computers

MDPI

Article

A Study on Evolution of Pull Request Template: How Are Pull Request Initial Contents Organised and Evolved?

Jungil Kim

Software Technology Research Center, Kyungpook National University, 80, Buk-gu, Daegu 41566, Republic of Korea; jikim424@gmail.com

Abstract

Pull request templates are used to reduce inconsistencies in information included in submitted pull requests in GitHub. A few studies have explored the effectiveness of employing pull request template. However, there is still a lack of how to evolve PR templates during software development. Knowledge is crucial to efficiently manage PR templates. To address this gap, we conducted a study on the organisation and evolution of pull request template initial content. For the study, 2689 target PR template files from 2614 public GitHub repositories were collected and 7 PR content categories including Description, Checklist, Reference, Test, Type, Additional Info, and Other were manually defined from the target files. Based on the defined categories, a pull request content classifier was built. By using the target dataset and the classifier, initial content organisation and its evolution were investigated. The results showed that 68% of target pull request templates organise their initial content with only two or fewer categories, and the initial content organisation remains in 71% of the entire pull request templates.

Keywords: computer science; software engineering; Mining software repositories; software development; pull request template

Check for updates

Academic Editor: George Angelos Papadopoulos

Received: 19 December 2025
Revised: 22 January 2026
Accepted: 22 January 2026
Published: 1 February 2026

Copyright: © 2026 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

GitHub (https://github.com/ (accessed on 20 January 2026)), currently one of the most actively used software development platforms, offers a pull-based development model which is a collaborative software development method [1]. The pull-based development model often suffers from inconsistencies in information included in PRs [2]. Submitted PRs provide a variety of information related to code changes, including summaries of changed code, related issue reports, and test methods. The quality of provided information may vary depending on PR authors. Expert contributors may provide condensed information in their preference form [3], while novice contributors may provide incorrect or omit necessary information [4]. These inconsistencies complicate PR review work and reduce efficiency of the pull-based development model [5].

To address the inconsistency problem, GitHub has encouraged the use of pull request template (PR template) since 2016 [6]. A PR template is a document file that predefines a PR format. Repository managers organise PR templates with contents required for PR review works. PR submitters can consistently create a PR from provided PR templates. Several previous studies [2,7] investigated the usage of PR templates. They showed that using PR template positively impacts repositories' PR review process. Although these studies provide an insight into the effectiveness of PR template usage, there is still a lack of how to evolve PR templates during software development. This knowledge is crucial to efficiently

manage PR template. For instance, it can contribute to building a proactive guideline to when PR template should be improved.

This paper presents a study on PR template initial content organisation performed to address the knowledge gap. In the study, we focus on investigating PR template content organisation and evolution at the categorical level. We collected 2689 target PR template files from 2614 public GitHub repositories for this study and defined 7 PR content categories including Description, Checklist, Reference, Test, Type, Additional Info, and Other by manually examining the contents of the target dataset. We then built a classifier that automatically categorises PR template content titles into these categories. By using the target dataset and the classifier, we investigated the following two research questions:

RQ1: ***How are the initial contents of PR templates organised?***

We conducted a clustering analysis to investigate RQ1. The initial content texts of the target PR template files were extracted from their first commit. Using the built classifier, we identified the content categories included in the extracted initial content texts. We then clustered the target PR template files with similar initial content categories. The results showed 13 clusters representing the types of initial content category organisation. We found that eight of the clusters have only two or fewer representative content categories.

RQ2: ***How have the initial contents of PR templates evolved over time?***

We performed a quantitative analysis of change in the initial contents. To perform the quantitative analysis, we implemented a method to identify PR template content title changes in commits. Using the implemented method, we extracted content category changes from the commits of the target PR template files. We then investigated evolutionary trends of the initial contents by analysing the extracted changes. The result showed that the constant trend is most common in the evolution of the initial contents. Furthermore, we compared the number of contributors between the repositories with the constant and non-constant PR templates and confirmed that the repositories with the non-constant PR templates has more contributors. We also observed that there is a negative correlation between the number of initial content categories and variation in the content category change in the non-constant PR templates. The findings imply that PR template evolution is likely related to collaboration complexity through PRs, and there is likely an optimal point in the number of content categories of PR template.

The following are the main contributions of this study:

- This work is the first study to analyse the initial contents of PR templates and its evolution. It extends previous studies [2,7].
- This study provides a method for identifying content title changes in commits of PR templates. The method can be applied to another analysis of software engineering artefacts' content changes. For instance, it can be used to analyse content changes in GitHub issue templates.
- The results of this study imply that PR template evolution is likely related to collaboration complexity through PRs, and there is likely an optimal point in the number of content categories of PR template.

The remainder of this paper is organised as follows. Section 2 introduces related works. Section 3 describes the data collection works performed in this study. Sections 4 and 5 present the motivation, approach, and results for RQ1 and RQ2, respectively. Section 6 discusses the study results, and Section 7 concludes this study.

2. Related Works

Several studies showed that using PR template has a positive effect on software repositories. Li et al. [7] investigated the effects of PR template in open-source projects on GitHub. They found that PR submissions tended to decrease after PR templates are adopted. Furthermore, they found that PR templates, which require much information make PR submission difficult. Zhang et al. [2] analysed the correlation between PR template usage and GitHub repository characteristics. They found that repositories using PR templates typically gained more popularity. They also found evidence that using PR templates reduced PR review times. The prior studies are limited to understand evolution of PR templates during software development, which is crucial to efficiently manage PR template. To fill this knowledge gap, our study extends these previous studies by analysing PR template initial content organisation and its evolution.

Similarly to PR template, issue report template is used to facilitate for writing issue reports on GitHub. Several previous studies investigated effects of issue report template in software development. Li et al. [8] performed a study on benefits of issue report template. They observed that issue reports written from templates were resolved faster. Sülün et al. [9] examined the impact of issue report template in large-scale open-source projects. They showed that issue report templates significantly reduced the resolution time of submitted issue reports. Zhang et al. [10] investigated the relationship between issue report templates and software development characteristics and found that the adoption of issue report templates was associated with increasing productivity. In another study on open-source projects [11], they showed that issue report templates are useful to identify duplicate issue reports. Patterns of content organisation and changes in issue report templates remain unknown. Our approach presented in this paper could be applied to find these patterns.

In software development projects, README.md and CONTRIBUTING.md files are important documentation files. To understand the content and structure of these two files, several studies have been performed. Ikeda et al. [12] investigated the README.md contents of JavaScript package projects. They revealed that usage, installation, and licence categories are commonly included in the README.md files. Liu et al. [13] compared the README.md content structures of open-source Java projects. They found 32 common README content structures. Fronchetti et al. [14] examined CONTRIBUTING.md of open-source software projects and found that these files provide inadequate information about newcomer onboarding barriers. These prior studies contributed to understanding the content and structure of README.md and CONTRIBUTING.md files used in software development projects. This study can contribute to understanding the initial content organisation and evolution of PR templates.

