03/15/2025

Will TuringBots replace software developers?

Nidhi Pareddy

Executive Summary

This project examines whether advanced AI-driven coding assistants (TuringBots) can meaningfully augment or potentially replace human software developers. Analyzing GitHub commit data provides insights into developer behaviors, trends in technology use, and the feasibility of AI-driven automation. Through thorough analysis, it was found that:

- GitHub commit activity has dramatically increased from 2005 to 2012, correlating with GitHub's founding, rise in open-source culture, and widespread adoption of Git.
- JavaScript leads among popular programming languages due to its versatility, but emerging AI-specific languages like Python have recently gained rapid traction.
- Most repositories seeing significant growth are associated with large tech firms, suggesting corporate sponsorship accelerates open-source adoption and commit activity.
- AI/Data Science projects frequently involve buzzwords like "Machine Learning" and "Big Data," with newer terms like "Generative AI" indicating ongoing technological advancements.
- Common reasons for commits primarily include bug fixing and feature addition, suggesting that software development remains focused on iterative improvement and functionality enhancements.
- Although most commit messages are unique, indicating individual developer input, consistent commit styles in some communities suggest areas where automation through AI assistants might feasibly replace human contributions.

Methodology

The dataset used consists of upwards of 1.36 TiB of GitHub data containing 5 different dataframes: software developers commits to repositories, its contents, the files in each repository, the coding language used, and the license.

Data Sources:

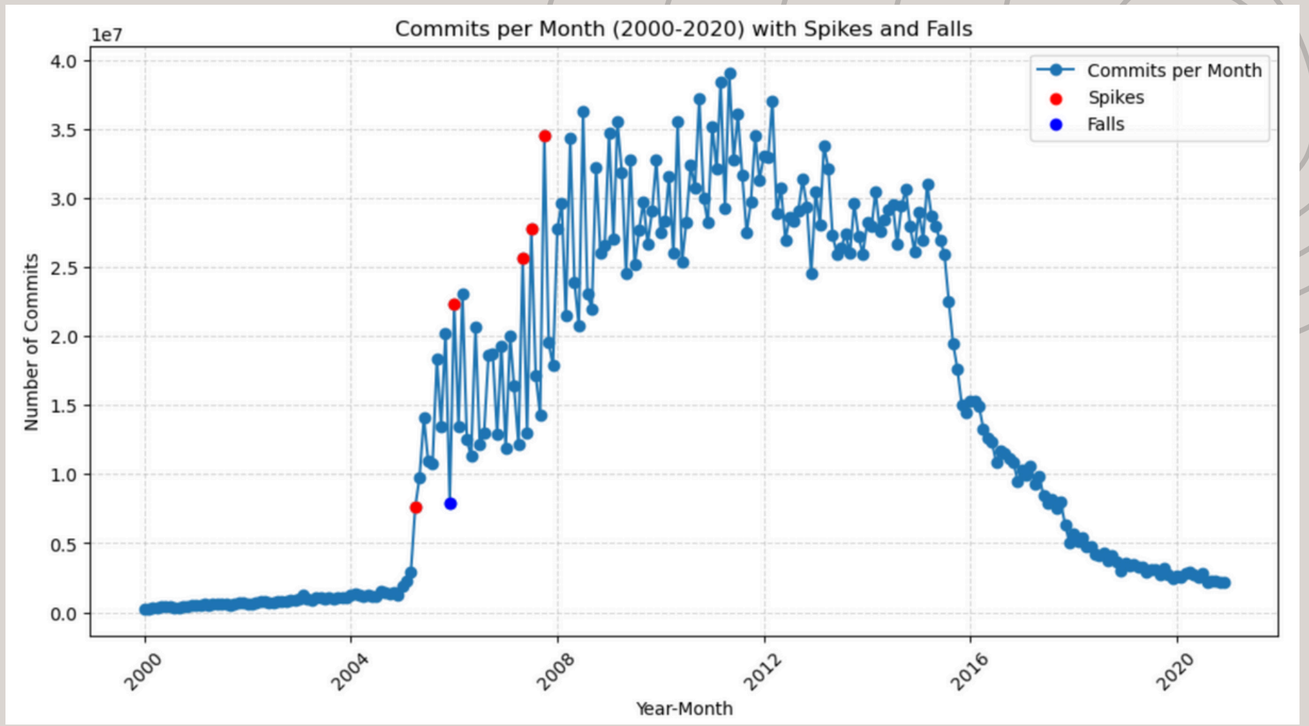
- GitHub programming languages information via GitHub API.
- Open-source licenses identified through GitHub's SPDX detection.
- Git commits data from open-source repositories grouped by repository.
- File metadata and contents (text files under 1 MiB) at the HEAD branch.

EDA and Data Cleaning:

- Removed rows with NULL or NA values to ensure completeness and quality of analysis.
- Took a 5% sample of the commit dataframe to represent the big data.
- Retained only distinct and matching repository names across datasets by linking dataframes, ensuring accurate and meaningful joins.

