1. Explanation of design choices in restructuring, cleaning, and manipulating the datasets

1.1. GitData.json:

Each document in the collection contains information about each file that has been changed whenever a commit is made. There is commit-level information summarising the details about the committer, author, project name, and total number of lines in files within the commit. The collection presents fields on a more granular file-level, for example the exact source code before and after the changes were made for each file or the exact additions and/or deletions made in the file. The commits, when pushed can signify a logical unit of change within a repository.

Existing:

- Structure: As described above, amongst other levels, the collections mainly have commit-level and file-level data. There are 60,334 documents and 15,124 unique hashes. As per Git structure all file changes done under the same commit share the same hash. The current structure of the collection is such that there exists a document for each file that has been changed, meaning that there are 60,334 files that have been changed in total. Thus, there is commit-level information repeating as many times as there are files in that commit across the entire collection.

The files can have same names but be belonging in different locations as well and this information can be verified through the old path and new path fields. These paths will be different for files in different locations, despite sharing the same name. In the total 9 project/repository names, the PyTorch organisation has various repositories based on the various modules handled by the project such as audio, elastic, text, vision, and PyTorch itself. These modules all belong to the one parental organisation. There are duplications of documents as well, for example, there are some documents with most information listed twice, however, certain fields exist only in one of them. None of the files was merged, all the commits happened in the main/master branch, there were 1102 committers perhaps each looking at more than one commits, lastly, more than one file were changed in each commit.

- Inefficiencies: With the repeating information about the commits, it is difficult to track each file associated to the commit in order to then appreciate how the project has evolved over time. More specifically, to query about a repository such as TensorFlow and to understand how new upgrades have been developing, the current structure would complicate the process because file-level information is spread across the whole collection. Each file would need to be inspected by looking at each document individually. The idea of the collection should be to explain changes occurring collectively at once, and not to manually pick up these changes and build them up into their respective units. The current storage size of the collection is 7989.91 MB.

Post-Restructuring:

- Structure: The new structure condenses the collection to 15,124 documents to represent each commit (hash) that has been generated from 60,334 documents representing each file. Further, author fields earlier such as name, date have been put into a nested author object, same for the committer information. The collections have a new field 'organisation' to show parental organisation handling the particular project and to make comparisons between Tensorlow and PyTorch more efficient.

Now, the major restructuring has been the addition of a new field called 'files_changes', which is a nested array of objects each comprising of fields related to changes in the files part within that commit. These fields reveal the old path, new path, the exact difference in lines before and after the change (including the source code). There is code-level metrics as well, such as Cyclomatic complexity, number of lines, and tokens – all under the file object. The duplicated documents were also deleted and the unique fields from both the documents were combined to represent a single entity.

- Value added: The new structure holistically combines first a high-level view of the commit, and then the possibility to go into individual file-levels, and even finer into the code in the files. Having each changed file stored within a list and as an object inside the commit they belong to will simplify retrieving the information across all the files that are part of that one logical unit of change, and therefore, help explain the change itself. The shared commit-level fields for each file can further reveal information about for example when the change was made, and by whom it was made – especially how these committers and changes shaped the repository over time or if the modular codes within the commit shared similar complexities. The new reduced storage size of the collection is 7948.46 MB.

- Recommendations: Further improvements in the structuring of the files can be done by merging code-level information together as another nested object within the files, however, this depends more upon the complexity and flexibility of the analysis goals. For example, it may not be relevant all the time to inspect individual codes but only the metrics revolving around them.

1.2. GitIssues.json:

Each document in this collection contains information about the issues that have been created in the PyTorch and TensorFlow organisation of GitHub. The fields provide information about the issues and the communication/comments that are left by users on a particular issue. Issues are integral in the development of a project as collaborators, users or creators can not only flag and resolve bugs but also present new ideas and feedback on existing work on GitHub, thus enabling a collective ecosystem of continuous development – an important feature in Open Source Software (OSS) development.

Existing:

- Structure: There are 32,329 documents in the collection, each of which represents the comments made across all the unique issues that are created within the two projects over the time period between 02/2016 and 12/2021. The number of unique issues were 5,400 i.e. the number of main titles. The structure repeats information in the documents that is related to the issue, the only changing parts between the documents belong to the comments. For example, if an issue has 11 comments, that issue-level information will exist 11 times within the 11 comment-level documents. The issue-level information comprises of title, status, description, created/updated/closed times, owners of the issue, total comments, total reactions and other things. The reactions were listed in an array and were repeated as many times as they were left by the users. The individual comment-level fields depict the comment creation/updating times, information of user making the comment, and the actual text of the comment.

- Inefficiencies: As mentioned, the issue-level information repeats as many times as there are comments inside that issue. The structure hinders the ability to assess the links and progress within a particular issue because querying n different comments in order to understand development of a particular solution or ideas within a common thread can be confusing. It is important to understand that it is the issue that binds the discussion, not the other way around. For example, in order to understand how users provided solutions to a failed instance of PyTorch installation, one would have to look into the 12 existing documents of that issue. The current size of the collection is 141.83 MB.

Post-Restructuring:

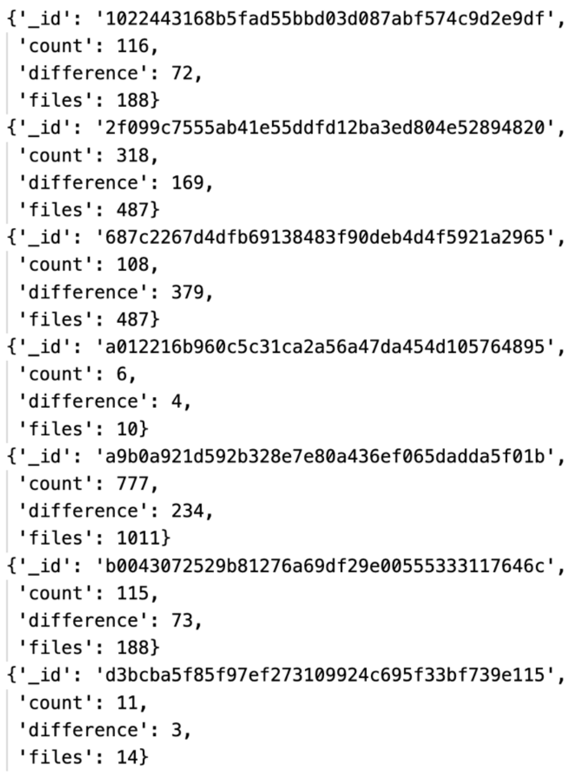
- Structure: The changes that were made to the collection reduced the number of documents to 5,400 documents i.e., the number of unique issues. The setup is such that each document, in addition to the meta information such as status or creator of the issue contains fields that are lists showing the assignees, descriptive labels, and the variety of reactions received by the issue. The reactions were changed from having repetitions in the list to showing the total counts for each reaction left by users. Moreover, the document provides a fourth array of nested objects for each comment made under the issue whose fields describe the comment maker, text, and creation/updating time. The organisation name was added for this collection as well. There was another object pertaining to the closing details of the issue that was added with fields ‘closed_at’ and ‘closed_by’.