3. Data Collection

3.1. Overview

To collect the dataset used in this study, we performed the data collection process shown in Figure 1. The overall data collection process consists of four steps: (1) We identified 2614 public GitHub repositories and (2) extracted 2689 PR template files from those repositories. (3) We then manually examined contents of the extracted files and defined seven PR template content categories. (4) Finally, we built a classifier to automatically identify the defined categories. The following subsections describe each of the data collection steps in detail.

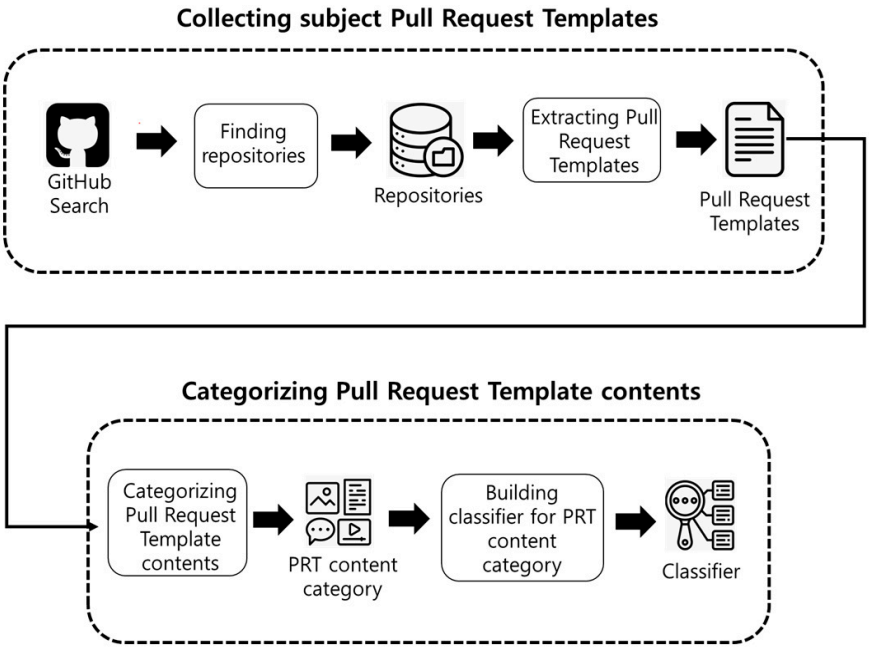


Figure 1. Overview of the data collection process. The process consists two phases, collection of subject pull request templates and categorization of pull request template contents, represented by dashed lines.

3.2. Collecting Target Pull Request Templates

To select a set of sample repositories for this study, we used the GitHub Search (GHS) dataset provided for supporting sampling of GitHub repositories [15]. Previous studies [16–20] also used the dataset to collect their sample repositories. We downloaded the latest version of the dataset (https://zenodo.org/records/4588464 (accessed on 20 January 2026)) and then randomly extracted 5315 repositories from the dataset to form an initial set of candidate repositories, and then excluded those that do not contain any PR template files or only provide non-English PR template files from the candidate repository set since those repositories are not suitable for this study. The remaining 2614 repositories were selected as our sample repository set.

After selecting the sample repository set, we searched for target PR template files in the selected repositories. We used a heuristic method based on the best practices suggested by the GitHub guide (https://docs.github.com/en/communities/using-templates-to-encourage-useful-issues-and-pull-requests/creating-a-pull-request-template-for-your-repository (accessed on 20 January 2026)) for locating the path of PR template files in repositories. The GitHub guide recommends locating PR template files in the root or under the ".guthub/" paths in a repository. We manually explored entire paths of the selected repositories and then identified that their PR template files are typically located in the paths of "/PULL_REQUEST_TEMPLATE.md", ".github/PULL_REQUEST_TEMPLATE.md", or ".github/PULL_REQUEST_TEMPLATE/PULL_REQUEST_TEMPLATE.md". By considering the paths, we retrieved the PR template files using the GitHub API (Get repository content (https://docs.github.com/en/rest/repos/contents?apiVersion=2022-11-28#get-repository-content (accessed on 20 January 2026))), which provides the contents of a specific path within a GitHub repository. As a result of this data collection, we collected 2689 target PR template files from the sample repositories. The target dataset is used for RQ1 and RQ2.

3.3. Categorising Contents of Pull Request Template

PR template files are typically provided in markdown format [21]. Figure 2 shows a PR template file in the GitHub repository britecharts-react. To define content categories of

PR templates, we analysed the texts of the target PR template files using an open coding approach [22], in accordance with the previous studies [7,23–25]. The author of this paper and two graduate students with expertise in software engineering served as annotators in the coding work. We randomly selected 1024 samples from the target PR template files and manually examined their texts. Each annotator carefully read the content titles in the texts and defined their meaning. Cohen's Kappa coefficient [26] was used to measure the degree of agreements on their decisions. Finally, we achieved 92% of a Cohen's Kappa coefficient in the end of our work. The coefficient value is interpreted as indicating strong agreement. Table 1 shows seven PR template content categories defined through the manual annotation works.

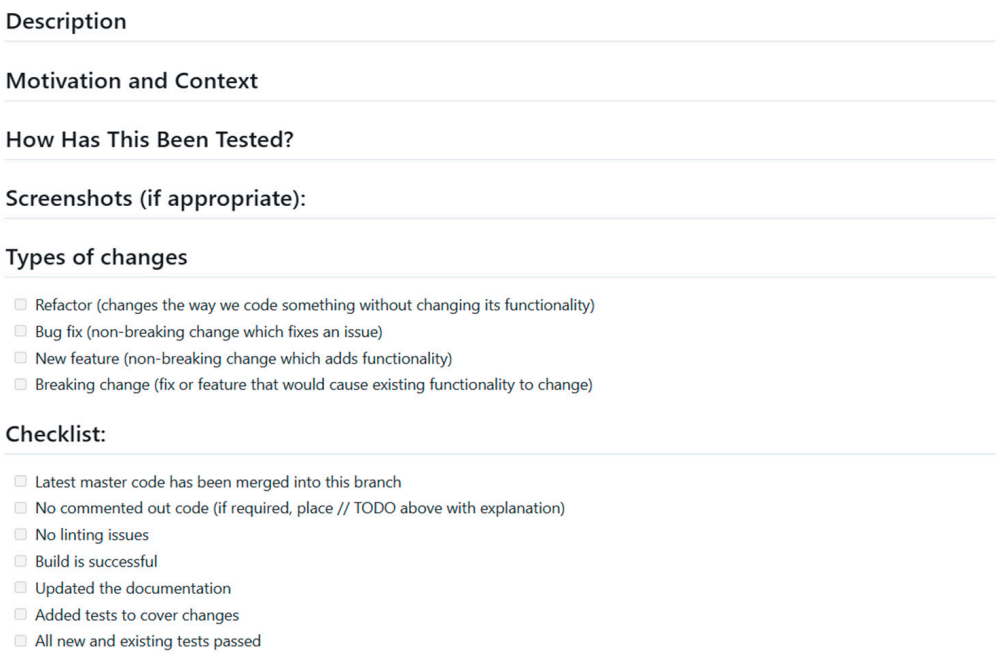


Figure 2. The content of PR template provided by the repository of britecharts/britecharts-react.