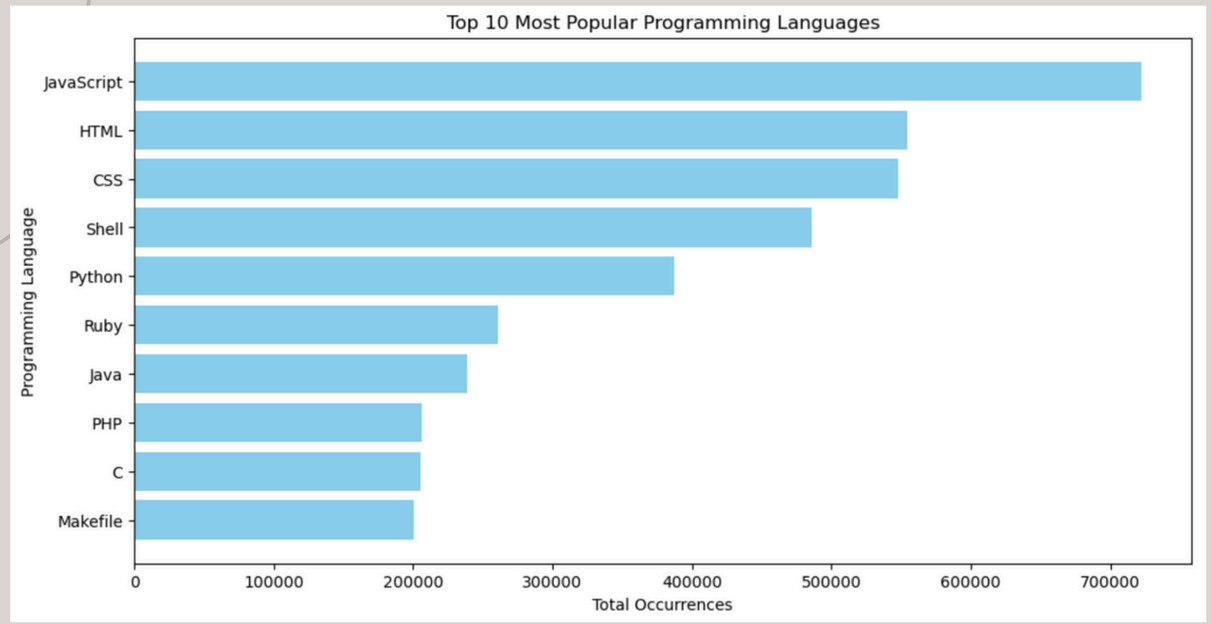
Timeline Analysis

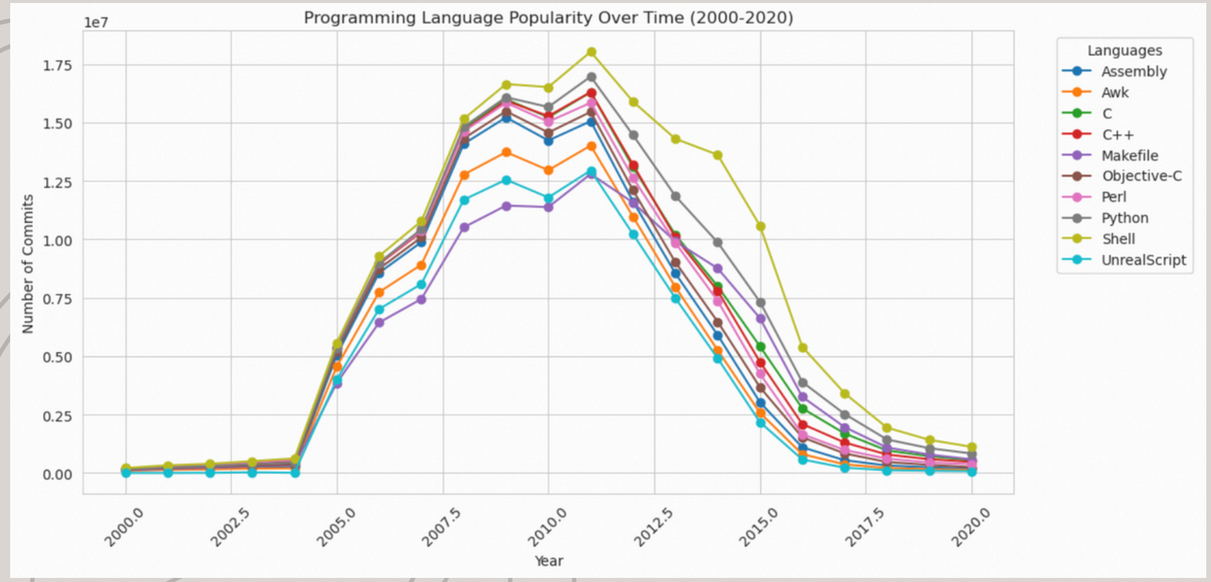
- Analyzed GitHub commit data from 2000 to 2020, initially using a representative 5% sample to manage data scale.
- Commit counts aggregated monthly and yearly to clearly identify overall activity trends and reduce data noise.
- Defined clear criteria—"spikes" as monthly commit increases of at least 100%, and "falls" as decreases by 50% or more—to highlight significant shifts or noteworthy events.
- Observed a steep increase in commits from 2005 to 2012, corresponding to widespread Git adoption (introduced in 2005), GitHub's founding (2008), and broader open-source adoption influenced by the Web 2.0 movement.



