Big Data Platforms Final Project

Will TuringBots replace human software developers?

Anubuthi Kottapalli

BIG DATA AND CLOUD COMPUTING

EXECUTIVE SUMMARY

1. The primary objective of this analysis was to explore key trends in GitHub repositories and commits, identify the most popular programming languages, understand the license distribution, and investigate the factors influencing repository activity.

2. Commit Trends: Spikes in activity align with key milestones like Git's release (2005), GitHub's launch (2008), and the COVID-19 pandemic (2020). Most commits are driven by feature development.
3. Language Popularity: JavaScript remains the most dominant language, with the permissive MIT license widely preferred for collaboration.
4. Repository Leaders: Linux and Google drive growth through innovation in open-source platforms like TensorFlow and Kubernetes.
5. AI's Impact: Tools like GitHub Copilot enhance productivity, focusing on feature development, but human developers remain essential.
6. Recommendations: Leverage AI and automation for repetitive tasks and improve commit documentation

Methodology and Source Data Overview

- Data Source:
  - GitHub Archive data from Google Cloud Storage (~1.36 TiB).
  - Includes commit history, programming languages, files, and licenses.
- Data Structure:
  - Commits: Commit metadata (author, date, message).
  - Languages: Language usage by repository.
  - Licenses: License types for repositories.
  - Files: File metadata (paths, modes).
  - Contents : content of the files in the repositories
- Key Variables:
  - committer_name, commit_year, commit_message, license, language_name.

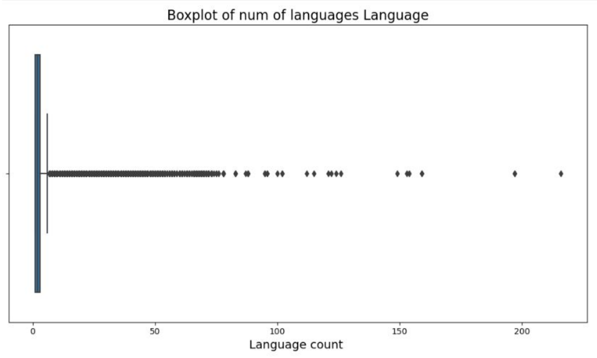
- Data Preprocessing:
  - Cleaned data by removing duplicates and irrelevant entries.
  - Handled missing data and text preprocessing.
- Analysis Approach:
  - EDA: Identify key variables and trends.
  - Time Series: Analyze commit trends and spikes.
  - Text Similarity: Measure commit message duplication.
- Tools & Technologies:
  - Apache Spark for data processing.
  - Python (matplotlib, pandas) for visualization.

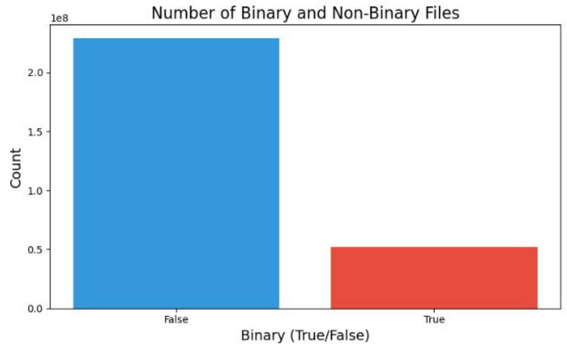
Data Cleaning and EDA (1)

Commits Languages Contents

- Pre processed the data and retained on required columns
- In the Commits folder The dates inputted by the committer started before 2008 and went on after 2023 .
  - Deleted data after 2023
  - Retained values before 2008 because github allows committers to input a specific commit date
- In the Languages folder there were a very small number of of repositories with very large number of languages and Deleted the outlier based IQR method .
- In the Contents folder there were a very small number of of repositories with very large number of copies of files , deleted the outliers based on IQR
- Plotted a graph to visualize the distribution of types of files (Binary code or script)