Table 1. The defined PR template content categories.

Content Category	Meaning	Example Content Head Text
Description	introduce summary and details of changes and behaviours	brief description, brief summary, code changes
Checklist	guide about submitting PR or other	before submitting a pr, notes, checklist before submitting
Reference	refer to related Issue reports and other documents	related issue, jira ticket number, what is the bug?
Test	describe testing and validation steps of submitted changes	test cases, test plan, validation steps performed, validation
Type	selects submitted change or PR types	type of pr, what type of pr is this?, types of changes
Additional info.	provides other useful information such as reviewer, screenshot, affecting area	screenshots for the change, component name, usage examples, how to review this pr
Other	other contents which do not involved above categories	learning, general, cla, other

After the annotation work, we constructed a training dataset to build a classifier that can automatically identify the defined content categories in all the target PR template files. We randomly selected 983 samples from the target PR template files. By using a markdown parser, we extracted the content titles from these samples and preprocessed the extracted content titles by using natural language processing techniques such as tokenisation, non-word token removal, and stemming. We then manually determined the labels of the preprocessed content titles in the defined content categories. Using this dataset, we performed 10-fold cross-validation on classification models such as multinomial naive Bayes (MNB) [27], linear SVC (L-SVC) [28], and logistic regression (LR) [29], which are commonly used in text classification problem, to find the best model in this study. The cross-validation results showed that the L-SVC-based model, LR-based model, and MNB-based model achieved 80%, 78%, and 68% of mean F1 scores, 82%, 84%, and 75% of mean precisions, and 80%, 77%, and 67% of mean recalls, respectively. Based on these results, we decided to use the L-SVC-based model as our classifier for this study. The L-SVC-based model was used to extract the PR template content categories in Table 1 from all of the target dataset.

4. RQ1: How Are the Initial Contents of PR Templates Organised?

Motivation: Through the data collection works, we defined seven content categories for PR template. The intention of RQ1 is to investigate common types of initial content organisation for PR templates. The answer of RQ1 could provide GitHub repository managers with information on the common content organisation types frequently used in PR templates. This is crucial to develop guideline for organising PR template initial contents.

Approach: To extract the initial contents of the target PR template files, we collected their commit dataset using the GitHub API (https://docs.github.com/en/rest/commits/commits?apiVersion=2022-11-28 (accessed on 20 January 2026)), which retrieves a set of commits related to its path parameter. We called the API by passing the path string of the target PR template files. As a result, we obtained 7881 commits containing the previous content texts of the target PR templates. For each of the PR template files, we sorted its commit dataset in chronological order and then found the initial content text in the "raw_url" field at the first commit.

We performed a clustering analysis to group the target PR template files with similar initial content categories. For the clustering analysis, we define feature vector ($X = [x_D, x_C, x_R, x_{TE}, x_{TY}, x_A, x_O]$) to describe content categories included in PR template initial content, where each of x_D, x_C, x_R, x_{TE}, x_{TY}, x_A, and x_O is a binary variable having value of 0 or 1, which indicate the presence or absence of Description, Checklist, Reference, Test, Type, Additional info., and Other in a PR template file, respectively. For example, if a PR template file has Description and Checklist categories, its feature vector is formed as X = [1, 1, 0, 0, 0, 0, 0]. We extracted content titles from the initial content texts of the target PR template files. The extracted content titles were preprocessed by using the natural language processing techniques employed in the PR template content categorization work described in Section 3.3. We classified the categories of the preprocessed content titles by using the built L-SVC based classifier. Based on the classification result, the feature vectors for the target PR template files were formed. We then applied agglomerative clustering [30], one of the hierarchical clustering methods, to the feature vectors to find hierarchy structure of the content categories. Based on the prior study [31], the similarity threshold was set to 12.5, which was decided by manually analysing the dendrogram resulted in the agglomerative clustering, and the Euclidean distance similarity function was used for the agglomerative clustering.

To interpret the clustering results, we define content category proportion (CCP) of a cluster as follow.

$$CCP(C_i,\ cc_j) = \frac{\#.PRTs\ in\ C_i\ with\ cc_j}{\#.PRTs\ in\ C_i} \quad (1)$$

where C_i and cc_j represent a cluster i and a content category j, respectively. $CCP(C_i,\ cc_j)$ ranges from [0, 1]. The minimum value indicates that none of the PR template files in C_i include cc_j, while the maximum value indicates that all PR template files in C_i include cc_j. In this study, we interpreted the level of proportion of $CCP(C_i,\ cc_j)$ in five degrees such as very small [0.0, 0.2], small [0.2, 0.4], moderate [0.4, 0.6], large [0.6, 0.8], and very large [0.8, 1.0]. We used 0.8 of CCP as the threshold to determine representative content categories of clusters.

Result: Figure 3 shows a dendrogram which visually represents the result of agglomerative clustering analysis. The y-axis refers to the Euclidean distance values between the clusters found by the clustering analysis, and the x-axis refers to unique identifiers randomly assigned to the clusters. With the similarity threshold which we set in the clustering analysis, 13 clusters were identified in the target PR template files.

