ChatGPT Interaction Analysis Report

Abstract: This report presents an in-depth analysis of developer interactions with ChatGPT, focusing on conversation structure, engagement metrics, and language usage trends. By loading and combining data from 12 JSON files, we systematically explored key fields such as conversation counts, upvotes, and programming language preferences. Statistical analysis and visualization techniques were employed to identify patterns and derive meaningful insights.

The findings revealed that conversations typically require 400 to 570 turns to reach a meaningful conclusion, with an average of 4.85 turns per conversation, indicating efficient discussion handling. Analysis of engagement metrics showed that most conversations have an UpvoteCount of 0, with occasional outliers receiving significantly higher upvotes. The distribution of the "Number" field predominantly fell between 20 and 122, though outliers raised the average to 1,510.

Language analysis highlighted Python, TypeScript, and JavaScript as the most popular programming languages, reflecting their versatility and strong ecosystem support for machine learning, server-side logic, and web development. These insights offer valuable perspectives for improving ChatGPT interactions and understanding developer preferences.

We systematically explored key prompt characteristics, including common opening words, prompt length, and high-frequency technical keywords. Through statistical analysis and visualization techniques, we identified trends in developer queries and their impact on issue resolution.

By systematically analyzed key variables, including programming language preferences, conversation length, and keyword patterns. Statistical techniques and machine learning models were employed to extract meaningful insights.

GitHub repository :https://github.com/PPSHOU/542-Group1.git

Questions Addressed

1. **What is the typical structure of conversations between developers and ChatGPT? How many turns does it take on average to reach a conclusion?**
2. **What is the distribution and average of UpvoteCount and Number?**
3. **What are the top 3 most popular languages?**
4. **Can we identify patterns in the prompts developers use when interacting with ChatGPT? Do these patterns correlate with the success of issue resolution?**
5. **How accurately can we predict the length of a conversation with ChatGPT based on the initial prompt and context provided?**

Methodology

1. Load and Combine Data

Utilized Python to efficiently load 12 JSON files containing data on discussions and issues. Leveraging the pandas library, combined these JSON files into a single cohesive dataset within a Jupyter Notebook environment. Conducted initial data exploration by displaying essential information, such as column names, data types, and memory usage of the combined DataFrame. Verified data integrity by checking for missing values and ensuring all files were correctly loaded.

2. Traverse Data and Extract Target Fields

Systematically traversed the combined dataset to extract specific target fields relevant to the analysis, including Type, RepoLanguage, Number, Prompts, Answer, Stat and UpvoteCount. Ensured data consistency by converting fields into appropriate types, such as integers for numeric fields and standardized categories for categorical fields. Special attention was given to the Conversations field, which contained list-structured data. Extracted the count of elements in each list and recorded this as a new field, Conversations_num, representing the number of conversations for each record. Furthermore, We split the prompts and answers, extract key words, and conduct deep analysis.

3. Perform Statistical Analysis and Visualization

Aggregated the extracted fields into a newly structured DataFrame for focused statistical analysis. Computed key statistical measures such as mean, median, and variance for numeric fields. Visualized the distribution of target fields by plotting bar charts and histograms, which provided a clear understanding of patterns and trends. Key insights derived from these visualizations were documented for meaningful interpretations, such as identifying the most common repository languages, trends in issue discussions, and the distribution of upvotes across records.

4. Machine Learning Analysis

We conducted keyword analysis to identify the most frequent keywords in developer prompts, explored correlations between prompt length, keyword occurrences, and upvote counts, and applied K-Nearest Neighbors (KNN) regression to predict answer token counts based on prompt structures and engagement metrics.

Results and Interpretation

Conversation Turn Analysis:

Upon analyzing the conversation data, it was observed that long conversations require **400 to 570 turns** to reach a meaningful conclusion or resolution. The **average number of turns per conversation** was found to be **4.85**, suggesting that although some conversations may involve a significant number of turns, most interactions are fairly concise. This pattern points toward efficiency in conversation handling, with many discussions being concluded after a relatively small number of turns.

Upvote Count and Number Distribution:

The analysis of the UpvoteCount revealed that the **majority of conversations have an UpvoteCount of 0**, indicating limited engagement or popularity for most conversations. Only a small proportion of conversations received upvotes greater than zero, highlighting a selective interest in certain discussions. Additionally, the **Number** distribution predominantly falls between **20 and 122**, but the average value is notably higher at **1,510**, primarily due to the influence of outliers. These observations underline the presence of **outliers** in both "UpvoteCount" and "Number" metrics, despite the majority of data points remaining within expected ranges. The presence of such outliers calls for further investigation to understand their impact and origin.

Language Popularity:

The top three most popular programming languages used in the dataset are **Python**, **TypeScript**, and **JavaScript**. These languages show a significantly higher usage frequency compared to other languages in the dataset. The dominance of these languages suggests that they are particularly suitable for the development and maintenance of ChatGPT. Their widespread popularity is likely due to their strong ecosystem support, versatility, and compatibility with machine learning, server-side logic, and web development tasks.

Prompt Patterns Analysis:

Common first words in prompts included **"how," "what," "can,"** indicating a strong preference for explanation-based queries. Debugging-related prompts contained keywords like **"error," "traceback"**, suggesting frequent troubleshooting requests. Action-oriented prompts often contained words like **"write," "find," "generate"**, reflecting a high demand for code generation.

From the plot, no strong correlation was found between prompt patterns and issue resolution success.

These findings suggest that while developers exhibit clear patterns in their prompts when interacting with ChatGPT, these patterns do not necessarily influence the resolution of issues. The analysis provides valuable insights for improving ChatGPT's response strategies and tailoring interactions based on developer needs.

Machine Learning Predictions:

The dataset was processed by defining key variables, including Type, RepoLanguage, Number, UpvoteCount, PromptTokens, AnswerTokens, Newline, and KeywordCount. Using these features, a K-Nearest Neighbors (KNN) regression model was trained to predict answer lengths. The optimal k value yield an R^2 score of 0.9157, demonstrating strong predictive capability for answer token counts based on prompt characteristics. Mean Absolute Percentage Error (MAPE): 15.82%, demonstrating the potential for modeling ChatGPT responses based on developer query structures.

These findings suggest that while prompt patterns influence engagement levels, they do not significantly impact issue resolution outcomes. Further research could explore additional contextual factors to refine predictive models for developer interactions with ChatGPT.