Technical Debt NLP Analysis

Tim Team (Group 4)

Jordi Puig Rabat

Patricia Cabot Álvarez

Raúl Higueras Serrano

Índex

4.1. Business understanding

4.1.1. Business objectives

Nowadays, software developing projects are divided into a lot of smaller tasks. Scheduling these tasks is always tough work, mainly due to the inability of humans to estimate the time required to solve problems.

The main goal we aim to achieve is to predict the difficulty of solving an issue. With this task comes some subtasks such as defining an index of difficulty, estimating the required time and giving a measure of commonness of the issue.

The success criteria will be:

- Give insights about the relationship between task descriptions and time consumption.
- Be able to implement a model with enough prediction power to estimate the difficulty of a specific issue.
- Provide a commonness metric of the specific issue according to the community.

4.1.2. Assessment of the current situation

In this section, we offer a detailed description of the project evaluation with the purpose of providing a wider view of the situation.

First, there is a list of the inventory of resources of the project.

- Personnel:
    - Tim Team is a group of three data scientists students from the Polytechnics University of Catalonia (UPC). The team is formed by Raúl Higueras, Jordi Puig and Patricia Cabot.
- Data: currently, we are working with the version 1 of the TechDebt dataset, due to the fact that the second version (the one we have been asked to work with) does not contain some of the attributes we are willing to work with. However, we ask the dataset developers to provide us an extended version 2 of the dataset with the desired attributes.
- Programming tools:
    - We will mostly program with Python, which offers a wide range of useful libraries for the task, such as implemented data mining tools.
    - In order to have the project code available to all the team in an easy way, we have created a GitHub repository.
    - Besides, the project report will be realised in Google Docs.

Following there is a list of the project requirements:

- The project is scheduled to be completed on the 29th of October.
- A report of the work done and the code repository shall be delivered at the end of the project. The results should be reproducible in a third-party computer.

- The data is open source, so we are allowed to use them under the Creative Commons Attribution-NonCommercial- ShareAlike 4.0 International license.

Following there is a list of the project assumptions:

- There exists a correlation between the task description and time spent to solve it. That is to say, the time spent to solve a task is independent of the group facing the problem and depends only on the task at hand.
- The starting and ending issue dates in the database correspond to the real times when the issue was found and when it was resolved.

Following there is a list of the project constraints:

- From the nature of the project, we cannot work with the version 2 dataset. However, we may obtain the attributes needed, so it may not be a problem in the future.

Following there is a table with the possible risks and their consequential contingencies.

Risks	Contingencies
Due to the nature of the data, the human-written attributes may show a lot of heterogeneity, which may make the problem difficult.	The main intention is to try different pre-trained text embedding techniques in order to find the most suitable for the text we are working with. Moreover, these pre-trained models can be fine-tuned and information from other domains than text can be added to improve the predictions.
Not having enough resources or potential to complete the expected task.	If the time prediction based on the issue description is powerful enough, then the project will be more focused on measurement of an issue's commonness.

Table 1. Table of risks and contingencies.

As the project is developed in the programming framework, we make use of precise terminologies of not common use. Here is a glossary with the most relevant words to make it more understandable:

- **Jira**: software developed by Altassian that allows the tracking of issues and an agile project management.
- **Issue**: an error or problem found in the software that needs to be fixed at some point. Each issue comes with a corresponding:
    - **Issue description:** text written by the developer explaining the problem they encountered.
    - **Issue summary:** brief abstract of the issue.
- **Spent time**: difference of time between the creation date and the resolution date of an Issue.

- **Commonness**: measure that explains how often the issue or a similar one appears in the dataset based on its context description.

These are the costs and benefits of the project.

- Costs: The cost of implementing this project can be calculated as the time spent on developing and documenting the results.
- Benefits: The potential benefits of achieving the desired results is a free open source tool to estimate the necessary time needed to solve code issues.

4.1.3. Data mining goals and success criteria

Once the assessment of the situation is determined, we can now present the project objectives in technical terms. That is to say, to specify the concrete goals we are willing to achieve in order to accomplish the business objectives.

