Tensorflow and Pytorch analysis

Data management systems (SMM695)

Group 24

Yong Shih Leong, Terry Tang, Te Lu

26 July 2022

Contents

1 Part 1: Creating databases, schemas and tables

1.1 Creating tables

1.1.1 Create table (GitData)

Taking a look at the gitData.csv file, we can divide the file into two tables which are commit data and files data. Before the files field, all of which are about commit-related data. A commit may involve multiple files, so when the same commit involves multiple files, the content of the fields before the files is repeated, hence it is more appropriate to build a separate table to prevent overlap. The fields before files would be able to use the hash field as the primary key, but the fields after it have no primary key hence we would give each file a unique id as the primary key.

1.1.2 Create table (GitIssues)

Likewise, the gitIssues.csv file can we split into 2 main parts. These are the section before and after the comment_id. The section before the comment_id is a description of the issue. The information before comment_id, has no unique identifiers, we can use "title" and the "create_at" time to create a unique identifier for issue id as the primary key. On the other hand, the section after comment_id is an annotation/comment on the issue, an issue can have multiple annotations/comments. The information after comment_id, references to comment_id as the primary key.

2 Part 2: Read, clean and load database

2.1 Data cleaning and feature engineering (GitData)

2.1.1 Adding a timezone to datetime

We add a timezone to the datetime to be able to be able to get more insights when we do the analysis in section 3 of the report.

2.1.2 Converting some columns into an integer

We also convert some columns such as *nloc*, *complexity* & *deleted_lines* into an integer to be able to easily do some data analysis on them as well.

2.2 Data cleaning and feature engineering (GitIssues)

2.2.1 Creating issue ID

As explained in section 1.1.2 the information before comment_id has no unique identifier therefore we create an issue ID to be able to gain more insights on the GitIssues csv.

2.2.2 Converting string columns into datetime

Converting columns such as *created_at*, *updated_at* & *closed_at* would allow us to create time series graphs to understand the flow of issues.

3 Part 3: Descriptive insights

3.1 Total commit count by project

First off even though it is stated that the data is from Pytorch and Tensorflow, when we look through the column *project_name* it contains other data such as *serve*, *vision* and *audio*. We would have to filter out the other projects and only focus on Pytorch and Tensorflow.

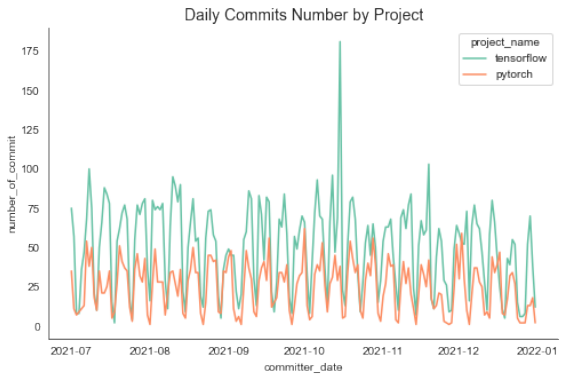

Figure 1: Overview of daily commit count

Based on figure 1 we can see that Tensorflow's commit count is much higher than Pytorch's throughout the duration from July 2021 to January 2022. This could suggest that the Tensorflows developers are much more active than the one at Pytorch. Or this could also suggest that Tensorflow has much more work to be done to be fully developed than Pytorch.

3.2 Total number of files and number of lines of code

Additionally, we could also take a look at the file number and lines of code which could tell us more about each of the packages Tensorflow and Pytorch.

Figure 2: Code lines and files number

Figure 2 shows that there are more number of files and lines of codes for tensorflow as compared to pytorch. This suggests that Tensorflow developers have a greater workload than those of Pytorch based on the number of lines of code and number of files.

3.3 Number of author comparison

Here we can begin to have insights into each of the OSS and how their workload is shared and divided.

	project_name	number_of_author	number_of_committer
0	pytorch	532	1
1	tensorflow	456	164

Table 1: Number of author comparison

Based on figure 2 we would assume that there would be a higher number of developers in Tensorflow as there is a higher workload. However it seems from table 1 that there are more Pytorch developers than Tensorflow. We can then conclude that a single Tensorflow developer would have a higher amount of workload than the developers at Pytorch.

