BIG DATA PLATFORMS FINAL PROJECT: CAN TURINGBOTS REPLACE HUMAN SOFTWARE DEVELOPERS?

A DATA-DRIVEN ANALYSIS OF GITHUB ACTIVITY AND PROGRAMMING TRENDS

By: Suyog Mahale

INDEX

EXECUTIVE SUMMARY

OBJECTIVE

We have analysed over 1.36 TB of GitHub Data to uncover trends in programming languages, commits, repository activity, developer behaviour, etc. to assess the feasibility of AI tools such as TuringBots in augmenting or replacing human developers

METHODOLOGY

Through the course of this analysis, we have derived insights about:

1. Timeline of commits. How commits have been distributed through the years, valleys and spikes, and missing data.
2. Language and Licenses. What languages and licenses are most popular and trends among licenses and languages.
3. Repository Analysis. Which are the most popular and rapidly growing repositories, and the technologies/languages associated with these repositories.
4. Most Prolific committers and reasons for committing to GitHub Repositories.
5. Subject and message similarity. Are the subject and message columns similar in the commits data frame and similarity trends across multiple programming languages.

INSIGHTS

AI has become an essential part of modern technology, transforming how developers tackle programming challenges. This shift is evident in the reduced reliance on traditional collaboration platforms like GitHub and the growing adoption of AI tools such as TuringBots.

While TuringBots significantly enhance productivity by automating tasks and optimizing workflows, they cannot fully replace human developers. These tools lack the creativity and critical thinking required for complex projects. Instead, they should be viewed as complementary assets that augment human effort. To maximize their potential, developers must balance AI integration with innovation, collaboration, and domain expertise in software development.

DATA OVERVIEW

We had access to approximately 1.36 TB of GitHub data separated as Commits, Contents, Files, Languages, and Licenses. The data was in the form of parquet files which was processed using Google Cloud Platform.

Commits - Data containing information abouts commits made to GitHub repositories. Provided information to repository names, authors, and information about the commits

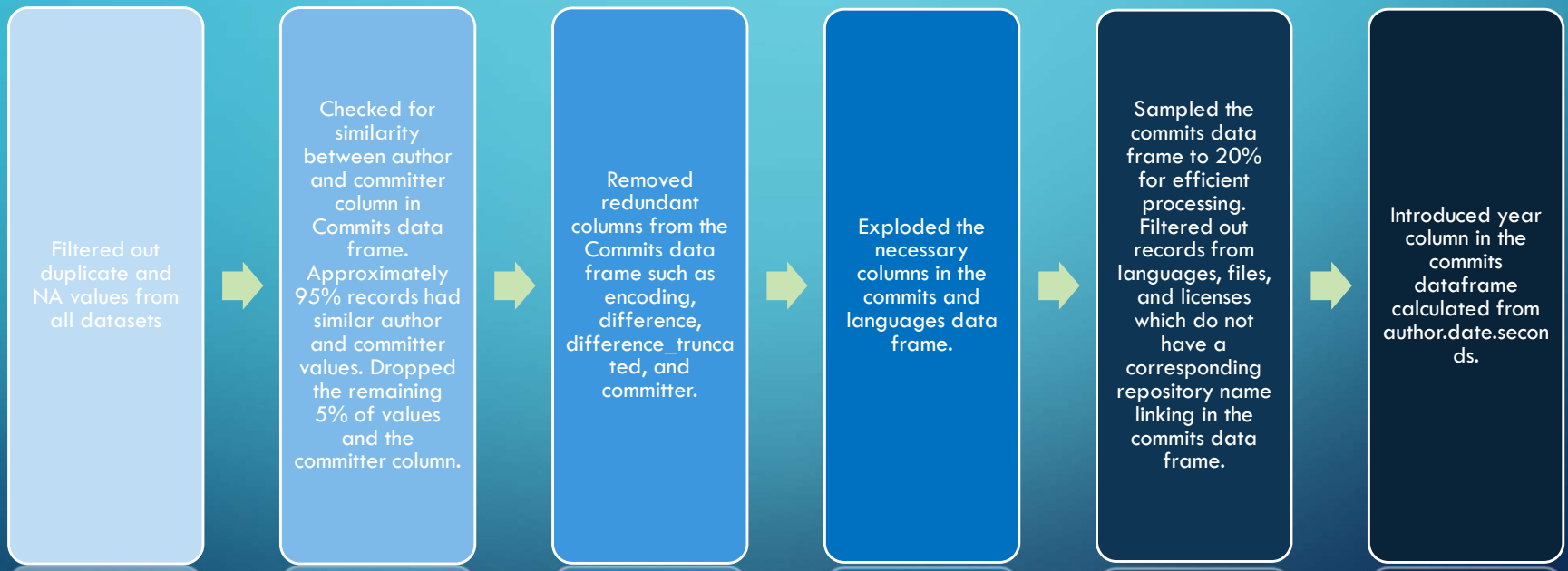
Contents - Data containing information about the contents of the GitHub repositories

Files - Information about the files present in the GitHub repositories

Languages - Data about the languages used by developers in various repositories.