Detected spikes between 2004 and 2006 possibly linked to influential tech conferences or major software releases by companies like Google and Apple, driving increased developer collaboration and commit activity.

Popular programming languages

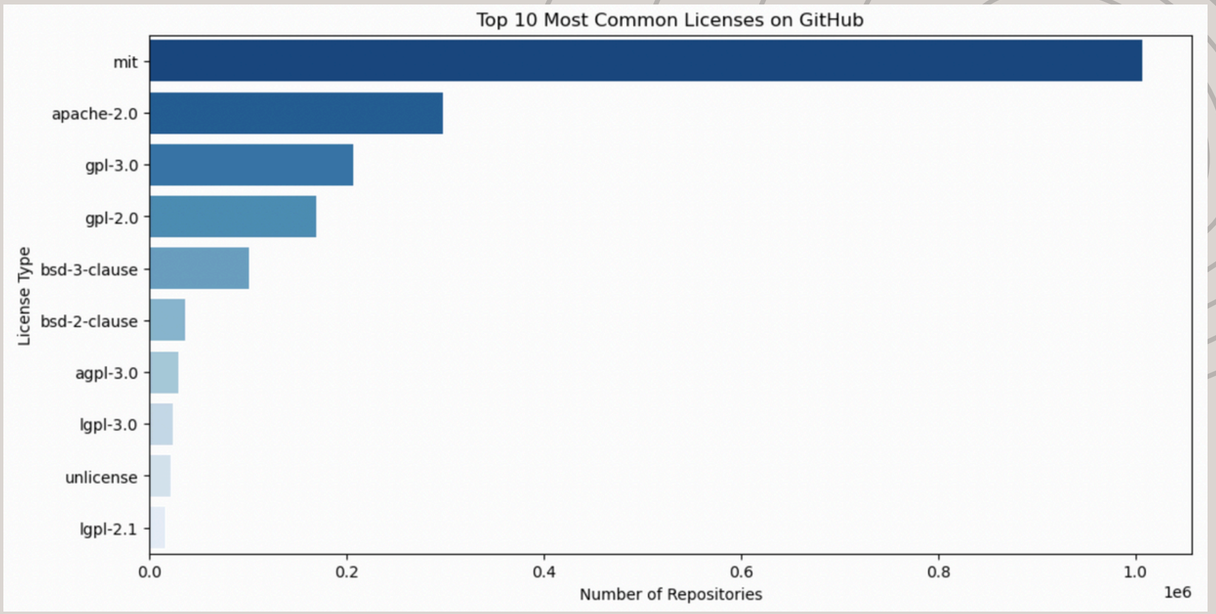


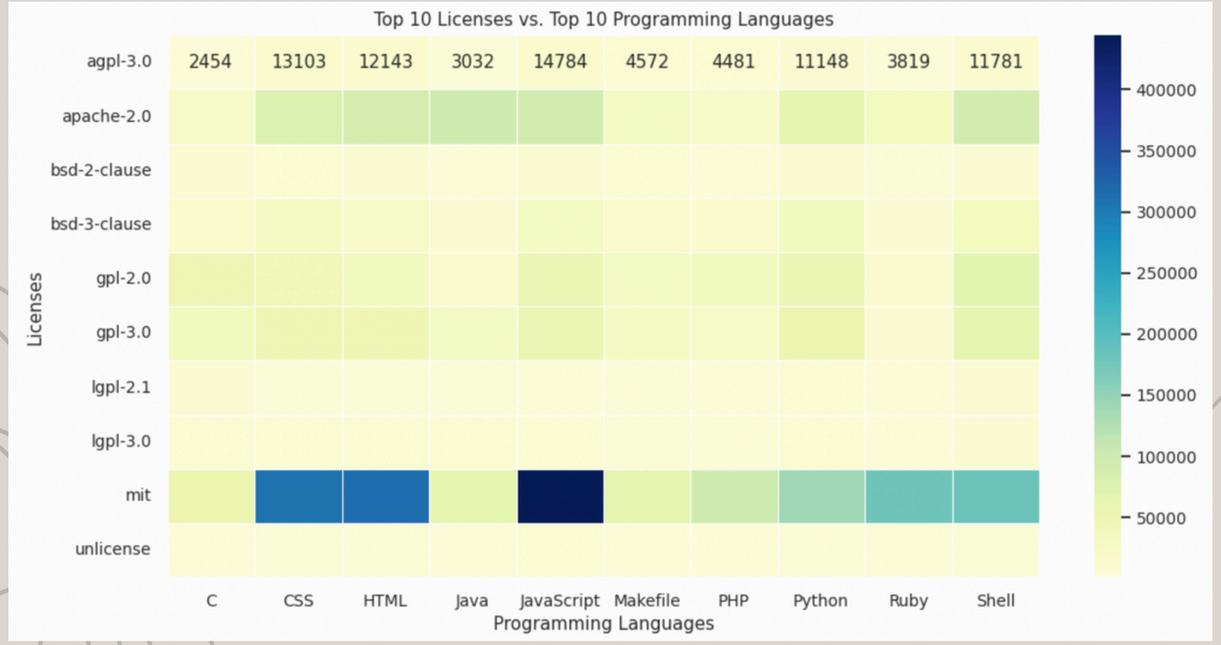


- JavaScript is the clear leader in the top 10 programming languages, indicating its widespread use across various domains.
- The line graph shows the use of programming languages over time. We see programming languages like Java and C dominate at the beginning, after which Python rises in popularity.
- These graphs suggest a dynamic ecosystem, where language popularity and use echo emerging technologies. TuringBots