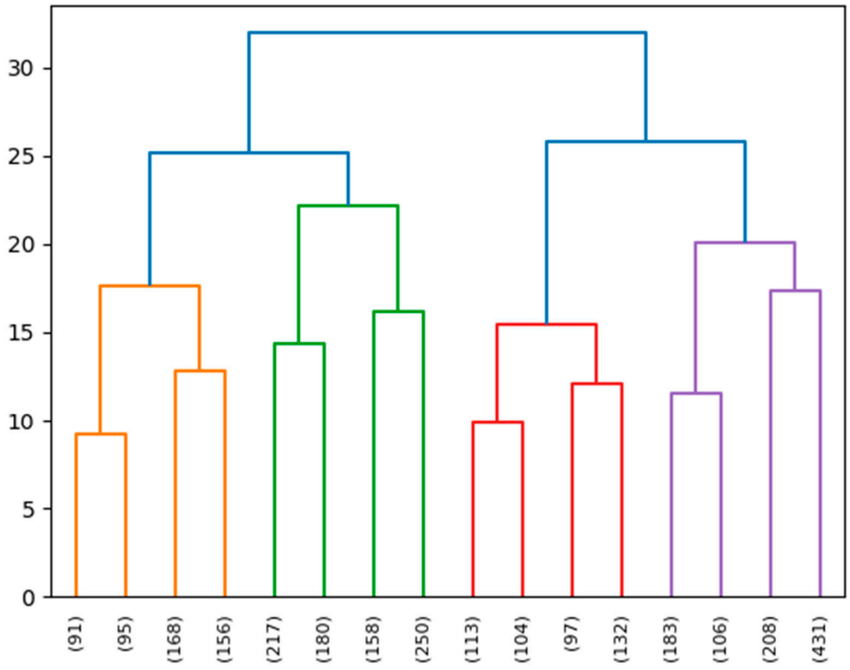


Figure 3. The result dendrogram for the performed agglomerative clustering analysis.

Table 2 shows the CCPs of the clusters. In Table 2, the bold numbers indicate values more than the CCP threshold (0.8≥). In the result, we identified representative content categories of the clusters which are shown in Table 3. We observed that the clusters, except for C1 and C4, exhibited distinct sets of representative content categories. C1 and C4 had the identical representative content category <Other> but different proportions in the content categories of Description, Checklist, and Reference. C1 had a higher proportion of these content categories than C4. C0 and C2 had one representative content category <Checklist> and <Description>, respectively. The remaining clusters had different sets, which consist of two or more representative content categories. C5, C10, C11, and C12 involved two representative content categories. They indicated <Description, Additional info.>, <Description, Checklist>, <Description, Test>, and <Description, Type>, respectively. C3, C7, and C9 had three representative content categories. Each of the clusters represented <Description, Checklist, Test>, <Description, Checklist, Reference>, and <Checklist, Type,

Additional info.>, respectively. C8 and C6 indicated representative content categories <Description, Reference, Test, Additional info.> and <Description, Checklist, Reference, Test, Type>, respectively.

Table 2. The CCPs of the result clusters. The second column (#. PRTs) indicates the number of PRs involved in each cluster.

Cluster	#. PRTs	Description	Checklist	Reference	Test	Type	Additional Info.	Other
0	208	0.000	**0.813**	0.423	0.125	0.014	0.168	0.120
1	229	0.764	0.764	0.576	0.000	0.192	0.210	**1.000**
2	289	**1.000**	**0.000**	0.367	0.000	0.000	0.000	0.000
3	180	**0.972**	**0.811**	0.000	**1.000**	0.289	0.533	0.117
4	217	0.415	0.000	0.000	0.000	0.000	0.194	**1.000**
5	250	**1.000**	0.392	0.520	0.000	0.000	**1.000**	0.000
6	186	**0.941**	**0.817**	**0.892**	**1.000**	**1.000**	0.597	0.156
7	181	**1.000**	**1.000**	**1.000**	0.348	0.000	0.000	0.000
8	158	**1.000**	0.392	**1.000**	**0.886**	0.000	**1.000**	0.576
9	156	0.404	**0.827**	0.006	0.231	**0.840**	**1.000**	0.000
10	250	**1.000**	**1.000**	0.000	0.000	0.000	0.000	0.000
11	217	**0.908**	0.203	0.502	**1.000**	0.009	0.000	0.359
12	168	**0.804**	0.607	0.524	0.000	**1.000**	0.327	0.119

Table 3. The representative content categories of the result clusters.

Cluster	Representative Content Categories
C0	C
C1, C4	O
C2	D
C3	D, C, TE
C5	D, A
C6	D, C, R, TE, TY
C7	D, C, R
C8	D, R, TE, A
C9	C, TY, A
C10	D, C
C11	D, TE
C12	D, TY

The result shows that 8 of the 13 clusters have two or fewer representative content categories. We confirmed that 68% of the target PR template files were included in these clusters. This means that initial contents of PR templates are typically insufficiently organised.

Based on the results, the answer to RQ1 is summarised as follows.

"The initial contents of PR templates are typically organised in two or fewer content categories."

5. RQ2: How Have the Initial Contents of PR Templates Evolved over Time?

Motivation: We noticed that the majority of the target PR template files (68%) fell into the clusters (8 out of 13), with two or fewer representative content categories in the result of RQ1. Similarly to other software artefact such as source files, the initial content of PR templates can be changed over time. In RQ2, we focus on the evolution of PR template initial content category organisation. RQ2 could contribute to understanding repository managers' effort to improve their PR templates.

Approach: Before starting the investigation for RQ2, we counted the number of commits of the target PR template files. We found that 1079 of the files have only one commit. These are not suitable for the RQ2, as they have not been modified since their first commit. Therefore, we excluded them, leaving 1610 in the target dataset.

To answer RQ2, we performed a quantitative analysis on the change in PR template content category organisation. PR template content category organisation may vary according to content category changes performed in subsequent commits. Content category changes performed in commits are classified into three types: addition <$\varepsilon \rightarrow$ c>, deletion <c $\rightarrow \varepsilon$>, and modification <c $\rightarrow$ c'> of a content category. The addition and deletion increase and decrease the number of content categories of existing PR template, respectively. The modification slightly changes the title of an existing content category, which has no effect on existing PR template content category organisation. Based on these changes, we define the content category volume change (CCVC) of a PR template as follows:

$$CCVC(prt) = \sum_{i}^{n} |Added_i| - |Removed_i| \quad (2)$$

where prt is a PR template. n refers to the number of commits of prt. $Added_i$ and $Removed_i$ represent the sets of content categories added and removed by the additions and deletions in commit i. Positive or negative $CCVC(prt)$ indicate an increasing or decreasing change in the content category volume of prt, respectively, while zero means no change.

We implemented a method to extract content title changes to identify content category changes in the target PR template files. Figure 4 presents the pseudocode for the method. The implemented method takes a pair of previous and current commits of a PR template and a string similarity threshold as its input, and outputs the sets of added, removed, and changed content titles identified in the input commit pair. The previous and current content title sets are extracted from the input commits by using a markdown parser (lines 4–5). The extracted content title sets are compared to identify the sets of added and removed content titles between the previous and current commits. It is assumed that the removed content titles do not exist in the current set, and conversely, the added content titles do not exist in the previous set. Therefore, the previous content titles not in the current set are included to the removed content title set (lines 6–8), and the current ones not in the previous set are included to the added content title set (lines 9–11). For each of the removed content title, string similarities with the added content titles are computed, and the most similar added content title is selected as a candidate changed content title (line 13). If the string similarity of the removed and selected content title is more than the input threshold, these content titles are determined as a change pair (*removed_c_title*, *added_c_title*) and are added to the changed set. They are then excluded from the removed and added content title sets (lines 14–16).