Licenses – Data containing information of licenses used by GitHub Developers

DATA CLEANING PROCESS



EXPLORATORY DATA ANALYSIS

We have used the following datasets and features for our analysis:

1. Commits
    1. repo_name – array column containing repository name
    2. Author – array column containing details about the committer more importantly containing name and date columns.
    3. Subject – string column with subject
    4. Message – string column with message
    5. Year – string column derived from author.date.seconds which will be used for timeline analysis
2. Languages
    1. Repo_name – string column containing repository name
    2. Languages – array column containing details about languages and information about size of commits
3. Licenses
    1. Repo_name – string column containing repository name
    2. License – string column containing license name

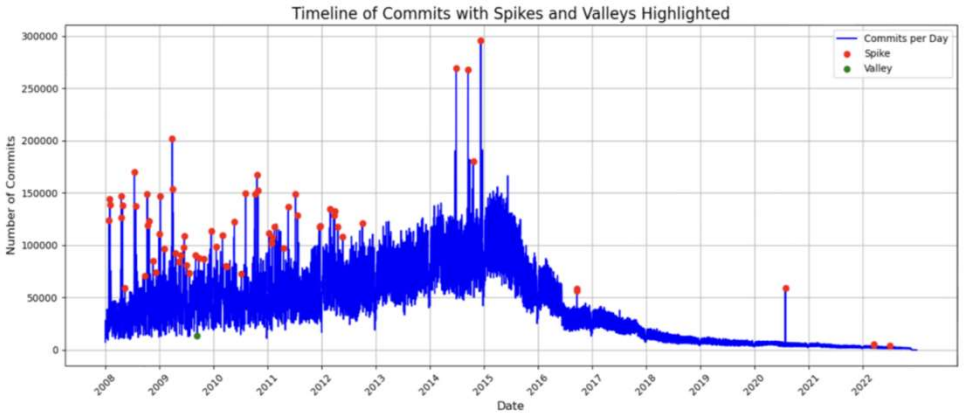
```
Commits: 265419190

Contents: 281191977

Files: 2309424945
Languages: 3325634
Licenses: 3325634
```

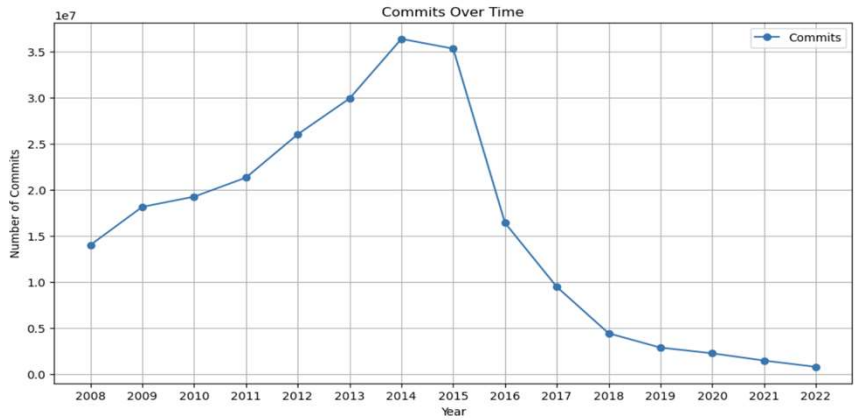
Initial count of all variables before cleaning

```
Total Commits: 346617644
Total Contents: 228154045
Total Files: 4868779
Total Languages: 7040164
Total Licenses: 2780910
```

Final count of all variables after cleaning and preprocessing

TIMELINE ANALYSIS OF COMMITS





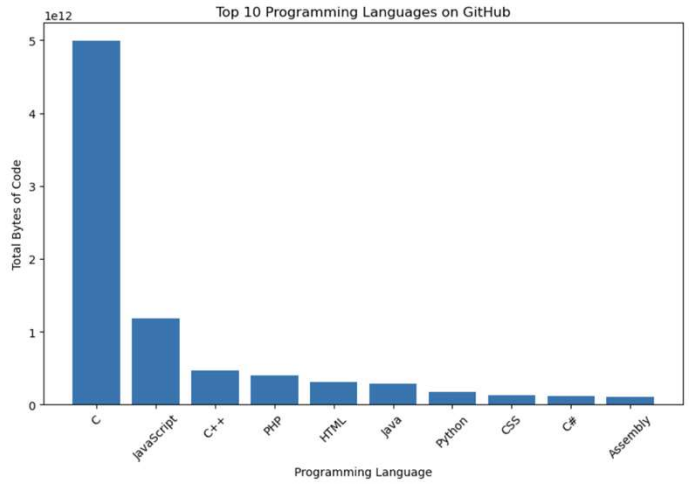
The graph displays the day-wise distribution of commits from 2008 to 2022, highlighting significant trends. We have filtered out years prior to 2008 as GitHub was founded in 2008 and also years after 2022 as there is not enough significant data for analysis. Red points represent spikes, which indicate unusually high commit activity, while green points represent valleys, indicating periods of significantly low activity.

