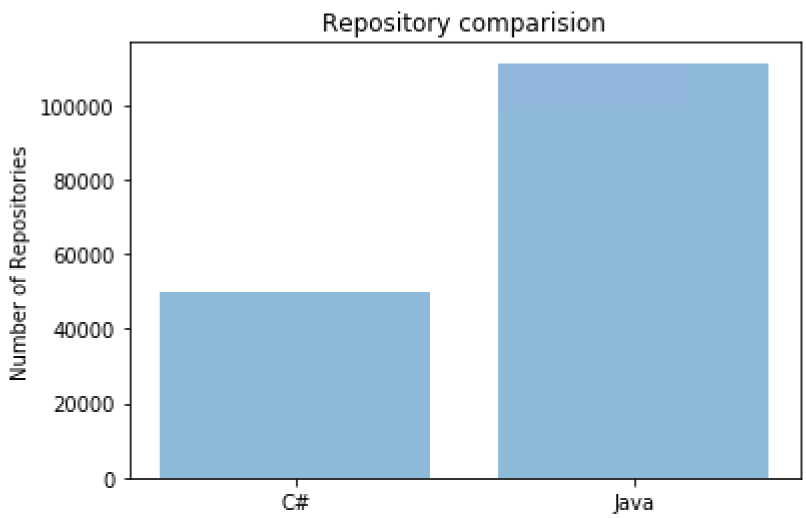

Figure 17: Comparison of repositories using C# or Java programming languages

To get number of issues, related to repositories having ‘C#’ and ‘Java’ programming language with more than one third of weightage in among source code, the subset files have been inner-joined with the filtered records [figure-24].

```
total_c_sharp_issues = 0
total_java_issues = 0
for x in range(0, 37):
    language_df = pd.read_csv('languageStats\\repoLanguages_{}.csv'.format(x), header=None,
                     names = ['row_index', 'Owner', 'RepoName', 'language', 'weightage'])
    language_df["weightage"] = language_df["weightage"].str.replace('%', '')
    #transforming weightage column
    language_df["weightage"] = language_df['weightage'].astype(str).astype(float)
    #filtering out dataset for java and c# with greater than 33% weightage
    java_df = language_df.loc[(language_df['weightage'] >= 33) & (language_df['language'] == 'Java')]
    c_sharp_df = language_df.loc[(language_df['weightage'] >= 33) & (language_df['language'] == 'C#')]
    #reading corresponding issues data set
    issue_df = pd.read_csv("multiplefilesWithLang\\subset_{}.csv".format(x))
    #joining both data set
    joined_java_df = pd.merge(issue_df, java_df, how='inner', on=['row_index', 'Owner', 'RepoName'],
                              left_on=None, right_on=None,
         left_index=False, right_index=False, sort=True,
         suffixes=['_i', '_l'], copy=True, indicator=False)
    joined_c_sharp_df = pd.merge(issue_df, c_sharp_df, how='inner', on=['row_index', 'Owner', 'RepoName'],
                                 left_on=None, right_on=None,
         left_index=False, right_index=False, sort=True,
         suffixes=['_i', '_l'], copy=True, indicator=False)
    #only body of issue is required, filtering out other columns
    joined_java_df = joined_java_df.filter(['Owner', 'RepoName', 'IssueNumber', 'body'], axis=1)
    joined_c_sharp_df = joined_c_sharp_df.filter(['Owner', 'RepoName', 'IssueNumber', 'body'], axis=1)
    #Saving it to a new csv file for analysis
    joined_java_df.to_csv('Java\\issues_data_{}.csv'.format(x), index=False )
    joined_c_sharp_df.to_csv('Csharp\\issues_data_{}.csv'.format(x), index=False )
    #adding counts for all files
    total_c_sharp_issues += len(joined_c_sharp_df)
    total_java_issues += len(joined_java_df)
```

Figure 18: Joining issues with programming languages, for repositories having C# and Java source code

Data Consolidation

Initial data was so huge, that processing of more than 5 million records always crashed computing systems. This was the reason to split the data and process them by taking one at time. After Data reduction and because of computation limitation, the size of target files has been reduced significantly. The target data has been prepared from only initial 37 parts out of total 106 parts.