As explained above in the business section, our aim is to determine the difficulty of an issue and its commonness. To do so, we defined two data mining goals, each of which addresses one of the two objectives:

1. Design a model to predict the difficulty/time consumption of an issue based on the task descriptions.
2. Discover the presence of topics among the issues.

According to this, the corresponding success criteria will be:

1. A prediction accuracy of ~70%.
2. There exists no clear metric to measure the success of unsupervised techniques such as clustering. That is why the success criteria for this task will be to manually find empirical examples of the system (such as manually testing common examples) and show that it works as expected.

4.1.4. Project plan

The set of tools we have predicted to use are:

- General-purpose data analysis tool: RStudio with the sqlite library.
- Data pre-processing and modeling tool: Jupyter Notebook with the pandas and pytorch for Python3.
- Scripting language for the final deployment: Python3.

The techniques will be slightly different for the two analyses we propose. For the time predictor, we will use:

- Natural Language Processing techniques for data preprocessing such as stop-word removal, tokenization, word and subword-embeddings.
- Linear regression and other classic ML models with word frequency models like tf-idf.

- Neural Network (NN) based models such as transformer arquitectures or Recurrent NN layers. The idea is passing as input argument the text information of an issue and predicting the consumption time to solve it, as a classification or a regression task.

For the clustering classification, we will use:

- Word embedding methods and NLP processing.
- Hierarchical clustering
- Dimensionality reduction techniques (like tSNE) for visualization purposes

In order to accomplish the project plan, we have divided it in the following steps with its own tasks to be solved.

Step 1	Problem and Data understanding
Definition	The objective first task of the project is to achieve a high level of comprehension of both the task ahead and the data, in order to make sure that the data provided is good enough to solve the task. These include learning about the magnitude of the problem, the domain of the texts, the format of the data and its quality.
Estimated Duration	1 week
Risks	1. The texts are not large or rich enough to extract a contextual meaning of the issues. 2. The attributes present in the issues table have not enough information to predict the estimated time.
Contingencies	1. If the texts are not suitable for the task, we will change the focus of the analysis to using the other attributes in the dataset. This is, analyzing the presence of lengthy tasks depending, for example, on the stage of the project, or the number or people working on it. 2. If attributes needed to predict the time are not adequate for the task, we will focus on the analysis of common issues, instead of time prediction.
Resources required	- A program able to parse the data and extract quick insights about the attributes such as number of missing values, statistical distributions and mean text length. For this purpose, we will use two different programmable environments: **RStudio** and **Jupyter Notebook**.
Inputs	- The dataset (in .db format or multiple .csv files).
Outputs	- A document with the project goal defined in a precise way. - A small set of visualizations and statistical computations that show the quality of the dataset.
Dependencies	None

Table 2. First step of the project plan.

Step 2	Data pre-processing
Definition	This step consists mainly in processing the raw text in the dataset in order to obtain a processable input to use later on. This is a common task in almost all data science projects, but it has a special relevance in NLP problems. Traditionally, to obtain the final formatted input, there are a set of transformations that are applied sequentially to the text. These tasks are: 1. Format and punctuation system unification. 2. Removal of non-informative words (stopwords) and symbols. 3. Synonym matching, syntax correction and similar techniques. 4. Numerical embedding, which can be based on (among others): a. Word frequencies: like the tf-idf b. Context: like neural word or subword embeddings. This task contains also the preprocessing of the target values: applying transformations to the multiple attributes to obtain the variables to predict in the desired format. However, this task is notably easier than the text processing step.
Estimated Duration	2 weeks
Risks	1. The text syntax is heterogeneous among the instances, hence a general pre-processing of the data is insufficient. 2. The text includes non-human-written text, such as automatic error.
Contingencies	1. If the text syntax is heterogeneous, it will require a more detailed pre-processing, taking special care to all the heterogeneities in the texts. This may slow down other tasks. 2. In that case, we will have to consider two different options: keep the text or design a way to automatically detect and remove that text. The best method will depend on the amount of automatic text found and their quality.
Resources required	- We will use **Jupyter Notebook**.
Inputs	- The raw text obtained directly from the dataset.
Outputs	- A clean dataset, containing only the input variables in a processable format (numerical vectors, not text) and the target variables. Only the attributes needed for the modeling step.
Dependencies	These tasks cannot be started until the project goals are defined in the first step. We also need a deep knowledge of the domain of the texts and their special traits, in order to guarantee a rich and correct processing.