  The yaxis represent the number of repositories that contain binary files

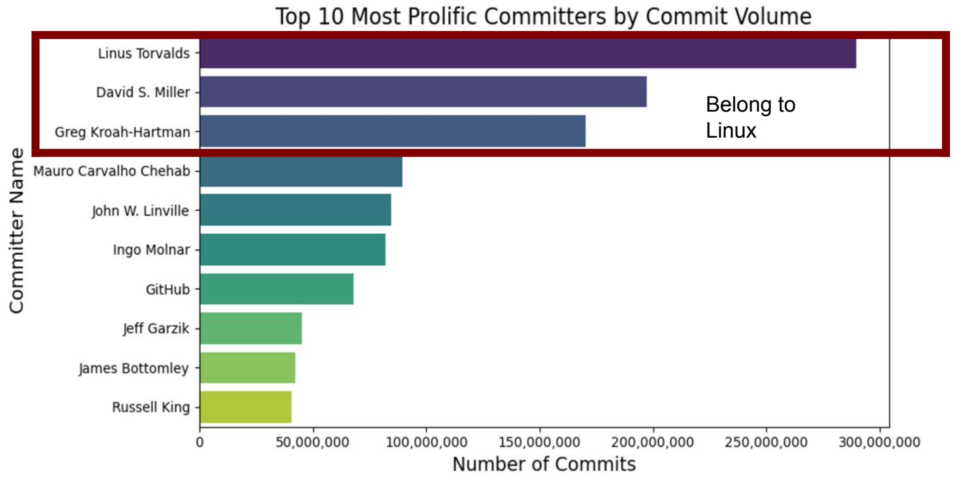




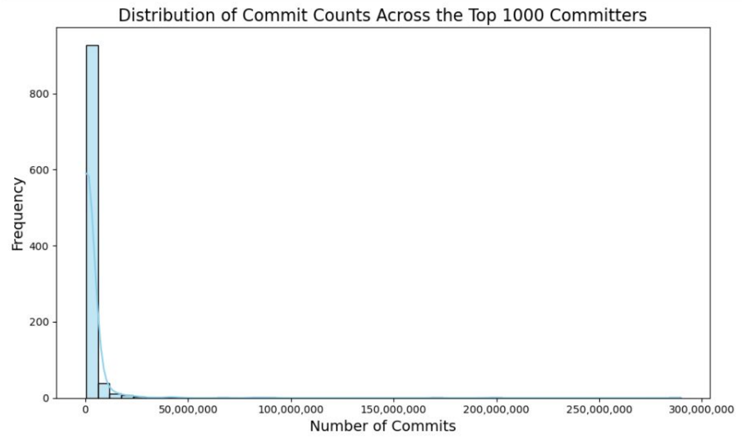
Data Cleaning and EDA (2)

Commits - Understanding the Committers

- The most prolific committers are:
  - Linus Benedict Torvalds is a Finnish and American software engineer who is the creator and lead developer of the Linux kernel
  - David S Milller an American software developer working on the Linux kernel,
  - Greg Kroah Hartman is a major Linux kernel developer



- By visualizing the distribution of these committers we can see that
  - It is a long tail distribution
  - Majority committers contributing a relatively small number of commits, while a few outliers have made an extraordinarily high number of contributions.



Timeline Analysis

Commit Activities over Time

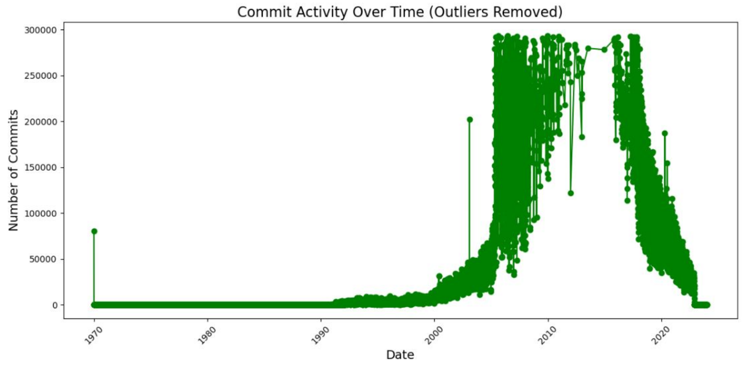
1. There are no evident gaps in the data , but there were some obvious outliers like dates >2023 .

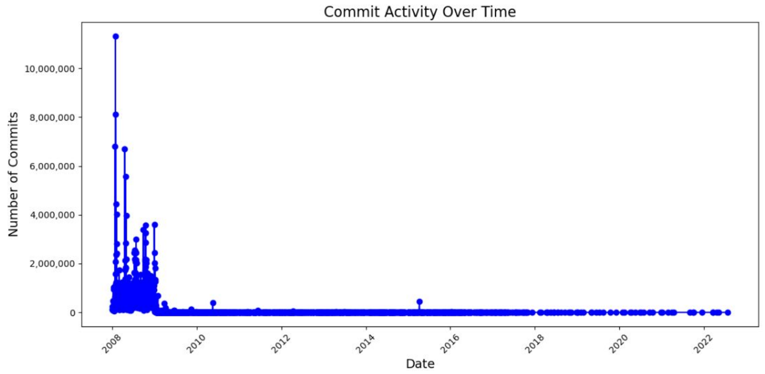
2. The data does have significant peaks they can be interpreted as follows:
   a. **Around 1970**: This could be due to a technical artifact, as timestamps in computing often default to Unix epoch time (`January 1, 1970`) when no valid timestamp is provided.

   b. **Late 1990s to early 2000s**::
      i. dot-com boom (1997–2000), The release of open-source platforms and tools
      ii. These could have been migrated to git later on.

   c. **Post-2005**: The consistent and significant activity in this period aligns with
      i. The release of Git (2005): Git revolutionized version control
      ii. The launch of GitHub (2008): The platform greatly popularized Git and provided an accessible way to share and collaborate on projects.

   d. **2010–2020**: The large spike and sustained activity during this period align with:
      i. Cloud adoption: Major cloud providers like AWS, Google Cloud, and Azure.
      ii. Rise of open-source projects
      iii. COVID-19 pandemic (2020): A potential peak in remote work during this time might have driven more digital collaboration and software contributions.