The issues have been filtered out for both Java and C# based source repositories. These are available in 37 parts, created from the original Data. Upon analyzing size of each part, it is possible to create one file for each language [figure-25].

```
In [165]: def combineResults(language):
     ...:     current_directory = path.join(r'C:\\github_issues',language)
     ...:     chdir(current_directory)
     ...:     print(getcwd())
     ...:     directory = listdir(current_directory)
     ...:     combined_results = pd.DataFrame([])
     ...:     for counter, file in enumerate(directory):
     ...:         temp_df = pd.read_csv(file,encoding = "ISO-8859-1")
     ...:         combined_results = combined_results.append(temp_df)
     ...:     combined_results.to_csv(language+'_issues.csv', index=False, encoding = "ISO-8859-1")
     ...:
     ...:

In [166]: combineResults('Java')
C:\github_issues\Java

In [167]: combineResults('Csharp')
C:\github_issues\Csharp
```

Figure 19: Merging subset files for each of both programming languages

Data Description

Number of available rules, collected from MSDN, to describe issues of a specific category are as per table-2 [6]. The data is textual data.

Table 2: Data Description for Rules & Categories

Rule Category	Number of Rules
Cryptographic	2
Design	62
Globalization	11
Interoperability	16
Maintainability	6
Mobility	2
Naming	23
Performance	19

Portability	3
Reliability	6
Security	51
Usage	43

Total number of issues for C# are 36952, while Java based repositories have 87920 number of issues [figure-26].

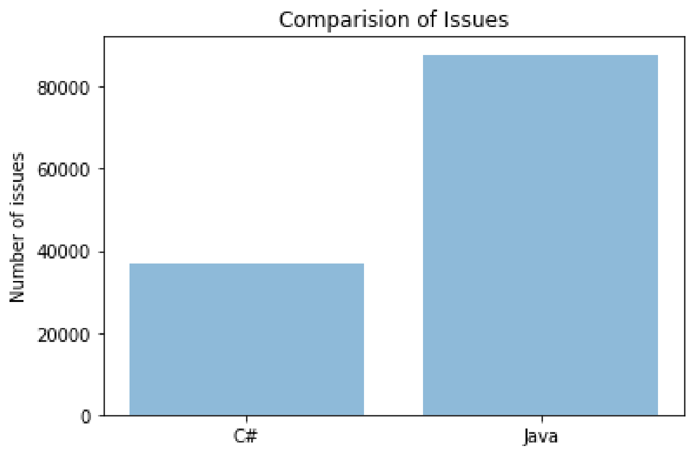

Figure 20: Comparison of number of issues among Java and C# based repositories

There are 4 columns as shown in figure-27 and Table-2.

```
     ...: df.columns
     ...:
Out[171]: Index(['Owner', 'RepoName', 'IssueNumber', 'body'], dtype='object')
```

Figure 21: Columns in the target data.

```
In [173]: df.head(10)
Out[173]:
               Owner              RepoName  IssueNumber  \
0            codenvy               codenvy         1477
1       jMonkeyEngine        jmonkeyengine          710
2              Azure    azure-sdk-for-java         1924
3  tunnelvisionlabs       java-immutable           25
4       opencharles          charles-rest          185
5       rzwitserloot                lombok         1461
6               red6            pdfcompare           15
7       confluentinc            kafka-rest          346
8          openbaton          generic-vnfm           31
9       spring-cloud      spring-cloud-gcp          131

                                             body
0  each time codenvy is started it creates volume...
1  meshcollidewith caches a bihtree for any mesh ...
2  this happens when creating a load balancer sec...
3  bunch of collections should have own collectio...
4  the puzzle 176a617e601 in srcmainjavacomamihai...
5  with v11612 fails to compile with custom nonnu...
6  i get this error when i run mvn eclipseeclipse...
7  the template properties only contain zookeeper...
8  i used the bootstrap method to install the dev...
9  currently checkstyle plugin is configured with...
```

```
In [175]: df.head(10)
Out[175]:
            Owner              RepoName  IssueNumber  \
0       james7132                Hourai           20
1          aspnet              Identity         1169
2     thestonefox                  VRTK         1158
3       LiveSplit             LiveSplit          867
4      Danielku15      BetterStartPage           25
5            mono            CocosSharp          417
6  textadventures                 quest          901
7      DevExpress  DevExtreme.AspNet.Data          142
8          RSuter                 NSwag          875
9     joaosilvapl                gabitv           21

                                             body
0   screenshot 20170604 at 6 14 00 pm\r\r\r\nserv...
1  i have a simple web application with two butto...
2  environment unity asset store 310 550f3 vive s...
3  if you copy a series of splits from the splits...
4  hi after a month of using bsp this happened to...
5  we have a somewhat simple game written using c...
6  as you can see in the following screenshot the...
7  c fact public void test1  datasourceloaderload...
8  i have an endpoint which returns a class class...
9   does windows 10 iot automatically detect the ...
```