Table 3. Second step of the project plan.

Step 3	Modeling
Definition	This step has two clear subtasks that can be parallelizable: ● **Time prediction:** generate a machine learning model able to predict with certain accuracy the estimated time to fix an issue given the processed text. ● **Commonness check:** design a system to derive how common is a given issue by its description, using the data present in the dataset. For the **Time prediction** task, only a subset of the dataset will be used, saving the remaining entries for Step 4 Validation. The **Commonness check** task is an unsupervised task (there is no target variable to predict), thus it will be performed using the complete dataset.
Estimated Duration	2 weeks
Risks	1. The processing of the text is insufficient. That is to say, the data presents formats incompatible with the models. 2. The model is not able to predict with enough accuracy.
Contingencies	1. Return to Step 2 Data pre-processing and apply a different processing strategy. 2. Try to detect the reason for the failure. If it is due to a bad pre-processing, do as in contingency plan 1. If it is due to the difficulty of the task, try more complex models or generate an analysis proving the difficulty.
Resources required	- We will use **Jupyter Notebook**.
Inputs	- The clean dataset with the needed attributes for modeling.
Outputs	- Trained models for predicting the time spent to solve an issue. - A system that classifies an issue based on its commonness among other issues.
Dependencies	The clean dataset resulting from Step 2 - Data pre-processing.

Table 4. Third step of the project plan.

Step 4	Evaluation
Definition	The objective of this task is to check whether the model and system resulting from Step 3 Modeling meet the project goals. To check that, we will test our model with unseen data records from the dataset and evaluate its performance.

Estimated Duration	1 week
Risks	None.
Contingencies	None.
Resources required	- We will use **Jupyter Notebook**.
Inputs	- The model and system designed in Step 3 Modelling. - The well-defined business goals and success criteria from Step 1.
Outputs	- A document reporting the results and explaining whether the success criteria are met or not.
Dependencies	The business goals and success criteria from Step 1, the clean dataset from Step 2 and the trained models and the system from Step 3

Table 5. Fourth step of the project plan.

Step 5	Deployment
Definition	Build an interface to allow the interaction with the system build, with the purpose of testing the performance with new data. As the project's scope is already big and the time is limited, this interface will be simple and small. It will consist of an executable file asking for a task description and it will return the predicted values.
Estimated Duration	1 week
Risks	None.
Contingencies	None.
Resources required	- A program to build a user interface with all the functionalities developed during the project.
Inputs	- A system that pre-processes the data from Step 1. - The model and system designed in Step 3 Modelling.
Outputs	- An executable file with the models loaded.
Dependencies	As this is the final step when all the work is gathered in a sole system, all the previous steps are necessary for carrying out the task.

Table 6. Fifth step of the project plan.

4.2 Data Understanding

4.2.1 Data Collection

In this particular project, the data collection has been an easy task for us given that all the tables can be found in a SQLite file in the github repository of the dataset we are working with. After downloading the .zip file and extracting to .db format, we can start querying the database from python using special libraries that make the process very easy.

4.2.2 Data Description

From the whole Technical Debt Dataset, our analysis will only focus on the following tables:

- `JIRA_ISSUES`: containing the description about each one of the issues reported on the jira system, its type, the corresponding project and their creation and closing dates.
- `GIT_COMMITS`: containing each one of the projects' commits with its descriptions and dates.
- `GIT_COMMITS_CHANGES:` containing statistics about each of the files changed by each commit in `GIT_COMMITS`.