3.4 Number of authors based on region (timezone)

By using the timezone difference, we can have a broad estimation of the geographical locations of the spread of developers despite not having geographical information.

From figure 3 shown above, we can conclude that the developers of the two open source projects both are mainly distributed in the GMT +7 and +8 (Southeast Asia). In addition, tensorflow

Figure 3: Timezone distribution

development is more widely distributed internationally as there are a few developers that are in different timezones.

3.5 Work hour distribution

3.5.1 Author work hour distribution

This next section, we use the author hours and number of commit to determine the times when developers have a high workload.

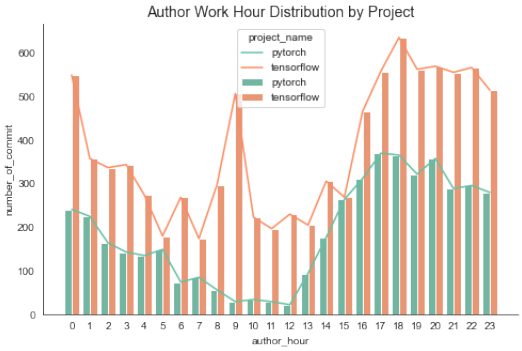

Figure 4: Author work hour

From figure 4 we can see a clear pattern for Pytorch, a decline in number of commits from hour 0 to hour 12 and then an increase from hour 13 to the peak in hour 17. Meanwhile Tensorflow's commit hours are more erratic, with a decline from hour 0 to 5 and rising back up to 500 commits in hour 9 and a sharp decline again only to reach its peak of 600 commits at hour 18. This can tell us that

the developers of Pytorch work on the development of the OSS as a part-time job as there are more commits during certain hours. Meanwhile for Tensorflow, the developers on average have a higher commit throughout all hours meaning that they develop the software as a full-time job.

3.5.2 Committer work hour distribution

We can also look at a different data *committer_hour* to see if it tells us a different story.

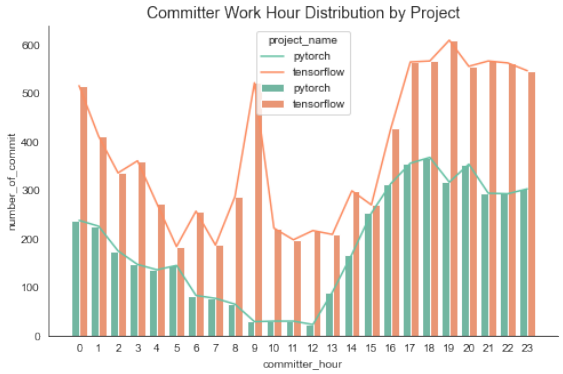

Figure 5: Committer work hour

Comparing figure 5 to the previous figure 4, we can see that committer work hour distribution is similar to that of author work hour.

3.6 Top authors

We further look into the top 10 developers with the highest contribution towards the projects, which is measured in the amount of code written (may not be optimal).

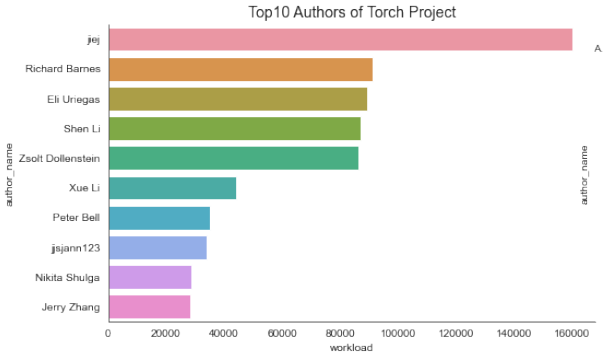

Figure 6: Pytorch authors

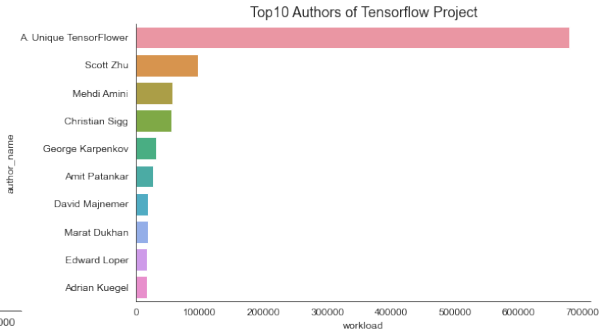

Figure 7: Tensorflow authors

Figures 6 & 7 compare the top 10 authors of Pytorch and Tensorflow respectively. The workload of the developers of Tensorflow's top 1 is extremely prominent, while the other members of top10 are not much different.

