A Human-Centred Approach to Designing Effective Large Language Model Based Tools for Writing Software Tutorials

Avinash Bhat

School of Computer Science
McGill University
Montreal, Quebec, Canada

December 2023

A thesis submitted to McGill University in partial
fulfillment of the requirements of the degree of
Master of Science (Thesis)

Contents

Abstract

Despite the proven capabilities of Large Language Models (LLMs) across diverse tasks, their effectiveness in technical writing – a domain that demands factual accuracy and specificity in addition to other writing goals, such as coherence and conciseness – remains unexplored. Moreover, a comprehensive understanding of the experiences and processes of humans when interacting with these models is lacking. To address this gap, we explored the opportunities in leveraging LLMs for tutorial writing from a human-centred perspective. Our exploration involved a study with professional software tutorial authors (N=7) on OpenAI playground to understand their existing tutorial writing practices and their potential approach to using LLMs for tutorial writing. Our study reveals that the interactions of the writers with these models are motivated by their existing practices of content research, editing, and verification, often guided by self-imposed quality standards. However, the unreliability of the LLMs and the poor usability of the interfaces leads to an incorrect mental model about the capability of the models. Guided by these insights, we propose design guidelines and a low-fidelity prototype emphasizing the transparency and usability of human-LLM interaction that can be integrated into the current tutorial writing workflows. Overall, our research underscores the significance of human-centric approach in designing language technologies to enhance tool usability and usefulness for users. Specifically for tutorial writing, our study identified key workflows that could be further explored for LLM integration, indicating potential directions for future research in this domain.

Résumé

Malgré les succès récents des grands modèles de langage (LLM, de l'anglais Large Language Models) dans diverses tâches, leur efficacité pour la rédaction de documents techniques – une tâche qui requiert de l'exactitude et de la spécificité en plus d'autres objectifs de style, tels que la cohérence et la concision – demeure inexplorée. De plus, nous manquons une compréhension globale de l'expérience et des processus humains lorsqu'ils interagissent avec les LLM. Pour combler ces lacunes, nous avons exploré des opportunités d'utiliser les LLM pour la rédaction de tutoriels, d'un point de vue centré sur l'humain. Notre exploration inclut une étude avec des auteurs professionnels de tutoriels (N=7) sur OpenAI playground pour comprendre leurs pratiques de rédaction actuelles et leur approche hypothétique quant à l'utilisation des LLM. Notre étude révèle que les interactions des auteurs avec les LLM sont motivées par leurs pratiques courantes de recherche, d'édition et de vérification de contenu, souvent guidées par des normes personnelles de qualité. Cependant, les lacunes des LLM et la convivialité déficiente des interfaces mènent à des modèles mentaux incorrects de la capacité des LLM. Guidés par ces apprentissages, nous proposons des suggestions de conception et des schémas de conceptions pour un prototype, mettant l'accent sur la transparence et l'ergonomie de l'interaction humain-LLM, qui peuvent être intégrés dans les flux de travail actuels de rédaction de tutoriels. Dans son ensemble, notre recherche souligne l'importance d'une approche centrée sur l'humain dans la conception des technologies linguistiques pour améliorer l'utilisabilité et l'utilité des outils pour les utilisateurs. Plus spécifiquement pour la rédaction de tutoriels, notre étude a identifié des processus clés qui pourraient être davantage explorés pour l'intégration des LLM, indiquant des directions potentielles pour

la recherche future dans ce domaine.

Acknowledgement

I extend my deepest gratitude to Professor Jin Guo for her invaluable mentorship and feedback throughout my master's journey, which massively influenced my academic perspective. Her consistent support and guidance were key to my work.

I thank my collaborator, Disha, without whom most of the ideas in this project would never take shape. I thank Prof. Guo, Prof. Robillard and my labmates at the Software Technology Lab for fostering a growth-oriented environment that greatly influenced my research. A huge shoutout to Jazlyn, Mathieu, Deeksha and Justine, who were my go-to people for anything related to research and beyond. I thank Mathieu for his help with the French résumé and Deeksha for her guidance on qualitative coding. I would like to thank the department staff, including Ann, Sheryl, Kamini, Ron and Corey, who patiently helped me with various administrative and technical tasks throughout the past two years.

I am indebted to my parents for their constant love and support and forever grateful for the safety net they provide that allows me to aim high. I cannot possibly list all my friends and their role in my successes. I thank all of them for cheering me in ways they might not even realize. A special shoutout to my roommates, Robin and Manas, who made the stress of being in a foreign country more bearable and, to a large extent, enjoyable.

Last but definitely the most, I am truly grateful to Divya for being by my side through the toughest times and loving me despite all my antics.