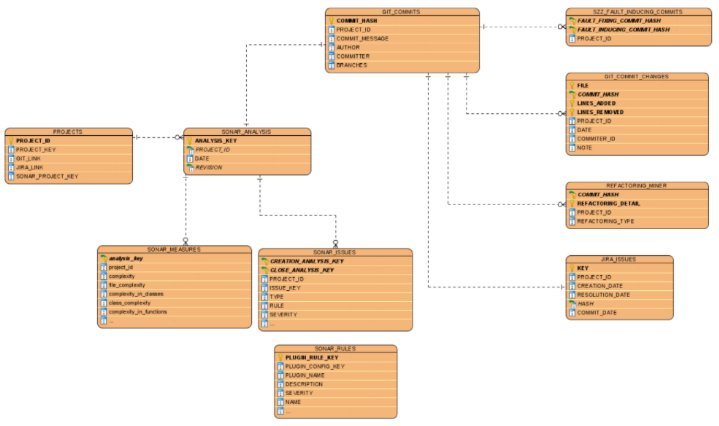


Image 1. Technical Debt Dataset scheme.

The most important information for our analysis will come from the Jira's system database. From there, we will extract the human-written text to be analysed along with the time duration of each issue. See *Table 7* for a summary of the attributes we will use.

Attribute	Format
projectID	Categorical
key **(PK)**	Identifier
creationDate	Datetime
resolutionDate	Datetime

Number of rows	30147
Number of columns	10

Table 8. Size of the resulting table after the selection

type	Categorical
summary	Text
description	Text

Table 7. Selected attributes from the `JIRA_ISSUES` table

The other two tables will be used to extract complementary information. Specifically, we are interested in the data about the number of files and lines of code changed for each issue. See *Table 9* for a summary of the attributes we will need.

Attribute	Format
commitHash **(PK)**	Identifier
file **(PK)**	Identifier
linesAdded **(PK)**	Integer
linesRemoved **(PK)**	Integer
note	Text

Table 9. Selected attributes from the remainder tables

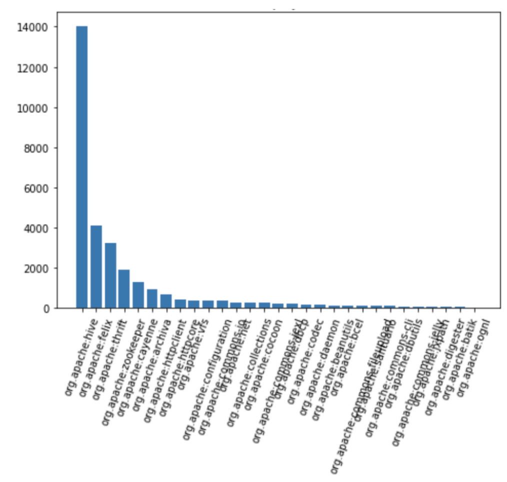
Number of rows	857740
Number of columns	8

Table 10. Size of the resulting table after the selection

4.2.3 Data Exploration

Distribution of Categorical Attributes

Firstly, we checked the distribution of the two categorical variables in the **jira_issues** table, corresponding to the belonging project and the issue tag.¡



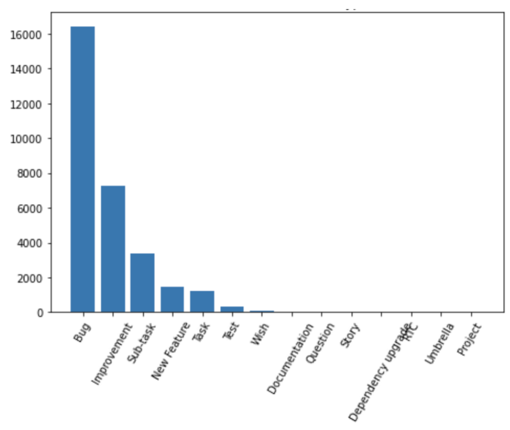


Image 2. Distribution of the projectID attribute

Image 3. Distribution of the type attribute

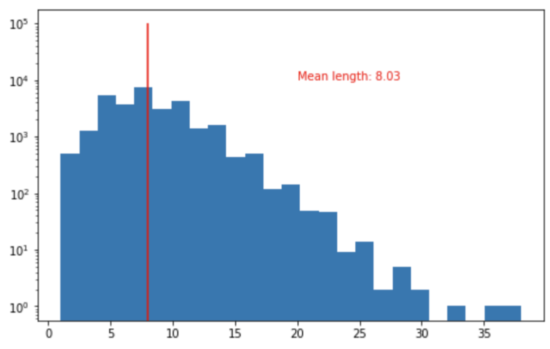
It can be observed that both variables show a noticeable unbalanced distribution, especially the first one, corresponding to the projects.

