# What's Inside a GitHub Repository? An Empirical Study on the Contents of 10K Projects

Andre Hora
*DCC, UFMG*
Belo Horizonte, Brazil
andrehora@dcc.ufmg.br

João Eduardo Montandon
*DCC, UFMG*
Belo Horizonte, Brazil
joao@dcc.ufmg.br

Diego Elias Costa
*REALISE Lab, Concordia University*
Montreal, Canada
diego.costa@concordia.ca

***Abstract*—GitHub is the largest code hosting platform, with millions of repositories spanning multiple technologies. Despite this, little is known about the actual contents of GitHub's repositories in the wild. This paper presents an initial empirical analysis to better understand the contents of real-world GitHub repositories. We analyze the files, directories, and extensions present in 10,000 GitHub repositories, as well as their evolution over ten years. Our results show major changes in GitHub over the last decade: (1) the consolidation of *README.md*, *.gitignore*, and *LICENSE* as standard artifacts; (2) the rise of GitHub Actions as the dominant CI/CD platform; (3) the growth of configuration formats such as TOML, YAML, and JSON, alongside a decline in XML; (4) new trends, such as the growth of *Dockerfile*; and (5) emerging content related to LLMs and generative AI (e.g., AGENTS.md and CLAUDE.md). Based on our findings, we discuss implications, including that open source is not only evolving organically but also increasingly guided by GitHub's standards, the rise and fall of technologies, and the potential support for mining software repository studies.**

***Index Terms*—Software evolution, Generative AI, GitHub**

## I. Introduction

GitHub is the largest code hosting platform, with millions of repositories spanning a wide range of technologies. Repositories are commonly expected to follow software development best practices, including testing, CI/CD pipelines, documentation, license information, installation files, and contribution guidelines, to name a few [1], [2]. Recently, repositories have also started incorporating generative LLMs and AI artifacts, such as configuration files for coding agents [3]–[6].

Despite this, little is known about the actual contents of GitHub's real-world repositories. For example, although contribution guidelines are recommended [1], [2], [7], [8], the extent to which they are actually adopted in practice remains unclear. The same observation applies to other important artifacts of software development, such as documentation, CI/CD, and licensing. Rather than assuming which software development artifacts are adopted, **practitioners, researchers, and educators should rely on empirical evidence of what developers actually include in their repositories**. For example, practitioners could use such evidence to benchmark their projects against common practices, researchers could design more representative empirical studies, and educators could align their teaching with what developers actually do.

This paper presents an initial empirical analysis to better understand the contents of real-world GitHub repositories.

TABLE I: Overview of the dataset [9].

| Year | Repositories | Files | Directories | Extensions |
|---|---|---|---|---|
| 2026 | 10,000 | 10,904,237 | 1,970,740 | 14,770 |
| 2021 | 6,371 | 5,709674 | 1,025,852 | 10,901 |
| 2016 | 2,591 | 2,650,192 | 402,451 | 9,487 |

Specifically, we analyze the files, directories, and extensions present in 10,000 GitHub repositories, as well as their evolution over ten years. Table I summarizes our dataset [9]. We propose two research questions to investigate the current state and the evolution of the repositories:

- **RQ1. What files, directories, and extensions are commonly found in GitHub repositories?** Overall, we detected that the most common files in GitHub repositories are *README.md*, *.gitignore*, and *LICENSE*. Commonly used Markdown files are related to open-source best practices, such as *CONTRIBUTING.md*, *CODE_OF_CONDUCT.md*, and *SECURITY.md*. Also, most emerging content is related to LLMs and generative AI (e.g., *CLAUDE.md* and *AGENTS.md*).
- **RQ2. How do files, directories, and extensions in GitHub repositories evolve?** Over the last decade, we observed major changes in GitHub: (1) the consolidation of *README.md*, *.gitignore*, and *LICENSE* as standard artifacts; (2) the rise of GitHub Actions as the dominant CI/CD platform; (3) the growth of configuration formats such as TOML, YAML, and JSON, alongside a decline in XML; (4) growth in Python and JavaScript extensions and a decline in C/C++, Java, and PHP; and (5) new trends, such as the growth of *package.json* and *Dockerfile*.

Based on our findings, we discuss implications for practitioners, researchers, and educators, including that open source is not only evolving organically but also increasingly guided by GitHub's standards, the rise and fall of technologies, and the potential support for mining software repository studies.

**Contributions:** The contributions of this study are threefold: (1) we conduct a large-scale empirical study to explore the actual content of GitHub repositories; (2) we derive actionable implications for practitioners, researchers, and educators; and (3) we provide a dataset of files, directories, and extensions from 10K repositories to support further research [9].

TABLE II: Files, directories, and extensions in 2026.