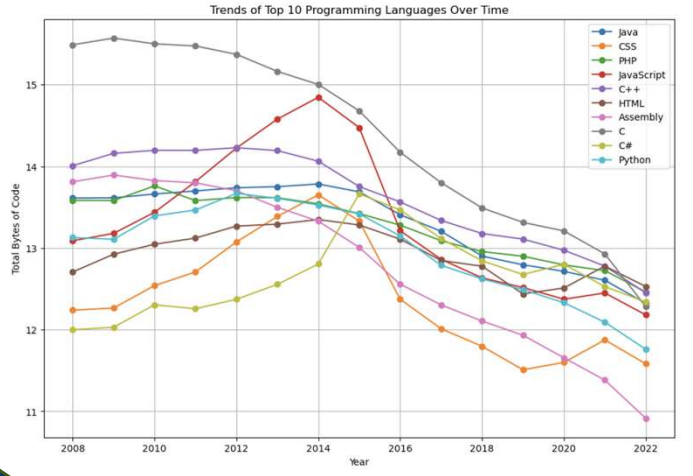
To identify spikes and valleys, a 30-day rolling window was applied to the commit data. The mean and standard deviation for each rolling period were calculated. Commits exceeding two standard deviations above the mean were classified as spikes, while commits falling below two standard deviations were classified as valleys. This method helps pinpoint periods of anomalous activity.

We have also visualized the year-wise distribution of commits for the same time period as depicted above.

The number of yearly commits shows an upward trend, peaking around 2014, 2015, and 2016. This period coincides with the surge in AI development, cloud computing, and big data technologies, fostering increased project collaboration among developers and driving innovation in new technologies.

However, commits start declining sharply from 2019 onward. This decline may be linked to the COVID-19 pandemic in 2020, which disrupted work patterns and slowed technological progress across industries.