List of Figures

List of Tables

Chapter 1

Introduction

The rapid development of large language models (LLMs) has significantly changed the technology landscape. LLMs have expanded capabilities essential for numerous language tasks [61]. This rise in capability has sparked innovations like prompt engineering, which refines how we communicate with these models [44, 37]. LLMs are now widely implemented across various applications [30] and are now pervasive, with several daily products integrating them[1]. As their influence grows, there's an increasing need to comprehend both their potential and the specifics of their functionality.

Regardless of the efficiency, the utility of these models is in producing human usable outputs. LLMs can unintentionally show biases due to their training data, and a high non-interactive effectiveness of the model does not always indicate a good human-LLM interaction [34]. Efforts like ChatGPT use human feedback to address this [11, 2]. It's clear that the effectiveness of LLMs isn't just about their capabilities but also how they're designed with users in mind and how they account for human input.

Human-AI collaboration and AI-assisted writing have been studied extensively [20, 47, 28]. While the focus in literature is generally on creative writing [20], professional writing tasks, such as software tutorial writing, which combines technical writing with creativity, are not as extensively studied. Acceptance of AI in such professional environments and tasks hinges on creating systems that are both transparent and aligned with user needs [46, 7].

Developing software tutorials is a complex task. Writers must comprehend intricate technical aspects, assess the knowledge level of their audience, and convey information

[1] https://openai.com/blog/chatgpt-plugins

clearly. Additionally, with software constantly changing, tutorials need frequent updates to remain up-to-date and accurate [27, 54]. Given the advancements in LLMs, their inclusion in tutorial writing is promising. LLMs possess the capability to generate relevant text and code examples, thereby streamlining the process for authors. Their capability to quickly generate insights from a large amount of information and produce coherent, contextually appropriate outputs can enhance the efficiency of the writing process. While LLMs seem like a valuable addition to the writing process, their integration presents challenges. A primary concern is their potential over-dependence on training data. This can lead to them suggesting outdated solutions or not recognizing newer best practices. There's also an issue of biases that could be inherent in their training data, risking the generation of incorrect techniques and perspectives. In order to effectively leverage the capabilities of the LLM, it's essential that LLMs operate interactively and allow for human input and feedback. A human-AI collaboration ensures that the produced content is both accurate and aligned with the author's intent. Therefore, while LLMs can be powerful tools, their integration necessitates a design that prioritizes human-AI collaboration.

Given the promising intersection of LLMs with tutorial writing and the evident gap in practical insights, we tried to understand the feasibility and challenges in integrating the LLMs in practice. Through directly interviewing software tutorial authors, we aimed to understand the unique workflows and challenges that exist in the tutorial writing space to identify the potential advantages offered by LLMs. Based on the reflections gathered during the interview phase and interactions observed during the hands-on exploration phase of the study, we propose two recommendations. First, a set of design guidelines that account for various workflows in the tutorial authoring domain and potentially leverage LLMs effectively and realize this in a conceptual design. Second, we discuss a framework that can potentially assist tool developers in categorizing different user interactions that can arise in a human-LLM interaction setting to design better LLM-based tools.

In summary, our contributions are as follows.

1. Based on direct interviews with software tutorial authors, we distill the feasibility and potential benefits and pitfalls of human-LLM collaboration for a tutorial writing task through an understanding of the unique workflows and challenges within the

domain.

2. We incorporate the insights from the interview and hands-on exploration phases to establish comprehensive design guidelines. These guidelines consider user preference in tutorial authoring workflows and highlight strategies for effective integration of LLMs, with the aim of bridging the gap between potential and practical applications.
3. We introduce and discuss a potential framework intended to guide both tool designers in enabling intuitive and effective LLM-based tools.

The organization of the thesis is as follows. In Chapter 2, we provide a background of LLMs and discuss their applications in the writing space. Following this, we discuss various cognitive models used in writing and interaction that motivate the design of AI-assisted writing tools. We also discuss related tools for tutorial authoring and maintenance-related activities. In Chapter 3, we discuss our study design, including the recruitment strategy, interviews and analysis. In Chapters 4 and 5, we present our findings. We develop the design guidelines and a conceptual design based on these findings, which we discuss in 6. Finally, we present a comprehensive discussion of our results, and a potential framework for designers in 7.

LLMs are reshaping the technological landscape, especially in language-related tasks. While they possess vast capabilities, they are not always straightforward to use, especially in producing content for human audiences. We aim to investigate the challenges and potential of using LLMs in software tutorial writing. Finally, we offer design guidelines derived from our findings and present a framework to help tool designers and developers optimize LLM-based tools.