3.7 File change type

3.7.1 Overview of file changes

We can compare the what has happened to the files based on the type of changes (*ADD*, *DELETE*, *MODIFY*, *RENAME*).

Figure 8: Overview of files change

Based on figure 8 we can see that the main work of a developer is to modify code. To bring this analysis further, we can add a time dimension to compare and see what the difference over time is.

3.7.2 File added based on date

Files added would indicate the (creation stage/ early stage) of OSS development.

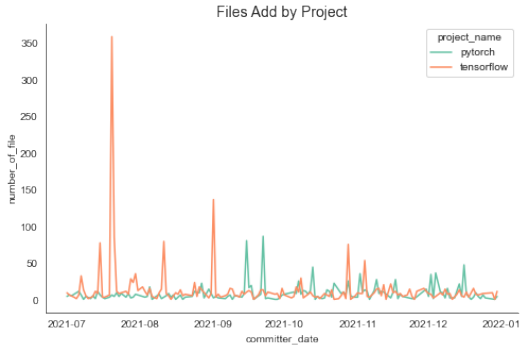

Figure 9: Files added

From a chronological perspective in figure 9, we can tell that the development period of tensorflow is July 2021. Meanwhile the development period of pytorch is September 2021, which is consistent with the actual situation.

3.7.3 Files modified by date

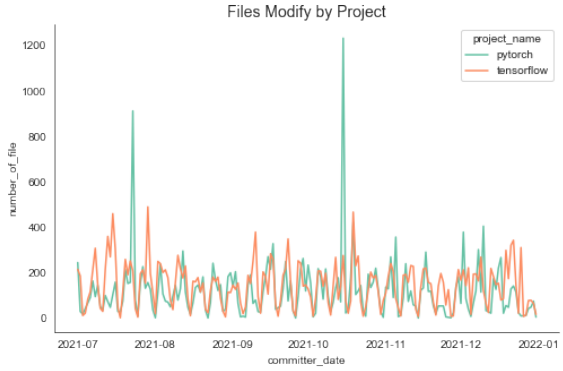

Figure 10: Files modified

From the perspective of file modification, pytorch has undergone two major revisions in July 21 and October 21 as seen in figure 10. It can be said that the development experience of Tensorflow was relatively smoother than that of Pytorch as the line graph of Tensorflow is less erratic than that of Pytorch's.

3.8 Issues

3.8.1 Number of issues found by date

In this section, we dig through the issues that were reported by the developers and get some insights from them.