(a) Files

| Pos | File | Repos | % |
|---|---|---|---|
| 1 | README.md | 9,532 | 95.3 |
| 2 | .gitignore | 9,498 | 95.0 |
| 3 | LICENSE | 7,309 | 73.1 |
| 4 | package.json | 3,333 | 33.3 |
| 5 | CONTRIBUTING.md | 3,170 | 31.7 |
| 6 | CHANGELOG.md | 2,787 | 27.9 |
| 7 | .gitattributes | 2,437 | 24.4 |
| 8 | index.html | 2,392 | 23.9 |
| 9 | Dockerfile | 2,390 | 23.9 |
| 10 | Makefile | 2,341 | 23.4 |

(b) Directories

| Pos | Directory | Repos | % |
|---|---|---|---|
| 1 | .github | 8,245 | 82.5 |
| 2 | workflows | 7,730 | 77.3 |
| 3 | src | 6,009 | 60.1 |
| 4 | docs | 3,740 | 37.4 |
| 5 | tests | 3,545 | 35.5 |
| 6 | scripts | 3,060 | 30.6 |
| 7 | test | 3,004 | 30.0 |
| 8 | ISSUE_TEMPLATE | 2,817 | 28.2 |
| 9 | utils | 2,608 | 26.1 |
| 10 | assets | 2,598 | 26.0 |

(c) Extensions

| Pos | Extension | Repos | % |
|---|---|---|---|
| 1 | md | 9,860 | 98.6 |
| 2 | gitignore | 9,498 | 95.0 |
| 3 | yml | 8,414 | 84.1 |
| 4 | json | 6,920 | 69.2 |
| 5 | png | 6,262 | 62.6 |
| 6 | txt | 5,689 | 56.9 |
| 7 | sh | 4,681 | 46.8 |
| 8 | yaml | 4,207 | 42.1 |
| 9 | js | 4,047 | 40.5 |
| 10 | py | 3,942 | 39.4 |

## II. Study Design

### A. Selecting Repositories

Our goal is to analyze real-world, actively maintained repositories hosted on GitHub. To this end, we rely on the SEART GitHub Search Engine (seart-ghs), a tool that allows researchers to sample repositories to use for empirical studies by using multiple combinations of selection criteria [10]. This tool maintains metadata for all GitHub repositories with at least ten stars. Based on seart-ghs, we selected the repositories that meet the following criteria: at least 100 commits, not being forks, having at least one commit in 2026, and having at least 100 stars (the star metric is primarily adopted in the software mining literature as a proxy of popularity [11], [12]).

This process yielded an initial set of 116,013 repositories, from which we randomly selected 10,000 for analysis. Random sampling was used to ensure a diverse set of repositories, rather than focusing only on the most popular ones, as would be the case if we had selected the top 10K. On the median, the selected repositories have 211 stars and 557 commits. They span 44 primary programming languages and include projects from organizations such as Microsoft, Google, and Facebook.

### B. Extracting Files and Directories

Next, we relied on the *GitHub REST APIs for Git trees*[1] to collect all files, directories, and extensions from the selected repositories. We collected data for the last 10 years, considering the snapshots from 2016, 2021, and 2026. Table I summarizes our dataset. For 2026, we analyzed the 10,000 repositories comprising over 10.9 million files, 1.9 million directories, and 14,770 distinct extensions. Next, we analyzed the repositories from the 2026 dataset that existed in 2021 and 2016, totalling 6,371 and 2,591 repositories, respectively. Our dataset is publicly available [9].

### C. Research Questions

We propose two research questions to explore the current state (RQ1) and the evolution of the repositories (RQ2). The rationale is to identify which software development artifacts are adopted in practice and how the usage changes over time to support practitioners, researchers, and educators. Practitioners can benchmark against common practices (e.g., typical repository structures), researchers can design more representative studies (e.g., building datasets that reflect current development artifacts), and educators can better align teaching with real-world development (e.g., motivating the usage of certain tools). In addition, this study is a first step toward a better understanding of GitHub content at an ultra-large scale.

[1]http://docs.github.com/en/rest/git/trees

## III. Current Content of GitHub (RQ1)

### A. Files, directories, and extensions

Table II provides an overview of the most commonly used files, directories, and extensions in 2026. It is interesting to note that, in all cases, the top three entries appear with considerably high frequency. Among the files, the three most frequent are *README.md* (documentation), *.gitignore* (version control), and *LICENSE* (licensing). Considering the directories, the three most common are *.github* (repository configuration), *workflows* (CI/CD), and *src* (source code). Lastly, regarding the extensions, the top three are: *.md*, *.gitignore*, and *.yml*.

> **Finding 1**: Currently, the most common files in GitHub repositories are *README.md*, *.gitignore*, and *LICENSE*. The most common directory is *.github*, and the most common extension is *.md*.