On the other hand, in the graphic showing the project identification there is one large project with almost 50% of the total issues (Hive) and two other projects with ~4000 issues (Felix and Thrift). The other ones are under 2000 issues. This imbalance can cause problems in the analysis if it is not taken into account.

On the other hand, the distribution of the issue type is less prominent, still it can be seen that the most frequent issues are bugs, improvements, sub-tasks, new features and tasks. The rest are barely present in the data.

Distribution of Text Data

As our project tries to extract features from human-written text, it is important to ensure that the strings of texts are long enough to contain information. That is why these two following plots were created.



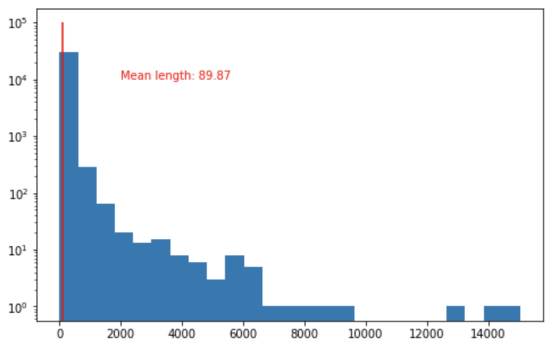


Image 4. Logarithmic distribution of the summary attribute lengths

Image 5. Logarithmic distribution of the description attribute lengths

As expected, the graphs show that people tend to summarize and write shorter texts rather than long ones when describing issues. The mean values of 8 words by summary and 90 words by description seem *a priori* long enough to extract meaningful representations.

Distribution of Target Variables

In order to determine whether the prediction task would be possible, we firstly checked the distribution of the task durations. The duration was computed as the difference between the resultionDate and the creationDate attributes and transformed to hours.

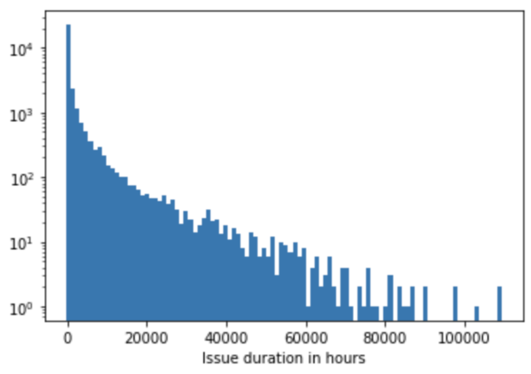


Image 6. Distribution of the issue duration

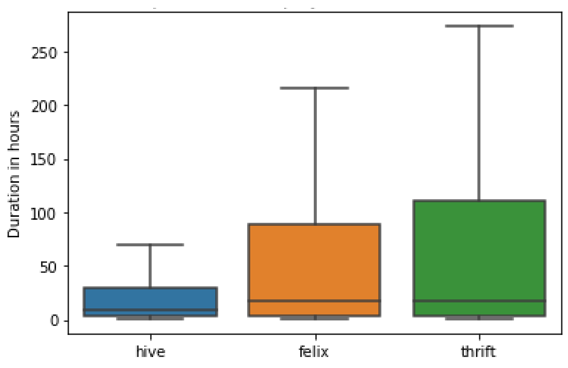


Image 7. Boxplot for the three projects with more issues

From the previous plots, we extracted two important ideas:

- Most of the issues are solved in little time (the first hours or days), even so there are cases of issues open for years and even more than a decade.
- The distribution of the durations can be very different depending on the project, so predicting directly the time may not be possible.

Regarding the other variables that could represent the effort of a commit, these are the distributions of the attributes LINES_ADDED and LINES_REMOVED.

