Analyzing DevGPT Dataset

Based on the NAIST DevGPT Dataset

Srinija Dharani
Computer Science
Florida State University
Tallahassee, FL
sd23be@fsu.edu

Roochita Ikkurthy
Computer Science
Florida State University
Tallahassee, FL
ri23b@fsu.edu

Chaitanya Guntupalli
Computer Science
Florida State University
Tallahassee, FL
cg21bb@fsu.edu

ABSTRACT

The project encapsulates our research endeavors centered on ChatGPT's interactions within the developer community, focusing on three pivotal areas. The first area of investigation involves a comprehensive analysis of programming language queries posed to ChatGPT, with an emphasis on identifying temporal shifts in language preferences. The second aspect delves into the variability of conversation lengths with ChatGPT, contingent upon the nature of the issues presented. This involves a methodical categorization of conversation types, a process that is presently being refined. The final facet of our research is concentrated on pinpointing the most prevalent prompts directed at ChatGPT by developers and critically analyzing the AI's response mechanisms. The research utilized sophisticated data analysis tools and adhered strictly to ethical standards in data handling. As we progressed, we anticipated generating actionable insights, with subsequent updates planned to chronicle ongoing developments in this multifaceted research initiative.

KEYWORDS

Data Analysis, Data Visualization, NLP, NLTK, Spacy

1 Code versus No-Code Solutions

1.1 Problem Statement

The objective of this research is to ascertain how the length of interactions with ChatGPT fluctuates in relation to the nature of the issues presented. This inquiry is significant as it offers critical insights for developers, illuminating the dynamics between the issue's complexity and the ensuing conversation duration. Understanding these patterns is pivotal for optimizing ChatGPT's efficiency and user experience.

1.2 Motivation

Understanding the most common questions posed to ChatGPT by developers and its responses can be highly beneficial. By examining how ChatGPT addresses various queries, one can refine their prompt-formulation skills. This improvement in crafting prompts can significantly enhance interactions with the tool, yielding more precise and useful answers. Additionally, exploring these interactions offers valuable insights into the behavior and capabilities of AI models such as ChatGPT, enriching our understanding of their functionality and potential applications.

1.3 Approach

1.3.1 Data Collection

For this analysis, we have employed data from Snapshot20231012. This specific version was chosen due to its comprehensive nature, as it integrates data from previous snapshots, thereby streamlining the research process. We took into consideration the following datasets:

1. Issue_Sharings
2. Commit_Sharings
3. Discussion_Sharings
4. HackerNews

1.3.2 Data Pre-Processing

1.3.2.1 Data Normalization

The dataset had a complex hierarchical structure with multiple levels of nesting. In order to isolate and extract particular data attributes, such as ChatGPTSharings, Conversations, Prompt, Answer, and ListOfCode, we applied a process of data normalization to each nested component within the dataset.

1.3.2.2 Extraction of English Rows

The dataset has multiple languages, which affected the analysis. So, we wrote a function to filter only the English rows.

1.3.3 Summary Generation

To generate the summary for each prompt, the following process was employed.

1. Tokenizing text into sentences, and further tokenizing the sentences into words.
2. Removing stopwords and punctuation
3. Calculating cosine similarity between sentences based on TF-IDF matrix.
4. Applying the page ranking algorithm to rank sentences for understanding each sentence's importance.
5. Storing the generated summary in a column in the dataset.

1.3.4 Defining Keywords

To define a set of keywords, I asked ChatGPT what developers normally seek its assistance in. Developers often approach ChatGPT with a wide range of requests, typically revolving around programming, debugging, learning concepts, and development tools. To identify these questions, you can focus on keywords that are common in such contexts. Here are some common types of requests and corresponding keywords:

1. **Coding Assistance:**
   Keywords: "code", "write", "script", "implement", "function", "algorithm".
   Example: "Can you write a Python function to sort a list?"
2. **Debugging and Error Resolution:**
   Keywords: "debug", "error", "fix", "issue", "resolve", "exception", "traceback".
   Example: "Help me debug this JavaScript error message I'm getting."
3. **Explanation of Concepts:**
   Keywords: "explain", "understand", "concept", "how does", "what is", "theory", "principle".
   Example: "Can you explain how recursion works in programming?"
4. **Learning and Educational Resources:**
   Keywords: "learn", "tutorial", "course", "resource", "guide", "documentation", "example".
   Example: "Recommend some resources to learn React."
5. **Best Practices and Design Patterns:**
   Keywords: "best practice", "design pattern", "architecture", "efficient", "optimization", "structure", "model".
   Example: "What are some best practices for REST API design?"
6. **Technology and Tool Recommendations:**
   Keywords: "recommend", "tool", "library", "framework", "technology", "platform", "software".
   Example: "Which libraries do you recommend for data visualization in Python?"
7. **Code Review and Optimization:**
   Keywords: "review", "optimize", "improve", "enhance", "refactor", "performance", "efficiency".
   Example: "Can you review and suggest improvements for my Ruby script?"
8. **Project Guidance and Planning:**
   Keywords: "plan", "approach", "strategy", "project", "structure", "organize", "develop". Example: "How should I plan the architecture for my mobile app project?"
9. **Integration and Configuration**:
   Keywords: "integrate", "configure", "setup", "install", "deploy", "connect", "environment".
   Example: "How do I integrate a payment gateway in my web application?"
10. **Version Control and Collaboration:**
   Keywords: "git", "version control", "collaborate", "merge", "branch", "commit", "repository".
   Example: "How do I resolve merge conflicts in Git?"

1.3.5 Checking Code or No-Code Solutions

Based on the ListOfCode column, we determined what each prompt generated – a code or no-code solution.

1.3.6 Findings

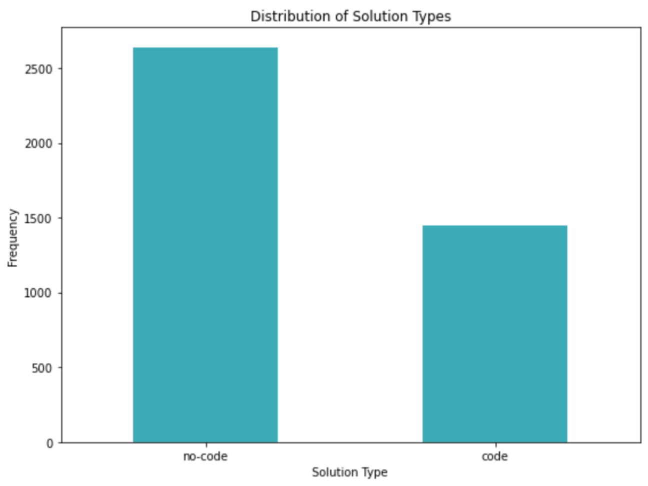


Figure 1.1. Distribution of Solution Types

We plotted a graph against Solution Type (Code or No-Code) and Frequency. We observed that more than 2500 prompts generated solutions with code. And around 1500 generated solutions with no code.

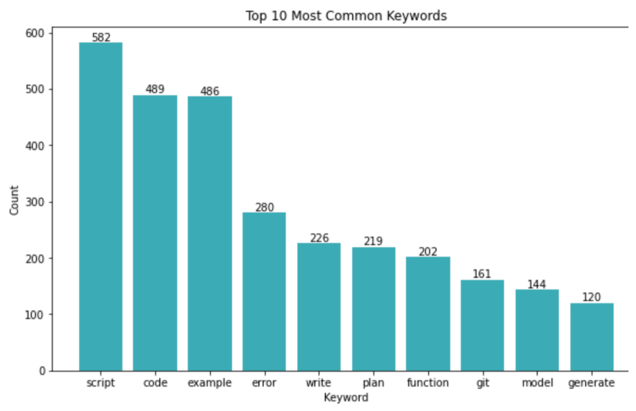


Figure 1.2. Top 10 Most Common Keywords with Frequencies

The above graph shows the 10 most commonly occurring keywords asked in all prompts that developers asked ChatGPT.

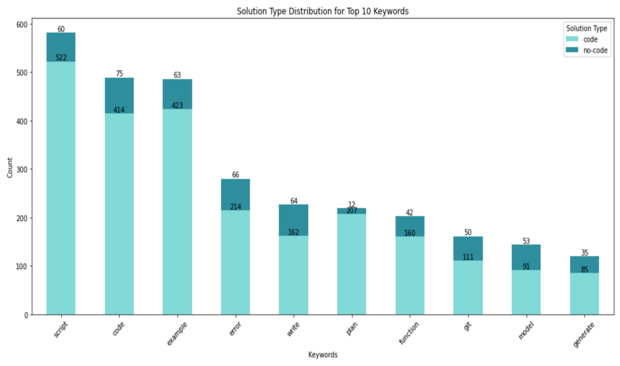


Figure 1.3. Solution Type versus Top 10 keywords

The above graph represents the number of code and no-code solutions given by ChatGPT for the top 10 most occurring keywords.