Figure 22: First 10 records in issues for Java (left) and Csharp (right)

Data Dictionary

Column Name	Data Type	Description	Number of Unique Values	
			C#	Java
Owner	String	It represents the owner of source code repository at Github.	28923	72555
RepoName	String	It is name of repository and can be repeated for different owners. It is unique per owner.	36487	85085
IssueNumber	Number [Categorical]	It is identification number of reported issue. It is unique for a single repository and owner combination.	16430	29785
body	Text	The description of reported issue.	36952	87920

Modeling Technique

The study is primarily focused on one question, which is to identify nature and intent of reported issues, with use of text analytics and utilize the intent to compare two very popular programming languages C# and Java. At some extent, the objective is similar to Topic Mining.

To meet the objective, the modeling technique is based on linguistic analysis and distance measurement of two documents. First document is treated as the standard reference, and then based on term frequencies and inverse document frequencies of both document, a distance index is being formed. The higher value of distance index or similarity index will confirm the strong relation with the category.

This process will be performed for each reported issue for repository based on C# and Java programming languages. The average score for a category for issues related to C# will be compared to that of Java.

Building Model

To find similarity in two document genism library packages is utilized. NLTK is used to tokenize documents from the definition of rules for each category. The generated tokens are mapped to a number in the dictionary [5]. Using the dictionary, a corpus is being created. With corpus a term frequency and inverse document frequency are created, and similarity index is measured using these frequencies.

```python
import gensim
print(dir(gensim))
from nltk.tokenize import  word_tokenize

mobility_df = pd.read_csv('C:\\github_issues\\Rules\\Mobility.csv')

gen_mobility_docs = [[w.lower() for w in word_tokenize(text)]
                        for text in (mobility_df["Description"].values)]

mobility_dictionary = gensim.corpora.Dictionary(gen_mobility_docs)

mobility_corpus = [mobility_dictionary.doc2bow(gen_doc) for gen_doc in gen_mobility_docs]

mobility_tf_idf = gensim.models.TfidfModel(mobility_corpus)

mobility_sims = gensim.similarities.Similarity('Mobility_temp',mobility_tf_idf[mobility_corpus],
                                      num_features=len(mobility_dictionary))
```

Figure 23: creating term frequency for reference documents (Rules for each categories)

```python
def getSimilarityArrayCryptography(query_text):
    query_doc = [w.lower() for w in word_tokenize(query_text)]
    print(query_doc)
    query_doc_bow = Cryptography_dictionary.doc2bow(query_doc)
    print(query_doc_bow)
    query_doc_tf_idf = Cryptography_tf_idf[query_doc_bow]
    print(query_doc_tf_idf)
    return max(Cryptography_sims[query_doc_tf_idf])
```