- Value added: The collection now fixes the redundancy that was due to the multiple documents for the multiple comments in one issue, and now contains 84% less documents than before. The more hierarchical approach stacks the comments in one place right inside the issues they belong to. Whilst analysing the data, the comments can be easily correlated with the type and title of the issue, and if needed, the textual data within the comments can then be analysed based on the analysis goals. Additionally, highly active users – potentially key to project development – contributing to the comments across different issues can be identified by obtaining and comparing the counts of unique users within and across issues. The downsized collection measures 37.33 MB.

2. Descriptive Insights

2.1. GitData.json

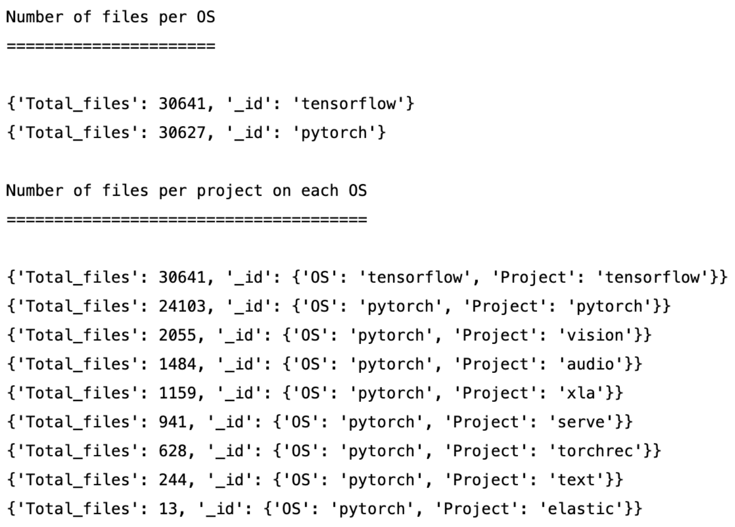
1. Inaccuracy in the number of files: The information such as the number of files per commit displayed is actually different from the number of files included inside the commit for some of the hashes. The difference between count of file objects (under ‘files_changes’) and the number of files (‘files’) was calculated, if it is not 0, then there is an inconsistency (Figure 1). 7 such instances were found; changing the ‘files’ to represent the number of files in the collection will inaccurately represent the actual state of the modularity of the project, and therefore, any analysis tied with it. It is recommended to collect the data on the missing files and add them to the collection.



```
{'_id': '1022443168b5fad55bbd03d087abf574c9d2e9df',
 'count': 116,
 'difference': 72,
 'files': 188}
{'_id': '2f099c7555ab41e55ddfd12ba3ed804e52894820',
 'count': 318,
 'difference': 169,
 'files': 487}
{'_id': '687c2267d4dfb69138483f90deb4d4f5921a2965',
 'count': 108,
 'difference': 379,
 'files': 487}
{'_id': 'a012216b960c5c31ca2a56a47da454d105764895',
 'count': 6,
 'difference': 4,
 'files': 10}
{'_id': 'a9b0a921d592b328e7e80a436ef065dadda5f01b',
 'count': 777,
 'difference': 234,
 'files': 1011}
{'_id': 'b0043072529b81276a69df29e00555333117646c',
 'count': 115,
 'difference': 73,
 'files': 188}
{'_id': 'd3bcba5f85f97ef273109924c695f33bf739e115',
 'count': 11,
 'difference': 3,
 'files': 14}
```

Figure 1: Number of inaccuracies in file size and included files.

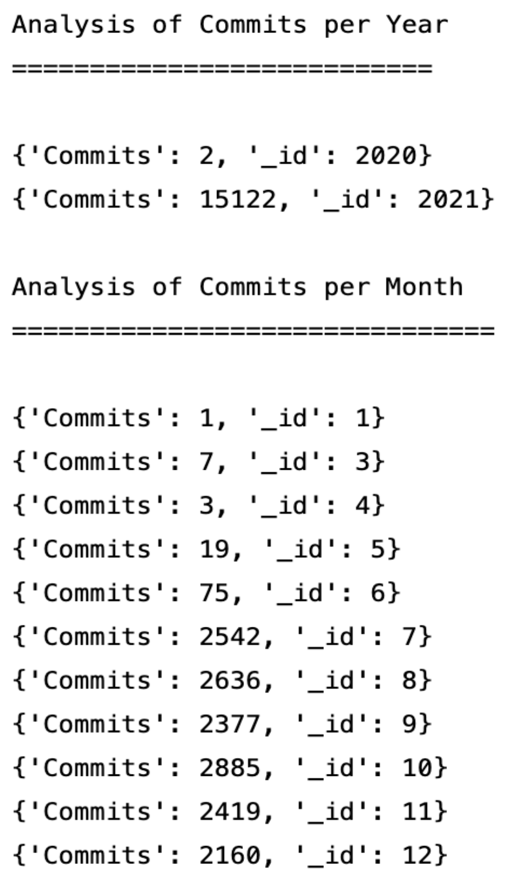
2. Modularity of projects:
   Large projects such as TensorFlow and PyTorch have been broken down into a large number of modules as can be seen in the number of files that are within these projects (Figure 2). That also explains the large number of collaborators that exist within these projects.



```
Number of files per OS
======================

{'Total_files': 30641, '_id': 'tensorflow'}
{'Total_files': 30627, '_id': 'pytorch'}

Number of files per project on each OS
======================================

{'Total_files': 30641, '_id': {'OS': 'tensorflow', 'Project': 'tensorflow'}}
{'Total_files': 24103, '_id': {'OS': 'pytorch', 'Project': 'pytorch'}}
{'Total_files': 2055, '_id': {'OS': 'pytorch', 'Project': 'vision'}}
{'Total_files': 1484, '_id': {'OS': 'pytorch', 'Project': 'audio'}}
{'Total_files': 1159, '_id': {'OS': 'pytorch', 'Project': 'xla'}}
{'Total_files': 941, '_id': {'OS': 'pytorch', 'Project': 'serve'}}
{'Total_files': 628, '_id': {'OS': 'pytorch', 'Project': 'torchrec'}}
{'Total_files': 244, '_id': {'OS': 'pytorch', 'Project': 'text'}}
{'Total_files': 13, '_id': {'OS': 'pytorch', 'Project': 'elastic'}}
```