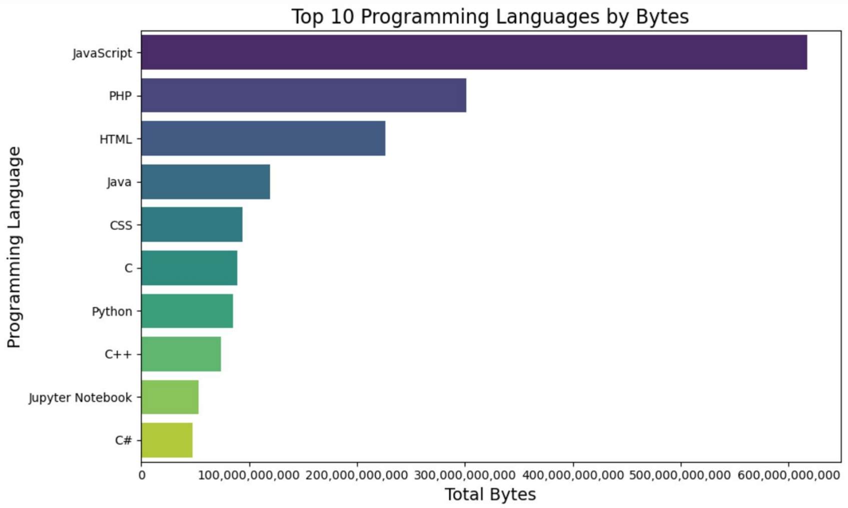


GIT HUB WAS LAUNCHED IN 2008

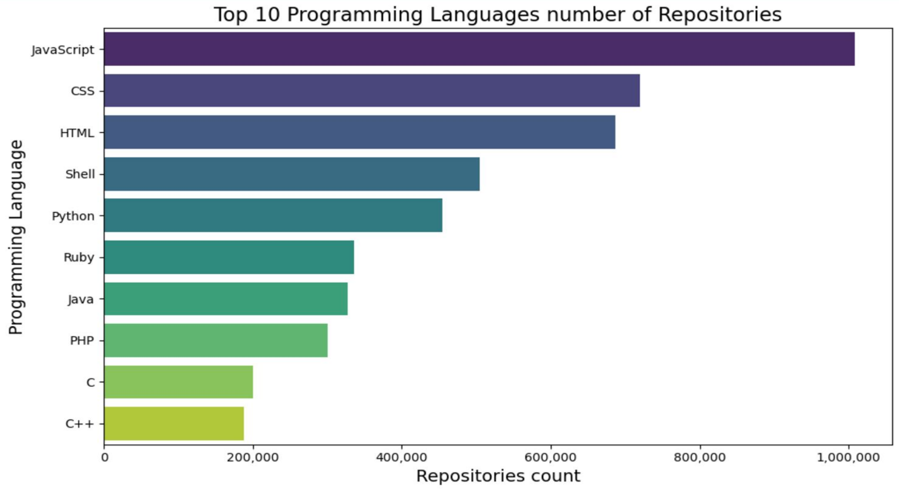
Timeline Analysis

Language Popularity over Time

The top 10 most popular languages by total Bytes



The top 10 most popular languages by number of repositories

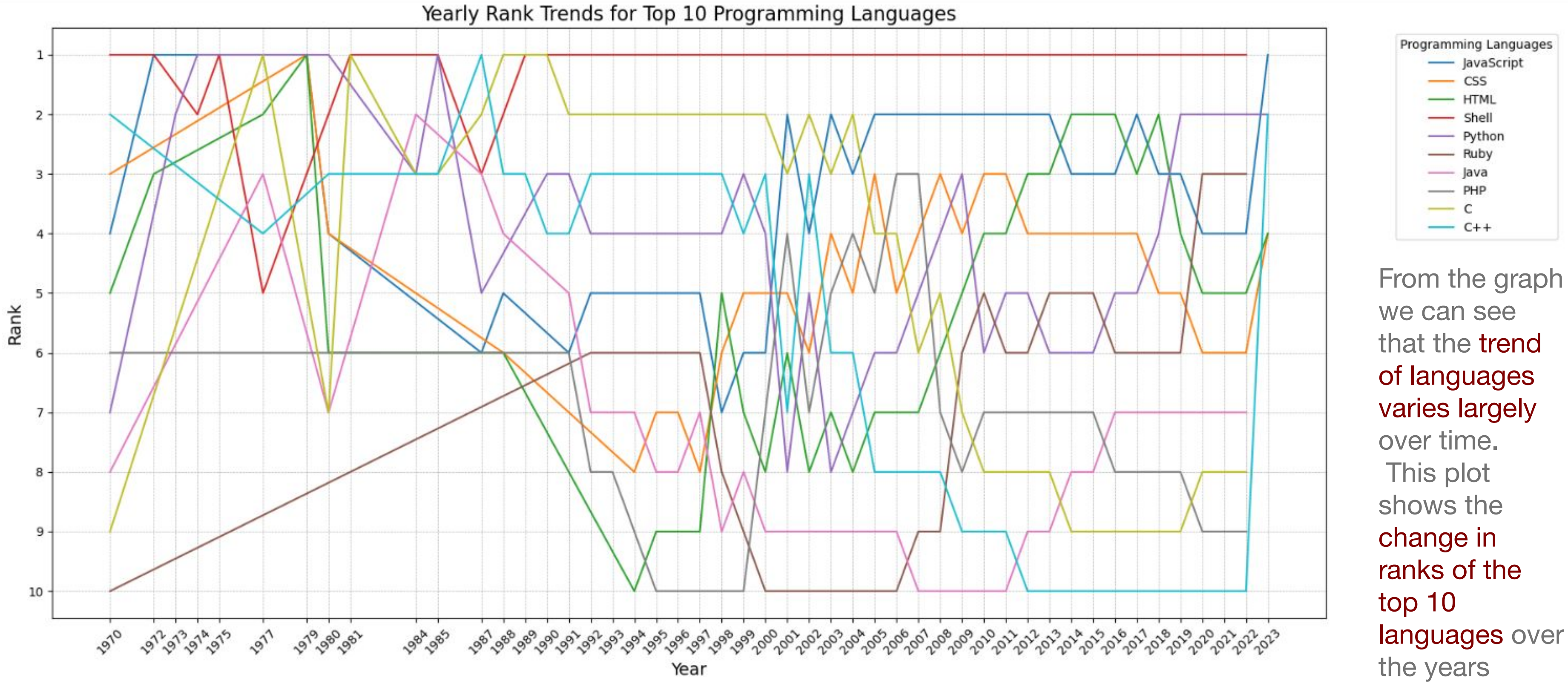


1. From both the graphs we can see that Web Development technologies are the most popular Languages
2. This highlights JavaScript's versatility and widespread adoption across varied project scales.