Chapter 2

Background and Related Work

Our work is inspired by the current body of research pertaining to the cognitive models and processes related to writing and interaction, the use of AI in writing, design of intelligent writing assistants, especially for software tutorial writing. In this section, we discuss how our work is related to works in those aspects in detail.

2.1 Cognitive Models of Writing and Interaction

Much of the literature on writing assistants builds on existing cognitive models of writing. Notably, four prominent models are the text-oriented model, the process model, the sociocognitive model, and the social practices model [14]. Each of these models emphasizes different aspects of writing. The text-oriented model focuses on the finished piece of text and its linguistic attributes. In contrast, the process model highlights the cognitive processes that an individual writer performs to produce text. The sociocognitive model sees writing as an intricate activity system, underscoring the interactions between the writer, technology, and the surrounding environment. The social practices model, with its anthropological roots, often relies on ethnographic studies of literature to understand how text production takes place at the intersection of society, culture, and language.

Given the clear implication to interaction design, HCI research primarily focuses on the process model, in particular the Cognitive Process Theory of Writing proposed by Flower and Hayes [19] and later refined by Hayes [26]. This theory delves deep into the various processes of writing. It highlights the task environment, which encompasses external elements like the writing topic, the evolving text as an output of the writing process, the tools used, and the sources of information. It underscores the importance of

the writer's long-term memory, which acts as a repository of the writer's knowledge on the subject, audience awareness, and various writing strategies. Within the active writing processes, three major components include planning – which deals with idea organization and goal-setting, translating – where ideas are converted into text and finally, reviewing – the stage of refining and revising the text. These processes are intertwined and can occur out of order. Hayes [26] later extended this model, introducing the role of working memory in the writing process and highlighting the importance of factors like motivation and transcription methods, such as typing versus handwriting. Several studies in intelligent assistant research [3, 49, 21, 12] draw heavily from the cognitive process theory to devise novel interactions between the writer and the AI assistant. While we acknowledge the importance and existence of various processes outlined by the Cognitive Process Theory in the tutorial writing task, we want to identify any additional processes corresponding to various additional artifacts that are developed during tutorial writing. This motivates the interview phase of the user study.

Coming to the cognitive models for interaction design, one of the most influential models has been the Seven Stages of Action framework [41]. The framework is a cyclical model that begins with goal formulation, followed by action planning, specification, execution, perception, interpretation, and finally, evaluation against the initial goal. The framework is supported by three levels of mental processing termed visceral, behavioural, and reflective. The visceral level captures the instinctual reactions and is often driven by a product's aesthetics. The behavioural level focuses on the product's usability, where the feedback arises from the ease or difficulty of use. The reflective level delves into reflections on the product's significance. In addition, seven design elements are proposed based on this interaction model. Discoverability ensures users can identify possible actions. Feedback provides clear action outcomes from the product to its user. A conceptual model is a mental model that the user forms about their potential interaction with the model. Affordances suggest potential interactions that are possible with the product. Signifiers provide clear action indicators of the affordances. Mappings ensure logical relationships between controls and outcomes. Constraints set user interaction boundaries. These design elements have extensively been used in design literature.

2.2 Language Models and Their Design Considerations

Recent AI advancements in natural language tasks are accelerated with the introduction of transformer architecture and self-attention mechanisms [52]. These, combined with the concept of pre-training and fine-tuning for context-aware word representations [42], led to models like BERT [13] and the GPT series [Radford et al.] of models. Scaling models based on computation, model parameters, and training data size improves their performance [31], and such scaled language models are termed as Large Language Models (LLMs). One notable trait of LLMs is their emergent ability [55], where, as models get bigger, they show new and unique capabilities not observed in smaller models. These emergent abilities are generally useful to achieving state-of-the-art results in several downstream natural language tasks [6]. Zhao et al. [61] identified three key emergent abilities of LLMs – in-context learning, instruction following, and step-by-step reasoning. The field of prompt engineering [44, 37] generally leverages these techniques to design effective prompts, which are user inputs used to elicit relevant content from the LLM. In-context learning involves giving the model few-shot examples [15]. For more complex tasks, step-by-step reasoning techniques like chain-of-thought prompting [56] have been shown to achieve good results. However, prompt engineering can also reveal biases from the pretraining data, sometimes producing problematic content. To counter this, researchers are incorporating human feedback into the training process [11, 2] which have been attributed for the recent success of models like ChatGPT [1].