Based on the method, we extracted added, removed, and changed sets of content titles from all the commits of target PR template files. In this work, we used the Jaro-Winkler algorithm implemented in the Python library strsimpy for the string similarity calculation and set the input threshold to 0.7, which was determined from the results of content title

change identification experiment described in Section 6.1. Then, we preprocessed the identified content titles using the identical preprocessing steps used in the categorization work described in Section 3.3 and classified the categories of the preprocessed content titles using the L-SVC-based classifier. After this classification task, we computed the CCVCs of the target PR template files using Equation (2).

Identifing PR template content title changes

Input : *a pair of commits* $(prev_commit, cur_commit)$ *of a pull request template and string similarity threshold* θ.
Output : *Added, Removed and Changed sets of content titles between the input commits.*

$Added \leftarrow \emptyset$
$Removed \leftarrow \emptyset$
$Changed \leftarrow \emptyset$
$prev_C_titles \leftarrow extract_content_category_titles(prev_commit)$
$cur_C_titles \leftarrow extract_content_category_titles(cur_commit)$
for each c_title *in* $prev_C_titles$
    if c_title *not in* cur_C_titles
        $Removed \leftarrow Removed \cup c_title$
for each c_title *in* cur_C_titles
    if c_title *not in* $prev_C_titles$
        $Added \leftarrow Added \cup c_title$
for each $remove_c_title$ *in* $Removed$
    $added_c_title \leftarrow \underset{added_c_title \in Added}{\arg\max} strsim(removed_c_title, added_c_title)$
    if $strsim(removed_c_title, added_c_title) \geq \theta$
        $Changed \leftarrow Changed \cup (removed_c_title, added_c_title)$
        $Added \leftarrow Added - added_c_title$
        $Removed \leftarrow Removed - removed_c_title$
return $Added, Removed, Changed$

Figure 4. The procedure of the method of identifying PR template content changes.

Result: From the calculated CCVCs of the target PR template files, we observed that 940 of the PR template files had 0 of CCVC, while 361 and 309 files had CCVCs greater and less than 0, respectively. Figure 5 is the distribution of the CCVCs of the target PR template files. This distribution indicates three characteristics related to the CCVCs of the target template files: (1) The majority of the template files have no CCVC. (2) There are slightly more template files with increasing content category volume than those with decreasing content category volume. (3) The range of decreasing content volume is slightly larger than the range of increasing content category volume.

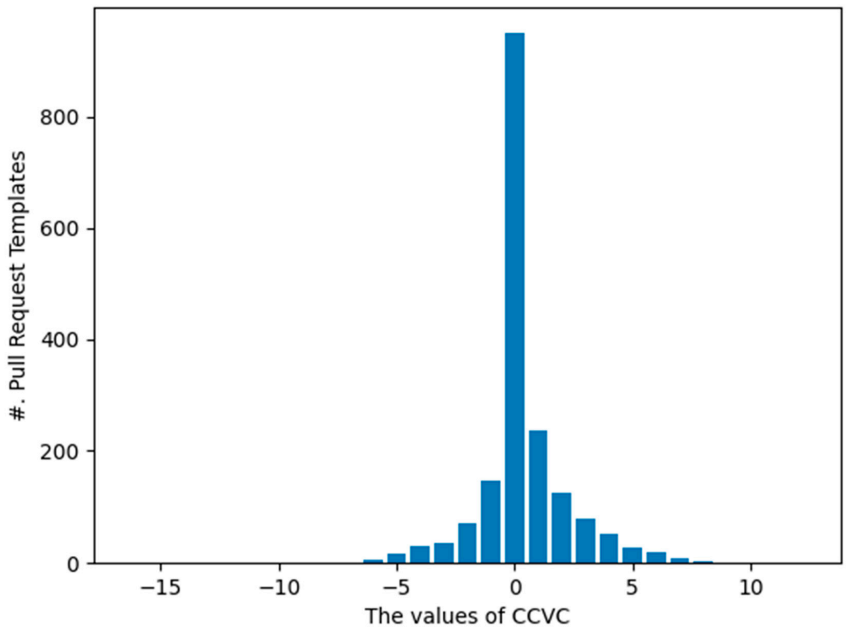


Figure 5. The CCVC distribution of the target PR template files. The y-axis indicates the number of pull request templates.

Based on the calculated CCVCs, we divided the target PR template files into four groups. The template files with zero of CCVC were grouped as the Constant group. In the group, the template files having their initial content category organisation changed were separately classified as the Changed group. The remaining template files with CCVC greater than 0 were included in the Increased group, while those with CCVC less than 0 were contained in the Decreased group. Figure 6 shows the proportions of the PR template files in these groups. Its mean, min, max, Q1 (25%), Q2 (50%), and Q3 (75%) percentile are 0.34, −16, 12, 0, 0, and 1, respectively. The Constant group was the largest among the groups. It had 53.8% of the target PR template files. The Increased group is larger than the Decreased group. They included 22.5% and 19.2% of the template files, respectively. The Changed group is the smallest and had only 4.5% of the template files. These results show that the constant trend is most common in the evolution of initial content of PR templates. It indicates that the initial contents of PR templates are rarely changed.

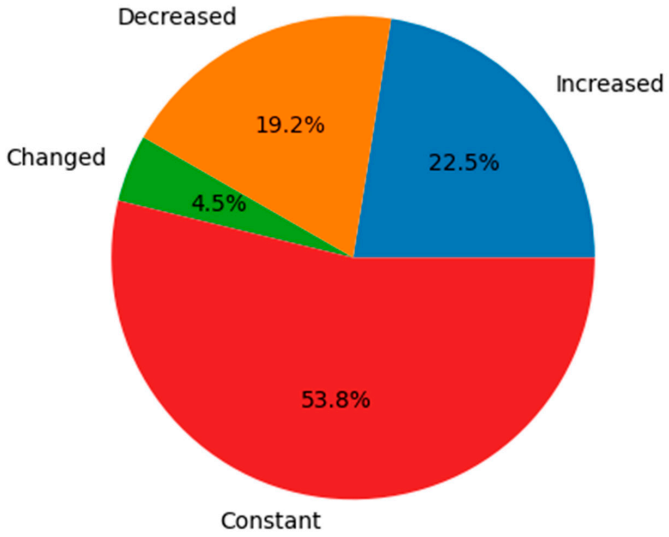


Figure 6. The proportion of the target PR template files having Increased, Changed, and Constant trends.