THE UNIVERSITY OF CHICAGO

Timeline Analysis

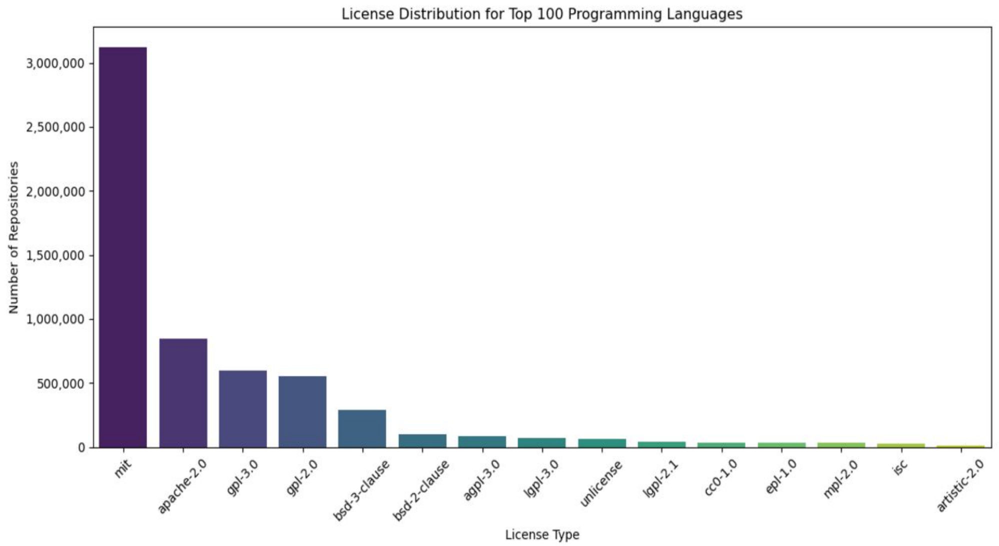
Language Popularity over Time

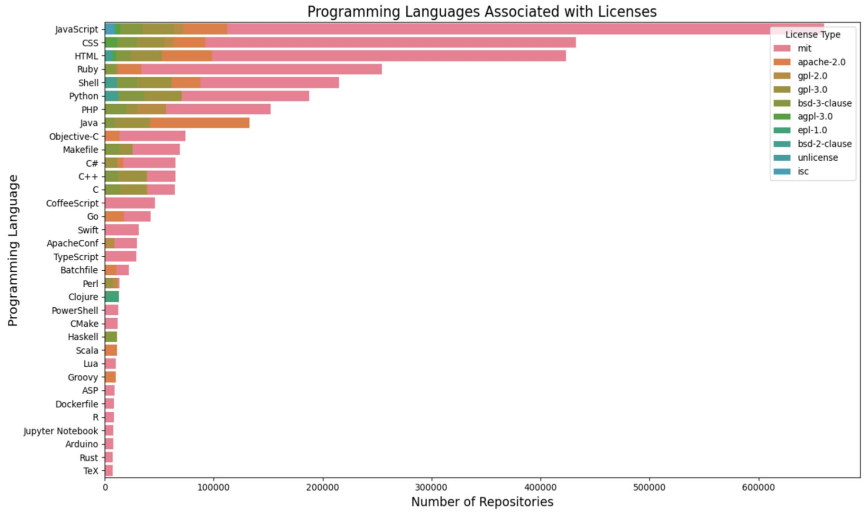


From the graph we can see that the trend of languages varies largely over time.

This plot shows the change in ranks of the top 10 languages over the years

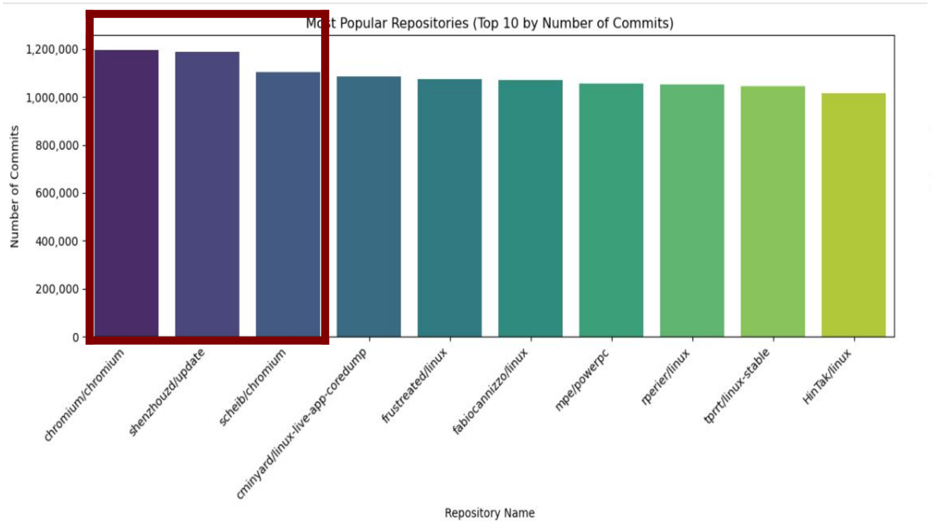
Programming Language and License Analysis

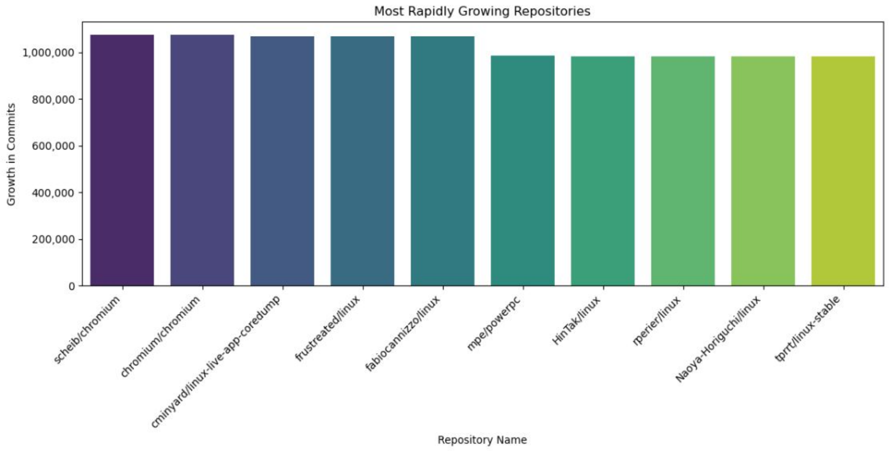




1. **MIT License Popularity**: The widespread adoption of the permissive MIT license highlights the open-source community's preference for licenses that promote collaboration and reuse .
2. **Language Flexibility**: The diversity of licenses for languages like Java and C++ suggests their versatility in both open-source and proprietary projects.
3. **Open-Source Ecosystem**: The dominance of licenses like MIT, Apache-2.0, and GPL reflects the foundational role of open-source principles in the programming ecosystem, fostering innovation and sharing.
4. **Diverse vs. Niche Licensing**: C++ and Java show diverse licensing, while niche languages like CoffeeScript and Go tend to align with single license types.