Improvement in language model capabilities necessitates a focus on usability research to foster better user acceptance of intelligent systems and acknowledge their potential [36]. AI's usefulness depends not only on its capabilities but also on users' willingness and ability to integrate AI into their workflow [46], which directly depends on their perceived usability. However, there are several issues pertaining to the usability of intelligent systems. Human-level issues, such as the need for humans to learn interaction methods or change their existing processes while performing tasks [29], can be challenging to address due to their overall diversity and user-specific nature. Conversely, issues primarily at the system level, like the interface's limited control over user

[1] https://openai.com/blog/chatgpt

interaction or the model's inherent unpredictability and complexity, are more addressable. The system-level issues generally inhibit any potential productivity gains [59]. While some research suggests that imperfections arising due to unpredictability can encourage creativity, it largely depends on the skill and confidence of the users and the availability of tools and interfaces to rectify AI-generated errors [58]. Moreover, users may value the quality of the end result over productivity and distrust the capability of the AI or its control mechanisms to match their style, resulting in an adoption barrier [5]. This underscores the importance of designing usable interfaces that amplify AI capabilities while respecting diverse user needs, including productivity, accuracy, control, ethics, and social implications.

The design of effective human-AI collaborative systems is a complex task due to the uncertainty of the AI's capabilities and the complexity of its output [59]. Over the years, several studies have proposed guidelines for designing human-AI interaction [1, 48, 8, 57] which have been received positively by the research and design community [60]. In the context of generative systems, Weisz et al. [57] suggest proactive design principles relating to generative AI's inherent traits, such as imperfection, exploration, control, and facilitating understanding through mental models and explanations. Conversely, Buschek et al. [8] highlight nine potential pitfalls in human-AI co-creation and propose measures to address these should they occur during collaboration with AI. Despite the contrasting perspectives, both studies stress more responsible and user-centric frameworks for human-AI collaboration.

Human-AI collaborative systems for writing have been studied across several dimensions, such as types of users, writing domains, and various interaction paradigms. Gero et al. [20] developed a design space for writing support tools guided by the level of constraint or specificity needed by the task and cognitive processes such as planning, translating and reviewing [19]. Their analysis of 33 systems from the literature identified a gap in planning and reviewing aspects, particularly for highly constrained tasks. However, their synthesis does not include tutorial writing, a domain characterized by open-ended while highly constrained writing tasks, presenting an opportunity to explore the potential for automation. Studying the specific processes of writing domains is essential for tailoring appropriate intelligent support mechanisms. For instance, an investigation into

expository writing revealed its evidence-driven nature, necessitating support requirements like evidence collection, information synthesis, and facilitated text composition [47]. Similar investigations have been conducted for poetry [23, 10] and drama scripts [38] outlining unique support requirements. User demographics are another aspect that contributes significantly to the decision process of intelligent system design. For example, a study involving parents assisting their children in an AI-aided interactive story-rewriting process underscored the potential of AI platforms to not only empower children but also foster parent-child relationships and navigate the complexities of the narrative [34]. Understanding user needs is critical for the effective integration of AI into human processes. Gero et al. [22] address this by focusing on writers' needs, perceptions of potential support actors, and their values, albeit such an approach caters to specific user groups. Conversely, a more general strategy involves understanding the systemic complexities that inhibit user comprehension, such as transparency, interpretability, and explainability. By identifying the corresponding user mindsets, levels of involvement, and expected knowledge outcomes, mechanisms to provide tailored support, broadening our ability to facilitate AI comprehension across diverse user groups can be devised [18, 7].

In the creative writing domain, the aspects of leveraging AI for maintaining the authorial style, improving writer inspiration, and assisting in various stages of writing have been extensively studied. It is widely observed that AI struggles to retain the distinctive style and voice of an author [28, 32, 5], leading to the development of novel interfaces like line sketching to provide better narrative control to the authors [12]. Additionally, Sun et al. [50] introduced an interactive writing tool that refines text using author-specified rhetorical tags, providing a means to maintain and accentuate the author's unique voice and style. Studies have investigated how leveraging AI into various writing stages influences writers. For instance, Roemmele [45] demonstrated that writers are generally inspired by AI-generated content, leading to improved content quality even if the AI content isn't directly used. This idea is further supported by Gero et al. [21], who found that writers often use AI-generated content for inspiration, translation assistance, and to gain diverse perspectives. In an analysis of pre-writing task, Wan et al. [53] identify the AI being used for ideation, illumination, and implementation while drafting initial versions of stories or slogans. Furthermore, they shed light on the division of roles and

initiatives between AI and humans, noting that humans appreciated AI's initiative in moments of creative blocks. The potential of AI as creative constraints, idea generators, and even as sources of antagonistic suggestions have been recognized [9]. Such insights have led to a recognition of AI as a collaborative and iterative partner in the creative process rather than a one-shot solution [16].