Figure 2: Breakdown of projects into modules (files)

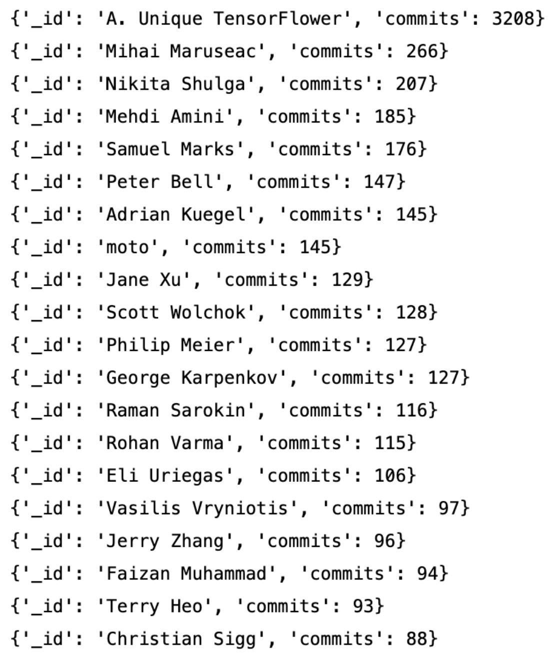
3. The commits have really been placed mostly in 2021. On a monthly level, the second half of the year is especially full of activity, which might indicate a certain level of increased engagement by developers and more free time at disposal after June (Figure 3).



```
Analysis of Commits per Year
===========================

{'Commits': 2, '_id': 2020}
{'Commits': 15122, '_id': 2021}

Analysis of Commits per Month
===============================

{'Commits': 1, '_id': 1}
{'Commits': 7, '_id': 3}
{'Commits': 3, '_id': 4}
{'Commits': 19, '_id': 5}
{'Commits': 75, '_id': 6}
{'Commits': 2542, '_id': 7}
{'Commits': 2636, '_id': 8}
{'Commits': 2377, '_id': 9}
{'Commits': 2885, '_id': 10}
{'Commits': 2419, '_id': 11}
{'Commits': 2160, '_id': 12}
```

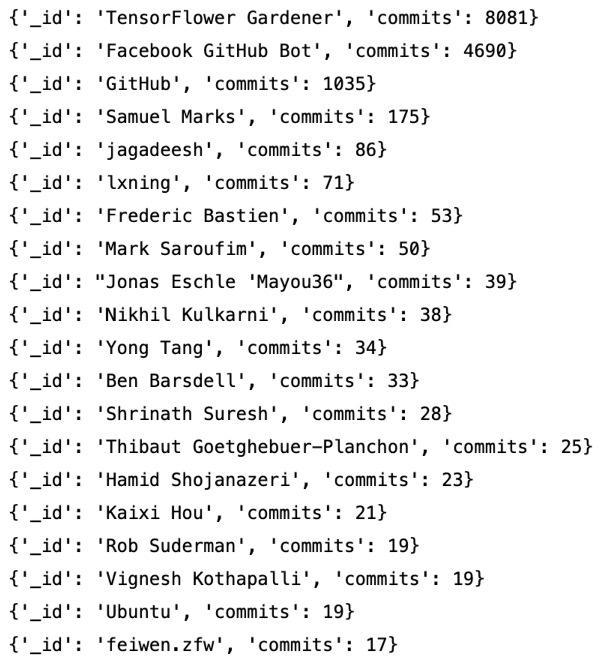
Figure 3: Time and frequency of commits

4. Top 20 authors vs top 20 committers ranked by the number of commits they created.
   The purpose of these figures was to understand if an author simply creates a patch and then reduces his/her own activity to leave it for other developers to make changes and contributions. Here, it can be observed that there 3 committers that create a sizeable number of commits and 1 author that initiates the most patches. There are no names appearing on both the sides (Figure 4). Therefore, there seems to be the case that some top active leaders in the ecosystem exist whilst the others show much less contribution despite being an OSS project.



```
{'_id': 'A. Unique TensorFlower', 'commits': 3208}
{'_id': 'Mihai Maruseac', 'commits': 266}
{'_id': 'Nikita Shulga', 'commits': 207}
{'_id': 'Mehdi Amini', 'commits': 185}
{'_id': 'Samuel Marks', 'commits': 176}
{'_id': 'Peter Bell', 'commits': 147}
{'_id': 'Adrian Kuegel', 'commits': 145}
{'_id': 'moto', 'commits': 145}
{'_id': 'Jane Xu', 'commits': 129}
{'_id': 'Scott Wolchok', 'commits': 128}
{'_id': 'Philip Meier', 'commits': 127}
{'_id': 'George Karpenkov', 'commits': 127}
{'_id': 'Raman Sarokin', 'commits': 116}
{'_id': 'Rohan Varma', 'commits': 115}
{'_id': 'Eli Uriegas', 'commits': 106}
{'_id': 'Vasilis Vryniotis', 'commits': 97}
{'_id': 'Jerry Zhang', 'commits': 96}
{'_id': 'Faizan Muhammad', 'commits': 94}
{'_id': 'Terry Heo', 'commits': 93}
{'_id': 'Christian Sigg', 'commits': 88}
```

Authors



```
{'_id': 'TensorFlower Gardener', 'commits': 8081}
{'_id': 'Facebook GitHub Bot', 'commits': 4690}
{'_id': 'GitHub', 'commits': 1035}
{'_id': 'Samuel Marks', 'commits': 175}
{'_id': 'jagadeesh', 'commits': 86}
{'_id': 'lxning', 'commits': 71}
{'_id': 'Frederic Bastien', 'commits': 53}
{'_id': 'Mark Saroufim', 'commits': 50}
{'_id': "Jonas Eschle 'Mayou36", 'commits': 39}
{'_id': 'Nikhil Kulkarni', 'commits': 38}
{'_id': 'Yong Tang', 'commits': 34}
{'_id': 'Ben Barsdell', 'commits': 33}
{'_id': 'Shrinath Suresh', 'commits': 28}
{'_id': 'Thibaut Goetghebuer-Planchon', 'commits': 25}
{'_id': 'Hamid Shojanazeri', 'commits': 23}
{'_id': 'Kaixi Hou', 'commits': 21}
{'_id': 'Rob Suderman', 'commits': 19}
{'_id': 'Vignesh Kothapalli', 'commits': 19}
{'_id': 'Ubuntu', 'commits': 19}
{'_id': 'feiwen.zfw', 'commits': 17}
```

