**Literature review**

**Journal of systems and software**

establishing link between GitHub repositories and academic papers (can be difficult)

* Establishing role of academic paper references contained in these repositories
* More than half the paper do not reference GitHub repositories
* Usually, research papers are published publicly to accelerate research
* Software developed based on public research are open source software (OSS)
* An argument can be made that if a paper has accessible code then it may have a bigger real life impact
* Readers can access, test and run the code to determine the full details of work
* Traceability also increases accountability of scientific results as due to a python glitch in approximately 100 papers, the research outcomes were deemed to be inaccurate
* Major contributors to scientific repositories are students and professors, the link between repositories and academic papers has not been well researched
* Machine learning is the most popular topic in repositories citing academic papers, individuals affiliated with academic background (university) tend to own these repositories

Study finds that **repositories which reference academic papers have the most impact, are most influential and align with academia**

**Examining the**

1. **Frequency of links from GitHub repositories to academic papers**
2. **Mixed-method study to identify public access, traceability, evolutionary aspects of such links**
3. **Availability of an online appendix containing qualitative coding results**

**Research objective:** finding relationship between GitHub repositories and academic papers

* **RQ1:** how many repositories link back to academic papers and how often these links are public access?
* **Rq2:** what is the relationship between GitHub repositories and academic papers, analyzing the relationship from six different aspects – domain (characteristic of depositories), affiliation of main contributors (industry of academia), whether repository contributors and paper authors are the same (helps us analyze the extent to which paper is being implemented by developers), whether repository papers link back to the original repository we found them in (acknowledging the existence of repository), finally looking at the impact of the papers being referenced (how influential academic papers are that are being referenced in repositories)
* **RQ3:** how does evolution affect the relationship between GitHub repository and academic paper – Do academic papers get updated after being referenced, are repository owners aware of when there is a new version of the academic paper

**Findings of RQ1:** although python files had the most references to academic papers, it would cover only 17% of repositories, whereas .md filed would cover every 9 out of 11 file. More than 98% of repositories had referenced an academic paper which was available to public for free while investigating the README.md files

**Finding of RQ2:** domain = related field, most common domain is deep learning (basically machine learning). moreover, majority of GitHub repositories referencing an academic paper are affiliated with universities, more than 50% repositories which referenced academic papers are not in any form related to the authors of the paper however 40% are, most referenced papers in GitHub repositories are highly cited in academia as-well, although more than 50% of the papers are referenced by single repository most referenced papers can be referenced in the repository written in a different language other than python

Considerations in **RQ3:** we consider a paper has an update if it was extended to a journal at a later point in time compared to when the paper is being referenced or if the paper was published on arxiv.org. changes in the content of paper are not considered

**RQ3:** most academic papers referenced on Github (84%) did not have any changes since they were published however a minority (16%) of them did have changes, moreover papers in GitHub README.mds do not attract any changes but small number of links are updated (repositories must have been changed too) however evolution is rare in the academic papers after being published.

**Limitations:**

1. The bias towards arxiv and doi links which are public access instead of those which are not, it would be better to extend the pattern of generalization
2. Another challenge would be to analyze the indirect impact of academic papers in GitHub repositories and what type of software changes would take place if the referenced academic papers was updated

**Evolution of technology within a simple computer model**

In its collective sense technology forms a network of elements in which novel elements are continually constructed from existed ones, while the new technology builds upon old and existing ones, it phases out the old technology

* New circuits or technology are constructed by randomly writing together the existing ones and see if they satisfy any existing needs
* If a circuit seems useful (satisfies some needs better than the previous ones) it replaces the one that previously satisfy the needs
* It is then added to the active collection of technologies and becomes an element for the construction of still further circuits
* In this way elements constantly gets added in active technologies and removed if deemed to be obsolete
* We cannot expect complex circuits to be created without the building blocks of simpler circuits and fulfilling simpler needs first (system arrives at complicated circuits only by first satisfying simpler needs and using the result as building blocks to bootstrap its way to satisfying more complex ones)

The experimental system

* Each run of the artificial system will be seen as experiment, each experiment starting with only primitive components (usually one, an elementary logic gate) and the computer generates new circuits by randomly wiring together several components in a noncyclic way.