### B. Emerging files, directories, and extensions

Table III presents the novel files, directories, and extensions identified in 2026. Most of the emerging content in GitHub repositories is related to LLMs, generative AI, and coding agents. In this context, we identified *CLAUDE.md*,[2] *AGENTS.md*,[3] *SKILL.md*,[4] and *copilot-instructions.md*[5] for guiding coding agents. For directories, we detected *skills*, *.claude*, *mcp*, *.cursor*, *.agents*, and *openai*.

It is also worth noting other emerging contents, such as *eslint.config* (configuration file used by ESLint to define linting rules and project settings), *uv.lock* (lockfile generated by uv that records the exact versions of Python dependencies), and

[2]https://code.claude.com/docs/en/best-practices

[3]https://agents.md

[4]https://agentskills.io

[5]https://docs.github.com/en/copilot/how-tos/copilot-on-github/customize-copilot/add-custom-instructions/add-repository-instructions

TABLE III: Emerging files, directories, and extensions in 2026.

(a) Emerging files

| Pos | File | Repos | % |
|---|---|---|---|
| 1 | CLAUDE.md | 898 | 9.0 |
| 2 | AGENTS.md | 846 | 8.5 |
| 3 | SKILL.md | 553 | 5.5 |
| 4 | eslint.config.mjs | 505 | 5.0 |
| 5 | eslint.config.js | 487 | 4.9 |
| 6 | uv.lock | 465 | 4.7 |
| 7 | vitest.config.ts | 413 | 4.1 |
| 8 | tsconfig.node.json | 381 | 3.8 |
| 9 | vite-env.d.ts | 377 | 3.8 |
| 10 | copilot-instructions.md | 313 | 3.1 |

(b) Emerging directories

| Pos | Directory | Repos | % |
|---|---|---|---|
| 1 | skills | 567 | 5.7 |
| 2 | .claude | 450 | 4.5 |
| 3 | mcp | 243 | 2.4 |
| 4 | .cursor | 161 | 1.6 |
| 5 | .vitepress | 122 | 1.2 |
| 6 | .agents | 120 | 1.2 |
| 7 | llm | 109 | 1.1 |
| 8 | [id] | 83 | 0.8 |
| 9 | openai | 82 | 0.8 |
| 10 | src-tauri | 79 | 0.8 |

(c) Emerging extensions

| Pos | Extension | Repos | % |
|---|---|---|---|
| 1 | mts | 224 | 2.2 |
| 2 | mdc | 131 | 1.3 |
| 3 | slnx | 121 | 1.2 |
| 4 | xcprivacy | 108 | 1.1 |
| 5 | astro | 82 | 0.8 |
| 6 | prisma | 64 | 0.6 |
| 7 | codespellrc | 61 | 0.6 |
| 8 | clangd | 53 | 0.5 |
| 9 | work | 48 | 0.5 |
| 10 | lycheeignore | 46 | 0.5 |

*vitest.config.ts* (configuration file used by Vitest to define testing settings). Regarding the extensions, the most frequent emerging extension is *.mts*, a TypeScript file extension used for ECMAScript modules.

> **Finding 2**: Most of the emerging files and directories in GitHub repositories are related to LLMs, generative AI, and coding agents such as *CLAUDE.md*, *AGENTS.md*, *SKILL.md*, *skills*, and *.claude*.

### C. Markdown files

The Markdown extension is the most common in GitHub repositories; therefore, we provide a detailed analysis of Markdown files in Table IV. Most of the top Markdown files are related to open-source best practices, including *README.md*, *CONTRIBUTING.md*, *CHANGELOG.md*, *CODE_OF_CONDUCT.md*, *SECURITY.md*, *LICENSE.md*, and *PULL_REQUEST_TEMPLATE.md*.

GitHub itself recommends the adoption of such files so that repository maintainers can establish guidelines that help collaborators make meaningful contributions [1]. For example, GitHub recommends creating contributing guidelines [2], code of conduct [13], and license [14] files.

TABLE IV: Markdown (*.md*) files in 2026.

| Pos | File | Repos | % |
|---|---|---|---|
| 1 | README.md | 9,532 | 95.3 |
| 2 | CONTRIBUTING.md | 3,170 | 31.7 |
| 3 | CHANGELOG.md | 2,787 | 27.9 |
| 4 | CODE_OF_CONDUCT.md | 1,625 | 16.2 |
| 5 | SECURITY.md | 1,354 | 13.5 |

> **Finding 3**: The most commonly used Markdown files are related to open-source best practices, including *CONTRIBUTING.md*, *CHANGELOG.md*, *CODE_OF_CONDUCT.md*, and *SECURITY.md*.