will find it hard initially to keep up with such dynamic behavior, however after training will be able to automate software development pipelines without innovation.

Distribution of Lisences

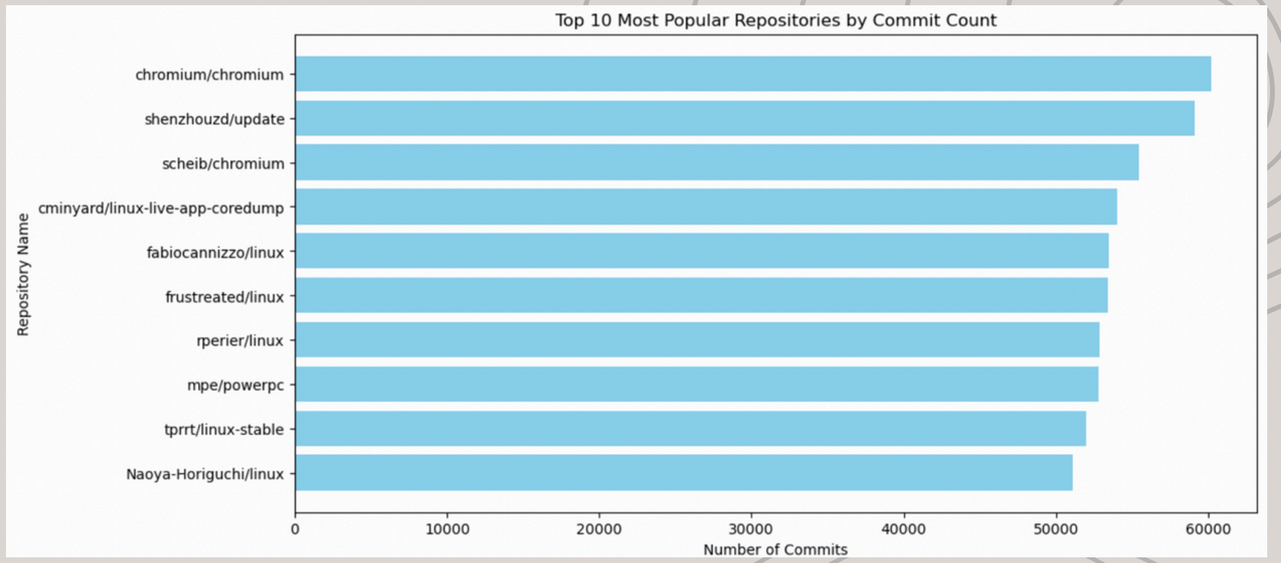
- MIT lisences dominates the other lisences reflecting its popularity and permissive nature, followed by licenses like Apache 2.0 and various GPLs.
- The heatmap highlights language–license correlations, indicating that while some licenses (like MIT) cut across multiple ecosystems, others cluster around specific communities or frameworks.
- Certain languages show clear preferences. For example, JavaScript often uses MIT, while projects in languages like Java or C++ may more frequently adopt Apache or GPL variants.