**Important note: while doing literature review GitHub rolled around a massive copilot update which enables users**

**Issues faced by users using copilot**

* Functionality usage issue: usually copilot gives code suggestion and “previous/next” and “all” suggestions however some users reported that it stopped giving suggestion on pycharm
* Setup/operation issue: causing copilot to crash unexpectedly
* Authenticaion failure: usually to use copilot services user needs to login with their account details however some people are unable to login

Differences between **literature review** and **copilot GuitHub**:

Study finds that repositories referencing academic papers have the most impact and are more influential however, the new GitHub copilot update might have changed everything as this study from GitHub and Accenture claims (Accenture & GitHub, 2024):

1. **Productivity Gains**:
   * Developers completed tasks 55% faster with GitHub Copilot
   * The tool helped reduce time spent on repetitive coding tasks
   * Developers using Copilot successfully completed more tasks overall
2. **Developer Experience Improvements**:
   * Significant increase in developer satisfaction
   * Reduced cognitive load and context switching
   * Developers reported feeling more focused on problem-solving rather than implementation details
3. **Code Quality**:
   * No negative impact on code quality observed
   * Some evidence suggesting potential improvements in code standardization
   * Reduced time spent on debugging and fixing syntax errors
4. **Enterprise Adoption Benefits**:
   * Faster onboarding for new developers
   * Knowledge sharing and standardization across teams
   * More efficient use of developer resources

The study suggests that AI coding assistants like GitHub Copilot provide significant value in enterprise environments, not just by increasing speed but by improving the overall developer experience and allowing developers to focus on higher-value work.

From the new update we can try and raise a lot of questions like:

* Are the repositories which reference academic papers still considered more impactful or influential?
* What are the engagement levels in repositories since the introduction of AI? Are people still engaging with each other or they are dependent on AI?

However, we decided to take up just one question:

**Q** If there is any difference between the codes before the introduction of AI and after the introduction of AI?

Our methodology to tackle the question is described in 7 steps below:

Our research methodology follows a systematic approach to identify, analyze, and detect AI-generated content in GitHub repositories. The procedure consists of seven interconnected phases designed to provide comprehensive insights into the prevalence and characteristics of AI-generated code.

**Phase 1: Repository Selection and Classification**

We begin by identifying trending repositories on GitHub using the platform's API. To ensure a balanced comparison, we select:

- Five top trending repositories explicitly utilizing AI packages/tools (identified through dependency files and workflow configurations)

- Five top trending repositories without AI-related tools

This selection is implemented through targeted search queries that filter repositories based on popularity metrics (star count) and programming language:

```python

search\_query = "language:python stars:>500"

```

For classification, we examine key indicator files including `requirements.txt`, `pyproject.toml`, and `.github/workflows/` configurations, specifically looking for AI-related dependencies such as OpenAI, Hugging Face, and TensorFlow packages.

**Phase 2: Commit History Extraction**

For each selected repository, we extract the complete commit history using the GitHub API. This step provides the foundational dataset for our analysis, capturing:

- Commit identifiers (SHA)

- Commit dates and timestamps

- Commit messages

- Author information

This historical data enables us to track the evolution of code contributions and identify patterns that may correlate with AI usage.

**Phase 3: Contributor Profile Analysis**

To develop a comprehensive understanding of contribution patterns, we expand our data collection to include the broader GitHub history of each contributor identified in Phase 2:

- All public repositories associated with each contributor

- Complete commit histories across these repositories

- Code contribution patterns and file modification types

This cross-repository analysis provides context for identifying individual coding styles and potential shifts that might indicate AI-assisted contributions.

**Phase 4: AI-Generated Content Detection**

We implement a multi-faceted approach to detect potentially AI-generated content within commits:

1. \*\*Multiple Detection Tools\*\*: We employ a combination of pre-trained detection models including DetectGPT, GPTZero, and RoBERTa-QA to improve reliability through consensus.

2. \*\*Perplexity Analysis\*\*: For each code snippet, we calculate perplexity scores using a code-specific language model:

```python

tokens = tokenizer(code, return\_tensors="pt")

loss = model(\*\*tokens, labels=tokens["input\_ids"]).loss

perplexity = torch.exp(loss)

```

3. \*\*Statistical Measures\*\*:

- Standard deviation of perplexity across code segments

- Burstiness calculation using the formula:

$\text{Burstiness} = \frac{\sigma - \mu}{\sigma + \mu}$

These complementary approaches allow us to identify statistical anomalies that may indicate AI-generated content.