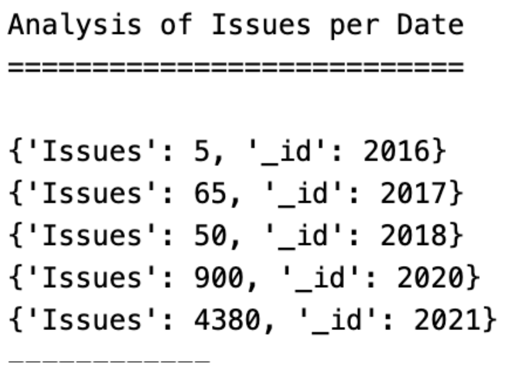
Committers

Figure 4: Difference between author and committer activity

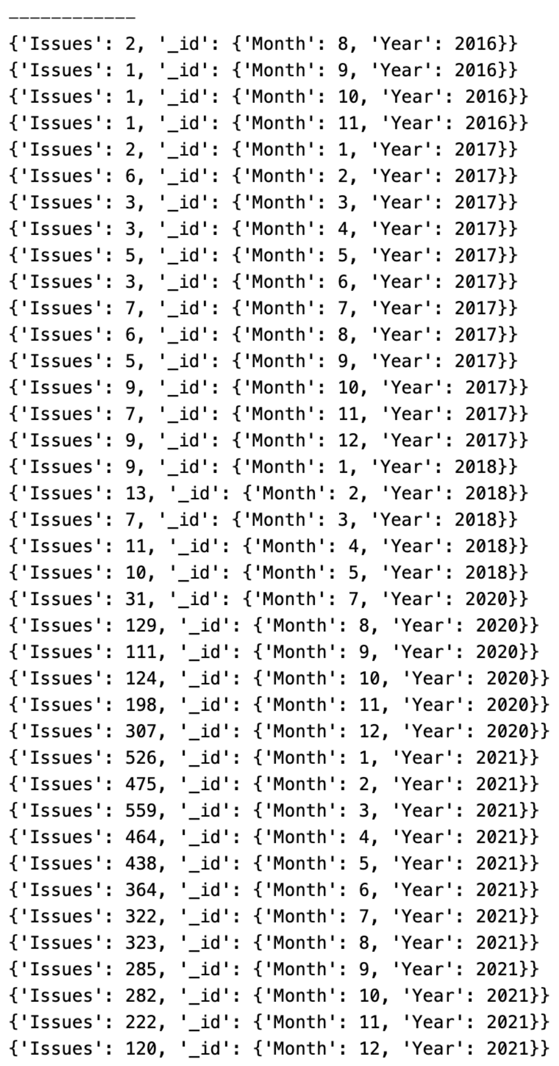
2.2. GitIssues.json

5. Frequency of issues by date:

   From a yearly perspective, the number of issues have only increased each year, and the jump from 2020 to 2021 has been 5x, perhaps due to the increasing AI development (Figure 5).



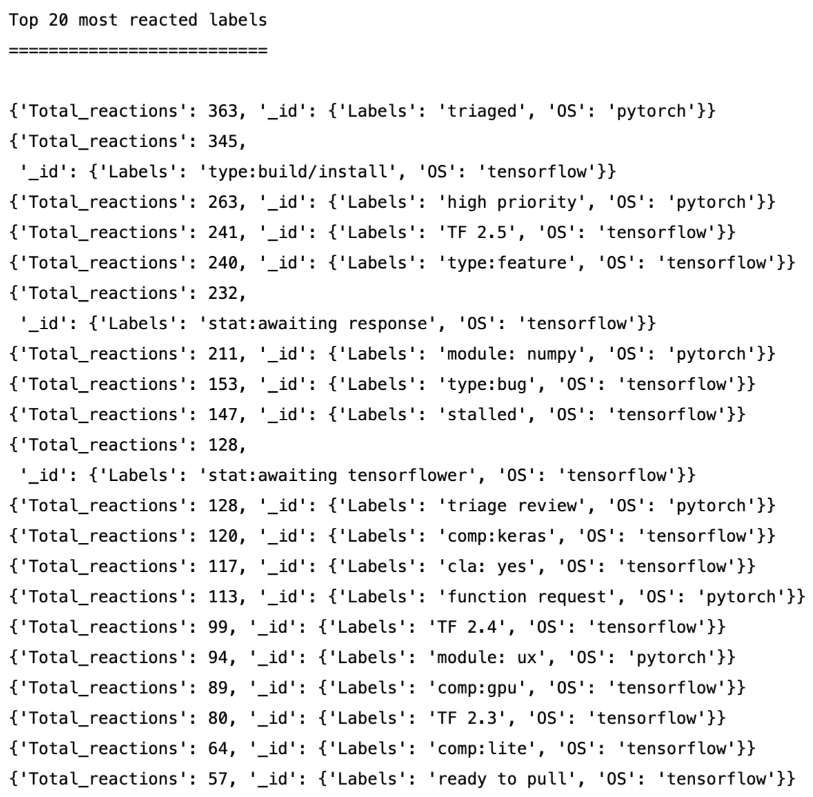
```
Analysis of Issues per Date
===========================

{'Issues': 5, '_id': 2016}
{'Issues': 65, '_id': 2017}
{'Issues': 50, '_id': 2018}
{'Issues': 900, '_id': 2020}
{'Issues': 4380, '_id': 2021}
____________
```



```
------------
{'Issues': 2, '_id': {'Month': 8, 'Year': 2016}}
{'Issues': 1, '_id': {'Month': 9, 'Year': 2016}}
{'Issues': 1, '_id': {'Month': 10, 'Year': 2016}}
{'Issues': 1, '_id': {'Month': 11, 'Year': 2016}}
{'Issues': 2, '_id': {'Month': 1, 'Year': 2017}}
{'Issues': 6, '_id': {'Month': 2, 'Year': 2017}}
{'Issues': 3, '_id': {'Month': 3, 'Year': 2017}}
{'Issues': 3, '_id': {'Month': 4, 'Year': 2017}}
{'Issues': 5, '_id': {'Month': 5, 'Year': 2017}}
{'Issues': 3, '_id': {'Month': 6, 'Year': 2017}}
{'Issues': 7, '_id': {'Month': 7, 'Year': 2017}}
{'Issues': 6, '_id': {'Month': 8, 'Year': 2017}}
{'Issues': 5, '_id': {'Month': 9, 'Year': 2017}}
{'Issues': 9, '_id': {'Month': 10, 'Year': 2017}}
{'Issues': 7, '_id': {'Month': 11, 'Year': 2017}}
{'Issues': 9, '_id': {'Month': 12, 'Year': 2017}}
{'Issues': 9, '_id': {'Month': 1, 'Year': 2018}}
{'Issues': 13, '_id': {'Month': 2, 'Year': 2018}}
{'Issues': 7, '_id': {'Month': 3, 'Year': 2018}}
{'Issues': 11, '_id': {'Month': 4, 'Year': 2018}}
{'Issues': 10, '_id': {'Month': 5, 'Year': 2018}}
{'Issues': 31, '_id': {'Month': 7, 'Year': 2020}}
{'Issues': 129, '_id': {'Month': 8, 'Year': 2020}}
{'Issues': 111, '_id': {'Month': 9, 'Year': 2020}}
{'Issues': 124, '_id': {'Month': 10, 'Year': 2020}}
{'Issues': 198, '_id': {'Month': 11, 'Year': 2020}}
{'Issues': 307, '_id': {'Month': 12, 'Year': 2020}}
{'Issues': 526, '_id': {'Month': 1, 'Year': 2021}}
{'Issues': 475, '_id': {'Month': 2, 'Year': 2021}}
{'Issues': 559, '_id': {'Month': 3, 'Year': 2021}}
{'Issues': 464, '_id': {'Month': 4, 'Year': 2021}}
{'Issues': 438, '_id': {'Month': 5, 'Year': 2021}}
{'Issues': 364, '_id': {'Month': 6, 'Year': 2021}}
{'Issues': 322, '_id': {'Month': 7, 'Year': 2021}}
{'Issues': 323, '_id': {'Month': 8, 'Year': 2021}}
{'Issues': 285, '_id': {'Month': 9, 'Year': 2021}}
{'Issues': 282, '_id': {'Month': 10, 'Year': 2021}}
{'Issues': 222, '_id': {'Month': 11, 'Year': 2021}}
{'Issues': 120, '_id': {'Month': 12, 'Year': 2021}}
```