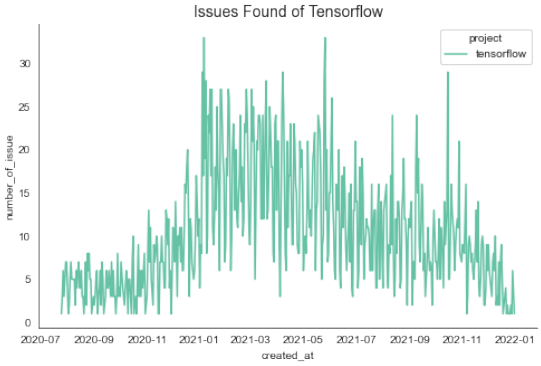

Figure 11: Number of issues in Tensorflow

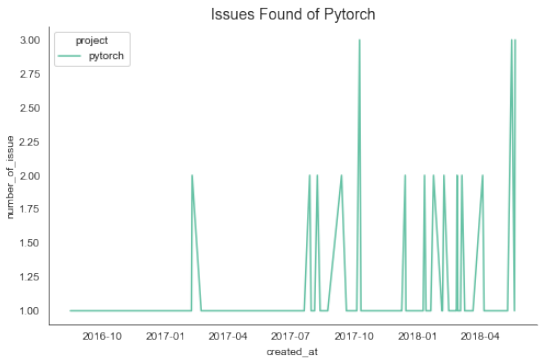

Figure 12: Number of issues in Pytorch

Based on figures 11 & 12, the data of the issues is mainly concerning tensorflow, and less about pytorch. We can see a trend where Tensorflow has a steady increase of issues throughout time and then a steady decline as well up till January 2021. It is not clear whether pytorch itself has no issue after 18 years (unlikely), or the issue reporting mechanism has been terminated, or was data not collected in this regard.

3.8.2 Issues fix time

Figure 13 & 14 depicts the time taken to resolve an issue within each of the OSS respectively.

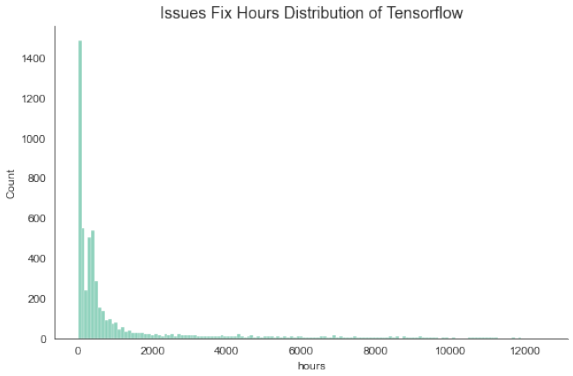

Figure 13: Time taken to fix issues Tensorflow

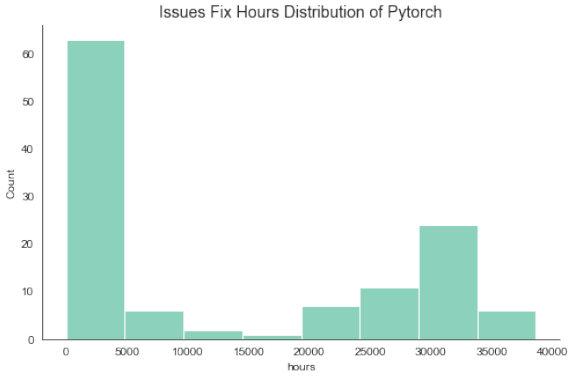

Figure 14: Time taken to fix issues Pytorch

Tensorflow hours of resolve on average is much shorter than Pytorch. However we have to note that Pytorch data itself is relatively small, and we can't rule out that there is a problem with this data (bias). Generally speaking, it is unlikely that an issue takes a few years to be resolved. We can see that there are some issues taking more than 12000 hours to be resolved in figure 13 which is a year and a half. This may be due to the fact that some issues that were solved were left open despite being solved.

3.8.3 Users that report the most issues

As we have information regarding the individual and the number of issues they report, we can put it onto a bar chart to have a picture of what that distribution looks like.