### D. Dotfiles

Dotfiles are hidden configuration files whose names start with a dot. Table V presents the most common dotfiles found in GitHub repositories. The majority of the dotfiles are Git-related, including *.gitignore*, *.gitattributes*, *.gitmodules*, *.pre-commit-config.yaml*, and *.gitkeep*. We also identified ignore-related files, such as *.gitignore*, *.dockerignore*, and *.prettierignore*, which are used to exclude files and directories from Git tracking, Docker build contexts, and Prettier code formatting.

TABLE V: Dotfiles in 2026.

| Pos | File | Repos | % |
|---|---|---|---|
| 1 | .gitignore | 9,498 | 95.0 |
| 2 | .gitattributes | 2,437 | 24.4 |
| 3 | .editorconfig | 2,013 | 20.1 |
| 4 | .dockerignore | 1,283 | 12.8 |
| 5 | .gitmodules | 995 | 9.9 |

> **Finding 4**: The majority of the dotfiles are Git-related, such as *.gitignore*, *.gitattributes*, and *.gitmodules*.

### E. Files without extension

Files without an explicit extension are also commonly found in GitHub repositories, as detailed in Table VI. The most common are: *LICENSE*, *Dockerfile*, and *Makefile*. Here, we observe two main groups of files: those related to build and automation processes (e.g., *Dockerfile*, *Makefile*, and *gradlew*) and those related to documentation and project governance (e.g., *LICENSE*, *CODEOWNERS*, and *AUTHORS*).

TABLE VI: Files without extension in 2026.

| Pos | File | Repos | % |
|---|---|---|---|
| 1 | LICENSE, | 7,309 | 73.1 |
| 2 | Dockerfile | 2,390 | 23.9 |
| 3 | Makefile | 2,341 | 23.4 |
| 4 | CODEOWNERS | 1,017 | 10.2 |
| 5 | gradlew | 700 | 7.0 |

> **Finding 5**: The most common files without explicit extensions in GitHub repositories are related to build and automation processes (e.g., *Makefile*) or documentation and project governance (e.g., *LICENSE*).

## IV. Content Evolution of GitHub (RQ2)

### A. Overview

In this research question, we investigate the evolution of the content in GitHub repositories. Table VII summarizes the median number of files, directories, and file extensions per repository. Overall, we observe that repositories tend to grow over time, increasing both the number of files (from 111 in 2016 to 211 in 2026) and directories (from 20 to 42). Interestingly, the median number of file extensions per repository also increased, from 12 in 2016 to 17 in 2026, suggesting greater diversity in the technologies being used.

TABLE VII: Overview of the content evolution (median).

| Year | Repos. | Files | Directories | Extensions |
|---|---|---|---|---|
| 2026 | 10,000 | 211 | 42 | 17 |
| 2021 | 6,371 | 131 | 27 | 14 |
| 2016 | 2,591 | 111 | 20 | 12 |

**Finding 6**: Overall, repositories have grown over the last decade in terms of the number of files, directories, and diversity of file extensions.

### B. Increasing/decreasing files, directories, and extensions

Table VIIIa presents the most common files with increasing prevalence, while Table VIIId shows the most common files with decreasing prevalence from 2016 to 2026. For example, the prevalence of *README.md* files increased from 77.8% of repositories in 2016 to 95.3% in 2026 (+17.5%). Other files with a large increase include *.gitignore* (from 84.5% to 95.0%, +10.5%), *LICENSE* (from 50.8% to 73.1%, +22.3%), *package.json* (from 17.2% to 33.3%, +16.1%), and *Dockerfile* (from 5.7% to 23.9%, +18.2%). Regarding the decreasing files, the most notable decline occurs for *.travis.yml*, which decreased from 46.4% in 2016 to 5.3% in 2026 (-41.1%). We recall that Travis CI was the dominant CI/CD platform before the rise of GitHub Actions [15].

Tables VIIIb and VIIIe list the most popular directories with increasing and decreasing prevalence, respectively. Two directories stand out with a remarkable increase in the last 10 years: *.github* (from 5.3% to 82.5%, +77.2%) and *workflows* (from 0.2% to 77.3%, +77.1%). Both are responsible for setting up CI/CD pipelines on GitHub, indicating a significant increase in this practice in the last decade [15].

Tables VIIIc and VIIIf feature the most common extensions with increasing and decreasing prevalence, respectively. Regarding the increasing extensions, *.yaml* (+35.9%), *.toml* (+33.0%), *.yml* (+30.9%), and *.json* (+30.3%) presented the highest increases. All these extensions are commonly used to configure the development environment, including CI/CD pipelines, linters, and package managers [15]–[17]. In contrast, the *.xml* extension decreased in usage by 6.4%. Interestingly, file extensions with decreasing prevalence are centered on source code files of traditional programming languages. For example, *.c*, *.h*, and *.in* extensions are the top-3 most affected ones, with 8.3%, 7.2%, and 7.0% reductions; these files are typically used in C/C++ projects. Source code extensions related to Java and PHP (i.e., *.java* and *.php*) decreased by 6.2% and 4.4%. In contrast, files related to Python (*.py*) and JavaScript (*.js*) increased by 10% and 7.5%.