2.3 Tool Support for Software Tutorial Authoring

Tooling support for tutorial authoring primarily tries to address the accuracy, executability and comprehensiveness of the tutorials by generating and integrating artifacts. Current research on text-based tutorial authoring focuses on code interspersed with textual explanations and other artifacts like videos, images and other linked content. The interview study performed by Head et al. [27] highlights the importance of maintaining consistency across the code artifacts, specifically code snippets and their explanations. They propose a tool, Torii, to simplify this process by mapping it to a source implementation and ensuring tutorial consistency by propagating any changes in the source code to the tutorial. Moreover, this guarantees the accuracy and executability of the tutorial. Colaroid [54] follows an alternative approach of allowing the authors to create content in stages rather than having to develop the reference implementation preemptively. This approach enables the tool to offer multiple code versions and capture snapshots for interactive reader engagement. A similar approach is followed by Ginosar et al. [24], where they allow propagating retrospective edits across versions in order to develop multi-stage tutorials.

Several techniques have been utilized to enrich the developed tutorials in order to ensure comprehensiveness and reader satisfaction. HelpViz [62] automatically generates visual elements for textual instructions using a transformer-based model [35] albeit for simple workflows. Studies have highlighted the value of user contributions in enriching tutorial content. FollowUs [33] emphasizes a collaborative approach, capturing video demonstrations from the community to offer varied perspectives and enrich the tutorial. Dubois et al. [17] focuses on textual annotations through user-contributed notes, adding insights and workarounds. While community contributions add value, such approaches necessitate quality control to ensure the accuracy and relevance of the added content.

Torta [39] automates tutorial generation for tasks across multiple interfaces, combining OS-wide activity tracing with screencasts to ensure comprehensiveness. Porta [40], while employing a similar system process tracing approach, tracks the learner's environment as they navigate the tutorial to identify their challenges while using the tutorial to learn content. It provides detailed insights like mouse movement heatmaps and event markers, aiding authors in refining content based on user interactions.

The study presents four limitations. First, the study was conducted with only seven participants, which brings into question the generalization of the results across a broader population of tutorial writers. We acknowledge that the insights derived might still be influenced by the specific experiences and backgrounds of these seven tutorial writers. However, the selected participants come from different wakes of the software engineering discipline and have extensive expertise (see Table 3.1), bringing in a wide range of experiences and insights. Additionally, around five users are shown to be sufficient to capture the usability-related issues. Second, most of the participants had not used the OpenAI playground prior to the study, indicating that the reactions might be influenced more by the novelty of the tool and the model than by its actual utility. We let the participants use the tool for multiple interactions and while the participants demonstrated this initially, the novelty wore off and they revealed deeper insights into the strengths and potential shortcomings. While being able to introduce participants to a new tool was unintentional, we were able to capture genuine first-time user experiences, which are invaluable for understanding the preconceived notions, initial barriers to adoption and usability issues corresponding to learnability.

Chapter 3

Method

We performed a user study with software tutorial writers to investigate how to design LLMs-enabled tutorial writing tools that can be integrated into their writing processes. The user study has three parts: a semi-structured interview about the tutorial writer's prior experiences, a hands-on activity to discover the potential of using LLMs for tutorial writing, and finally, a follow-up interview reflecting their attitude on tutorial writing with LLMs. We discuss the recruitment and study process below, which is approved by the research ethics board of the authors' university.

3.1 Participants and Recruitment

We aimed to engage a diverse set of individuals with extensive practical experience in writing and publishing technical tutorials, ensuring they could provide valuable insights into the challenges, strategies, and opportunities in this area. Therefore, we posted advertisements on technical writing communities (see Appendix for details) on Slack, Reddit, and LinkedIn between August and September 2022. The selection criteria included a prerequisite that each participant had previously published a technical tutorial in English. Participants were requested to share links to their published tutorials and were assessed by one of the authors to ensure they were relevant technical articles. In total, we had 33 signups for the study, of which 19 were excluded for not sharing links to a published tutorial. Of the 14 who shared the links, four participants were excluded after the vetting process, while three did not proceed with the interview scheduling process. Finally, we ended up with seven participants (henceforth referred to as P_1-P_7). Table 3.1 provides detailed demographic and professional information about the participants, such

as their years of experience, number of tutorials written, and familiarity with AI-enabled tools.