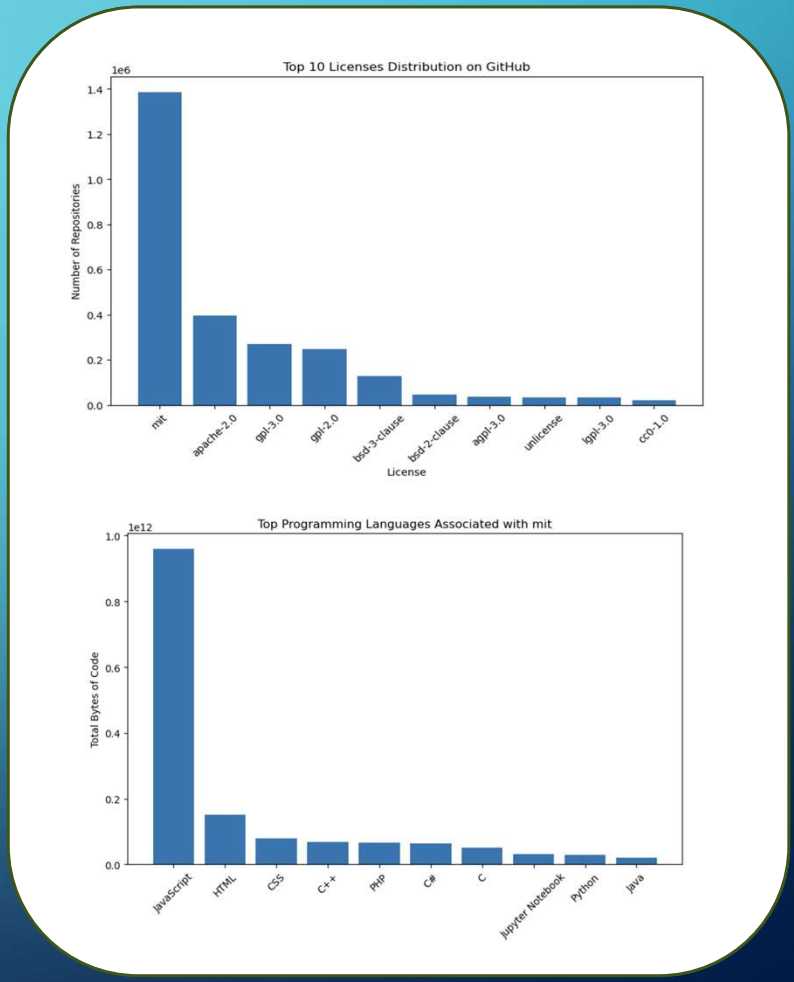


PROGRAMMING LANGUAGE TRENDS

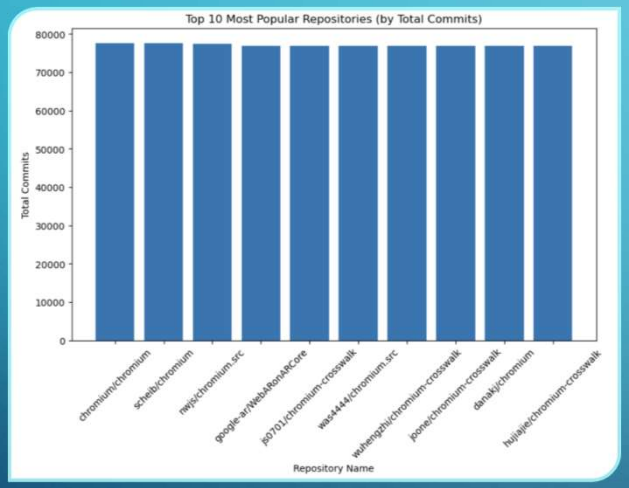
- **Grouping by Total Bytes Committed:** This analysis focused on programming language trends by aggregating the total bytes of code committed for each language across repositories, providing a measure of their relative usage and popularity. For visualization purposes, the log of the total bytes was used to effectively represent programming language trends and handle the wide range of values.
- **Top Languages Identified:** The top five programming languages, based on the total bytes of code committed to GitHub, were C, JavaScript, C++, PHP, and HTML, highlighting their dominance in the coding landscape.
- **Trends Over Time:** C emerged as the most consistently popular programming language throughout the entire analysis period, while JavaScript, C#, and CSS gained popularity steadily until they peaked around 2014–2015.
- **Post-2016 Decline:** A significant decrease in the popularity and volume of commits was observed for all programming languages after 2016, reflecting a shift in programming activity or repository contributions.

LICENSE DISTRIBUTION AND ANALYSIS

- **Analysis of Popular Licenses:** This study analyzed the most widely used licenses by examining the number of repositories each license is associated with. From this analysis, the top five most popular licenses emerged as MIT, Apache-2.0, GPL-3.0, GPL-2.0, and BSD-3-Clause.
- **License-Specific Language Trends:** A deeper exploration revealed that certain programming languages are more prevalent under specific licenses. For instance, repositories using the MIT license predominantly feature JavaScript as the leading language, followed by HTML and CSS. This suggests that the MIT license is heavily favored in web development projects.
- **Dominance of Specific Languages:** Among the top 10 licenses analyzed, half were characterized by one or two dominant programming languages, indicating a strong association between specific licenses and particular types of development activities