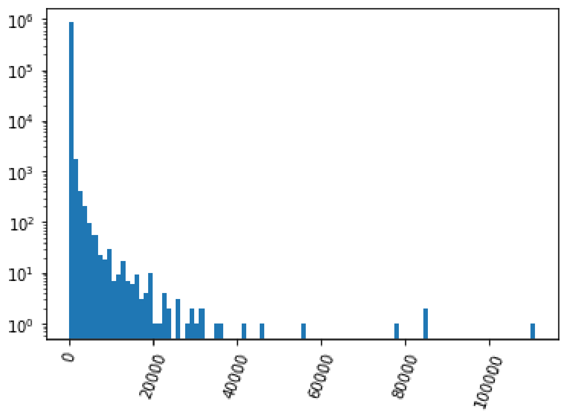


Image 8. Distribution of the added lines per file changed

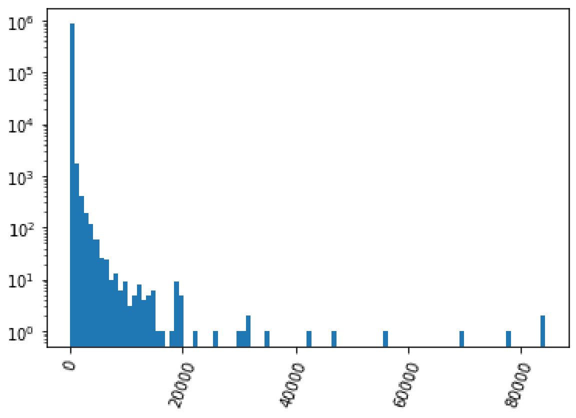
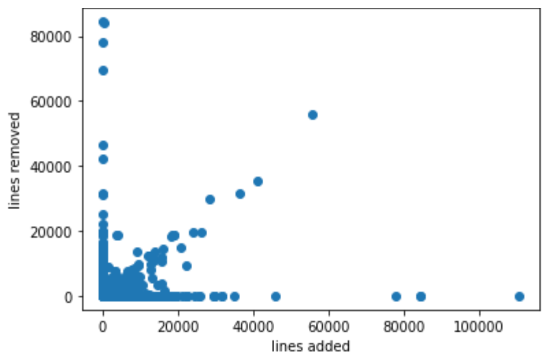


Image 9. Histogram of the removed lines per file changed

Both plots are quite similar, and show that the most frequent line additions or removals are of a few lines only, while big changes are far less common.

We also checked the relation between the two variables using two scatterplots.



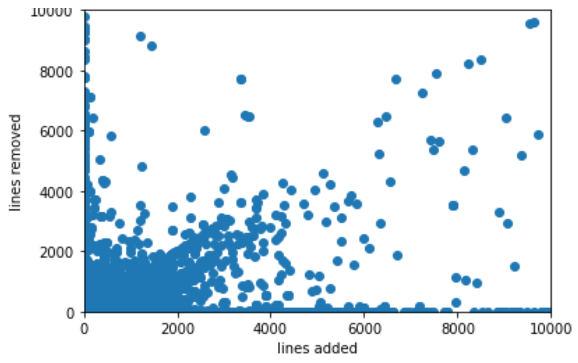


Image 10. Scatterplot of the added lines and the removed lines

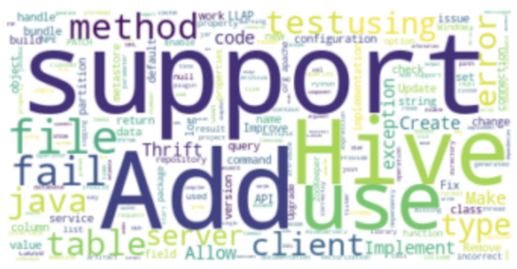
Image 11. Zoom in of the scatterplot of the added lines and the removed lines

From the scatterplots, we can observe that the relationship between the two variables follows three different patterns: files with additions and no deletions, files with deletions and no additions and files with both of them.

Getting an idea of the task difficulty

During our problem understanding phase, we feared that the task would be too difficult. To obtain a first idea of whether the problem would be possible to solve, we designed the following composed visualization. It consists of two word clouds, an image containing words in different positions, their size representing the amount of times they appear in the corpus.

The text used to generate the left image was the summaries of the 25% quickest issues of each project, while the one in the right was made with the 25% most time-consuming issues of each project.