Participant	Years in Software Engineering	Tutorial Authoring Frequency (Past 3 Yrs)	Tutorials Written	Experience with AI tools	Current Occupation	English proficiency
$P1$	<5 years	Weekly/biweekly	5	Not used previously	Lead, technical documentation	Professional Working Proficiency
$P2$	<5 years	Once a month	2	VS Code IntelliSense	University Student (Computer Engineering)	Native/Bilingual Proficiency
$P3$	11-15 years	Once a month	20	Not used previously	Technical Writer	Professional Working Proficiency
$P4$	11-15 years	2-3 times a week	50	GPT-3 based tools (Jasper AI)	CEO (of a technical writing agency)	Native/Bilingual Proficiency
$P5$	1-2 years	Once in several months	20	VS Code IntelliSense	Student, Technical Writer	Native/Bilingual Proficiency
$P6$	>15 years	Once a month	50	Not used previously	Technical Writer	Native/Bilingual Proficiency
$P7$	11-15 years	Once a month	50	VS Code IntelliSense	Software Engineer, SRE	Full Professional Proficiency

Table 3.1: Background of the User Study Participants

3.2 Study Design

We design the user study as a mixed-methods investigation, structured into three parts. The study for each participant lasted around one hour and was screen-recorded.

Semi-structured interview. The study started with a semi-structured interview where participants shared their overall experiences and the use of existing tools for writing, organizing and maintaining the tutorials. In particular, we asked them to contextualize their discussion using (but not limited to) the tutorials they submitted during the recruitment phase. This part of the study helped us gain a deep understanding of our participants' current workflow, important considerations when scoping, writing, and

maintaining software tutorials, and primary challenges they encountered that might have major implications for the design of LLMs-enabled writing tools.

Hands-On Exploration. We asked the participants to explore the scenario of writing a hypothetical tutorial on topics they were familiar with, assisted by the LLM, i.e. Codex[1] in this case, using the OpenAI playground (now renamed to OpenAI platform).[2] We used the Playground interface since it provides access to text and code based models which are sufficient for an initial exploration for the tutorial writing task, without the need for fine tuning. The ChatGPT interface[3] was unavailable at the time of the study. A screenshot of the interface during the study is showing in figure 3.1.

By default, the playground features Complete mode of interaction, which presents a large textbox along with a panel where the users can choose the playground settings, notably, mode (one of Complete, Edit, or Instruct) of interaction, models (like text-davinci-003, text-curie-001)[4], maximum length token (default value of 256) which indicates the number of tokens generated by the LLM per request, and temperature (default value 1). These settings were introduced to the participants, and they were free to modify them at any point during the exploration.

Since most of the participants had not used OpenAI's playground prior to this study, we started with a brief introduction to this tool. We then asked participants to follow the "think aloud" protocol [25] during the exploration – encouraging them to voice their thoughts, actions, and expectations as they interacted with the tool. The interviewer occasionally prompted participants with questions regarding their actions and impressions of the interaction with the tool. If needed, the interviewer also clarified the tool's features and encouraged participants to experiment with different aspects of the interface. The objective of this part of the study was to observe how participants might incorporate the LLM into their tutorial writing process and to identify any challenges or benefits that arose.

Follow-up Interview. After the hands-on exploration, we asked the participants to reflect

[1] https://openai.com/blog/openai-codex
[2] https://platform.openai.com/playground
[3] https://openai.com/blog/chatgpt
[4] https://platform.openai.com/docs/models