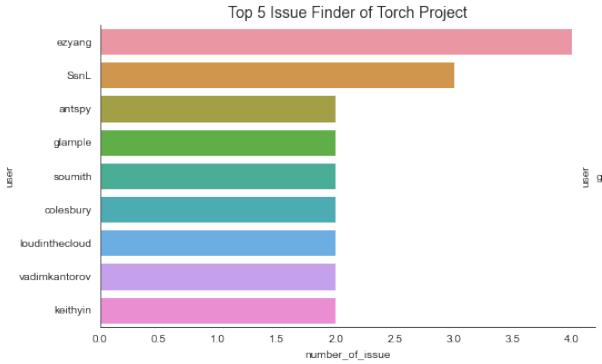

Figure 15: Top issue finder Pytorch

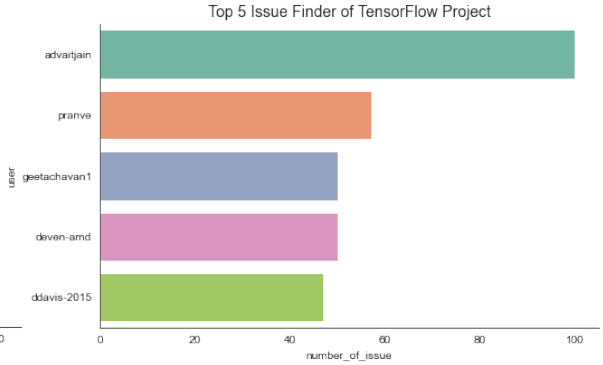

Figure 16: Top issue finder Tensorflow

Looking at figures 15 & 16 we can see a trend with both Tensorflow and Pytorch, there usually is one developer that finds the more issues than the other developers in both Tensorflow and Pytorch.

3.8.4 Users that close the most issues

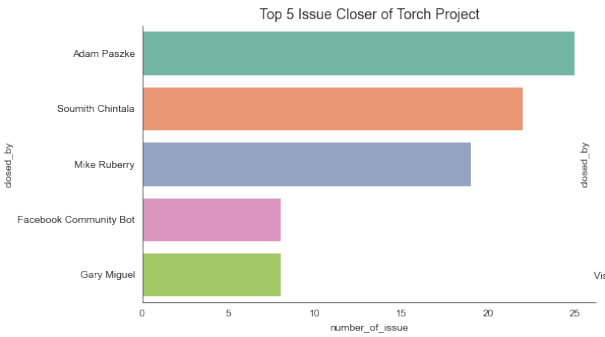

Figure 17: Top closer Pytorch

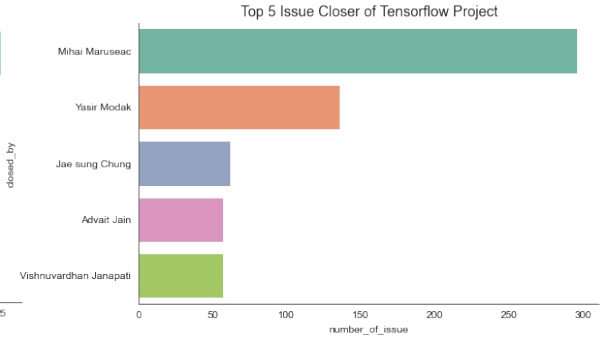

Figure 18: Top closer Tensorflow

In figures 17 & 18 for Tensorflow it is similar to part 3.8.3 where one developer closes / solves the most issues compared to the rest. Interestingly, for PyTorch there are 3 developers that close an almost equal amount of issues.

3.8.5 Top issue commenter

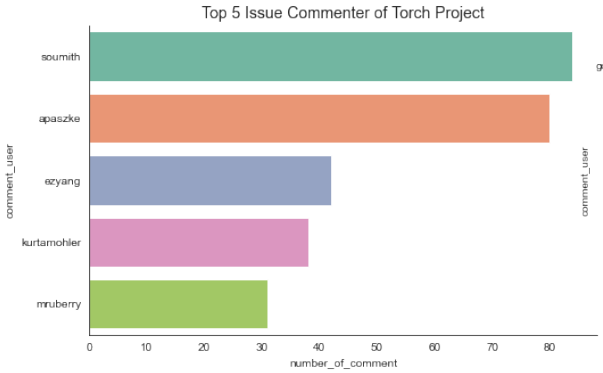

Figure 19: Top commenter Pytorch

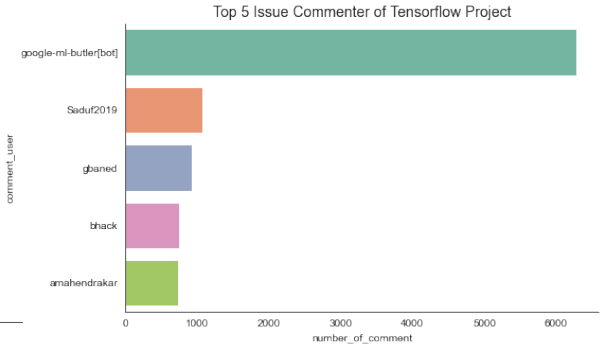

Figure 20: Top commentor Tensorflow

Moving on, we take a look at who are the top commenters in each of the OSS of Pytorch and Tensorflow. We can see from figures 19 & 20 that it tells a similar story to that of figures 17 & 18. Even though the top issue closer users are different to that of the issue commenter users, comparing the shape of the bar chart, it looks similar. The top 1-2 developers always contribute significantly more than the rest of the developers.