**Phase 5: Data Organization and Storage**

All collected data is systematically organized in a relational database with the following structure:

- \*\*Repositories table\*\*: Metadata including repository name, star count, primary language, and AI-tool classification

- \*\*Commits table\*\*: Detailed commit information with detection metrics (perplexity, AI detection scores)

- \*\*Contributors table\*\*: Contributor profiles with cross-repository statistics

This structured approach facilitates comprehensive analysis and enables efficient querying of relationships between variables.

**Phase 6: Validation and Reliability Assessment**

To ensure the robustness of our findings, we implement several validation techniques:

- \*\*Inter-rater reliability\*\*: Quantitative comparison of results from different AI detection tools using Cohen's Kappa and Fleiss' Kappa coefficients

- \*\*Cross-validation\*\*: Application of our metrics to known human-written and AI-generated code samples to establish baseline performance

- \*\*Temporal consistency\*\*: Analysis of detection patterns over time to identify potential anomalies

These validation steps are critical for establishing confidence in our detection methodology and findings.

**Phase 7: Analysis and Reporting**

The final phase involves comprehensive analysis of the collected data to identify patterns and trends:

- Longitudinal analysis of perplexity and burstiness across repositories

- Comparative analysis between AI-tool-using and non-AI-tool-using repositories

- Contributor-level analysis to identify changing patterns in coding style

Findings are visualized through appropriate charts and graphs to effectively communicate trends, with all data exported in accessible formats (CSV, JSON) and interactive dashboards for further exploration.

Through this methodical approach, we aim to provide a comprehensive understanding of the prevalence, characteristics, and impact of AI-generated code in GitHub repositories, contributing valuable insights to the ongoing discussion around AI tools in software development.

**Where did we reach and what is left?**

We did not end up completing all the 7 steps mentioned above, as the data extraction would take too much time. However, I will run down the viewer as to how far we were able to reach what is left.

**Step 1: Identify the top 5 trend**

Using **REST API** we can search specific items on github

Limitations (Taken from the website):

* To keep the REST API fast for everyone, we limit the number of repositories a query will search through. The REST API will find up to 4,000 repositories that match your filters and return results from those repositories.
* To keep the REST API fast for everyone, we limit how long any individual query can run. For queries that exceed the time limit, the API returns the matches that were already found prior to the timeout, and the response has the incomplete\_results property set to true.
* You need to successfully authenticate and have access to the repositories in your search queries, otherwise, you'll see a 422 Unprocessable Entry error with a "Validation Failed" message. For example, your search will fail if your query includes repo:, user:, or org: qualifiers that request resources that you don't have access to when you sign in on GitHub.
* You need to successfully authenticate and have access to the repositories in your search queries, otherwise, you'll see a 422 Unprocessable Entry error with a "Validation Failed" message. For example, your search will fail if your query includes repo:, user:, or org: qualifiers that request resources that you don't have access to when you sign in on GitHub.

**Creating the script to classify trending repositories into “AI” and “Non-AI”**

With the help of “GitHub\_Trend.py” we identified top 5 AI repositories and top 5 non AI repositories which were:

Top 5 AI Repositories:

freeCodeCamp/freeCodeCamp https://github.com/freeCodeCamp/freeCodeCamp

tensorflow/tensorflow https://github.com/tensorflow/tensorflow

ohmyzsh/ohmyzsh https://github.com/ohmyzsh/ohmyzsh

Significant-Gravitas/AutoGPT https://github.com/Significant-Gravitas/AutoGPT

flutter/flutter https://github.com/flutter/flutter

Top 5 Non-AI Repositories:

EbookFoundation/free-programming-books https://github.com/EbookFoundation/free-programming-books

sindresorhus/awesome https://github.com/sindresorhus/awesome

codecrafters-io/build-your-own-x https://github.com/codecrafters-io/build-your-own-x

public-apis/public-apis <https://github.com/public-apis/public-apis>

jwasham/coding-interview-university https://github.com/jwasham/coding-interview-university

**Now that we have our target repositories, we can move on to the 2nd step and start extracting Commit history and details of these repositories**

1. At first I started the process and tried including all 10 repositories however the code “Commit\_extraction\_and\_users.py” did not work and gave the error “Failed to fetch commit details for freeCodeCamp/freeCodeCamp with SHA: XXX”. This back and forth kept on happening for quite a while and I tried changing the code and tried by only extracting one repository at a time. However by going on to DeepSeek, it finally resolved my issue and the code “Extraction\_AUTOGPT.py” finally worked. But only for extracting Commit history of the repository, it did not work for the extraction of Commit details.