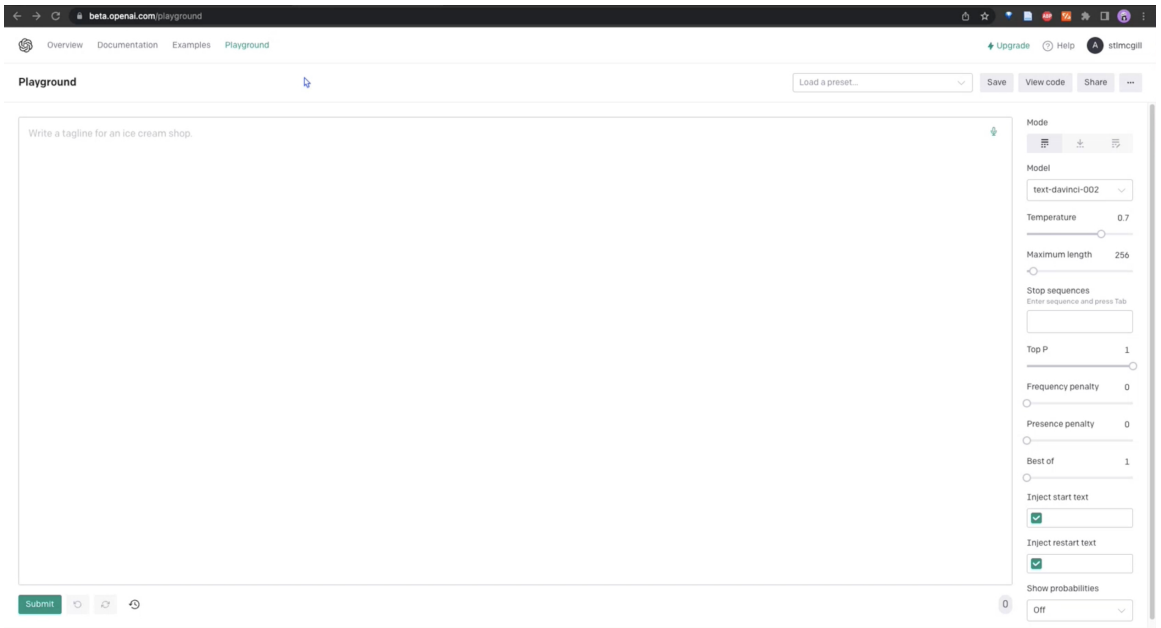


Figure 3.1: A screenshot of the OpenAI Playground interface in August and September, 2022, as used in the study.

on their interaction with the LLM, including the perceived usefulness of the tool, any difficulties they encountered, any advantages they considered the LLM might provide in their writing process, or any other aspects they deem relevant. In the end, we asked the participants about potential features they expected to have for an LLM-based tutorial writing tool.

3.3 Data and Analysis

We used the audio transcript[5] of the study to analyze the interviews before and after the hands-on exploration stage. Our analysis is primarily qualitative[6], focused on participants' experiences and attitudes towards using LLMs in tutorial writing, their feedback on the LLM's performance, and how it might influence their current writing process. Special attention was paid to areas where LLMs added value and where improvements could be made.

[5]extracted using Microsoft Teams
[6]Performed using Atlas.ti (https://atlasti.com/)

3.3 Data and Analysis

We used the screen recordings in addition to the audio to observe the participants' interactions with the playground during the hands-on exploration. Here, we leveraged a hybrid thematic analysis approach to make rich and reflective observations [51]. First, the first author went through the audio transcripts and screen recordings to make annotations on the salient themes from the comments of the participants or their interaction with OpenAI's playground. After the first author generated the initial codes, the remaining two authors further critiqued and joined the discussion to ensure robustness. The results are presented in Chapter 5 which motivate our design guidelines outlined in Chapter 6.

Chapter 4

Processes in Tutorial Writing

This section summarizes our investigation of how tutorial writers approach their writing tasks in practice. The task often involves the authors performing multiple iterative workflows, including understanding the audience, researching the topic, and developing and verifying artifacts (e.g., code or screenshots) with the goal of producing reader-centric instructional material for a certain technology or software. The workflows are motivated by the authors' aspiration to create tutorials that adhere to mostly self-imposed considerations and quality standards concerning accuracy, specificity, completeness, and clarity. We discuss four essential workflows mentioned by our participants when writing tutorials and how they demonstrate the interdependent and iterative nature of the writing task.

4.1 Understanding the information needs of the target audience

Software tutorials, whether published to the public or distributed within the software team, have particular target readers. The information needs of those target readers directly influence our participants to decide on the scope, writing style, and artifacts incorporated into their tutorials. Sometimes, the tutorials stemmed from notes written for personal consumption, e.g., to learn new technology and documentation for future reference. These tutorials are often less detailed in nature, as mentioned by $P7$: *"[I] keep notes ... I document my steps basically ... so that's it"*. Even when these self-targeted tutorials are shared with others, it is with an understanding that they are best suited for individuals with similar expertise levels (e.g., *"If it's something I'm just doing for myself [I] might be less*

worried about who exactly the audience is, it might just be people like me" ($P4$)).

On the other hand, writers invest additional effort in tutorials intended for a wider audience to make the content more applicable and accessible to the reader depending on their levels of technical skills (e.g., *"What differs is the amount of detail ... With a very junior-level reader I might want to include every step ... but with a more senior person I might just skip step[s] ... I might jump right to the things that are a little more relevant to them" ($P4$)*). Ideally, writers aim to interact directly with their readers to gather a detailed understanding of their informational needs, but obtaining such firsthand insights is often not feasible (e.g., *"the ideal situation, which I've rarely had is that I'll be able to sit next to the end-user or the target audience and say, What keeps you up at night and how can we help you?" ($P6$)*). Given this challenge, writers often resort to various alternate methods to find out the information needs of their readers. For instance, when they are part of technical writing teams in a company, they obtain tutorial requirements from other user-oriented teams (e.g., tech support, sales, client success and user experience teams) or reach out to software development teams anticipating a tutorial for a forthcoming product release. Writers also engage the system as end users themselves to gain firsthand insight into the potential difficulties other users may face so that they can address this in their tutorials. Finally, with extensive domain knowledge, technical writers can identify gaps in publicly available information about a concept or software feature. By proactively working to fill these gaps, they ensure that their tutorials are both comprehensive and relevant to their intended audience.