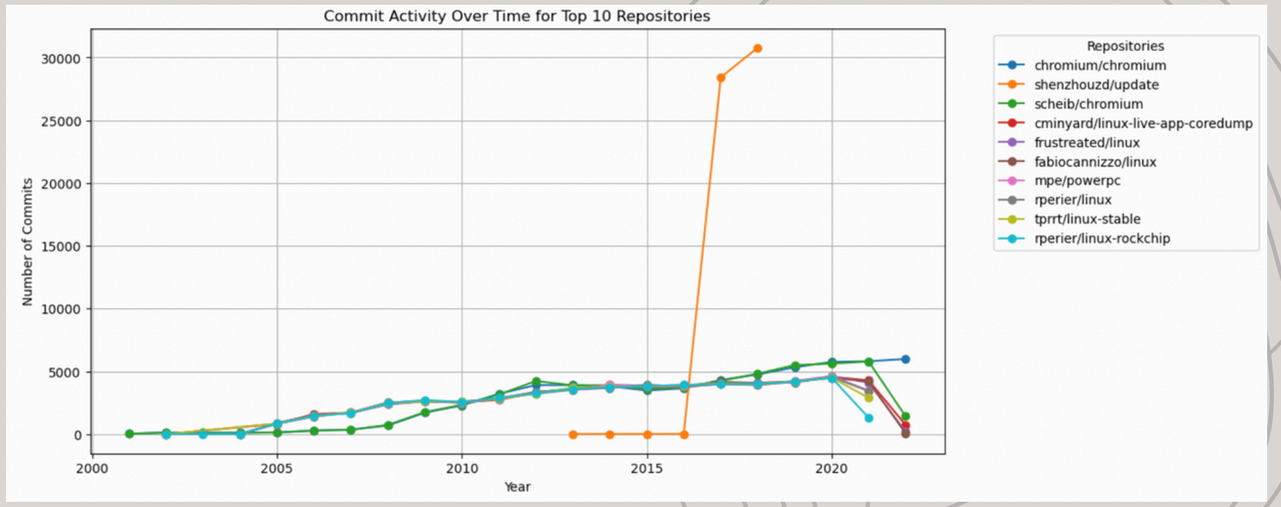




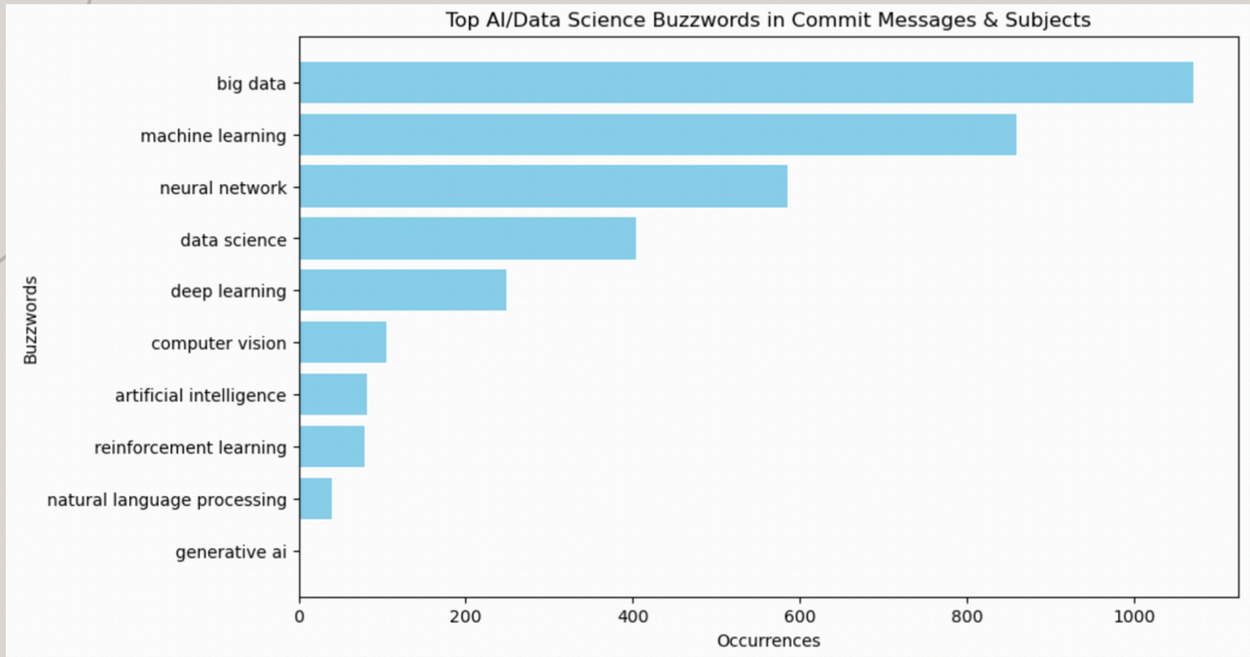
Popular and Growing Repositories

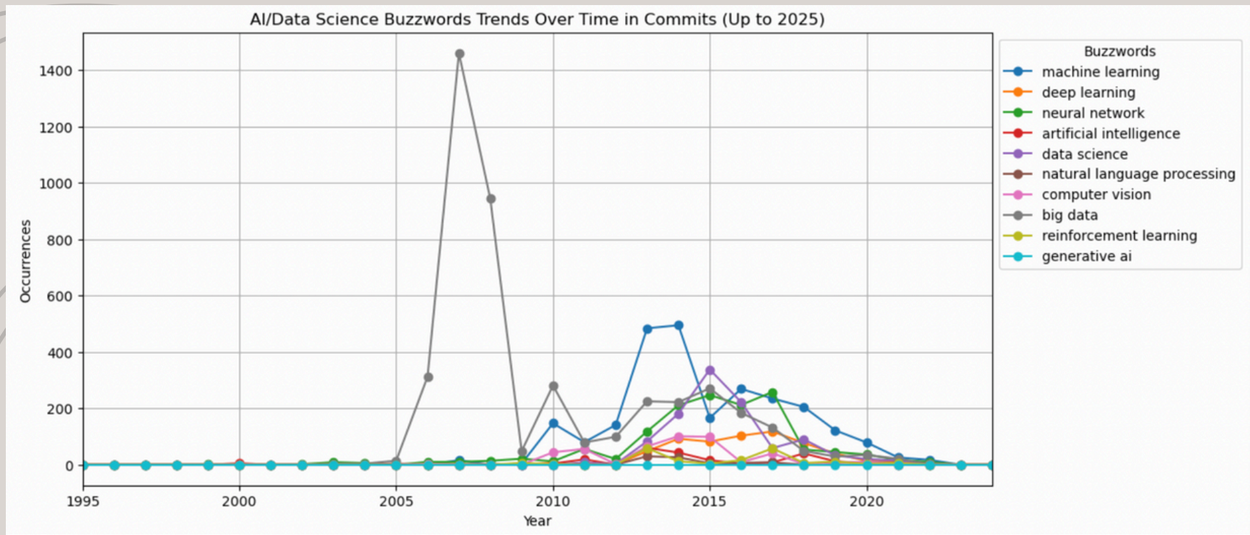
- Many of the top repositories by commit count are backed by large tech companies or well-known open-source foundations, suggesting corporate support and a strong community.
- Machine learning frameworks, containerization/orchestration tools, and modern web development libraries often show sharp spikes in commit activity, reflecting high demand and rapid innovation.
- Big Tech open-sourcing key projects fuels active collaboration and accelerates adoption, as seen in the brisk commit growth for these repositories once they're publicly released.