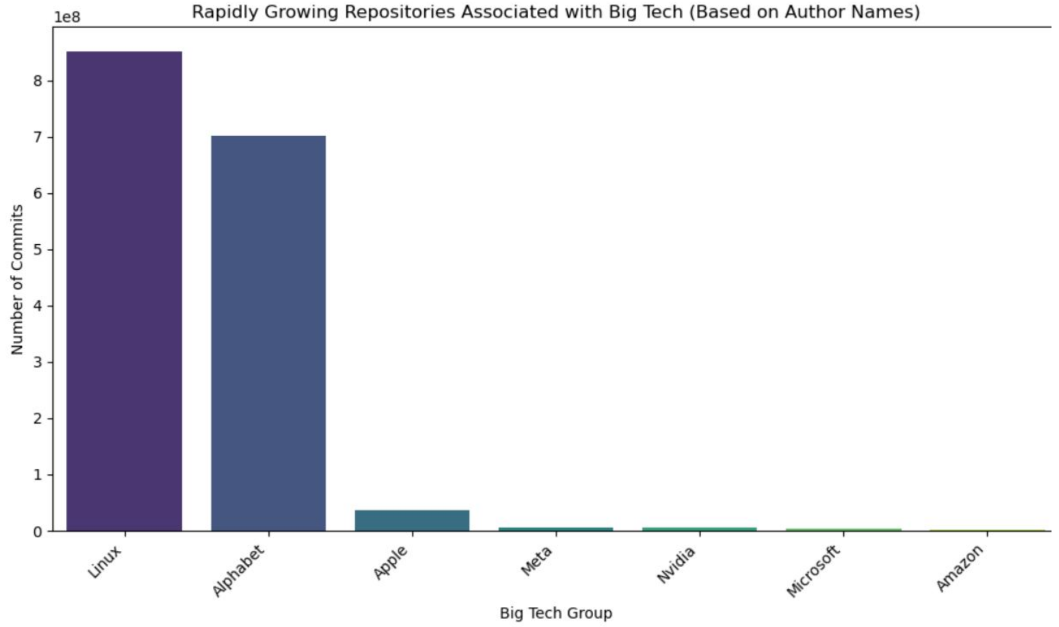
Most popular technology and repositories analysis





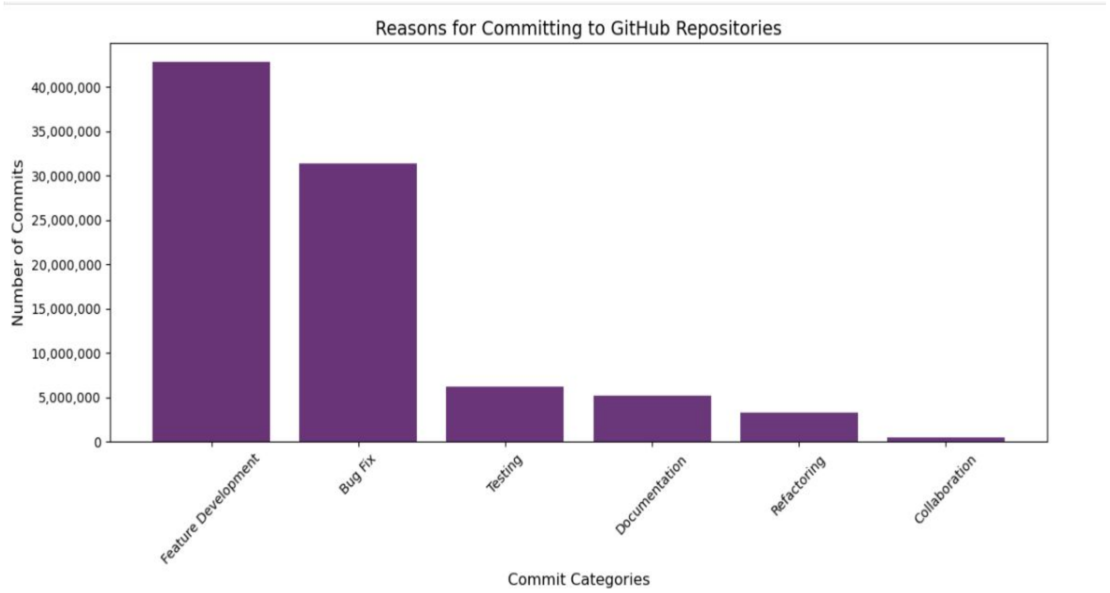
The top 3 most popular repositories are open source.
Widespread adoption: Their popularity suggests that they serve significant user bases, and their open-source nature allows for wide-scale improvements and bug fixes from contributors worldwide. Which is what they are also most rapidly growing
Community-driven development: The high activity around these repositories indicates robust community involvement, which is often seen in open-source projects with large, active developer communities.

