Investigating Developer Interactions with ChatGPT
DATA 542

Team 6 Members:
Vidal Mendoza Tinoco, Jane Shen, Qijia Zheng

Introduction

This project explores the DevGPT dataset (NAIST, 2023), which captures developer conversations with ChatGPT sourced from various GitHub issues and discussions, with the goal of identifying interaction patterns. The analysis focuses on understanding the types of topics developers bring to ChatGPT, whether prompt structures or linguistic features influence the likelihood of issue resolution, and whether the initial prompt and context can predict the length of a conversation. To investigate these questions, multiple natural language processing (NLP) techniques were applied, along with supervised and unsupervised learning methods to analyze conversational trends and prompt characteristics. The results highlight key insights into developer interactions with ChatGPT, as well as the challenges associated with processing and extracting meaningful information from unstructured text data.

Pre-processing

The conversations analyzed in this project were sourced from GitHub issues and discussions, as these were most relevant to the research questions. To manage computational load, the dataset was restricted to a snapshot from August 31, 2023. To improve NLP performance, only conversations where the primary detected language of the prompts was English were retained. The final issues dataset contained 1,431 prompts from 313 unique English conversations, while the final discussions dataset contained 154 prompts from 41 unique English conversations.

Code Reuse

The code for tokenization, bag-of-words conversion, and LDA model construction is adapted from Samuel Cortinhas' Kaggle notebook "NLP6: Topic Modelling with LDA" (available under the Apache 2.0 license). Modifications were made to suit the specific requirements of this project.
A copy of the Apache 2.0 license can be found at http://www.apache.org/licenses/LICENSE-2.0.

Analysis

Section I: Commonly Presented Issues

This section explores the research question, **“What types of issues (bugs, feature requests, theoretical questions, etc.) do developers most commonly present to ChatGPT?”** To address this question, the analysis focuses on the issues through the Conversation data consisting of Prompts and Answers using NLP techniques. After cleaning, tokenizing and lemmatizing the data we can convert the words into a bag-of-words representation. We trained the LDA models to identify 10 topics each for Prompts and Answers to find the specific types of issues developers most commonly present to ChatGPT.

Methodology

The data extraction and cleaning process begins with removing newline characters from prompts and answers, ensuring the text is formatted correctly for further processing. To prepare the text for analysis, a custom tokenizer is applied, incorporating lemmatization to convert words to their base forms while filtering out non-alphabetic characters, punctuation, and stop words. Only adjectives, nouns, and verbs are retained to preserve meaningful linguistic structures. The tokenized data is then mapped to unique IDs using Gensim's Dictionary, with infrequent words, those appearing fewer than three times or with a frequency below 0.5, being removed to reduce noise. This refined dataset is then transformed into bag-of-words representations, where each document is represented as a list of word ID-frequency tuples.

To uncover underlying topics in developer interactions with ChatGPT, Latent Dirichlet Allocation (LDA) models are trained using the bag-of-words representation, extracting 10 distinct topics from the dataset. After training, topic distributions are calculated by aggregating probabilities across documents. Finally, word clouds are generated to visually represent the most significant words within each topic, providing an intuitive way to interpret the primary themes discussed in developer prompts and responses.

Results

The top ten topics for Prompts and Answers in issue Conversation can be found in the Appendix. Based on the analysis result, the most common issues developers present to ChatGPT are:

1. Code Debugging and Errors
2. Code Implementation and Optimization
3. Theoretical and Conceptual Questions
4. Database and Data Handling
5. Tool and Workflow Assistance
6. Feature Requests and Enhancements

Conclusion

The analysis of Prompts and Answers using NLP techniques successfully identified 20 key themes in the conversations. The results highlight common issues to seek assistance with, such as code debugging, implementation, theoretical questions, database handling, tool assistance, and feature requests.

Section II: Patterns in Prompts and Issue Resolution

This section explores the research question, **"Can we identify patterns in the prompts developers use when interacting with ChatGPT, and do these patterns correlate with the success of issue resolution?"** Initial data exploration revealed significant variation in developer prompts, ranging from specific troubleshooting queries to unclear directives and raw error traces. Based on this variability, it was hypothesized that successful issue resolution might correlate with clear, well-structured questions and directives. To test this, LDA and k-means clustering were employed to examine whether prompts from closed issues displayed distinct structural patterns compared to prompts from open issues, indicating a link to successful issue resolution.