Based on these results, the answer to RQ2 is concluded as follows.

"The constant trend is most common in the evolution of PR template initial contents. The initial contents of PR templates are rarely changed."

6. Discussion

6.1. Finding the Threshold for Identifying PR Template Content Title Change

To address RQ2, we implemented a PR template content title change identification method. This method determines content title changes based on string similarity threshold, which is given as its input parameter. As the threshold influences the determination, the results of the method vary according to the given threshold. Hence, it is necessary to find the optimal string similarity threshold to ensure the reliability of the method.

To find the optimal threshold, we conducted a simple experiment of content title change identification. We randomly selected 101 content title changes from the commits of target PR templates and labelled 58 correct and 43 incorrect content title changes by manually checking these content title changes. For example, changes such as like "Checklist" -> "Checklist:" and "Testing instructions" -> "Testing information" were classified as the correct changes, while others such as "References" -> "PR Checklist" and "Description" -> "Checklist" were classified as the incorrect changes. We then performed the experiment by using the labelled dataset. The work was repeated nine times with increasing test string

similarity threshold by 0.1, from 0.1 to 0.9. For each iteration, we computed the Jaro-Winkler similarity of the content title changes in the dataset. The changes were determined as correct if their computed similarities are more than the test threshold, otherwise as incorrect. The accuracy of this work was evaluated by the following equation.

$$accuracy = \frac{\#\ correct\ decisions}{|CS| + |IS|} \tag{3}$$

where CS and IS refer to the change sets identified as correct and incorrect, respectively. Figure 7 shows the results of this experiment. We confirmed that the highest accuracy was achieved at 0.7 of test threshold. Based on this result, we decided to set the input threshold of the content title change identification method to 0.7 in RQ2.

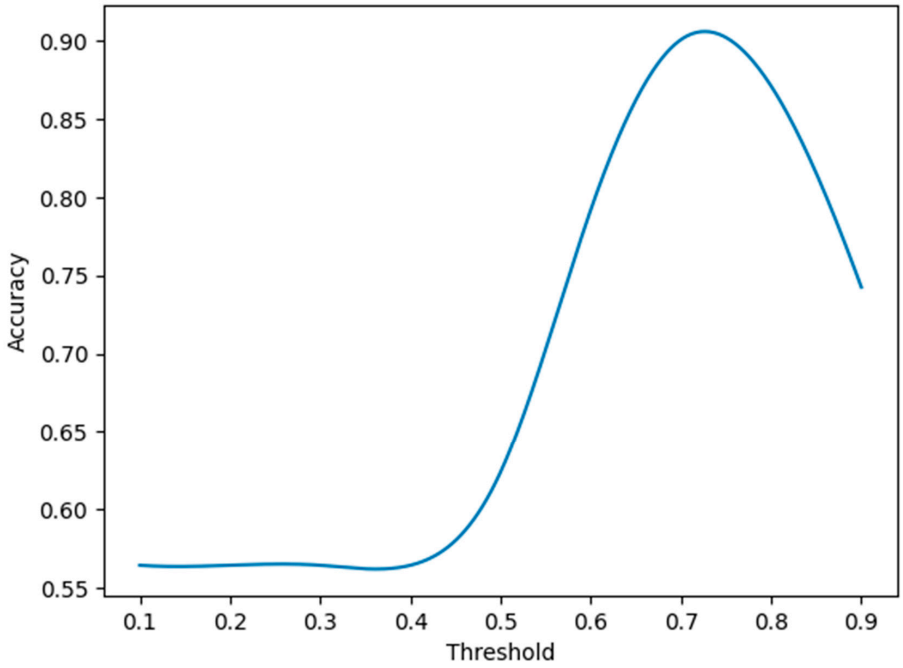


Figure 7. The result of experiment of identifying content category change.

6.2. Suitability of the CCP Threshold

The result of RQ1 relies on the used CCP threshold to determine the representative PR content categories in the clustering result. To validate the suitability of the threshold, we applied KMode clustering [32] to the target PR template dataset used in RQ1. KMode clustering algorithm is suitable for clustering categorical binary data and shows representative categories of its result clusters. KMode clustering requires three parameters, the number of clusters, the initial number of iterations for selecting centroids, and the maximum number of iterations. We set the parameters to 13, 20, and 100. The hamming distance was used because it is suitable for comparing binary categorical data. Table 4 shows the result clusters obtained from the clustering and their representative PR content category sets. By comparing these results with the result clusters in Table 3, we confirmed that there were nine identical centroid clusters (C0, C1, C2, C3, C4, C5, C6, C7, and C9). It indicates that the clusters are close to those found in the previous clustering analysis with a CCP threshold of 0.8.

Table 4. The representative PR content categories of the result clusters of KMode. The * mark indicates a cluster having identical representative PR content categories with a cluster in Table 3.

Cluster	Representative PR Content Categories
C0 *	C
C1 *	D, C, R
C2 *	D, C
C3 *	D, A
C4 *	O
C5 *	C, TY, A
C6 *	D, C, R, TE, TY
C7 *	D, TE
C8	D, C, R, TE
C9 *	D
C10	R, O
C11	D, R, O
C12	D, C, TE, TY

6.3. Comparison of PR Template Evolution

RQ2 shows that the majority (53.8%) of the target PR template files remains their initial content organisation. We performed a statistical analysis to investigate whether repository characteristics such as active days and number of contributors are related to the PR evolution. Based on the result of RQ2, we divided the target repositories into two groups, repositories with the PR templates having the Constant trend as the constant group and the others as the non-constant group. We then calculated the distributions of their active days and number of contributors and compared the means by using the Mann–Whitney U test, which is a nonparametric statistical test method. To measure the difference between these mean values, we used the Cohen's d effect size. Table 5 shows the five number summaries and the results of statistical test. The result for active days showed that the p-value is larger than 0.01, and the Cohen's d effect size is -0.01. This indicates that the groups have no difference in their active days. Meanwhile, the result for the number of contributors shows that the p-value is less than 0.01, and the effect size is 0.2. It indicates that they have statistically significant difference in the number of contributors (average difference of 168 contributors). The result suggests that the PR template evolution is related to the number of contributors rather than active days.

Table 5. The distribution summary and statistical test result for active days and number of contributors of the Non-changed and Changed groups.