**Finding 7**: Over the last decade, we observed major changes in GitHub: **(1)** the consolidation of *README.md*, *.gitignore*, and *LICENSE* as standard artifacts; **(2)** the rise of GitHub Actions as the dominant CI/CD platform (and the decline of Travis CI); **(3)** the growth of configuration formats such as TOML, YAML, and JSON, alongside a decline in XML; **(4)** growth in Python and JavaScript extensions and a decline in C/C++, Java, and PHP; and **(5)** new trends, such as the growth of *package.json* and *Dockerfile*.

## V. Discussion and Implications

### A. Current state of GitHub

Nowadays, a typical GitHub repository contains *README.md* (growth from 77.8% to 95.3%; +17.5%), *.gitignore* (84.5%–95%; +10.5%), and *LICENSE* (50.8%–73.1%; +22.3%) files. Interestingly, these files are suggested by the GitHub platform when creating a new repository [18] and can be automatically generated, which may help explain their prevalence in practice. Other highly prevalent artifacts (each above 70%) include the *.github* and *workflows* directories, indicating widespread adoption of GitHub Actions for CI/CD, even though GitHub Actions is not currently suggested by GitHub when creating a new repository.

**Implication #1.** These findings highlight the strong influence of GitHub in shaping open source. In practice, open source is not only evolving organically, but also increasingly guided by the platform's standards. Given the widespread use of artifacts such as *workflow* and *ISSUE_TEMPLATE* directories, GitHub could extend support for new repositories with CI/CD and issue management templates.

### B. Rise and fall of technologies

Our study allows quantifying how the usage of certain artifacts evolves. For example, we observed the decline of Travis CI and the rise of GitHub Actions [15]. Similarly, we identified that lightweight data formats, such as TOML, YAML, and JSON, increased in favor of traditional ones, like XML. There are also specific cases, such as the decline of *setup.py* (-5.1%) (build scripts for Python projects), possibly in favor of more modern alternatives as *pyproject.toml* (+11.5%).

**Implication #2.** At first glance, these trends suggest that maintainers are continuously adapting their repositories to new technologies. Researchers can further investigate the reasons for such changes, as well as propose solutions to support these transitions automatically [19]. This information can also be used by educators and practitioners to guide the technology selection for teaching and adoption.

TABLE VIII: Increasing and decreasing files, directories, and extensions over time (2016–2026).

(a) Increasing files (%)

| Pos | File | 2016 | 2026 | Δ |
|---|---|---|---|---|
| 1 | README.md | 77.8 | 95.3 | 17.5 |
| 2 | .gitignore | 84.5 | 95.0 | 10.5 |
| 3 | LICENSE | 50.8 | 73.1 | 22.3 |
| 4 | package.json | 17.2 | 33.3 | 16.1 |
| 5 | CONTRIBUTING.md | 15.7 | 31.7 | 16.0 |
| 6 | CHANGELOG.md | 12.6 | 27.9 | 15.3 |
| 7 | .gitattributes | 14.4 | 24.4 | 10.0 |
| 8 | index.html | 18.4 | 23.9 | 5.5 |
| 9 | Dockerfile | 5.7 | 23.9 | 18.2 |
| 10 | __init__.py | 15.9 | 22.3 | 6.4 |

(b) Increasing directories (%)

| Pos | Directory | 2016 | 2026 | Δ |
|---|---|---|---|---|
| 1 | .github | 5.3 | 82.5 | 77.2 |
| 2 | workflows | 0.2 | 77.3 | 77.1 |
| 3 | src | 46.7 | 60.1 | 13.4 |
| 4 | docs | 15.4 | 37.4 | 22.0 |
| 5 | tests | 25.5 | 35.5 | 10.0 |
| 6 | scripts | 14.2 | 30.6 | 16.4 |
| 7 | ISSUE_TEMPLATE | 0.1 | 28.2 | 28.1 |
| 8 | utils | 10.5 | 26.1 | 15.6 |
| 9 | assets | 9.1 | 26.0 | 16.9 |
| 10 | images | 16.3 | 23.9 | 7.6 |