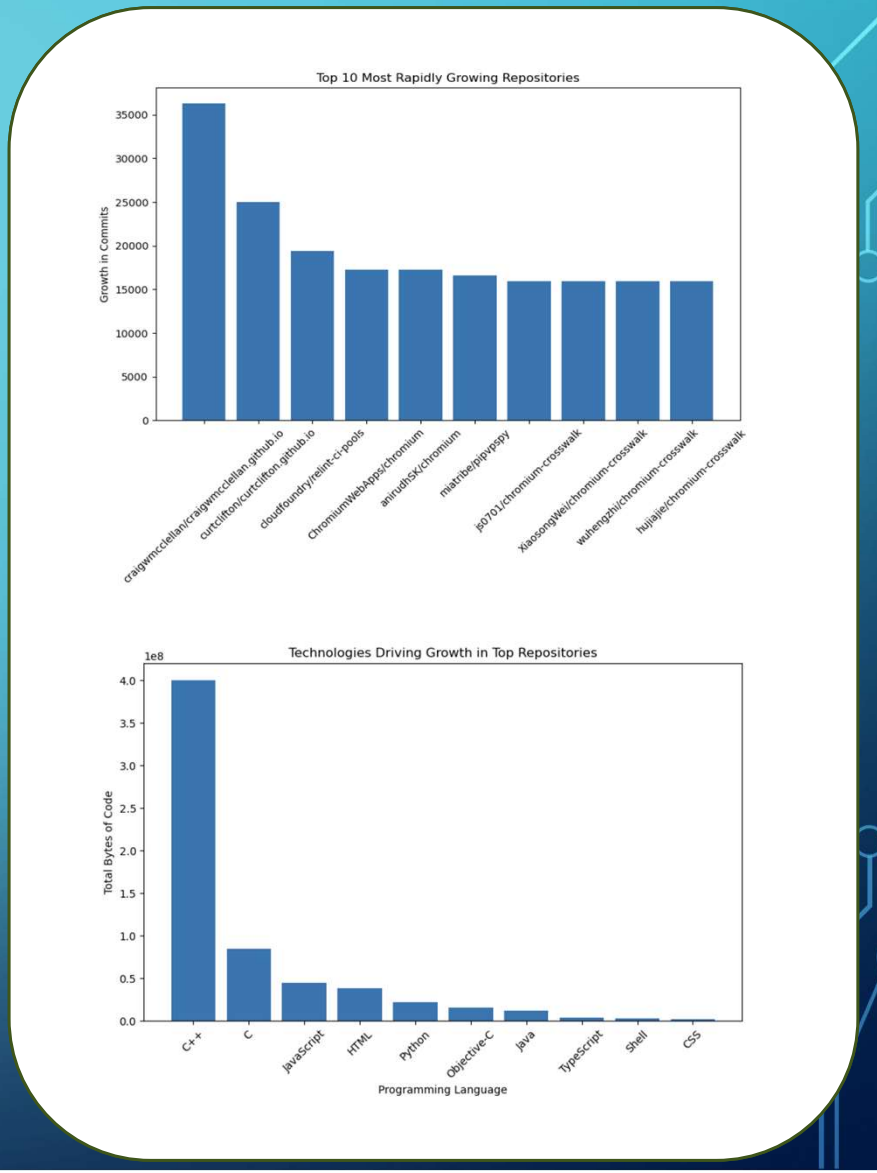
MOST POPULAR REPOSITORIES



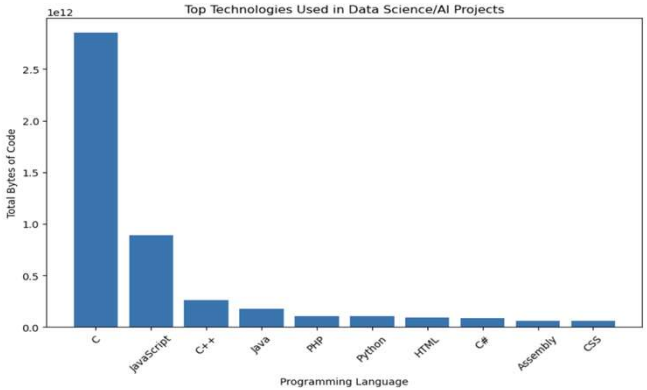
- **Analysis of Popular Repositories:** When visualizing the top 10 most popular repositories based on the number of commits, it was observed that 9 out of the 10 repositories are related to the Chromium project.
- **Understanding Chromium's Role:** Chromium is an open-source web browser project that serves as the foundation for several widely used browsers. It is primarily developed and maintained by Google, making it a pivotal codebase. Chromium underpins not only Google Chrome but also other popular browsers such as Microsoft Edge, Samsung Internet, and Opera, highlighting its extensive influence in the web development ecosystem.
- **Dominance of Google-Associated Repositories:** This analysis underscores that majority of the most popular repositories are closely associated with Google, a major player in the tech industry. The prevalence of Chromium-related repositories in this ranking reflects Google's significant contributions to open-source software development and its widespread adoption in browser technologies.

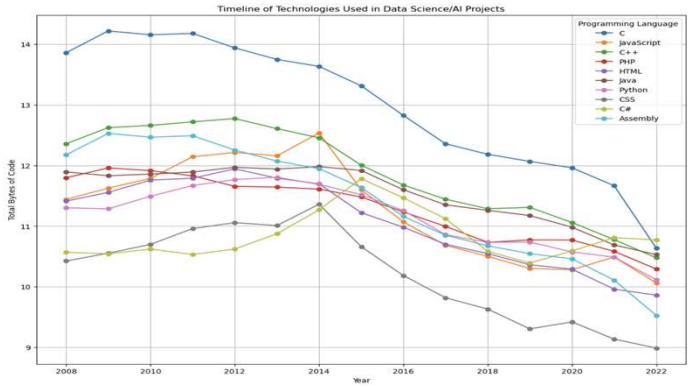
MOST RAPIDLY GROWING REPOSITORIES

- **Analysis of Rapidly Growing Repositories:** To identify the most rapidly growing repositories, we analyzed the year-over-year difference in the number of commits. The top 10 repositories with the largest growth in commits are highlighted in the accompanying visualization on the left.
- **Technology Insights:** Upon examining the technologies driving development in these repositories, it was found that C++ is the predominant programming language. This is a notable deviation from C, which remains the most popular language overall. This trend suggests that developers may prefer C++ in these repositories due to its optimization and suitability for modern development needs.
- **Association with Chromium:** Among the top 10 rapidly growing repositories, six are linked to Chromium, an open-source platform developed by Google. Chromium serves as the foundation for several major web browsers, further emphasizing its pivotal role in the open-source community.
- **Google's Dominance:** The strong presence of Chromium-related repositories in this ranking highlights the significant influence of Google in driving the growth of open-source development. This demonstrates the company's leading role in fostering innovation and adoption of cutting-edge technologies through its projects