Variable	Group	Mean	Min	Q1	Q2	Q3	Max	p-Value	Effect Size
Active days	Constant	3050	0	2320	2940	3659	6149	0.01 > (0.47)	−0.04
	Non-constant	3003	0	2287	2888	3658	5933		
Num. contributors	Constant	169	0	32	72	166	6292	0.01<	0.2
	Non-constant	337	0	42	98	194	10,554		

To investigate the relationship between the initial content category organisation and CCVC in PR template evolution, we performed Spearman rank-order correlation analysis, a nonparametric correlation analysis, between the number of initial content categories and CCVC in the non-constant group. The result shows that the p-value is less than 0.01, and the correlation coefficient is -0.35. This indicate that there is a statistically significant weak negative correlation between PR initial content category organisation and CCVC. It suggests that initial PR template organisation with few content categories tends to increase according to time and vice versa.

6.4. Implication

The result of RQ1 indicate that PR templates tend to be initially simply organised. It is known that complexity of PR template affects developer contribution through PR submission [2]. PR templates with many content categories make the writing work harder [2,7], while ones with few content categories reduce the hurdle of PR submission. Therefore, repository managers should minimally optimise PR template organisation to reduce contributor's workload in writing PRs. It could lead to a positive effect in the recruitment of long-term contributors. The finding in RQ1 may be behind the intention of repository managers for growing their software repository.

Furthermore, we observed that the constant trend is most common in the evolution of the target PR template files in RQ2. PR templates have the effect of communication burden reduction between contributors and PR reviewers regarding submitted PRs [2,7]. The communication burden in a software repository tends to increase as the number of contributors increases [1]. The observed changes in this study may be related to the increment of the communication burden. This is evidenced in the statistical test on the number of contributors between the two groups of repositories with constant and non-constant PR templates described in Section 6.3. It implies that repository managers should consider reconstructing PR template organisation when the number of contributors increases. In addition, we noticed the evolution of the non-constant groups found in RQ2. PR templates may be evolved to reduce their information insufficiency and overload. The insufficiency and overload of PR template information cause to make communication between contributors and PR reviewers difficult. To avoid these issues, repository managers can add additional content categories or remove unnecessary and redundant content categories in existing PR templates. This could lead to a positive effect reducing the communication burden to improve existing PR template content category organisation. The correlation analysis result for the number of initial content categories and CCVC described in Section 6.3 implies such evolution.

6.5. Threats to Validity

Internal validity: The threat to internal validity of this study is related to PR template authors' preferences about PR validation. We carefully suppose that contents of PR template files are typically determined by the preferences. Such preferences may lead to differences in the content organisation of PR template files. This characteristic is not considered in this study because no method has been found to explicitly represent the characteristic as a target factor. Previous studies [2,7] also neglected to account for the characteristic.

External validity: The threat to external validity of this study is related to the selection of the target platform. All the datasets used in this study were collected from GitHub. Hence, the findings of this study may be limited to be applied to other software development platforms. However, GitHub is currently the largest software development platform that uses PR template. We believe there are no significant barriers to applying the findings to repositories in GitHub.

Another threat to external validity is related to the languages of the target PR template files. PR template files can be written in various languages. As explained in Section 3.2, this study excluded PR template files written in non-English languages. Therefore, the findings of this study do not reflect non-English PR template contents. However, the majority of repositories on GitHub support English. Previous studies [2,7] also ignored non-English PR template files. Thus, this threat is not significant.

Construct validity: The threat to construct validity is related to the content category classification method used in this study. This study employed the L-SVC-based classification model to identify the content categories organised in the target PR template files. The results of RQ1 and RQ2 rely on the accuracy of the classification model. To find the most appropriate classification model for this study's classification problem, we performed the 10-fold cross-validations on MNB-, L-SVC-, and LR-based classification models. The validations showed that the L-SVC-based classification model achieved the highest accuracy.

Another threat is related to the string threshold used as an input parameter of the PR template content title change identification method implemented in this study. The results of the method may vary according to the parameter. We found the optimal threshold by conducting the experiment described in Section 6.1. We believe that the efforts could significantly reduce this construct validity threat.

7. Conclusions

This paper presents a study on the initial content organisation of PR templates. We collected 2689 PR template files from 2614 public repositories as the target dataset for this study. By manually examining the contents of the collected PR template files, we defined seven PR template content categories. We developed an L-SVC-based classification model to automatically classify these content categories. We then investigated the following two research questions: (RQ1) How are the initial contents of PR templates organised? and (RQ2) how have the initial contents of PR templates evolved over time? To answer RQ1, we performed a clustering analysis on the initial content category organisation of the target PR template files. The results showed that the initial contents of the target PR template files were typically organised with two or fewer content categories. Furthermore, we conducted a quantitative analysis of the change in the initial content category volume of the target PR template files to address RQ2. The results showed that the constant trend is most common in the evolution of the target PR template files. By comparing the number of contributors between repositories with the constant and non-constant PR templates, we confirmed that the repositories with non-constant PR templates have a statistically significantly larger number of contributors. It implies that PR template evolution is likely related to collaboration complexity through PRs. We also observed a negative correlation between the number of initial content categories and the CCVC of the non-constant PR templates. The result implies that there is likely an optimal point in the number of content categories of PR template. In future work, we plan to conduct an additional study to reveal PR template evolution patterns related to the collaboration complexity and the optimal point. We believe that such patterns can help understand the appropriate time and methods for improving PR templates, and it can also contribute to developing tools to support PR template management and modification works.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2023-00213733).

Data Availability Statement: All the datasets and scripts used in this study are currently provided in https://github.com/aldidos/es_prt_init_content (accessed on 20 January 2026).