Figure 5: Frequency of issues by dates

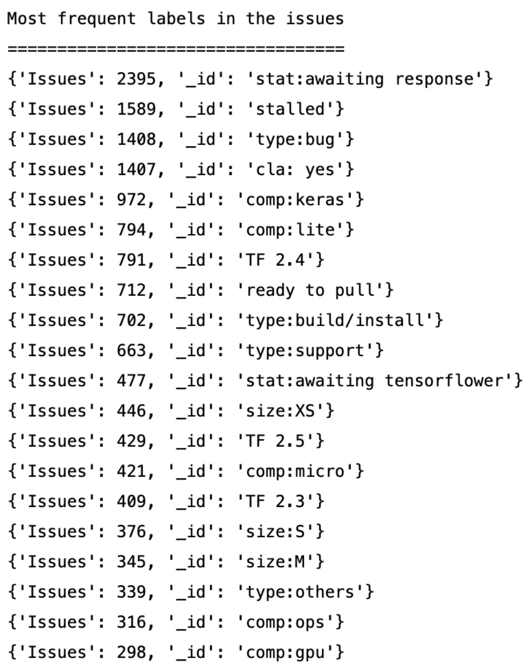
6. Labelling of issues that have the highest reactions:
   The most reactions have been accumulated by the 'triaged' label within PyTorch. According to the repository, this label signifies that a team member has acknowledged the issue and prioritised it to a task board (Figure 6). It is quite natural to assume that issues with more reactions tend to be pertinent and worthy of developer priority. Even the third label that received high reactions is to do with high priority for PyTorch. Other common labels with high reactions are related to features, installation, and bugs.



```
Top 20 most reacted labels
==========================

{'Total_reactions': 363, '_id': {'Labels': 'triaged', 'OS': 'pytorch'}}
{'Total_reactions': 345,
 '_id': {'Labels': 'type:build/install', 'OS': 'tensorflow'}}
{'Total_reactions': 263, '_id': {'Labels': 'high priority', 'OS': 'pytorch'}}
{'Total_reactions': 241, '_id': {'Labels': 'TF 2.5', 'OS': 'tensorflow'}}
{'Total_reactions': 240, '_id': {'Labels': 'type:feature', 'OS': 'tensorflow'}}
{'Total_reactions': 232,
 '_id': {'Labels': 'stat:awaiting response', 'OS': 'tensorflow'}}
{'Total_reactions': 211, '_id': {'Labels': 'module: numpy', 'OS': 'pytorch'}}
{'Total_reactions': 153, '_id': {'Labels': 'type:bug', 'OS': 'tensorflow'}}
{'Total_reactions': 147, '_id': {'Labels': 'stalled', 'OS': 'tensorflow'}}
{'Total_reactions': 128,
 '_id': {'Labels': 'stat:awaiting tensorflower', 'OS': 'tensorflow'}}
{'Total_reactions': 128, '_id': {'Labels': 'triage review', 'OS': 'pytorch'}}
{'Total_reactions': 120, '_id': {'Labels': 'comp:keras', 'OS': 'tensorflow'}}
{'Total_reactions': 117, '_id': {'Labels': 'cla: yes', 'OS': 'tensorflow'}}
{'Total_reactions': 113, '_id': {'Labels': 'function request', 'OS': 'pytorch'}}
{'Total_reactions': 99, '_id': {'Labels': 'TF 2.4', 'OS': 'tensorflow'}}
{'Total_reactions': 94, '_id': {'Labels': 'module: ux', 'OS': 'pytorch'}}
{'Total_reactions': 89, '_id': {'Labels': 'comp:gpu', 'OS': 'tensorflow'}}
{'Total_reactions': 80, '_id': {'Labels': 'TF 2.3', 'OS': 'tensorflow'}}
{'Total_reactions': 64, '_id': {'Labels': 'comp:lite', 'OS': 'tensorflow'}}
{'Total_reactions': 57, '_id': {'Labels': 'ready to pull', 'OS': 'tensorflow'}}
```