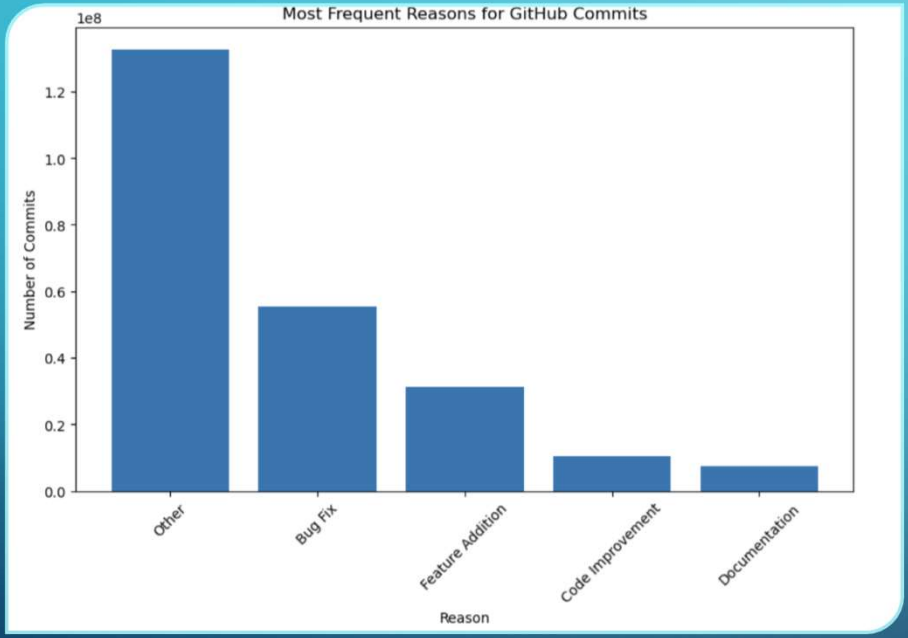
DATA SCIENCE AND AI REPOSITORIES





- **Identifying Top Repositories in Data Science and AI:** To pinpoint the leading repositories employing Data Science and AI technologies, we conducted a targeted search within the message and subject fields of commit data, using a curated list of keywords related to Data Science and AI. This approach allowed us to identify repositories where these technologies are explicitly referenced in development activity.
- **Dominance of C in Data Science and AI Repositories:** From the analysis, C emerged as the most frequently used programming language in repositories focused on Data Science and AI. This highlights its continued relevance in these fields, likely due to its efficiency and widespread use in foundational AI libraries and high-performance computing tasks.
- **Code Volume Analysis by Language:** The total bytes of code for each programming language used in top Data Science and AI repositories were analyzed and visualized on a logarithmic scale. This visualization provides a clearer comparison of code volume across programming languages and highlights the significant contributions of certain languages to these repositories.
- **Trends in Language Popularity Over Time:** When analyzing the historical trends, JavaScript and C# showed a steady rise in usage within Data Science and AI repositories, peaking around 2015. However, a sharp decline in the popularity of most languages was observed after this period, reflecting a possible shift in development priorities or methodologies. Interestingly, C#, JavaScript, and Python had a slight upward trajectory after 2019, indicating their increasing adoption for certain applications within Data Science and AI, such as enterprise solutions and specialized frameworks.

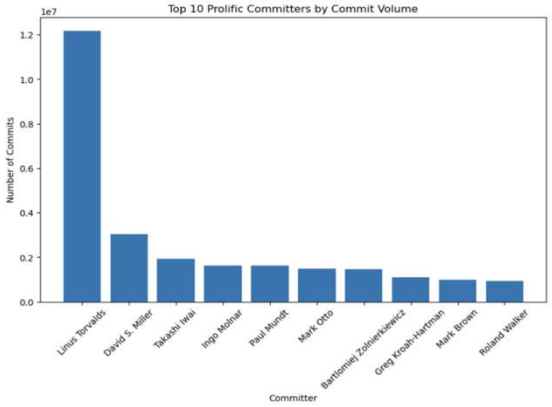
REASONS FOR COMMITTING

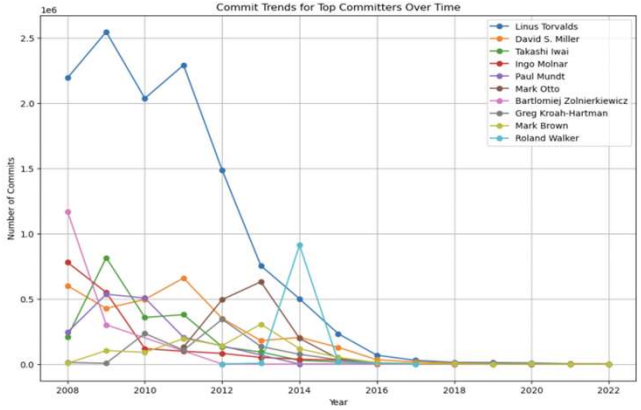


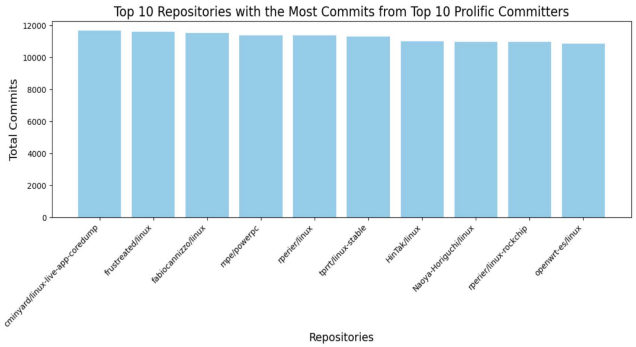
- **Objective of Commit Analysis:** To gain insights into the nature of commits made by developers to GitHub, we performed an in-depth analysis of the subject and message columns within the commits data frame. These fields were examined to identify patterns and classify the primary reasons behind commit activities.
- **Keyword-Based Categorization:** Using a keyword search, we filtered and analyzed records from the subject and message columns. While some records did not match the predefined keywords, for the sake of analytical clarity, we categorized the reasons for commits into four primary groups: Bug Fixes, Feature Additions, Code Improvements, and Documentation Updates.
- **Findings from Commit Analysis:** The results of the analysis indicate that the most frequent reason for commits is Bug Fixes, highlighting the critical role of maintaining software reliability. This is followed by Feature Additions, reflecting the continuous enhancement of functionalities, and Code Improvements, which aim to optimize existing implementations. Documentation Updates accounted for the smallest proportion of commits, suggesting a relatively lower focus on documentation in comparison to other development activities.