Figure 24: Function to query term frequencies for each issue, and utilize it get maximum similarity index

```
csharp_df["Mobility_Arr"] = csharp_df["body"].astype(str).apply(getSimilarityArrayMobility)
csharp_df["Cryptography_Arr"] = csharp_df["body"].astype(str).apply(getSimilarityArrayCryptography)
csharp_df["Portability_Arr"] = csharp_df["body"].astype(str).apply(getSimilarityArrayPortability)
csharp_df["Maintainability_Arr"] = csharp_df["body"].astype(str).apply(getSimilarityArrayMaintainability)
csharp_df["Reliability_Arr"] = csharp_df["body"].astype(str).apply(getSimilarityArrayReliability)
csharp_df["Globalization_Arr"] = csharp_df["body"].astype(str).apply(getSimilarityArrayGlobalization)
csharp_df["Interoperability_Arr"] = csharp_df["body"].astype(str).apply(getSimilarityArrayInteroperability)
csharp_df["Performance_Arr"] = csharp_df["body"].astype(str).apply(getSimilarityArrayPerformance)
csharp_df["Naming_Arr"] = csharp_df["body"].astype(str).apply(getSimilarityArrayNaming)
csharp_df["Usage_Arr"] = csharp_df["body"].astype(str).apply(getSimilarityArrayUsage)
csharp_df["Security_Arr"] = csharp_df["body"].astype(str).apply(getSimilarityArraySecurity)
csharp_df["Design_Arr"] = csharp_df["body"].astype(str).apply(getSimilarityArrayDesign)


java_df["Mobility_Arr"] = java_df["body"].astype(str).apply(getSimilarityArrayMobility)
java_df["Cryptography_Arr"] = java_df["body"].astype(str).apply(getSimilarityArrayCryptography)
java_df["Portability_Arr"] = java_df["body"].astype(str).apply(getSimilarityArrayPortability)
java_df["Maintainability_Arr"] = java_df["body"].astype(str).apply(getSimilarityArrayMaintainability)
java_df["Reliability_Arr"] = java_df["body"].astype(str).apply(getSimilarityArrayReliability)
java_df["Globalization_Arr"] = java_df["body"].astype(str).apply(getSimilarityArrayGlobalization)
java_df["Interoperability_Arr"] = java_df["body"].astype(str).apply(getSimilarityArrayInteroperability)
java_df["Performance_Arr"] = java_df["body"].astype(str).apply(getSimilarityArrayPerformance)
java_df["Naming_Arr"] = java_df["body"].astype(str).apply(getSimilarityArrayNaming)
java_df["Usage_Arr"] = java_df["body"].astype(str).apply(getSimilarityArrayUsage)
java_df["Security_Arr"] = java_df["body"].astype(str).apply(getSimilarityArraySecurity)
java_df["Design_Arr"] = java_df["body"].astype(str).apply(getSimilarityArrayDesign)
```

Figure 25: Processing each issue to collect similarity score for each Category.

```
                                                body  Mobility_Arr  \
0  screenshot 20170604 at 6 14 00 pm   server ko...      0.000000
1  i have a simple web application with two butto...      0.401478
2  environment unity asset store 310 550f3 vive s...      0.368101
3  if you copy a series of splits from the splits...      0.324443
4  hi after a month of using bsp this happened to...      0.355409

   Cryptography_Arr  Portability_Arr  Maintainability_Arr  Reliability_Arr  \
0          0.000000         0.000000             0.000000         0.000000
1          0.250640         0.440722             0.217947         0.257243
2          0.277350         0.277733             0.227629         0.277717
3          0.358057         0.356683             0.202875         0.311441
4          0.358974         0.477655             0.281871         0.306430

   Globalization_Arr  Interoperability_Arr  Performance_Arr  Naming_Arr  \
0           0.000000              0.290808         0.451539    0.241976
1           0.220180              0.368070         0.257937    0.199759
2           0.189182              0.340548         0.212642    0.234814
3           0.297229              0.305862         0.243345    0.211676
4           0.160109              0.212098         0.338132    0.254889

   Usage_Arr  Security_Arr  Design_Arr
0   0.000000      0.165120    0.174212
1   0.220224      0.193680    0.233413
2   0.296591      0.297931    0.146545
3   0.212755      0.142085    0.213449
4   0.329749      0.466870    0.402720

In [92]:
```

Figure 26:Maximum similarity Indices for issues of repository based on C# Language

```
                                               body  Mobility_Arr  \
0  each time codenvy is started it creates volume...      0.231455
1  meshcollidewith caches a bihtree for any mesh ...      0.362738
2  this happens when creating a load balancer sec...      0.229416
3  bunch of collections should have own collectio...      0.374634
4  the puzzle 176a617e601 in srcmainjavacomamihai...      0.410391

   Cryptography_Arr  Portability_Arr  Maintainability_Arr  Reliability_Arr  \
0          0.369800         0.453889             0.300691         0.307429
1          0.330534         0.498572             0.238651         0.310931
2          0.392232         0.378229             0.268902         0.203638
3          0.339683         0.580389             0.204214         0.333537
4          0.323575         0.334215             0.249863         0.257358

   Globalization_Arr  Interoperability_Arr  Performance_Arr  Naming_Arr  \
0           0.150355              0.284255         0.271123    0.271255
1           0.205824              0.285035         0.264160    0.211515
2           0.189343              0.233746         0.259230    0.232842
3           0.155835              0.255279         0.183986    0.215153
4           0.162530              0.269126         0.172041    0.260947

   Usage_Arr  Security_Arr  Design_Arr
0   0.251831      0.163291    0.340887
1   0.200599      0.216931    0.273619
2   0.236257      0.175199    0.170762
3   0.200483      0.282090    0.150072
4   0.174678      0.160000    0.238826

In [97]:
```

Figure 27: Maximum similarity Indices for issues of repository based on Java Language

Model Assessment

To compare the strong or loose bonding of a category with a programming language, average of similarity index is used. If score of similarity is lower for a category, then the programming language has less number of issues and vulnerability. The programming language rarely leads to an error related to that specific category.