Methodology

Additional preprocessing was applied to the issues dataset to remove non-ASCII characters, non-English prompts, and null prompts. Due to the high variability in prompt structure, traditional regex-based methods were insufficient for distinguishing natural language from code snippets. To address this, the Google Gemini API (Google, 2025) was used for more accurate identification and separation of code chunks from raw prompts. Final cleaning steps included the removal of URLs and file paths using regex to improve data quality.

LDA was first applied to assess the feasibility of clustering by identifying distinct topics within the prompts. A custom tokenizer was implemented to retain specific position tags: Adjectives (ADJ), Nouns (NOUN), Verbs (VERB), Auxiliaries (AUX), Pronouns (PRON), Adverbs (ADV), Determiners (DET), Subordinating Conjunctions (SCONJ), Interjections (INTJ), Adpositions (ADP), Particles (PART).

Following LDA, K-means clustering was applied to the closed issue prompts, aiming to group them by similarity based on position tags. Both embeddings and TF-IDF were explored as feature representations, with TF-IDF proving more effective. The TF-IDF vectorizer limited the maximum number of features to 100, filtering out infrequent words and reducing noise while preserving key patterns. This also enabled n-gram-based clustering for more precise prompt characterization (e.g., distinguishing "how can" from "explain this").

Results

Careful selection of relevant position tags, combined with modified stopwords, improved LDA model coherence from -3 to -5.63, indicating better topic separation. The resulting word clouds for the top ten topics highlighted clear patterns, showing groupings distinguished by WH-question words (e.g. “why”, “what”, “how”), imperative verbs (e.g. “give”, “implement”, “fix”), and technical domain nouns (e.g. “code”, “server”, “model”).

The optimal clustering resulted in five distinct clusters (sizes: 139, 62, 59, 55, 31), confirmed through visual analysis. To characterize the clusters, positional tag distributions were analyzed, and key distinguishing words were identified:

1. Code-focused queries: Lower word usage, highest occurrence of "code."
2. Implementation queries: Dominated by "how," "can," and "use."
3. Clear user intent: Strong use of "want" indicating expected results.
4. Definition-focused: Primarily "what," with shorter, probing prompts.
5. Exploratory reasoning: Higher usage of "why," "which," "if," "would," and "could," indicating comparative and hypothetical considerations.

Details of each cluster, along with sample prompts, can be found in Appendix A. Visualizations comparing position tag and target word frequencies across subsets and clusters can be found in Appendix B.

Conclusion

Comparing overall word counts in prompts between closed and open issues revealed that closed issue prompts contained significantly more structured questions, despite having fewer prompts per conversation on average. Specifically, closed issue prompts exhibited a higher rate of well-defined inquiries, such as direct code-related questions, implementation guidance,

clarifications of concepts, and exploratory reasoning. This suggests a correlation between the prevalence of focused, purposeful questions in closed issue prompts and successful issue resolution, whereas open issue prompts tended to be less structured, often lacking clear intent or direction, which may contribute to unresolved issues.

Section III: Predicting Conversation Length

This section addresses the research question, **"How accurately can we predict the length of a conversation with ChatGPT based on the initial prompt and context provided?"** The research investigates whether engineered features derived from textual content and contextual metadata can reliably predict conversation length. Three models, random forest, gradient boosting (with hyperparameter tuning), and a neural network, were explored and evaluated using Mean Absolute Error (MAE) and the R squared score.

Methodology

Data from discussion and issue datasets were merged to create a unified, non-redundant dataset. Each conversation was uniquely identified by adding a source prefix, and the data underwent thorough cleaning, including date conversion and the handling of missing values, to define the target variable as the number of prompts (conversation length). A comprehensive set of features was engineered from the text. These included textual metrics (word and character counts), indicators of code presence derived via regular expressions, sentiment scores computed with TextBlob, and advanced NLP features, such as counts of nouns, verbs, adjectives, lexical diversity, and named entities, extracted using spaCy. Additionally, with suggestions from OpenAI (2024) and DeepSeek (2024), the code was refined to be better formatted and simplified, improving readability and efficiency.

In addition, embeddings (Word2Vec and SentenceTransformer) and topic features (via Latent Dirichlet Allocation) were generated. The code was organized into clear sections covering data loading and cleaning, feature extraction, embedding and topic modeling, and model training and evaluation. The dataset was split into training and test sets, models were trained, and performance metrics were computed. Visualization of results was accomplished by generating and saving plots (for MAE, R squared, and neural network residuals) alongside a CSV summary of model results. For the implementation of the neural network model, references such as Keras Sequential Model Guide (Chollet, 2023) were used, along with suggestions from OpenAI (2024) and DeepSeek (2024) to optimize the architecture and training process.