(c) Increasing extensions (%)

| Pos | Extension | 2016 | 2026 | Δ |
|---|---|---|---|---|
| 1 | md | 83.4 | 98.6 | 15.2 |
| 2 | gitignore | 84.5 | 95.0 | 10.5 |
| 3 | yml | 53.2 | 84.1 | 30.9 |
| 4 | json | 38.9 | 69.2 | 30.3 |
| 5 | png | 43.0 | 62.6 | 19.6 |
| 6 | sh | 38.0 | 46.8 | 8.8 |
| 7 | yaml | 6.2 | 42.1 | 35.9 |
| 8 | js | 33.0 | 40.5 | 7.5 |
| 9 | py | 29.4 | 39.4 | 10.0 |
| 10 | toml | 2.6 | 35.6 | 33.0 |

(d) Decreasing files (%)

| Pos | File | 2016 | 2026 | Δ |
|---|---|---|---|---|
| 1 | .travis.yml | 46.4 | 5.3 | -41.1 |
| 2 | Makefile | 24.7 | 23.4 | -1.3 |
| 3 | README | 19.5 | 7.0 | -12.5 |
| 4 | LICENSE.txt | 14.7 | 10.6 | -4.1 |
| 5 | setup.py | 12.7 | 7.6 | -5.1 |
| 6 | COPYING | 12.7 | 4.4 | -8.3 |
| 7 | README.txt | 10.8 | 4.8 | -6.0 |
| 8 | AUTHORS | 10.5 | 3.9 | -6.6 |
| 9 | pom.xml | 8.7 | 4.5 | -4.2 |
| 10 | appveyor.yml | 7.6 | 1.8 | -5.8 |

(e) Decreasing directories (%)

| Pos | Directory | 2016 | 2026 | Δ |
|---|---|---|---|---|
| 1 | test | 34.0 | 30.0 | -4.0 |
| 2 | doc | 16.3 | 9.0 | -7.3 |
| 3 | bin | 14.4 | 12.4 | -2.0 |
| 4 | java | 13.4 | 11.3 | -2.1 |
| 5 | css | 13.0 | 12.0 | -1.0 |
| 6 | util | 12.9 | 11.8 | -1.1 |
| 7 | js | 11.9 | 9.8 | -2.1 |
| 8 | include | 10.3 | 7.8 | -2.5 |
| 9 | source | 7.7 | 6.8 | -0.9 |
| 10 | org | 7.4 | 4.0 | -3.4 |

(f) Decreasing extensions (%)

| Pos | Extension | 2016 | 2026 | Δ |
|---|---|---|---|---|
| 1 | xml | 33.1 | 26.7 | -6.4 |
| 2 | h | 25.4 | 18.2 | -7.2 |
| 3 | c | 20.9 | 12.6 | -8.3 |
| 4 | in | 19.5 | 12.5 | -7.0 |
| 5 | java | 17.0 | 10.8 | -6.2 |
| 6 | cpp | 15.2 | 12.2 | -3.0 |
| 7 | properties | 14.2 | 13.1 | -1.1 |
| 8 | cfg | 12.5 | 8.4 | -4.1 |
| 9 | rst | 11.4 | 8.4 | -3.0 |
| 10 | php | 10.2 | 5.8 | -4.4 |

### C. Support to mining software repository studies

Many mining software repository studies rely on detecting specific files and directories. For instance, to study coding agents, researchers must first identify repositories containing files such as *AGENTS.md* and *CLAUDE.md* [5], [6], [20]–[24]. Similarly, studies on contribution guidelines depend on detecting repositories with *CONTRIBUTING.md* files [7], [8], [25]. There is a plethora of other files that can support software mining tasks, such as *LICENSE* for licensing [26], *CHANGELOG.md* for break changes [27]–[29], *package.json* and *pom.xml* for dependencies [17], [30]–[32], *setup.py* and *pyproject.toml* for Python packaging [16], *Dockerfile* for containerization [33], [34], to name a few.

**IMPLICATION #3.** This study is a first step toward a better understanding of GitHub content at a large scale and in continuously updated settings. We plan to develop a taxonomy of repository artifacts, e.g., common files related to software testing, CI/CD, and software dependencies. This taxonomy can guide researchers in selecting an initial set of artifacts to consider when mining software repositories.

## VI. Threats to Validity

We analyzed files, directories, and extensions from a random sample of 10,000 open-source GitHub repositories that are actively maintained in 2026 and have at least 100 stars. Thus, our findings may not directly generalize to other contexts, such as less actively maintained projects, less popular projects, closed-source projects, or other code-hosting platforms like GitLab or Bitbucket.

## VII. Related Work

As a key host of open-source software development, GitHub remains a primary source of empirical studies in SE, powering more than 70% of MSR studies [35]. At a meta-level, many studies have analyzed GitHub project content to report on the challenges of conducting SE research using its data [36] and to recommend data curation strategies [35], [37] and tooling [10]. In comparison, fewer studies have focused on characterizing repository metadata as we have. Gonzalez et al. characterized a decade of AI and ML repositories on GitHub, surfacing unique structural and workflow patterns in this community [38]. Further studies have used repository content to investigate specific software practices, focusing on project README documentation [39], contribution guidelines [7], licensing [40], and dependency management. In contrast to these analyses of specific repository content, our study takes a broad, cross-cutting view of *all* files, directories, and extensions across 10,000 actively maintained repositories, and tracks their evolution over ten years.