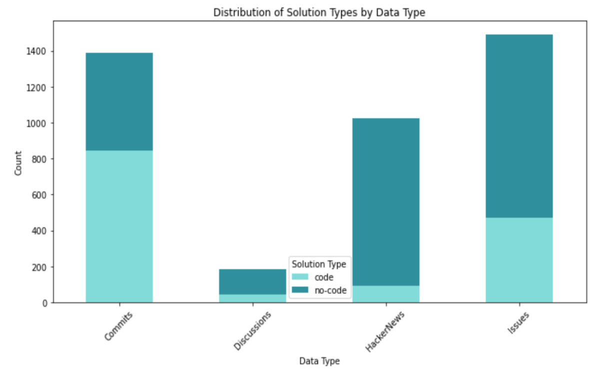


Figure 1.4. Solution Types versus Data Types

The datasets – discussions and hackernews denote a high frequency of no-code solutions.

Final findings are as follows.

1. The keyword "script" seems to have the highest overall co-occurrence with other keywords, as indicated by the generally darker colors in its row and column. This suggests that "script" is a common term that is often mentioned alongside various other keywords.

2. The pairs ("code", "example") and ("write", "plan") also show a higher degree of co-occurrence, which can be seen from the darker squares at their intersections. This might indicate that these pairs of concepts are often discussed together.

3. Other keywords like "function", "git", and "model" seem to have moderate to low co-occurrence with other keywords, as the corresponding squares are lighter in color. This could imply that these terms are more specialized or occur in more specific contexts.

4. The keywords "error" and "write" also show a significant degree of co-occurrence. This could suggest a common theme or discussion topic related to writing code and encountering or handling errors.

5. The keyword "generate" has moderate co-occurrence with "script" and "code" but is less frequently associated with other keywords like "error" or "plan".

1.4 Result

- Developers are most frequently seeking assistance with scripting, coding examples, debugging, and understanding or implementing specific functions or code.
- ChatGPT's responses are tailored to the nature of the prompt, with a significant number of responses containing code. However, there is also a substantial number of 'no-code' responses, which include explanations, guidance, or conceptual discussions.
- The balance between 'code' and 'no-code' responses highlights the AI's ability to adapt its output to the needs of the user, whether they require hands-on coding help or conceptual understanding.

1.5 About the Primary Contributor of this Research Question

Name: Srinija Dharani
FSU ID: sd23be@fsu.edu

2 Programming Languages Versus Categories

2.1 Problem Statement

This analysis aims to identify the most occurring programming languages that developers ask ChatGPT to resolve their bugs. More specifically, what mostly occurring programming languages do developers ask ChatGPT to rectify their bugs and solve their issues in?

2.2 Motivation

This investigation aims to identify prevalent programming languages for which developers seek resolution of bugs and other issues through interactions with ChatGPT. Additionally, it seeks to discern the programming languages that developers predominantly inquire about over time when soliciting assistance from ChatGPT.

2.3 Approach

2.3.1 Introduction

In this analysis, we conducted a comprehensive study on the most recent snapshot (10-12-2023) of GitHub repository interactions. The actual dataset comprises six files: Commit, PR (Pull Request), File, Issue, and Discussion Sharing. Our primary focus was on extracting relevant information such as RepoLanguage, Title, and Prompt from these files to gain insights into the nature of interactions within the repositories.
Hence, we had to take only five files:

1. Commit Sharing
2. Discussion Sharing
3. Issue Sharing
4. File Sharing
5. PR Sharing

2.3.2 Data Cleaning and Preprocessing

To ensure the quality of our analysis, we initiated the process by carefully handling missing data. The Hacker News sharing file was omitted from our analysis due to the absence of the crucial RepoLanguage attribute. After removing NaN entries, the remaining data was consolidated into a final dataset, which consisted of approximately 6000 rows—an ample sample size for our analysis.

2.3.3 Language Detection

One of the key challenges in our analysis was handling non-English prompts. To address this, we employed the Language Detect tool to identify non-English words within the prompts. We subsequently categorized these prompts as non-English, contributing to a more accurate analysis.

2.3.4 Prompt Categorization

For a nuanced understanding of the prompts, we leveraged the nltk.tokenize library. Specifically, we imported word_tokenize to categorize prompts into distinct categories, namely:

1. Bugs
2. Features
3. Errors
4. Tests.

Each category was associated with a set of keywords to ensure robust categorization.

2.4 Result

Finally, our analysis can be shown through visualizations. We have shown visualizations for each of the files for better understanding of the data.