Results

The relatively low R^2 values, approximately 0.20 to 0.35 for the best-performing model, indicate that the current feature set explains only about 20–35% of the variability in conversation lengths. This outcome is not entirely unexpected, given the inherent complexity of predicting conversation length based solely on the initial prompt and available context, where many unobserved factors, such as user behavior and external influences, play a role.

The MAE, representing an average error of about 9–10 prompts, must be interpreted in context. If the typical conversation length is around 100 prompts, the error corresponds to roughly a 10% deviation, which can be considered reasonable for a baseline. However, if conversations are significantly shorter, the same absolute error would be more critical. Overall, the tree-based models, especially random forest, appear to capture the available information

better than the neural network, suggesting that ensemble methods are particularly well-suited for this task.

Model	MAE	R2
Random Forest	9.683	0.345
Gradient Boosting	9.977	0.315
Neural Network	10.766	0.211

Conclusion

In conclusion, predicting the length of a ChatGPT conversation based solely on the initial prompt and context is a challenging task. While the best model (random forest) achieved an MAE of approximately 9.68 and explained about 35% of the variance, these metrics highlight both the potential and the limitations of the current approach. The low R squared values reflect the complexity of the problem and the likelihood that important factors remain unmodeled, whereas the MAE must be interpreted relative to the overall scale of conversation lengths. These results provide a baseline for further research, suggesting that more advanced feature extraction and modeling techniques are needed to improve predictive accuracy. Future enhancements, including richer data and more sophisticated models, are likely to yield better performance and deeper insights into the dynamics of conversation length.

Conclusion

This study examined how developers interact with ChatGPT, focusing on discussion topics, prompt structures, issue resolution patterns, and the ability to predict conversation length. The analysis revealed that clear, well-structured prompts are more likely to lead to successful issue resolution, whereas vague or ambiguous queries often remain unresolved. Topic modeling and clustering techniques identified key themes in developer queries, ranging from debugging and implementation guidance to theoretical and exploratory questions. These insights provide a deeper understanding of how developers utilize ChatGPT and the types of assistance they seek in software development workflows.

In predicting conversation length, machine learning models, particularly random forest, outperformed other approaches but still exhibited limited predictive accuracy (R squared = 0.35, MAE = 9.68 prompts). This suggests that while certain textual and contextual features contribute to conversation length, many unmodeled factors, such as user behavior and follow-up interactions, play a significant role. Future research should explore advanced NLP techniques, transformer-based embeddings, and richer contextual metadata to improve predictive performance. Additionally, integrating conversational flow analysis and user engagement patterns could provide deeper insights into AI-human interactions, refining the modeling of ChatGPT discussions.

References

Chollet, F. (2023, June 25). The Sequential model. Keras. Retrieved from https://keras.io/guides/sequential_model/

DeepSeek. (2024). DeepSeek (R1 version) [Large language model]. https://chat.deepseek.com/

Google. (2025). Gemini API | google AI for developers. Google. Retrieved from https://ai.google.dev/gemini-api/docs

OpenAI. (2024). ChatGPT (4o version) [Large language model]. https://chat.openai.com/chat

Software Engineering Lab @ NAIST. (2023). *DevGPT* [GitHub Repository]. GitHub. Retrieved from https://github.com/NAIST-SE/DevGPT

Appendix A: Tables

Table A1: Top 10 Topics of Prompts

Topic	Top Keywords	Proportion
0	line, module, import, file, kwargs, fix, quickedit, args, jekyll, lib	10.4%
1	string, public, new, class, use, code, need, set, private, database	18.1%
2	token, language, stream, datum, text, option, term, error, include	9.6%
3	def, return, null, response, const, value, loss, import, false, output	7.5%
4	usage, range, span, return, list, text, item, server, client, info	9.0%
5	src, pm, student, local, schedule, language, model, program, operator, test	9.6%
6	return, file, user, plugin, create, image, message, code, email, use	12.2%
7	line, package, lib, opt, site, transformer, let, model, return	7.1%
8	line, lib, package, local, run, dist, build, error, file, issue	7.5%
9	type, thread, task, string, title, page, matrix, use, mapping, connection	9.1%

Table A2: Top 10 Topics of Answers

Topic	Top Keywords	Proportion
0	user, message, datum, stream, use, need, handle, protocol, provide, key	11.9%
1	use, need, method, request, update, server, application, component, example, tag	9.2%
2	search, document, phrase, source, datum, number, unique, word, bit, provide	9.1%
3	table, query, department, salary, column, string, name, use, base, employee	8.5%
4	file, code, directory, run, error, issue, package, use, module, command	21.6%
5	model, code, provide, method, function, class, file, database, logic, use	12.0%
6	context, issue, thread, use, ensure, provide, object, access, need, error	6.5%
7	die, der, click, den, dass, file, option, migration, window, und	6.6%
8	library, use, code, event, load, second, version, check, build, dynamic	7.3%
9	code, step, image, project, generate, prompt, create, error, run, package	7.4%

Table A3.