* For Commit details of AUTOGPT, I made another python script (with the help of DeepSeek) “AUTOGPT\_details.py” and it helped extracting the Commit details for the said repository.

1. For Jawsham I used the script “Commit\_history\_and\_details\_jawsham.py” to extract both Commit history and Commit details
2. For codecrafters I used the script: “Commit\_history\_and\_details\_codecrafters.py”
3. For Publicapis I used the script: “Commit\_history\_and\_details\_publicapis.py”
4. For Flutter I used the script: “Commit\_details\_and\_history\_flutter.py”
5. For Sindresorhus I used the script: “Commit\_details\_and\_history\_sindresorhus.py”
6. For Ebooks I used the script: “Commit\_history\_and\_details\_ebooks.py”
7. For Ohmyzsh I used the script: “Commit\_history\_and\_details\_ohmyzsh.py”
8. For FreeCodeCamp I used the script: “commit\_history\_freeCodeCamp\_freeCodeCamp.json”
9. For Tensorflow the script is: “Commit\_history\_and\_details\_tenserflow.py” however I was not able to complete the extraction for the repository as it was too long and would have taken atleast a week and a half

**Note: All the data has been shared with the Supervisor on the GitHub.**

**Step 3: Extraction of Commit details of users/contributors of above repositories**

1. For AUTOGPT, to extract contributors’ commit details and history the script used: “User\_history\_for\_AutoGPT.py”
2. For Sindresorhus, to extract contributors’ commit details and history the script used: “User\_history\_for\_sindresorhus.py”
3. For Codecrafters, to extract contributors’ commit details and history the script used: “User\_history\_for\_codecrafters.py”
4. For Publicapis, to extract contributors’ commit details and history the script used: “User\_history\_for\_publicapis.py”
5. For Jwasham, to extract contributors’ commit details and history the script used: “User\_history\_for\_jwasham.py”
6. For Flutter, to extract contributors’ commit details and history the script used: “User\_history\_for\_flutter.py”

While Extracting data for Contributors I realized that the code was only giving data of up to 30 contributors at max because of GitHub's API pagination. By default, the GitHub REST API returns a maximum of 30 items per page for most endpoints that return arrays. This includes the contributors endpoint. Hence, I checked how many contributors were there for AUTOGPT with the help of the script “python3 count\_contributors\_2.py” and it turned out that there were approximately 438 contributors. Hence I tried running a new script which will give out commit details and history for top 80 contributors – “Commits\_of\_every\_contributor\_AUTOGPT.py” but it failed to pull out data for all the contributors hence I left it at that

Due to time constraints I decided to move on and implement step 4 on the already available data:

**Step 4: Building the code to successfully analyse the extracted JSON files**

With the help of Claude I built the script “simplified\_ai\_detector.py” which utilises ChatGPT 2 and older AI models. It analysed the files and gave out “json\_analysis\_20250219\_211917.csv” stating an average AI score of 0.60

A computer screen with white text

AI-generated content may be incorrect.Later I asked Claude to integrate ChatGPT 4 and above models to detect AI in the JSON files and it gave me the script “modern\_ai\_detector.py”

However it was not able to detect any AI activity hence I asked Claude to tweak the script and make it more sensitive to the usage of AI and ended up with “modern\_ai\_detector\_4.py”. The results of the most recent script are saved in the file “json\_analysis\_20250221\_161743.csv”

**End of research from my end however some drawbacks are:**

* I accidentally did not factor in the other generic data in the JSON file and ran the code on the file as a whole, later I realized I only needed to run the script on the code which was extracted from the repositories and not on the generic data hence that is why I was unable to find any usage of AI in the data
* The script of the most recent AI detector “modern\_ai\_detector\_4.py” has some flaws in it which needs to be reconfigured
* Maybe extraction of the data needs to be done again from repositories and this time it should be just the code
* JSON file data needed to be clearer and not filled with just the generic data

Reference for the GitHub article: -

<https://github.blog/news-insights/research/research-quantifying-github-copilots-impact-in-the-enterprise-with-accenture/>