For discussion data, we observed that python was most occurring programming languages used in discussion sharing and around 1267 prompts were related to bugs.

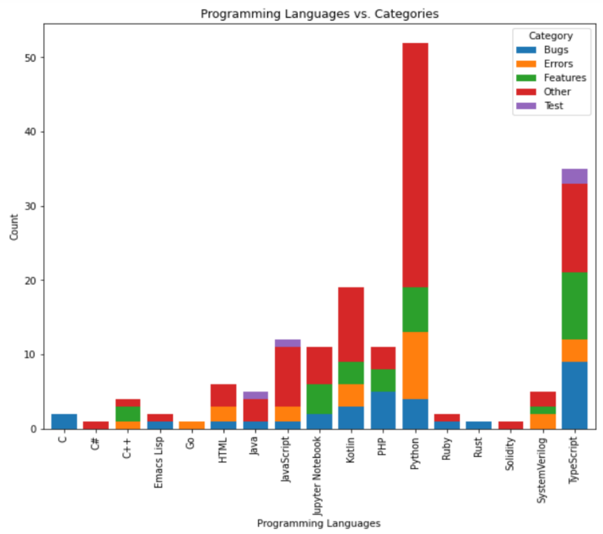


Figure 2.1. Programming Language vs Categories – Discussions

For commit data, we observed that CSS was most occurring programming languages used in commits and around 541 prompts were related to bugs.

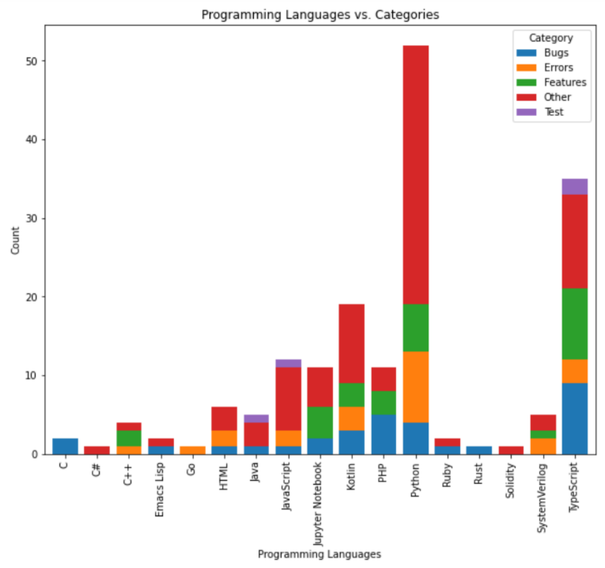


Figure 2.2 Programming Language vs Categories – Commits

For issue data, we observed that Python was most occurring programming language used in issue sharing and around 311 prompts were related to bugs.

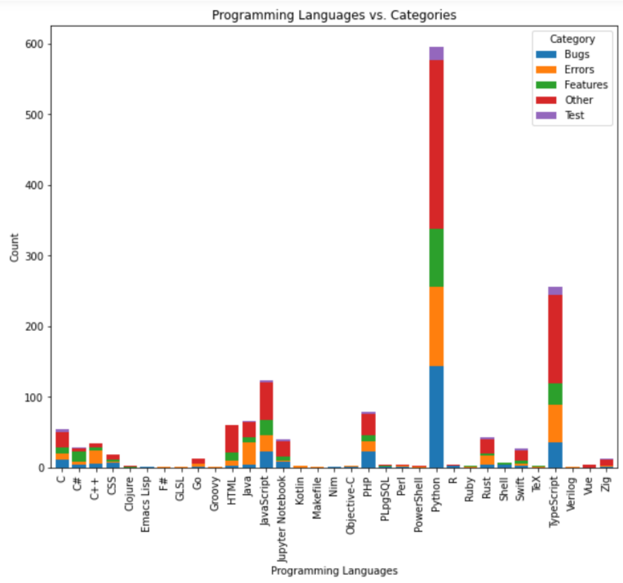


Figure 2.3 Programming Language vs Categories – Issue

For file sharing data, we observed that Python was most occurring programming language used in file sharing and around 20% of the prompts are Errors and 19% were Bugs.