## VIII. Conclusion and Further Steps

This paper presented an initial empirical analysis to better understand the contents of real-world GitHub repositories. We analyzed the files, directories, and extensions present in 10,000 GitHub repositories, as well as their evolution. In short, our results revealed major changes in GitHub over the last decade.

**Further Steps:** As future work, we plan to extend this analysis to an ultra-large scale, refine our analysis by better characterizing data across domains, and develop a web platform that enables practitioners, researchers, and educators to easily track the prevalence of files, directories, and extensions.


## Acknowledgments

This research was supported by CNPq (process 403304/2025-3), CAPES, and FAPEMIG.

## REFERENCES


[1] GitHub: Setting up your project for healthy contributions , https://docs.github.com/en/communities/setting-up-your-project-for-healthy-contributions, May, 2026.

[2] GitHub: Setting guidelines for repository contributors , https://docs.github.com/en/communities/setting-up-your-project-for-healthy-contributions/setting-guidelines-for-repository-contributors, May, 2026.

[3] A. Fan, B. Gokkaya, M. Harman, M. Lyubarskiy, S. Sengupta, S. Yoo, and J. M. Zhang, "Large language models for software engineering: Survey and open problems," in *International Conference on Software Engineering: Future of Software Engineering*, 2023, pp. 31–53.

[4] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, and H. Wang, "Large language models for software engineering: A systematic literature review," *ACM Transactions on Software Engineering and Methodology*, 2023.

[5] R. Robbes, T. Matricon, T. Degueule, A. Hora, and S. Zacchiroli, "Promises, perils, and (timely) heuristics for mining coding agent activity," in *International Conference on Mining Software Repositories*, 2026.

[6] ——, "Agentic Much? Adoption of Coding Agents on GitHub," *ACM Transactions on Software Engineering and Methodology*, 2026.

[7] O. Elazhary, M.-A. Storey, N. Ernst, and A. Zaidman, "Do as I do, not as I say: Do contribution guidelines match the GitHub contribution process?" in *International Conference on Software Maintenance and Evolution*, 2019, pp. 286–290.

[8] B. Falcucci, F. Gomide, and A. Hora, "What do contribution guidelines say about software testing?" in *International Conference on Mining Software Repositories*, 2025, pp. 434–438.

[9] Dataset: files, directories, and extensions, https://doi.org/10.5281/zenodo.20185536, May, 2026.

[10] O. Dabic, E. Aghajani, and G. Bavota, "Sampling Projects in GitHub for MSR Studies," in *International Conference on Mining Software Repositories*, 2021, pp. 560–564.

[11] H. Borges, A. Hora, and M. T. Valente, "Understanding the factors that impact the popularity of GitHub repositories," in *International Conference on Software Maintenance and Evolution*, 2016, pp. 334–344.

[12] H. Borges and M. T. Valente, "What's in a GitHub star? understanding repository starring practices in a social coding platform," *Journal of Systems and Software*, vol. 146, pp. 112–129, 2018.

[13] GitHub: Adding a code of conduct to your project , https://docs.github.com/en/communities/setting-up-your-project-for-healthy-contributions/adding-a-code-of-conduct-to-your-project, May, 2026.

[14] GitHub: Adding a license to a repository , https://docs.github.com/en/communities/setting-up-your-project-for-healthy-contributions/adding-a-license-to-a-repository, May, 2026.

[15] A. Decan, T. Mens, P. R. Mazrae, and M. Golzadeh, "On the use of GitHub actions in software development repositories," in *International Conference on Software Maintenance and Evolution*, 2022, pp. 235–245.

[16] M. Alfadel, D. E. Costa, and E. Shihab, "Empirical analysis of security vulnerabilities in python packages," *Empirical Software Engineering*, vol. 28, no. 3, p. 59, 2023.

[17] J. E. Montandon, L. Lourdes Silva, and M. T. Valente, "Identifying Experts in Software Libraries and Frameworks Among GitHub Users," in *International Conference on Mining Software Repositories*, pp. 276–287.

[18] GitHub: Creating a new repository, https://docs.github.com/en/repositories/creating-and-managing-repositories/creating-a-new-repository, May, 2026.

[19] M. Macedo, Y. Tian, P. Nie, F. R. Cogo, and B. Adams, "InterTrans: Leveraging transitive intermediate translations to enhance LLM-based code translation," in *International Conference on Software Engineering*, pp. 1153–1164.