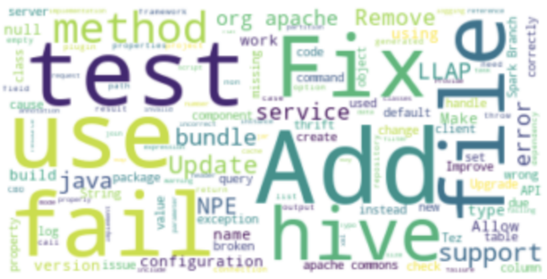


Image 12. Word cloud of the summaries of the 25% quickest issues per project

Image 13. Word cloud of the summaries of the 25% most time-consuming issues per project

The main idea that we can take from the visualization is that the distribution of words of the two corpus is indeed different. There are some words that appear similarly on both clouds, while others appear with more emphasis in only one (like *support* in the quick issues, and *fix* in the time-consuming ones). All in all, this means that the text may contain enough information to be able to classify the issues by time consumption, thus the objectives may be achievable.

4.2.4 Data Quality Verification

The step of verifying the data quality is crucial to a project, due to the fact that if it is insufficient, the results may be biased or even incorrect.

On the one hand, we checked the quality of the `JIRA_ISSUES` table. To do so, we first checked the number of missing values of the selected attributes. Due to how the data is collected, the only attribute with missing values is the description attribute. However, they only represent 8.51% of the total entries, so will not cause a problem.

Then, as we will work with written text (description and summary attributes), we considered the potential difficulties related to quality we could encounter in the preprocessing step. Here is a list of them:

- As the text is written by humans, it may contain orthographic errors, such as typing errors.
- As the text addresses programming issues, it may contain non-human written text, such as the terminal errors obtained by the developers or the code itself that produces the error.
- As we are working in the software engineering domain, we may encounter out-of-dictionary words, as they may be too specific on the topic.

One the other hand, we checked the quality of the remaining tables. Although there were no missing values because the important attributes we needed were primary key, we identified some errors in the database that can generate issues in the future:

1. There are commit changes that present files with no addition or deletion of lines of code, due that git is not able to track that information from certain file extensions.
2. There were missing COMMIT_HASH values. That is, some values present in the other tables (like `JIRA_ISSUES`) had not the referenced commit on the `GIT_COMMITS` table, or the hash identifier was wrong.

With this, we roughly identified that only 50% of the issues in the jira table could be matched with the corresponding commits, and therefore our data for the complete analysis becomes heavily limited.

4.3 Data Preparation

4.3.1. Data selection

The data we will work with comes from three different tables of The Technical Debt Dataset. The most important one is `JIRA_ISSUES` where we can obtain the independent variables (description and summary) that will be used for prediction and also the time the issue was created and solved. The two auxiliary tables `GIT_COMMITS` and `GIT_COMMITS_CHANGES` are used to obtain information useful to determine whether an issue was hard to solve. This is the number of lines added or removed and the number of files changed. We use other attributes from these tables that are only used to join them in the following way:

- A commit corresponds to an issue if the commit_message contains the issue_key inside the text.
- A commit change corresponds to a commit if they both have the same commit_hash.

Table	`JIRA_ISSUES`	`GIT_COMMITS`	`GIT_COMMITS_CHANGES`

Variables	● key ● project_id ● creation_date ● resolution_date ● summary ● description ● type	● commit_hash ● commit_message	● commit_hash ● file ● lines_added ● lines_removed

Table 11. Tables from the Dataset and its attributes used in the project

4.3.2. Data cleaning

For our analysis, the main data cleaning problem is dealing with the missing values. We have two ways of dealing with them.

For the direct analysis of text, the problem comes from the missing values in the Description attribute of the `JIRA_ISSUES` table. Our way of dealing with the text is combining the strings of both attributes Summary and Description. And, as the Summary attribute cannot be blank, samples without Description contain only the Summary string, but they are not blank.

For the difficulty prediction, we need the data of the `GIT_COMMIT_CHANGES`, which generates a lot of missing values when joining with the issues. As the amount of missing values is too large to impute values, we decided to just remove all of the issues with no associated commits. This leaves us with a much smaller dataset, but with information about the lines of code modified for each issue.