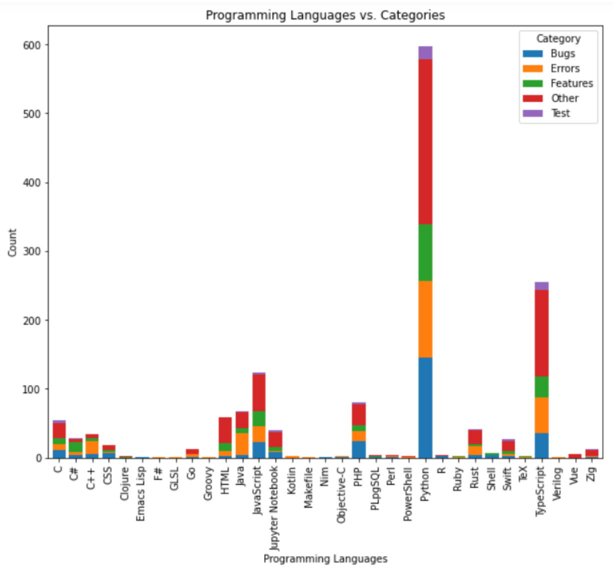


Figure 2.4 Programming Language vs Categories – File Sharing

For PR sharing data, we observed that C++ was most occurring programming language used in PR sharing and around 18% of the prompts are Features and 14% were Errors.

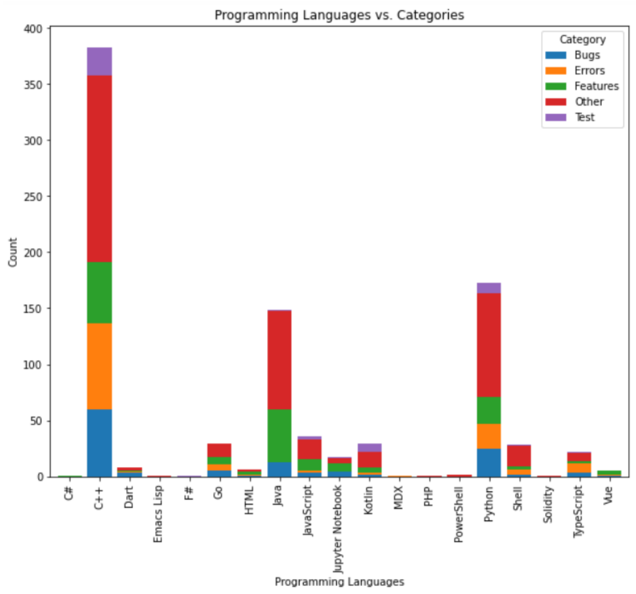


Figure 2.5 Programming Language vs Categories – PR

For the final dataset, here are the visualizations:

Python and C++ are the most occurring programming languages observed. Python to be 1503 prompts and C++ to be 1061 prompts.

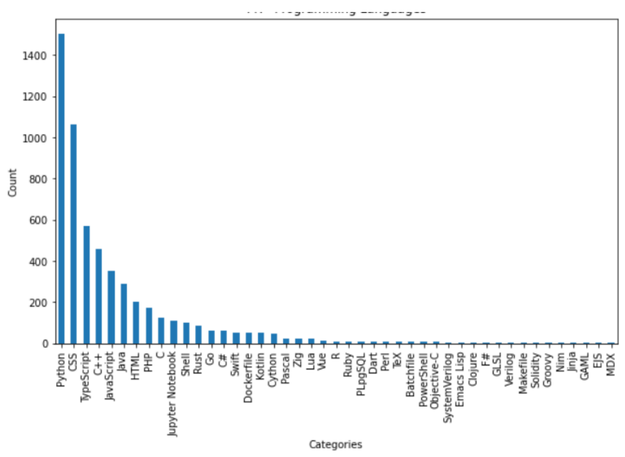


Figure 2.6 Most Occurring Programming Language

The category pie chart tells us that 27.5% of the prompts are of "other" category. As per our observation, the prompts are to be in other language other than English and some are code, which is difficult to add them into any of the categories. Hence, we can say that the majority of the prompts are related to Bugs and the least are related to Tests.

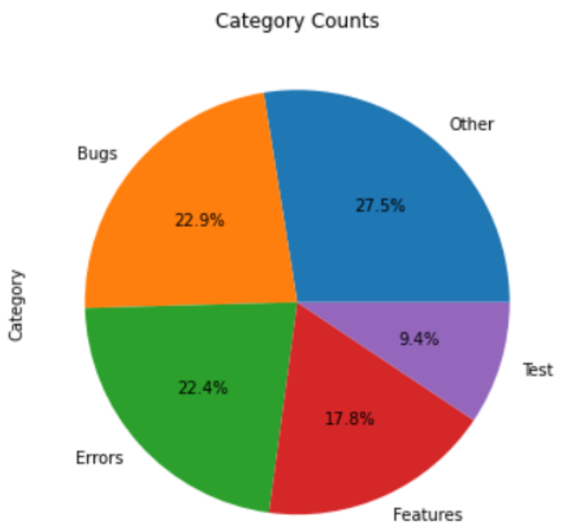


Figure 2.7 Category Counts

In the visualization for final dataset. We can say that developers asked more questions related to errors in pythons programming language and questions related to bugs in C++ programming language. The third occurring programming language to be observed is TypeScript.