Figure 6: Issue labels and the level of reactions received by them

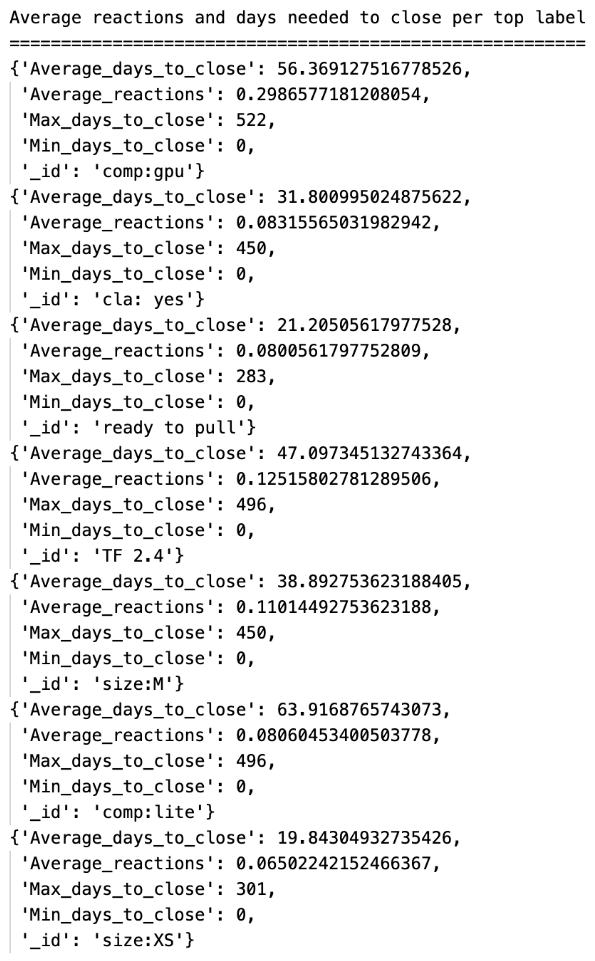
7. Most frequently occurring labels within the issues.
   'awaiting response' and 'stalled' are the two most used labels in the issues, showing the potential waiting or hold-up in closing/concluding an issue (Figure 7). Collaborators also discuss specific modules or add labels to show status within repositories or even to signify that a particular developer's response is being awaited ('awaiting TensorFlower').



```
Most frequent labels in the issues
==================================
{'Issues': 2395, '_id': 'stat:awaiting response'}
{'Issues': 1589, '_id': 'stalled'}
{'Issues': 1408, '_id': 'type:bug'}
{'Issues': 1407, '_id': 'cla: yes'}
{'Issues': 972, '_id': 'comp:keras'}
{'Issues': 794, '_id': 'comp:lite'}
{'Issues': 791, '_id': 'TF 2.4'}
{'Issues': 712, '_id': 'ready to pull'}
{'Issues': 702, '_id': 'type:build/install'}
{'Issues': 663, '_id': 'type:support'}
{'Issues': 477, '_id': 'stat:awaiting tensorflower'}
{'Issues': 446, '_id': 'size:XS'}
{'Issues': 429, '_id': 'TF 2.5'}
{'Issues': 421, '_id': 'comp:micro'}
{'Issues': 409, '_id': 'TF 2.3'}
{'Issues': 376, '_id': 'size:S'}
{'Issues': 345, '_id': 'size:M'}
{'Issues': 339, '_id': 'type:others'}
{'Issues': 316, '_id': 'comp:ops'}
{'Issues': 298, '_id': 'comp:gpu'}
```

Figure 7: Most used labels in the issues

8. Days needed to close an issue, based on labels.
   It can be seen that the highest average and maximum number of days to close an issue is for the label ‘comp:gpu’, which shows how training and hardware related issues can take longer to figure out (Figure 8).



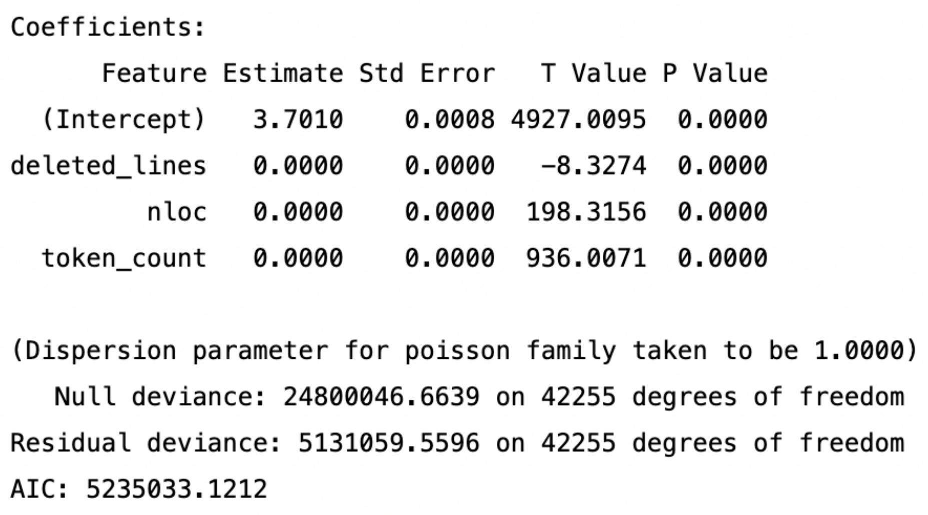
```
Average reactions and days needed to close per top label
========================================================
{'Average_days_to_close': 56.369127516778526,
 'Average_reactions': 0.2986577181208054,
 'Max_days_to_close': 522,
 'Min_days_to_close': 0,
 '_id': 'comp:gpu'}
{'Average_days_to_close': 31.800995024875622,
 'Average_reactions': 0.08315565031982942,
 'Max_days_to_close': 450,
 'Min_days_to_close': 0,
 '_id': 'cla: yes'}
{'Average_days_to_close': 21.20505617977528,
 'Average_reactions': 0.0800561797752809,
 'Max_days_to_close': 283,
 'Min_days_to_close': 0,
 '_id': 'ready to pull'}
{'Average_days_to_close': 47.097345132743364,
 'Average_reactions': 0.12515802781289506,
 'Max_days_to_close': 496,
 'Min_days_to_close': 0,
 '_id': 'TF 2.4'}
{'Average_days_to_close': 38.892753623188405,
 'Average_reactions': 0.11014492753623188,
 'Max_days_to_close': 450,
 'Min_days_to_close': 0,
 '_id': 'size:M'}
{'Average_days_to_close': 63.9168765743073,
 'Average_reactions': 0.08060453400503778,
 'Max_days_to_close': 496,
 'Min_days_to_close': 0,
 '_id': 'comp:lite'}
{'Average_days_to_close': 19.84304932735426,
 'Average_reactions': 0.06502242152466367,
 'Max_days_to_close': 301,
 'Min_days_to_close': 0,
 '_id': 'size:XS'}
```