4.3.3. Data construction

Creation of new attributes

To generate our final dataset, we have to create two derived attributes from the data:

- **Issue duration**: derived attribute from the difference between start and completion time.
- **Lines added** and **lines removed:** aggregated attributes adding the lines added and removed for each issue.
- **Files:** aggregated attribute counting the number of files modified for each issue.

Natural Language Processing

The most important part of this section has been the processing of the text data, stored in the description and summary attributes. Following, we present the steps we have taken. This steps have been performed using regular expressions and the help of the Natural Language Toolkit python package (nltk).

Firstly, we have unified the text style. That is to say, replacing the uppercase letters with lowercase and removing special terms. The latter consisted in the following actions:

- Remove the special characters *\r* and *\n*, due to the fact that they are used to format text and we are not interested in them.
- Remove the text present inside the symbols *{}* and *[]*.

- Replace special expressions: versions of libraries or packages, links, repository paths, filenames and dates with their expression type in uppercase (VERSION, URL, PATH, FILE, DATE).
- Remove punctuation symbols.
- Replace multiple spaces for a single space.

Secondly, we have removed the stop words present in the texts, due to the fact that they are very frequent but contain no important information.

Thirdly, we have removed project-specific terms (such as project names) from the texts, since our project goal is to extrapolate issue difficulty, independent from the project they belong to.

Here are some examples of the same text (from the description attribute) in raw form and after applying each of the steps:

Raw text	`There are a bunch of files who have either a plain wrong number or at least something that is not really consistent with https://thrift.apache.org/docs/committers/HowToVersion`
	
	`Additionally, the DOAP.rdf is slightly outdated.`

Step 1	`there are a bunch of files who have either a plain wrong number or at least something that is not really consistent with URL additionally the FILE is slightly outdated`
Step 2	`bunch files either plain wrong number least something really consistent URL additionally FILE slightly outdated`
Final text	`bunch files either plain wrong number least something really consistent URL additionally FILE slightly outdated`

Table 12. Example of a description after the different processing steps.

4.3.4. Data integration

The join between the GIT tables is relatively easy as we have one key (commit_hash) that relates each row in `GIT_COMMITS_CHANGES` to another row in `GIT_COMMITS`.

The hardest part (in the engineering and computational way) is to join the commits with the issues. We used the assumption that every commit that was used to solve an issue (partially or completely) had the issue key in its message. Thus, we joined the issues with the commits if the key was present in the commit message. As we work with around 30.000 issues, this operation is computationally expensive if no transformation is used. To speed up the join process we created a Full Text Search (FTS3) table that indexes by the text in the commit message and allows us to easily query those commits that contain some specific text.

4.3.5. Dataset description

The resulting dataset in a single table with the following attributes:

Attribute	Description	Required
issue_key	Issue identifier	Yes
project_id	Project identifier	Yes
duration	Time (in hours) the issue was open	Yes
summary	Issue summary text	Yes
description	Issue description text	No
type	Issue type (bug, task, etc)	Yes
files	Number of files involved	Yes
lines_added	Number of added lines	Yes (can be 0)
lines_removed	Number of removed lines	Yes (can be 0)

Table 13. Attributes of the final dataset.

The dataset will be saved in a simple comma-separated file (*data.csv*) containing one entry per row. Additionally, we will save a secondary file (*texts.csv*) with only the attributes **project_id**, **summary**, **description** and **type**, but without removing the samples with missings during the join. This way, we can use the whole dataset of texts for the tasks that do not need the `GIT_COMMITS_CHANGES` attributes.

4.4 Good Practices

The three good practices we are using throughout the project are:

1. **Use a directory structure:** we are using the cookie-cutter data science template (https://drivendata.github.io/cookiecutter-data-science/)
2. **Use linters to check code quality**: we are using lint8, which is a linter for python integrated in the cookie-cutter data science template to ensure the code quality of our source code and notebooks.
3. **Use interpretable models when possible:** in the modelling step, we will try to use interpretable models for both the clustering phase and the prediction problem, and provide an interpretation of the model alongside the testing results. This way, the results are not only a numeric value indicating success but a set of insights explaining how the model works.