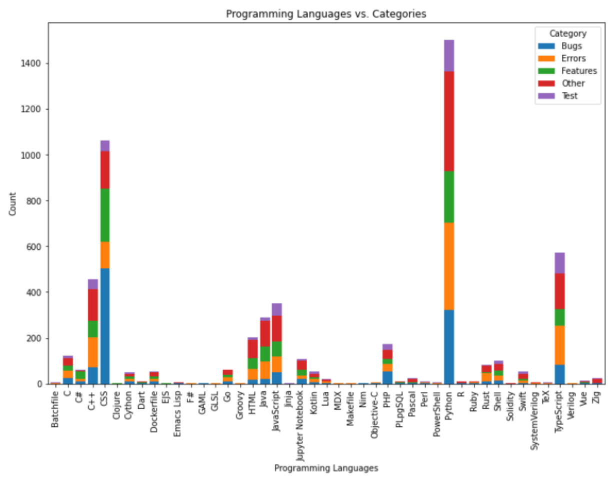


Figure 2.7 Programming Language vs Categories

2.5 About the Primary Contributor of this Research Question

Name: Roochita Ikkurthy
FSU ID: ri23b@fsu.edu

3 Conversation Length

3.1 Problem Statement

The objective of this research question is to provide an analysis of how the length of a ChatGPT conversation changes based on the type of issue presented this will provide significant insights for developers to understand the duration of conversation of a particular kind is expected. This analysis also helps in improving the user experience and improving ChatGPT.

3.2 Motivation

The primary motive of this research is to ascertain how the length of interactions with ChatGPT fluctuates in relation to the nature of the issues presented. This inquiry is significant as it offers critical insights for developers, illuminating the dynamics between the issue's complexity and the ensuing conversation duration.

3.3 Approach

3.3.1 Data Collection

For this analysis, we employed data from Snapshot20231012. This specific version was chosen due to its comprehensive nature, as it integrates data from previous snapshots, thereby streamlining the research process. After finalizing the snapshot to work, we decided to use all six snapshots. We went through the data structure to understand my required elements ("Prompts" and "Answers"), thereby understanding, and extracting them from the deeply nested data structure.

We started working on the "commitsharing.json" file, and the general idea at the start was to label the prompts in this data into categories and use this as an SVM-labeled data set to enhance it for all six JSON files combined files.

3.3.2 Data Preprocessing

In this phase of the project, we had to ask ChatGPT to provide some generic keywords and categories that could closely fit the prompts in them. This was just to preprocess the data into a phase from which we could set up the keyword generation using SpaCy NLP.

3.3.3 Issue categorization

For categorizing the issues, we used SpaCy NLP, which is a Python library for advanced Natural Language Processing (NLP), it is efficient and known for its speed. We used the preprocessed data that is cleaned and tokenized, which are the only requirements to be met before extracting features and training a model using SpaCy. This SpaCy NLP upon reading the preprocessed data of "Prompts", "Answers", and "Categories", provided clusters of keywords for each topic discussed in the prompts. It basically understands the sentiment and scenario where a conversation is taking place and extracts keywords that are both frequently used and meaningfully conveying the situation. We've generated 100 topics which have all the keywords associated with the prompts.

From these keywords that are generated, we manually determined a category name that best describes the issue. With these newly generated keywords and categories, we wrote a "categorization_prompt" function, which will provide the enhanced categories.

We checked the count of each category before generating the new keywords and the number of prompts in the category "Other" is dominatingly more than the rest of them. As explained in the presentation, the goal is to decrease the number of prompts that are going down into this category, thereby we kept checking the count frequently while we were adding the new keywords.

3.3.4 Length of Conversation Calculation

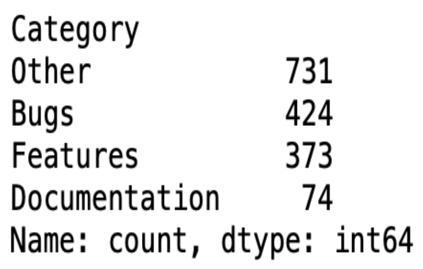
We calculated the length of conversation by counting the number of times a conversation goes to and from during the discussion. This was accurate as it is calculated based on the "Conversations" section of "Source" code.