This categorization provides a structured understanding of developer activity on GitHub, enabling a clear view of the priorities and focus areas within repository development.

MOST PROLIFIC COMITTERS



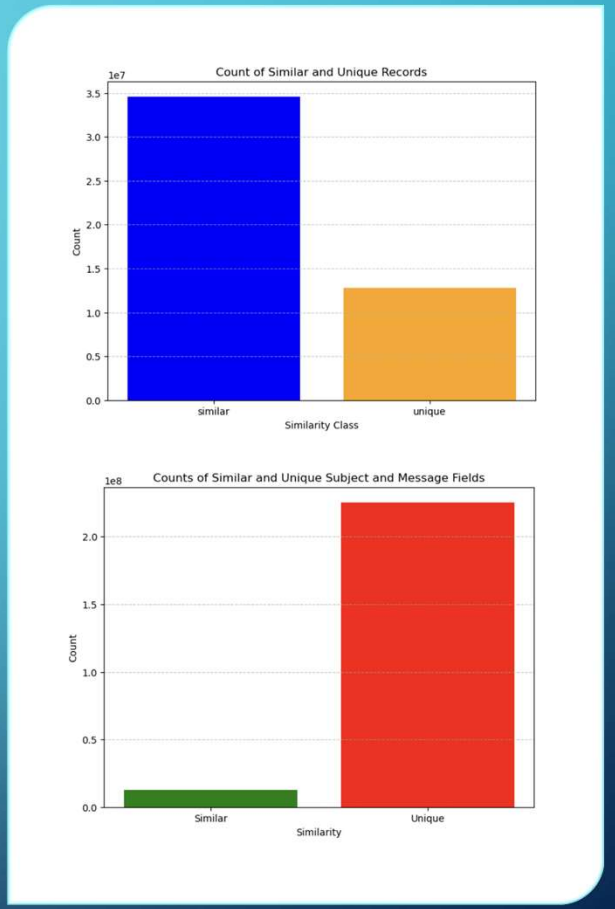




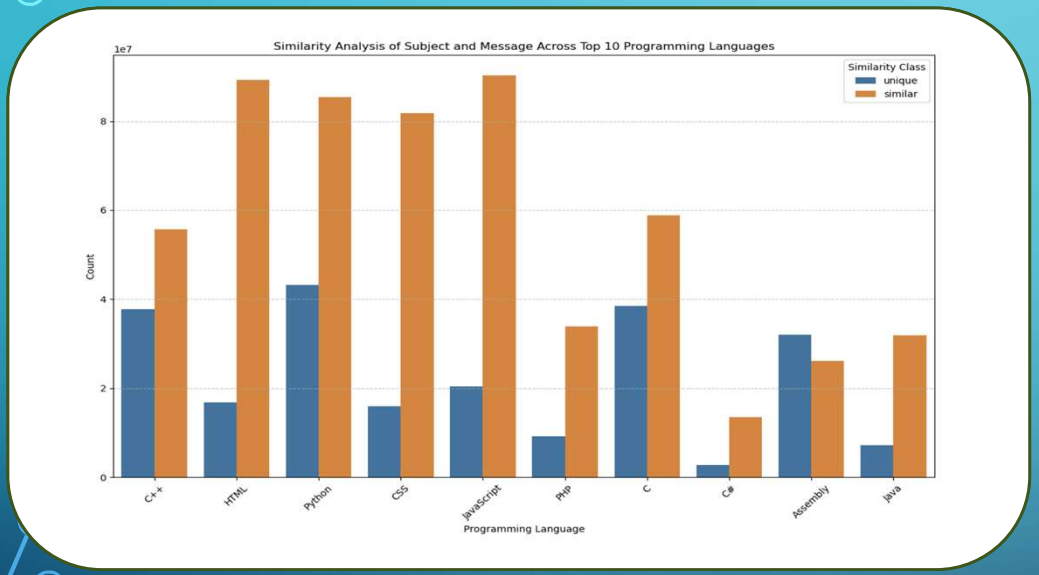
- **Analysis of Prolific Committers:** This analysis focused on identifying the most prolific and frequent committers by evaluating the number of unique commits each individual contributed to GitHub repositories. This approach provided insights into the activity levels and consistency of developers over time.
- **Top Committers Identified:** Based on the total number of commits, the top five committers were identified as Linus Torvalds, David S. Miller, Takashi Wai, Ingo Molnar, and Paul Mundt. These individuals consistently demonstrated high levels of activity, contributing significantly to GitHub repositories.
- **Temporal Trends in Commit Activity:** The ranking of the most prolific committers exhibited notable variations over time. For instance, in 2014, Roland Walker emerged as the top committer, surpassing all other developers in terms of contributions. However, this was an exception to the general trend.
- The repositories with the most commits from the top 10 prolific committers are heavily associated with large-scale and widely-used projects, such as those related to Linux kernel development and open-source system software, reflecting the significant impact and focus of these contributors on foundational, high-demand technologies.