- if TuringBots keep up with emerging technologies for companies, they can replace those popular committers and push code into repos at a faster rate than humans can.





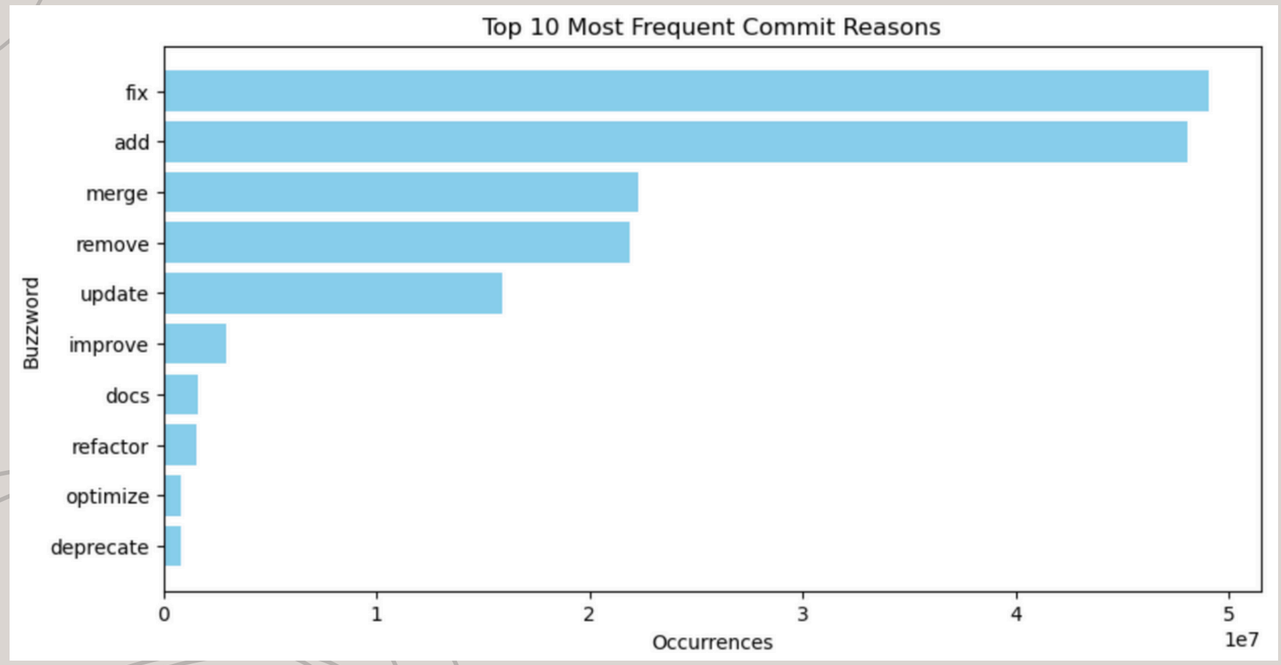
Technologies associated with AI





- “Big Data” and “Machine Learning” appear most frequently in commit messages and project descriptions, highlighting their central role in data science and AI initiatives.
- Terms like “Neural Network,” “Computer Vision,” and “Reinforcement Learning” show notable usage, reflecting deeper, more specialized AI domains gaining traction.
- The timeline graph indicates that while “Big Data” and “Machine Learning” have been popular for a longer period, newer technologies (e.g., “Generative AI”) start appearing more recently, suggesting evolving developer interests and breakthroughs in AI.