3.3.5 Data Visualization

We used matplotlib and seaborn to interpret the data in a horizontal bar graph, which demonstrates the categories and average length of conversations.

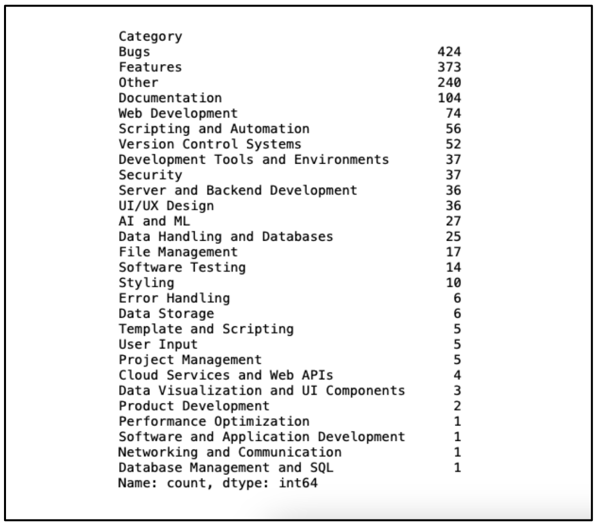
3.4 Findings

We measured the category count after the average conversation length calculation and found that the "bugs" and "features" are the top two categories in the number of prompts.



```
Category
Other            731
Bugs             424
Features         373
Documentation     74
Name: count, dtype: int64
```

Figure 3.1. Initial category count



```
Category
Bugs                                  424
Features                              373
Other                                 240
Documentation                         104
Web Development                        74
Scripting and Automation               56
Version Control Systems                52
Development Tools and Environments     37
Security                               37
Server and Backend Development         36
UI/UX Design                           36
AI and ML                              27
Data Handling and Databases            25
File Management                        17
Software Testing                       14
Styling                                10
Error Handling                          6
Data Storage                            6
Template and Scripting                  5
User Input                              5
Project Management                      5
Cloud Services and Web APIs             4
Data Visualization and UI Components    3
Product Development                     2
Performance Optimization                1
Software and Application Development    1
Networking and Communication            1
Database Management and SQL             1
Name: count, dtype: int64
```

Figure 3.2. Category Count after executing the new categories.

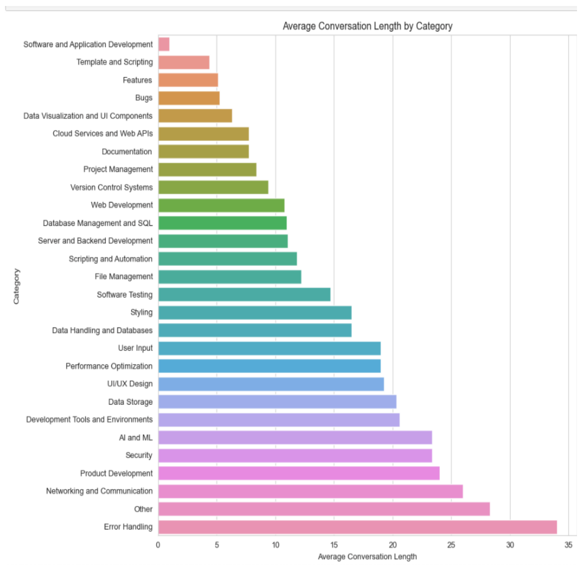
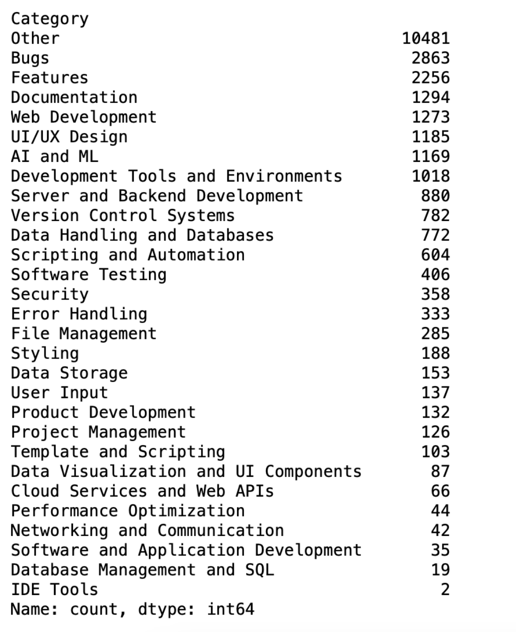


Figure 3.3. Results graph providing the average conversation length by category.