SUBJECT AND MESSAGE ANALYSIS

- **Overview of Subject and Message Similarity**: The subject and message columns in the commits data frame exhibit noticeable similarities. To quantify this, we applied cosine similarity analysis, a widely used technique in natural language processing to measure the similarity between two text vectors.
- **Cosine Similarity Analysis Methodology**: The text data from the subject and message columns were transformed into numerical vectors using vectorization techniques. Cosine similarity was then calculated between these vectors to determine how closely related the two columns are. Records with a cosine similarity score of 0.7 or higher were classified as duplicates, while scores below 0.7 were categorized as unique. This approach enables the identification of subtle overlaps or patterns in textual data that might not be immediately apparent.
- **Findings on Similarity**: The graph in the top-right corner illustrates the distribution of rows in the commits data frame where subject and message columns are either duplicate or unique. From the analysis, it was observed that approximately 70% of all records have high similarity, indicating that a significant portion of the subject and message content overlaps, though not identically.
- **Analysis of Exact Matches**: To investigate whether developers are directly copying and pasting the subject and message contents, we calculated the count of records where the two columns are exactly identical. As shown in the bottom-right graph, only 4% of the records are exact matches, suggesting that while the subject and message columns share a high degree of similarity, they often differ slightly, likely due to minor variations or elaborations in the message content. This highlights that while there is a strong resemblance, developers are not merely duplicating content but making small adjustments to one or both fields.



SUBJECT AND MESSAGE ANALYSIS ACROSS PROGRAMMING LANGUAGES



- **Subject and Message Duplication Analysis:** This study examined the extent of duplication between subject and message records across different programming languages, providing insights into patterns of redundancy.
- **High Duplication in Web Development Languages:** Web development languages like HTML, CSS, JavaScript, and PHP showed the highest similarity between subject and message records. This is likely because these records often include documentation details or repetitive information that is directly copied.
- **Low Duplication in Object-Oriented Languages:** In contrast, object-oriented languages such as C++, Python, and C exhibited the least amount of duplication between subject and message records, suggesting more distinct and deliberate structuring of these elements in such repositories.

CONCLUSION

This project provided an in-depth analysis of GitHub repositories to explore whether AI-powered tools like TuringBots could replace human software developers. By examining five key datasets i.e., commits, files, licenses, languages, and contents, we uncovered valuable insights into the patterns, technologies, and practices shaping the open-source ecosystem. The timeline analysis revealed distinct peaks and valleys in contribution trends, indicating seasonal or event-driven spikes in activity. Additionally, while C remained the most popular language, the most rapidly growing repositories demonstrated the dominance of C++, Python, and JavaScript, highlighting where AI-powered development tools could have the most impact. The study of licenses showed a strong preference for permissive licenses like MIT and Apache, suggesting the importance of AI tools that respect these licensing requirements.

The similarity analysis of commit messages and subjects identified areas where redundancy and repetitive tasks occur, providing actionable insights for developing AI tools that automate these processes. Across the top programming languages, we observed varying levels of uniqueness in commit activity, suggesting opportunities for AI to optimize workflows in specific communities. By combining these findings, this project not only sheds light on the current state of open-source development but also provides a roadmap for how AI tools like TuringBots can augment human developers by increasing efficiency, enhancing productivity, and driving innovation in the software development landscape.

RECOMMENDATIONS

With the rise of AI tools, it is inevitable that TuringBots will play a significant role in assisting developers with programming tasks. However, there are several proactive steps we can take to ensure this collaboration is both effective and beneficial.

Embrace AI as a Collaborative Tool: Leverage AI tools like TuringBots to enhance productivity and automate repetitive tasks, while retaining human oversight for creativity and critical problem-solving.

Invest in Developer-AI Training: Equip developers with skills to effectively collaborate with AI tools, ensuring seamless integration into workflows.

Promote Collaboration: Encourage human collaboration alongside AI to address complex challenges that require creativity and domain expertise.

Focus on Ethical AI Practices: Establish guidelines for responsible AI use, ensuring transparency and fairness in development processes.

Monitor Trends and Adapt: Continuously evaluate the evolving role of AI in software development to stay ahead of industry changes and maintain a balanced developer-AI ecosystem.

Implementing these recommendations will not only accelerate the adoption of AI-powered tools like TuringBots but also foster collaboration, innovation, and productivity in the software development ecosystem.

THANK YOU!