Defining cluster characteristics and sample prompts

Cluster	Key Distinction	Sample prompts
0	Lower overall word usage but highest occurrence of "code"; possibly code-focused queries	i. “Can you explain this piece of code?” ii. “I currently have this code:”
1	Dominated by "how," "can," and "use," focused on implementation queries	i. “How do I docker build this?” ii. “Using sql.js, how can I load extensions such as generate_series?”
2	Strong use of "want," indicating clear user intent and expected results.	i. “I want us to engage into solving a bug:” ii. “I want to add an option to my CLI tool for importing CSV files into a database”
3	Primarily "what"; prompts in this cluster were shorter and focused on definitions and probing questions.	i. “What kind of vector databases exist?” ii. “What does an invalid or unexpected response look like?”
4	Higher usage of "why," "which," "if," "would," and "could," suggesting a focus on exploratory reasoning, comparative evaluation, and hypothetical consideration.	i. “Why not implement it as a http proxy” ii. “Why does `(*it).a` work but `it->a` doesn't compile?”

Appendix B: Figures

Figure B1

Word Cloud of Prompts

Figure B2

Word Cloud of Answers

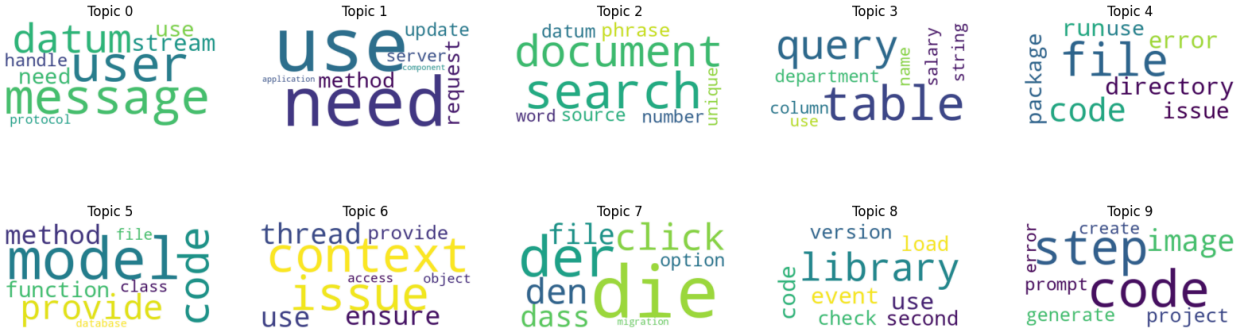

Figure B3.

Comparison of total normalized counts of position tags and target words in closed and open issue prompts

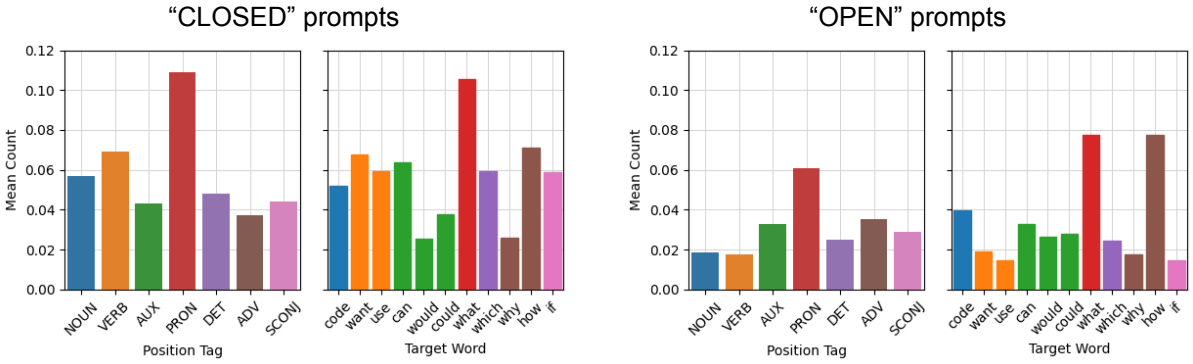

Figure B4.

Comparison of normalized counts of position tags and target words between clusters of closed issue prompts