From the above bar graph, we understood that the top two categories with more length are "Error handling: and "other". This provides valuable insight into the developer's expectations of how the conversation length changes based on the type of issue presented, even though the "bugs" and "features" are more in the count they do not have conversations that are long enough as most of the prompts based of these categories are straight forward and requires minimum length to answer, while on the other hand, the "Error handling" and "other" categories might have the complex prompts that require more information sharing and obtaining, it can take more time to answer these queries and hence they have more length of conversation than any other category.

In the same way, here are results and findings from the analysis done on the entire Snapshot, including all the six Json files;



```
Category
Other                                   10481
Bugs                                     2863
Features                                 2256
Documentation                            1294
Web Development                          1273
UI/UX Design                             1185
AI and ML                                1169
Development Tools and Environments       1018
Server and Backend Development            880
Version Control Systems                   782
Data Handling and Databases               772
Scripting and Automation                  604
Software Testing                          406
Security                                  358
Error Handling                            333
File Management                           285
Styling                                   188
Data Storage                              153
User Input                                137
Product Development                       132
Project Management                        126
Template and Scripting                    103
Data Visualization and UI Components       87
Cloud Services and Web APIs                66
Performance Optimization                   44
Networking and Communication               42
Software and Application Development       35
Database Management and SQL                19
IDE Tools                                   2
Name: count, dtype: int64
```

Figure 3.4. category count of all the six Json files included

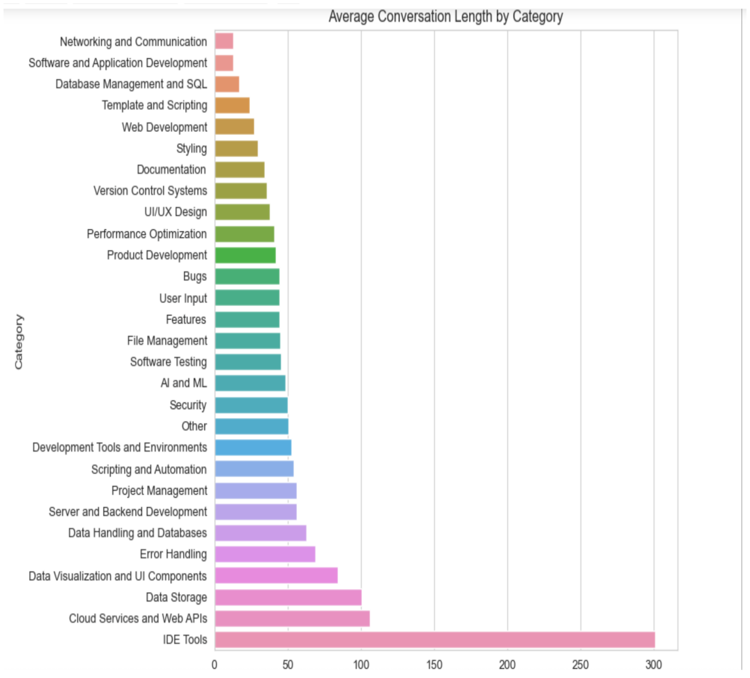


Figure 3.5 Average Conversation Length by category of all Json files included.

3.5 About the Primary Contributor of this Research Question

Name: Chaitanya Guntupalli
FSU ID: cg21bb@fsu.edu

4 Links

Video Presentation:
https://youtu.be/UH3dNdCywug

GitHub Repository:
https://github.com/srinijadharani/SEngg-Project/

5 Conclusion

The research we conducted resulted in the following findings.

1. **Code Versus No-Code Solutions:** We analyzed what kinds of prompts developers normally asked ChatGPT. We found out that ChatGPT was asked to write scripts, generate code, and give examples of a particular concept. ChatGPT mostly gave code solutions to questions like these, but there were also very predominant no-code solutions where the developer and ChatGPT had conversations about some concepts.
2. **Programming Language Versus Categories:** Our analysis provides a detailed overview of the recent GitHub repository interactions, shedding light on language diversity, prompt categorization, and potential areas for further exploration.
3. **Conversation Length:** The results indicate that the type of issue presented enforces conversation length change and provides insight for developers about how to predict the length based on the issue presented.

6 Acknowledgements

We sincerely thank Dr Sonia Haiduc and Mr. Shamim Seraj for their continuous support and encouragement. This research would not have been complete without their help.