3.8.6 Top issue labels

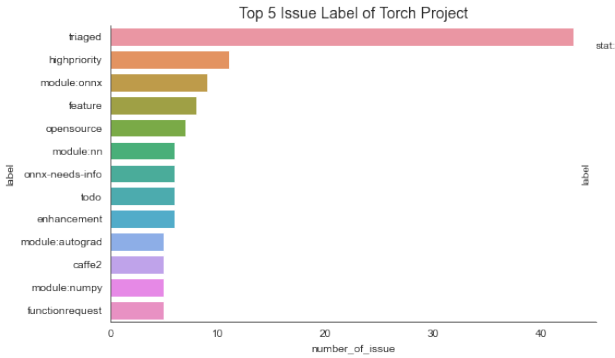

Figure 21: Top issue label for Pytorch

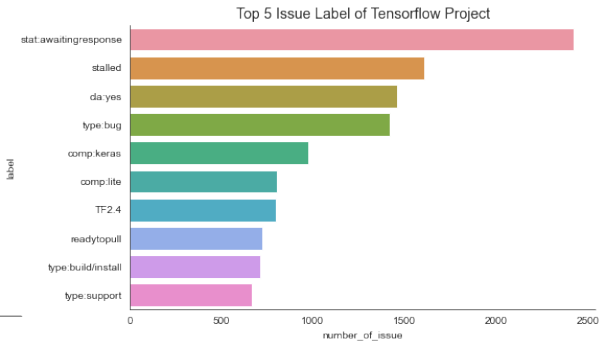

Figure 22: Top issue label for Tensorflow

Looking at figure 21 we can see that the label *triaged* is reported the most at about 45 reports. This number is small compared to the smallest issue *type:support* in Tensorflow which has more than 500 reports and the most reported issue is *stat:awaitingresponse* which has about 2500 reports.

4 Part 4: Using Pyspark for analysis

Pyspark was used to build a regression model for machine learning, and functionally allowed us to predict the "issues" count for the upcoming weeks for tensorflow. Weeks was chosen instead of days as the daily fluctuation in the data may be too large.

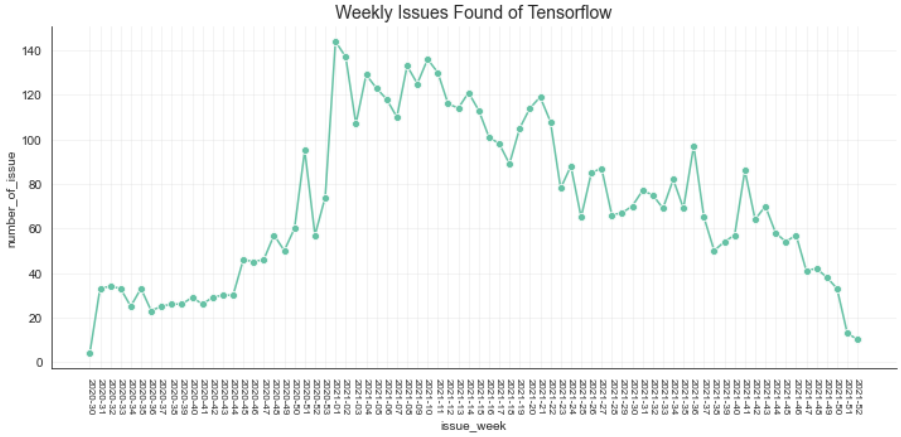

Figure 23: Issues found per week in Tensorflow

The figure above (23) shows the trend of number of issues over the weeks in the development process. We proceed to use the number of issues in the past 5 weeks (by generating them as features) to predict the number of issues in the next week. We then create a feature vector and fit it into the ols model. Lastly, we use the model to predict the number of issues in the coming week, which is 38.