Figure 8: Days needed to close the issues per each label

3. Further Data Analysis

1. <u>Regress complexity</u> on other features to see if it can be explained by deletions, insertions, token count, lines of code. Essentially, with cyclomatic complexity, higher numbers are bad and lower numbers are good. A high complexity indicates that the code might be hard to maintain in terms of structure and also more prone to producing errors (Microsoft 2022). In a collaborative environment with developers from diverse areas, ensuring a smooth structure may be challenging. Generalised Linear Regression with the assumption that Poission complexity follows a Poisson distribution was done to understand if the cyclomatic complexity can be predicted by the mentioned features and to determine how a change in for example number lines of code may affect the complexity of the overall code.

   Poisson was chosen due to the non-negative property of complexity parameter, and log link due to some values being on a much higher end than the others. The regression results were such that all 3 predictors were statistically significant in their p-values (Figure 9) to reject the null hypothesis that they do not impact the complexity parameter. However, the extent of impact is really small according to the dataset. The coefficients are all very close to zero and after exponentiating them, ‘*deleted_lines*’, will contribute to a 0.01% decrease in the complexity, whereas ‘*nloc’* and ‘*token_count’* will both contribute to less than 0.0001% increase in the complexity. With regards to the overall fit, the null and residual deviances give a Chi-Square p-value of almost 0, implying that model may not be a good fit and needs more features to explain the complexity.



```
Coefficients:
      Feature Estimate Std Error   T Value P Value
  (Intercept)   3.7010    0.0008 4927.0095  0.0000
deleted_lines   0.0000    0.0000   -8.3274  0.0000
         nloc   0.0000    0.0000  198.3156  0.0000
  token_count   0.0000    0.0000  936.0071  0.0000

(Dispersion parameter for poisson family taken to be 1.0000)
   Null deviance: 24800046.6639 on 42255 degrees of freedom
Residual deviance: 5131059.5596 on 42255 degrees of freedom
AIC: 5235033.1212
```

Figure 9: Regression Summary

2. <u>Network of collaboration </u>between programmers has been created as per whiche the nodes represent developers as authors/committers and they are connected if they have either authored or committed a patch on the same hash (Figure 10). The node colour represents the organisation they belong to (blue for TensorFlow, and orange for PyTorch). PyTorch and TensorFlow, while open source, are competing projects. The idea behind the network graph was to analyse if unexpected allies are formed whilst dedicating time to these competing projects or in an OSS ecosystem (O’Mahony and Beckhy 2008). As the graph shows, there are isolated developers on the periphery who only participate with few people or just author something and leave for few other people to follow on. There are clusters with a focal point of collaborators within each cluster which might indicate that some developers are contributing in many places by authoring or committing on various patches.

   Lastly, there are other hubs that connect these clusters and enable flow of information. Interestingly, however weak, or strong, there exists a connection between isolated developers or group of developers and there are many crossovers between people working on competing projects. It may be safe to conclude that in an OSS ecosystem, developers may be agnostic to the projects and unify towards a collective and collaborative environment to enable industry-wide improvements, showing a good representation of coopetition (Nguyen Duc et al 2017).

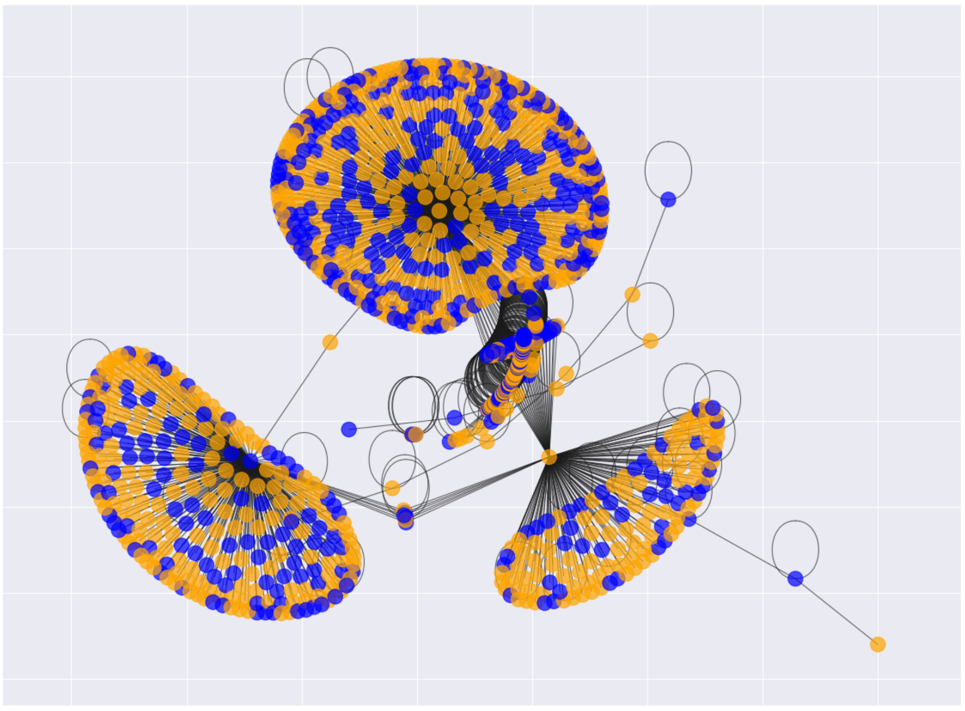


Figure 10: Developer collaboration network (Blue: TensorFlow, Orange: PyTorch)