Most popular technology and repositories analysis



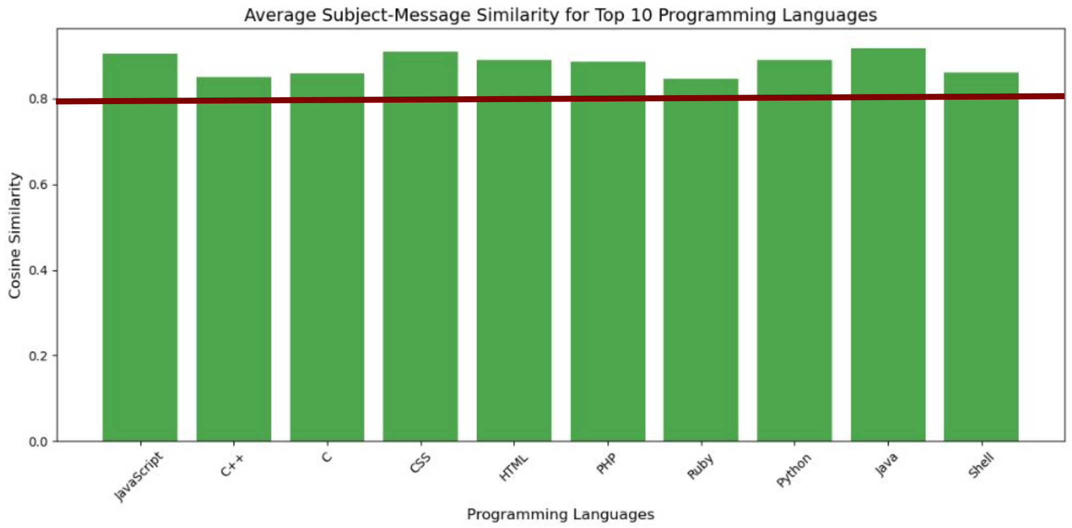
1. **Linux and Alphabet (Google)** are key players in the open-source ecosystem, driving the growth of repositories through constant contributions and technological breakthroughs.
2. **Linux**'s dominance in server infrastructure, cloud computing, and mobile (via Android) has led to its rapid adoption and ongoing development.
3. **Alphabet (Google)** is pushing the envelope in **cloud** and **AI** technologies, with significant contributions from open-source projects like **TensorFlow** and **Kubernetes**.The **3 most popular repositories belong to Google.**
4. **Big Tech** companies are increasingly adopting open-source to accelerate innovation, enhance scalability, and foster cross-industry collaboration.

Reasons for Commiting to Git



- **Focus on Innovation and Stability**: Most commits are driven by new feature development, followed by bug fixes, indicating a strong focus on growth and maintaining software stability.
- **Quality Assurance**: Testing ranks third, showing an emphasis on ensuring that new features and bug fixes meet high-quality standards before integration.
- **Supportive Role of Documentation**: While documentation is essential, it is less frequent compared to development and bug fixing, suggesting that technical improvements take priority over updating project documentation.

Similarity between Subject and Message

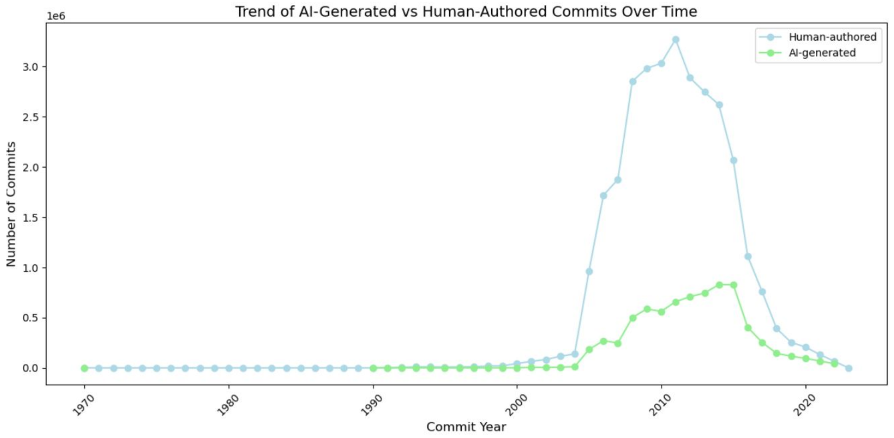


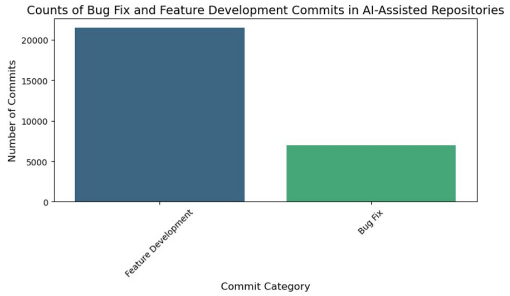
- The similarity between subject and message does not depend on the languages in the repository,

- **High Similarity Between Subject and Message**: A cosine similarity greater than 0.8 across most repositories indicates that the commit subject and message are often highly aligned, **suggesting that developers tend to repeat or copy-paste similar content in both fields.**

- While this could be seen as an efficient way to document commits, it may also reflect a lack of detailed, individualized explanations for the changes, potentially reducing the clarity of commit records.