[20] J. L. Lulla, S. Mohsenimofidi, M. Galster, J. M. Zhang, S. Baltes, and C. Treude, "On the Impact of AGENTS.md Files on the Efficiency of AI Coding Agents," *arXiv preprint arXiv:2601.20404*, 2026.

[21] H. V. F. Santos, V. Costa, J. E. Montandon, and M. T. Valente, "Decoding the Configuration of AI Coding Agents: Insights from Claude Code Projects," in *International Workshop on Agentic Engineering*, 2026, pp. 1–5.

[22] A. Hora and R. Robbes, "Are coding agents generating over-mocked tests? an empirical study," in *International Conference on Mining Software Repositories*, 2026.

[23] M. Galster, S. Mohsenimofidi, J. L. Lulla, M. A. Abubakar, C. Treude, and S. Baltes, "Configuring agentic ai coding tools: An exploratory study," in *International Conference on AI-Powered Software*, 2026, pp. 11–20.

[24] T. Gloaguen, N. Mündler, M. Müller, V. Raychev, and M. Vechev, "Evaluating agents.md: Are repository-level context files helpful for coding agents?" *arXiv preprint arXiv:2602.11988*, 2026.

[25] J. Tsay, L. Dabbish, and J. Herbsleb, "Influence of social and technical factors for evaluating contribution in GitHub," in *International Conference on Software Engineering*, 2014, pp. 356–366.

[26] C. Vendome, G. Bavota, M. D. Penta, M. Linares-Vásquez, D. German, and D. Poshyvanyk, "License usage and changes: a large-scale study on github," *Empirical Software Engineering*, vol. 22, no. 3, pp. 1537–1577, 2017.

[27] L. Xavier, A. Brito, A. Hora, and M. T. Valente, "Historical and impact analysis of API breaking changes: A large-scale study," in *International Conference on Software Analysis, Evolution and Reengineering*, 2017, pp. 138–147.

[28] A. Brito, L. Xavier, A. Hora, and M. T. Valente, "Why and how Java developers break APIs," in *International Conference on Software Analysis, Evolution and Reengineering*, 2018, pp. 255–265.

[29] A. Hora, R. Robbes, N. Anquetil, A. Etien, S. Ducasse, and M. T. Valente, "How do developers react to api evolution? the pharo ecosystem case," in *International Conference on Software Maintenance and Evolution*, 2015, pp. 251–260.

[30] A. Decan and T. Mens, "What do package dependencies tell us about semantic versioning?" *IEEE Transactions on Software Engineering*, vol. 47, no. 6, pp. 1226–1240, 2019.

[31] D. Pinckney, F. Cassano, A. Guha, and J. Bell, "A large scale analysis of semantic versioning in npm," in *International Conference on Mining Software Repositories*, 2023, pp. 485–497.

[32] L. B. de Alcântara Júnior and J. E. Montandon, "GivenWhenThen: A Dataset of BDD Test Scenarios Mined from Open Source Projects," in *International Conference on Mining Software Repositories*, 2026, pp. 1–5.

[33] J. Henkel, C. Bird, S. K. Lahiri, and T. Reps, "A dataset of dockerfiles," in *International Conference on Mining Software Repositories*, 2020, pp. 528–532.

[34] J. Cito, G. Schermann, J. E. Wittern, P. Leitner, S. Zumberi, and H. C. Gall, "An empirical analysis of the Docker container ecosystem on GitHub," in *International Conference on Mining Software Repositories*, 2017, pp. 323–333.

[35] M. Vidoni, "A systematic process for mining software repositories: Results from a systematic literature review," *Information and Software Technology*, vol. 144, p. 106791, 2022.

[36] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "An in-depth study of the promises and perils of mining GitHub," *Empirical Software Engineering*, vol. 21, no. 5, pp. 2035–2071, Oct. 2016.

[37] N. Munaiah, S. Kroh, C. Cabrey, and M. Nagappan, "Curating GitHub for engineered software projects," *Empirical Software Engineering*, vol. 22, no. 6, pp. 3219–3253, 2017.

[38] D. Gonzalez, T. Zimmermann, and N. Nagappan, "The state of the ML-universe: 10 years of artificial intelligence & machine learning software development on GitHub," in *International Conference on Mining Software Repositories*, 2020, pp. 431–442.

[39] G. A. A. Prana, C. Treude, F. Thung, T. Atapattu, and D. Lo, "Categorizing the content of GitHub README files," *Empirical Software Engineering*, vol. 24, no. 3, pp. 1296–1327, 2019.

[40] J. Wu, L. Bao, X. Yang, X. Xia, and X. Hu, "A large-scale empirical study of open source license usage: Practices and challenges," in *International Conference on Mining Software Repositories*, 2024, pp. 595–606.