3. Time-Frequency of commits was created to understand the motivations behind developers' level of contribution within the OSS ecosystem. The commits have been summed up for each month and visualised in the plot below (Figure 11). The idea was to understand if there is seasonality or any patterns that affect development depending upon when collaborators may have relatively more time outside their daily jobs, and how individual motivation affects the outcome (von Krogh et al 2012). It is important to highlight that the commits data ranges from 11/2020 to 12/2021 and whilst that does not provide much seasonal insights, it does show a clear surge in commit activity during the second half of 2021. This can be corroborated with the fact that TensorFlow, for example had very minimal version updates/releases between January 2021 to May 2021, compared to the months that followed. (Refer to the update history here: https://github.com/tensorflow/tensorflow/releases?page=4)

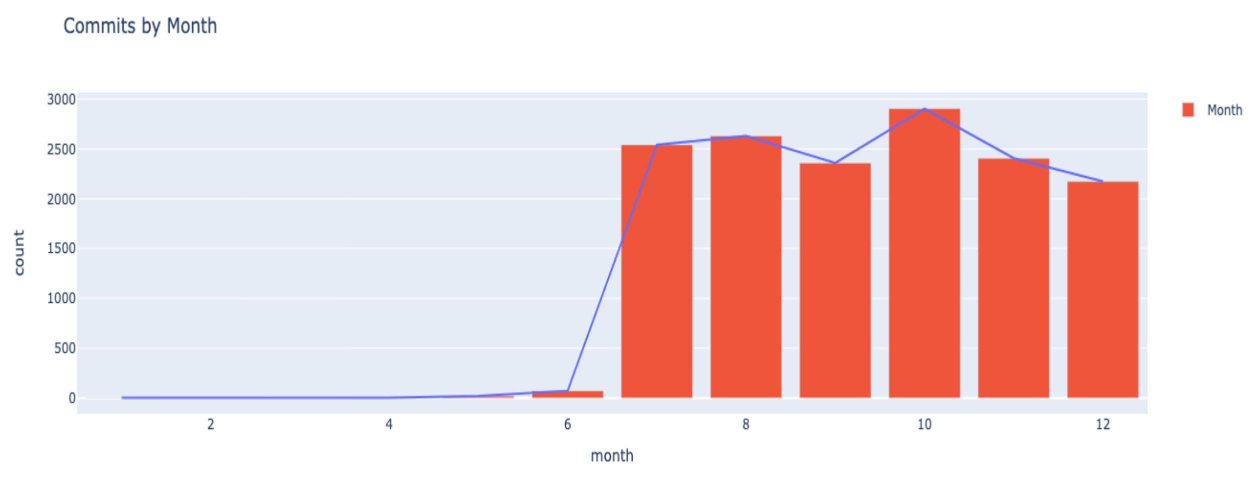


Figure 11: Time-Frequency of commits

4. <u>Understanding prioritisation of issues and developer engagement</u> by plotting the duration an issue has been opened for (Closing Time – Opening Time) against the number of comments each issue has received, and coloured based on the intensity of reactions accumulated Figure 12). The motivation behind this visualisation was to understand how a community of developers while having differences on an individual level, eventually drive efforts on a community level to prioritise and resolve issues (He et al 2020). This phenomenon can be observed in the fact that the issues – although a vast majority get closed in less than 200 days – having the most comments are actually being closed in much less time. Initially this may seem counterintuitive since issues opened for longer may presumably accumulate more comments over time, but the correlation is actually the opposite – the issues with most engagement are closed much sooner (even in less than an day) because the engagement might be occurring due to the urgency of the issue at hand.

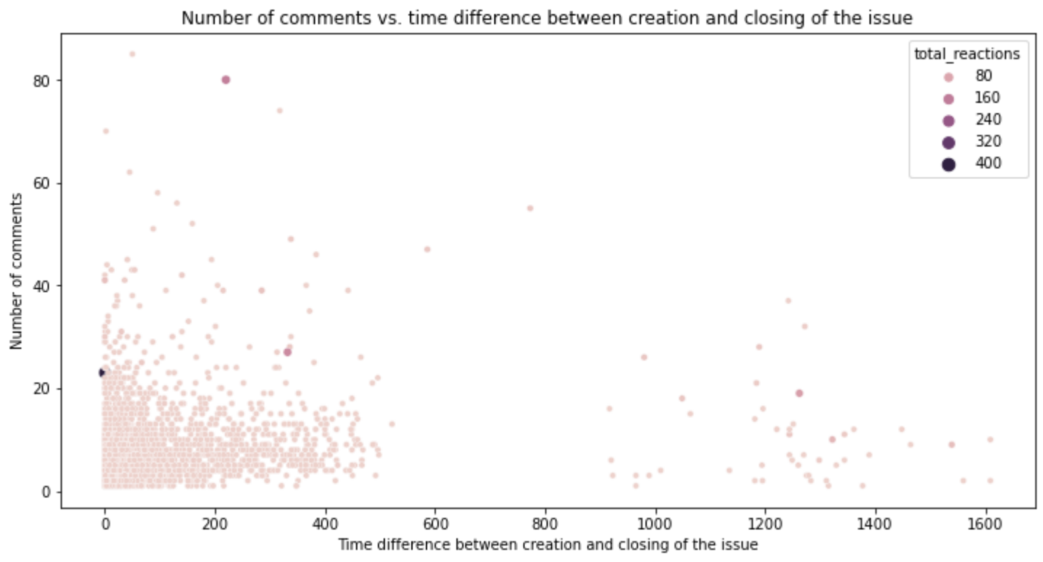


Figure 12: Issues -> Number of comments vs time to close

References

He, V., Puranam, P., Shrestha, Y. and Krogh, G., 2020. Resolving governance disputes in communities: A study of software license decisions. *Strategic Management Journal*, 41(10), pp.1837-1868.

Microsoft, 2022. *Code metrics - Cyclomatic complexity - Visual Studio (Windows)*. [online] Docs.microsoft.com. Available at: <https://docs.microsoft.com/en-us/visualstudio/code-quality/code-metrics-cyclomatic-complexity?view=vs-2022> [Accessed 22 July 2022].

Nguyen Duc, A., Cruzes, D., Hanssen, G., Snarby, T. and Abrahamsson, P., 2017. Coopetition of Software Firms in Open Source Software Ecosystems. *Lecture Notes in Business Information Processing*, pp.146-160.

O'Mahony, S. and Bechky, B., 2008. Boundary Organizations: Enabling Collaboration among Unexpected Allies. *Administrative Science Quarterly*, 53(3), pp.422-459.

von Krogh, Haefliger, Spaeth and Wallin, 2012. Carrots and Rainbows: Motivation and Social Practice in Open Source Software Development. *MIS Quarterly*, 36(2), p.649.