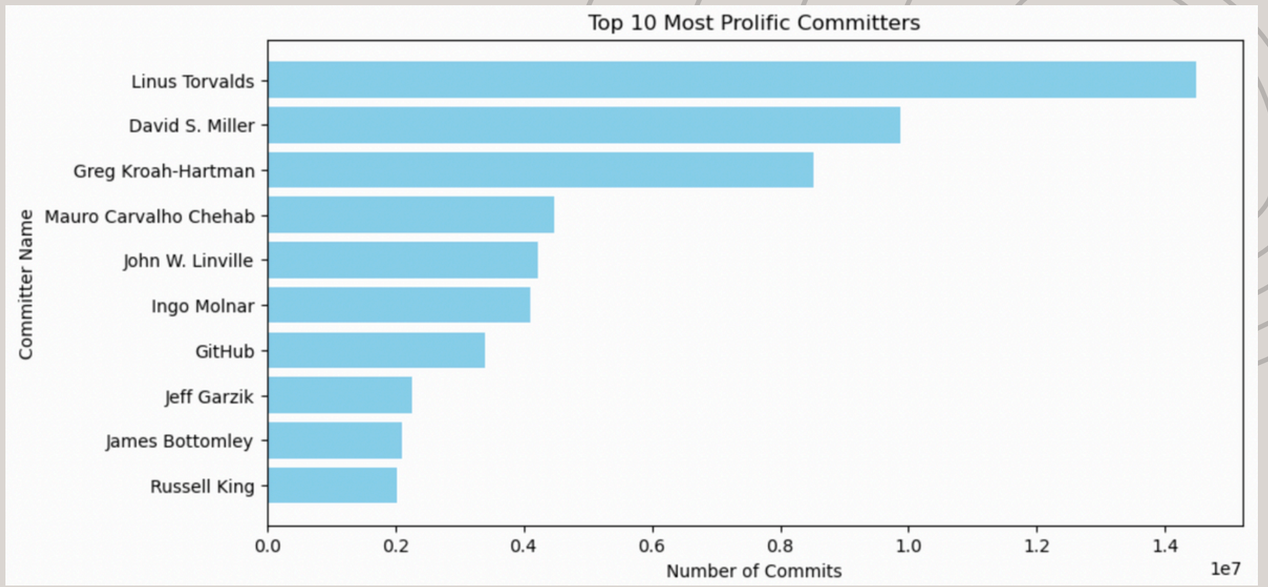
Frequent Commit Reasons

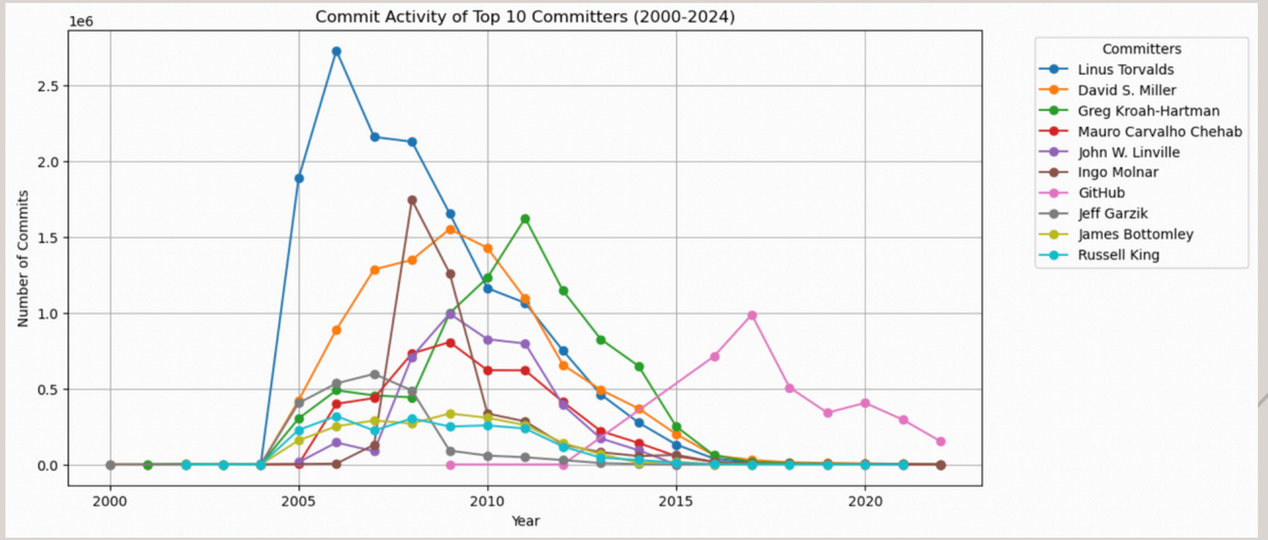


- The most common commit reasons revolve around resolving issues (“fix”) and introducing new functionality (“add”).
- Frequent merges and code removals suggest active collaboration, integration of multiple branches, and ongoing cleanup of obsolete or redundant code.
- Updates to documentation (“docs”), refactoring, and optimization indicate a consistent focus on improving code quality, performance, and clarity.
- These functions are easy for TuringBots to replicate as the code already exists, only changes are being made. This means that if developers train Bots on these specific functions such as “add” or “fix”, TuringBots have the capacity to replace software developers.