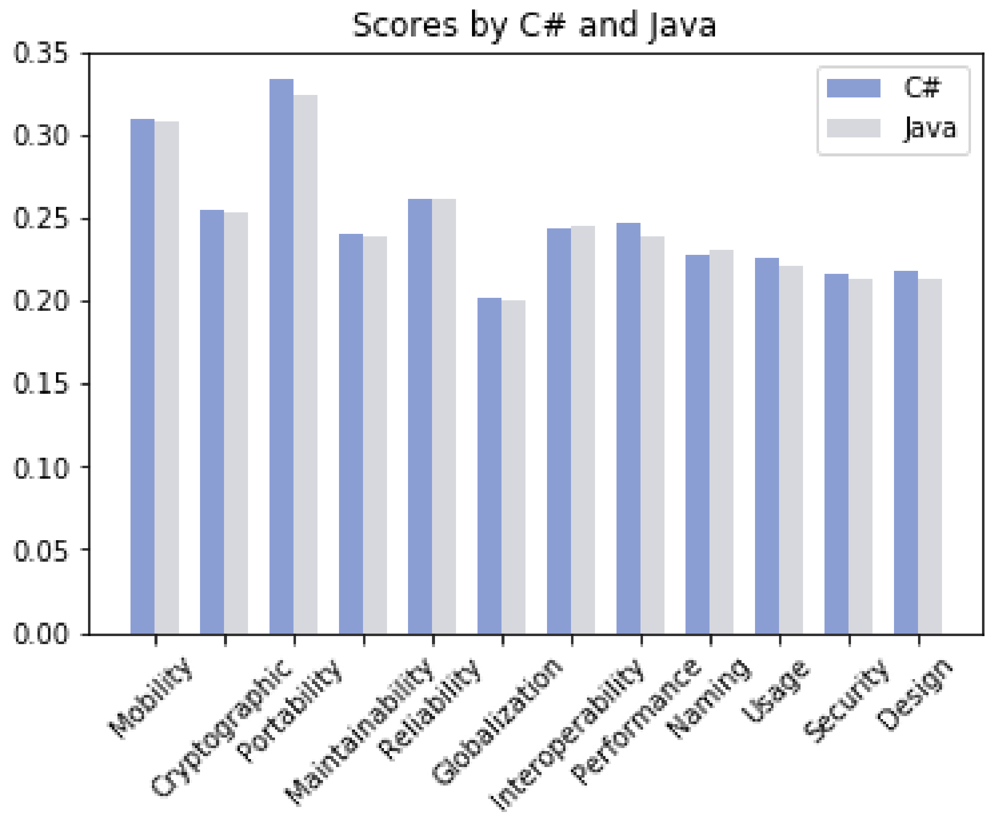

Figure 28: Average Issue Similarity Indices for each category

Conclusion

With the results, obtained by assessing the average value of similarity index, for both C# and Java, there is not much of difference in both languages. It concludes that both have similar type of issue to solve.

From the result it can also be observed that most of issues are related to Cryptography, Mobility and Maintainability.

Future Work

With 2/3 part of data still to extracted over next 6-8 months, the study can reveal newer information. In this study, rules are adapted from MSDN website [6]. Same set of rules have been used for both C# and Java. If, rules can be transformed for any specific language, yield will change.

References

1. GitHub Archive - https://www.gharchive.org/
2. Kaggle Dataset Github Issues - https://www.kaggle.com/davidshinn/github-issues
3. David Shinn Profile at Medium - https://medium.com/@david.shinn
4. Python Library to detect languages - https://github.com/Mimino666/langdetect
5. https://www.oreilly.com/learning/how-do-i-compare-document-similarity-using-python
6. MSDN Code Analysis - https://msdn.microsoft.com/en-us/library/ee1hzekz.aspx