CONCLUSIONS: AI BOTS AND RECENT TRENDS

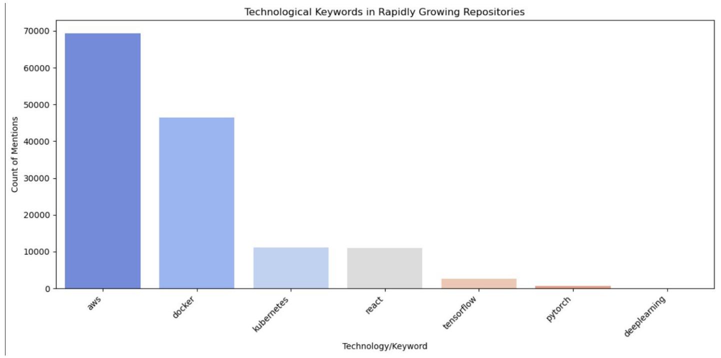




- AI/Bot generated commits increased significantly around the late 2000s and peaked around 2010-2015. However, there has been a notable decline post-2015.
  - **Possible Reason**: The decline could reflect changes in how AI tools are utilized, such as the rise of automated pipelines and fewer, higher-quality commits per contribution. It might also result from dataset limitations or a shift in industry practices.
- feature development commits far exceed bug fix commits in AI-assisted repositories.
  - **Implication:** This suggests that AI tools may enhance productivity by enabling developers to focus on creating new features rather than fixing bugs. It also points to improved initial code quality due to AI assistance.
- From the trends we can see that even though there is an increase in trend of Bot generated Commits , human generated commits is still comparatively large

Conclusions

- AI-driven tools like GitHub Copilot, Code Llama, and others are significantly influencing development practices, contributing to a growing number of commits made with AI assistance.
- These tools have made meaningful improvements in developers' productivity, possibly due to their ability to generate more efficient code, reduce bug rates, and increase feature development speed.
- The trend of increased AI-related commits reflects the broader shift toward automated software development processes.



- The key AI technological trends in the growing repositories are AWS and Docker which are foundational technologies for building and scaling AI solutions. Their prominence suggests a strong focus on cloud-native and containerized environments for deploying AI workloads.

THE UNIVERSITY OF CHICAGO

Conclusions

1. Dominance of JavaScript: JavaScript remains the most popular programming language on GitHub, heavily associated with the permissive MIT license. This indicates its wide adoption for web development and open-source contributions.
2. License Preferences Reflect Ecosystem Needs:The MIT license is the most frequently used, aligning with the open-source ethos of simplicity and permissiveness. Other licenses, such as Apache-2.0 and GPL
3. Popular Reasons for Commits:The majority of commits focus on feature development, followed by bug fixes, testing, and documentation. This highlights the prioritization of innovation over maintenance tasks in open-source projects.
4. Most Growing Repositories:Big Tech companies like Alphabet (Google) and Linux Foundation drive the most rapidly growing repositories, reflecting their role in releasing cutting-edge technologies and fostering open-source collaboration.
5. Text Similarity in Commit Messages:High similarity between commit subjects and messages indicates that many committers reuse text, likely to save time, but also suggests opportunities for better commit documentation practices

Recommendations

1. Leverage AI for Quality and Innovation:
   a. Promote the adoption of AI tools for creating robust initial code, enabling developers to prioritize feature development.
   b. Train teams to maximize the potential of AI-assisted tools for productivity and innovation.
2. Enhance Automation Strategies:
   a. Integrate bots for repetitive tasks such as testing, dependency updates, and documentation to streamline workflows and minimize human error.
   b. Continuously optimize automation tools to align with modern development practices.
3. Investigate the post-2015 decline in AI-assisted commits to identify gaps or opportunities for adopting modern AI tools.

Will TuringBots replace human software developers?

TuringBots and AI-assisted tools can significantly enhance developer productivity by automating repetitive tasks and aiding in code generation, but they are unlikely to fully replace human software developers. Human developers are still essential for creative problem-solving, strategic decision-making, and understanding complex requirements that go beyond the capabilities of AI.

THANK YOU