Most Prolific Commiters

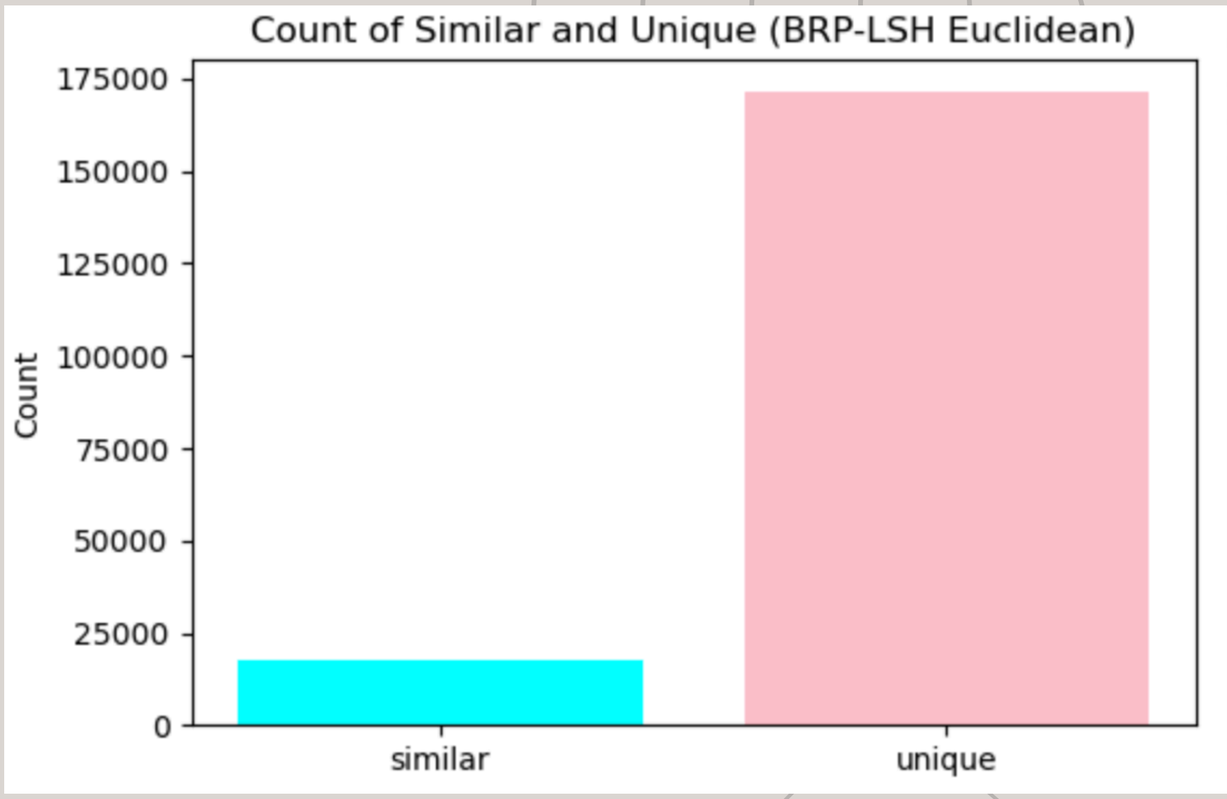
- Many on the list (e.g., Linux kernel contributors) play crucial roles in large, collaborative open-source projects, driving high commit volumes.
- The line chart reveals some committers steadily contributing over many years, while others exhibit intense bursts of commits in shorter windows.
- Their commit history highlights the depth of involvement—ranging from introducing key features to extensive maintenance—underscoring their influence in the open-source ecosystem.
- With the growth of open source ecosystems, it becomes easier to find data to train AI Bots on, making this type of advancement a double edged sword.





The Uniqueness of “Subject” and “Message”

- According to the LSH-based analysis, most commit subjects and messages appear to be unique, indicating that developers typically write custom commit descriptions.
- A smaller portion of commits shows repeated or near-duplicate messages, which could stem from boilerplate templates, automated tools, or repetitive workflows.
- The overall trend suggests that while some teams rely on standard commit formats, the majority of contributors still use individualized commit messages across different projects and languages.
- The unique nature of most commits will make the training of TuringBots hard, and once TuringBots are used more for software development, we will see a rise in similar counts and a decrease in unique counts.



Conclusion

- AI-driven coding assistants (TuringBots), trained extensively on standardized tasks such as bug fixing or feature additions, have the potential to significantly enhance developer productivity by automating routine tasks.
- However, the predominantly unique nature of commit messages suggests substantial limitations in fully replicating nuanced human developer input.
- While TuringBots and similar technologies can substantially augment software engineering and data science workflows, complete replacement of human developers remains unlikely, particularly for tasks demanding creativity, strategic planning, and innovative solutions.

Thank You