Acknowledgments: The author has reviewed and edited the outputs and takes full responsibility for the content of this publication.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Zhang, X.; Yu, Y.; Wang, T.; Rastogi, A.; Wang, H. Pull request latency explained: An empirical overview. *Empir. Softw. Eng.* **2022**, *27*, 126. [CrossRef]
2. Zhang, M.; Liu, H.; Chen, C.; Liu, Y.; Bai, S. Consistent or not? An investigation of using pull request template in GitHub. *Inf. Softw. Technol.* **2022**, *144*, 106797. [CrossRef]
3. Bao, L.; Xia, X.; Lo, D.; Murphy, G.C. A large scale study of long-time contributor prediction for GitHub projects. *IEEE Trans. Softw. Eng.* **2019**, *47*, 1277–1298. [CrossRef]
4. Rehman, I.; Wang, D.; Kula, R.G.; Ishio, T.; Matsumoto, K. Newcomer candidate: Characterizing contributions of a novice developer to GitHub. In Proceedings of the 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME), Adelaide, Australia, 28 September–2 October 2020. [CrossRef]
5. Jiang, J.; Lv, J.; Zheng, J.; Zhang, L. How developers modify pull requests in code review. *IEEE Trans. Reliab.* **2021**, *71*, 1325–1339. [CrossRef]
6. GitHub: Creating a Pull Request Template for Your Repository. Available online: https://docs.github.com/en/communities/using-templates-to-encourage-useful-issues-and-pull-requests/creating-a-pull-request-template-for-your-repository (accessed on 4 November 2025).
7. Li, Z.; Yu, Y.; Wang, T.; Lei, Y.; Wang, Y.; Wang, H. To follow or not to follow: Understanding issue/pull-request templates on github. *IEEE Trans. Softw. Eng.* **2022**, *49*, 2530–2544. [CrossRef]
8. Li, H.; Yan, M.; Sun, W.; Liu, X.; Wu, Y. A first look at bug report templates on GitHub. *J. Syst. Softw.* **2023**, *202*, 111709. [CrossRef]
9. Sülün, E.; Metehan, S.; Eray, T. An empirical analysis of issue templates usage in large-scale projects on github. In *ACM Transactions on Software Engineering and Methodology*; Association for Computing Machinery: New York, NY, USA, 2024; Volume 33, pp. 1–28.
10. Zhang, J.; Peng, M.; Zhang, Y. Empirical study on github issue report templates. In Proceedings of the IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC), Osaka, Japan, 2–4 July 2024; pp. 1284–1289.
11. Zhang, J.; Liu, Z.; Bao, L.; Xing, Z.; Hu, X.; Xia, X. Inside Bug Report Templates: An Empirical Study on Bug Report Templates in Open-Source Software. In Proceedings of the 15th Asia-Pacific Symposium on Internetware, Macau, China, 24 July 2024; pp. 125–134.
12. Ikeda, S.; Ihara, A.; Kula, R.G.; Matsumoto, K. An empirical study of readme contents for javascript packages. In *IEICE TRANSACTIONS on Information and Systems*; Institute of Electronics: Tokyo, Japan, 2019; Volume 102, pp. 280–288.
13. Liu, Y.; Noei, E.; Lyons, K. How ReadMe files are structured in open source Java projects. *Inf. Softw. Technol.* **2022**, *148*, 106924. [CrossRef]
14. Fronchetti, F.; Shepherd, D.C.; Wiese, I.; Treude, C.; Gerosa, M.A.; Steinmacher, I. Do contributing files provide information about oss newcomers' onboarding barriers? In Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, San Francisco, CA, USA, 3–9 November 2023; pp. 16–28.
15. Dabic, O.; Aghajani, E.; Bavota, G. Sampling projects in github for MSR studies. In Proceedings of the 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), Madrid, Spain, 17–19 May 2021; pp. 560–564. [CrossRef]
16. Tufano, R.; Masiero, S.; Mastropaolo, A.; Pascarella, L.; Poshyvanyk, D.; Bavota, G. Using pre-trained models to boost code review automation. In Proceedings of the 44th International Conference on Software Engineering, Pittsburgh, PA, USA, 21–29 May 2022; pp. 2291–2302. [CrossRef]
17. Rosalia, T.; Luca, P.; Gabriele, B. Automating Code-Related Tasks Through Transformers: The Impact of Pre-training. In Proceedings of the 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), Melbourne, Australia, 14–20 May 2023; pp. 1–13. [CrossRef]
18. Klivan, S.; Höltervennhoff, S.; Panskus, R.; Marky, K.; Fahl, S. Everyone for themselves? A qualitative study about individual security setups of open source software contributors. In Proceedings of the IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2024; pp. 1065–1082. [CrossRef]
19. Nikeghbal, N.; Kargaran, A.H.; Heydarnoori, A.; Schütze, H. Girt-data: Sampling github issue report templates. In Proceedings of the IEEE/ACM 20th International Conference on Mining Software Repositories (MSR), Melbourne, Australia, 15–16 May 2023; pp. 104–108. [CrossRef]
20. Nikeghbal, N.; Kargaran, A.H.; Heydarnoori, A. Girt-model: Automated generation of issue report templates. In Proceedings of the 21st International Conference on Mining Software Repositories, Lisbon, Portugal, 2 July 2024; pp. 407–418. [CrossRef]

21. GitHub. About Issue and Pull Request Templates. 2016. Available online: https://docs.github.com/en/github/building-a-strong-community/about-issue-and-pull-request-templates (accessed on 20 January 2026).
22. Glaser, B.G. Open coding descriptions. *Grounded Theory Rev.* **2016**, *15*, 108–110.
23. Zimmermann, T. Card-sorting: From text to themes. In *Perspectives on Data Science for Software Engineering*; Morgan Kaufmann: Burlington, MA, USA, 2016; pp. 137–141. [CrossRef]
24. Elazhary, O.; Storey, M.-A.; Ernst, N.; Zaidman, A. Do as I do, not as I say: Do contribution guidelines match the github con-tribution process? In Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME), Cleveland, OH, USA, 29 September–4 October 2019; pp. 286–290. [CrossRef]
25. Li, Z.; Yu, Y.; Wang, T.; Yin, G.; Li, S.; Wang, H. Are you still working on this? An empirical study on pull request abandonment. *IEEE Trans. Softw. Eng.* **2021**, *48*, 2173–2188. [CrossRef]
26. Więckowska, B.; Kubiak, K.B.; Jóźwiak, P.; Moryson, W.; Stawińska-Witoszyńska, B. Cohen's Kappa Coefficient as a Measure to Assess Classification Improvement Following the Addition of a New Marker to a Regression Model. *Int. J. Environ. Res. Public Health* **2022**, *19*, 10213. Available online: https://www.mdpi.com/1660-4601/19/16/10213 (accessed on 20 January 2026). [CrossRef] [PubMed]
27. Yang, F.J. An implementation of naive bayes classifier. In Proceedings of the International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 12–14 December 2018; pp. 301–306.
28. Suthaharan, S. Support vector machine. In *Machine Learning Models and Algorithms for Big Data Classification*; Integrated Series in Information Systems; Springer: Boston, MA, USA, 2016; Volume 36, pp. 207–235. [CrossRef]
29. Best, H.; Wolf, C. Logistic regression. In *The SAGE Handbook of Regression Analysis and Causal Inference*; Sage: Los Angeles, CA, USA, 2018; pp. 153–171.
30. Tokuda, E.K.; Comin, C.H.; Costa, L.D.F. Revisiting agglomerative clustering. *Phys. Stat. Mech. Its Appl.* **2022**, *585*, 126433. [CrossRef]
31. Boyko, N.I.; Tkachyk, O.A. Hierarchical clustering algorithm for dendrogram construction and cluster counting. *Inform. Math. Methods Simul.* **2023**, *13*, 5–15. [CrossRef]
32. Goyal, M.; Shruti, A. A Review on K-Mode Clustering Algorithm. *Int. J. Adv. Res. Comput. Sci.* **2017**, *8*, 725–729. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.