Tutorial authors identify information need of the target readers through feedback from user-facing or development teams and the experience of the authors' when using the software and their knowledge of existing gaps in the publicly available documentation; their understanding of the information need become dominant influence on the topics of the tutorials.

4.2 Researching the tutorial topics

After determining the information needs of the intended readership of the tutorial, writers proceed to an extensive research phase surrounding the identified topics by consulting and

performing a deep dive into existing resources (e.g., documentation, text or video tutorials) to strengthen their understanding of the topic (e.g., *"I see some YouTube videos about YouTubers doing it, and how it works on the internet, and on Reddit. I've looked for the code [to] make it work" ($P2$)*). Often, writers leverage their access to developers for further insights and clarifications currently not covered in existing resources (e.g., asking developers about design intentions for developed APIs or software).

The participants highlighted two major challenges related to the interaction with developers. First, developers often assume that the writers already possess a foundational understanding of background concepts during technical discussions (e.g., *""[Developers] expect us to understand certain things in the development area ... they don't have much awareness that we are totally new to this"" ($P3$)*). This expectation necessitates writers to rapidly acquire a nuanced understanding of background concepts, which can be considerably difficult for those new to the technology. The second challenge involves securing time with developers for these discussions (e.g., *""Getting a developer's time is sometimes difficult, especially during the sprint or a deadline"" ($P6$)*). Participants acknowledged that the recent shifts towards remote work have facilitated convenient and productive collaborations, with tools like Slack and Zoom ensuring quicker responses by the developers.

> Writers research existing resources and consult with developers before writing a tutorial but face challenges in acquiring background knowledge and securing developers' time, the latter being mitigated by collaboration tools.

4.3 Developing tutorial artifacts

Writers produce several artifacts, often as a byproduct of research which often occurs along with the development process. These artifacts are subsequently incorporated into tutorials or used as a reference throughout the development process. Participants primarily created text (e.g., tutorial outlines and software configuration steps), code, and screenshots. They suggested that the use of videos, while valuable, is less frequent due to the high maintenance requirements. For certain topics, participants report using artifacts like links to GitHub repositories, plots and figures, and comparison tables for different

software, which demonstrate the writers' ability to tailor content to meet specific software and audience needs (e.g., *"Typically a tutorial is gonna have text screenshots, code samples and then ideally like a whole GitHub repo[sitory] with all the code used if that makes sense." ($P4$)* The primary role of source code in software documentation has been reported in previous studies [27, 24]. Our study result reinforced those findings from the author's perspective – our participants reported that developing a reference code implementation formed an integral part of their writing process and was sometimes even explicitly requested by clients. The primary motivation for developing a reference code implementation includes creating a guide that determines which software features to highlight, outlining the steps to include in a tutorial, and verifying and proving the tutorial's accuracy and reliability.

During the implementation of source code, writers employ a reader-centric approach to understand target implementation areas and conduct collaborative testing with developers. One strategy that participants often employ is anticipating potential difficulties readers may face and then crafting clear, straightforward implementations to address these challenges. For example, $P7$ documents problem-solving strategies and corresponding solutions, which then inform the structure of the tutorial. This practice enables writers to determine the aspects of the code that need to be explained, thereby enhancing the clarity and specificity of the tutorial. Writers allocate significant time to test the reference implementations thoroughly to ensure accuracy. The testing is performed in collaboration with the developers and serves dual purposes: writers report and help developers resolve any bugs identified during the development of reference implementations, and developers aid writers by offering solutions and workarounds to facilitate effective reference implementation.

In developing source code, writers often encounter challenges such as complex software configurations, interfacing with unfamiliar technologies, and steep learning curves associated with IDEs and tools. Writers frequently have to maintain versions of successful and even unsuccessful code implementations as a reference and rerun them to validate their correctness. This is particularly problematic when a tutorial requires extensive configuration; missing steps or executing code in the wrong order often necessitates reverting to the beginning of the configuration process, which can be

time-consuming. Another challenge arises when clients request tutorials that integrate technologies that have not been combined previously. In such cases, writers need to perform independent research to create a novel implementation, which introduces an additional layer of complexity in the process for which they might not have the necessary expertise. A further challenge pertains to the tools utilized in developing the reference implementations. Participants reported using integrated IDEs, text editors, and in one case, even traditional pen and paper for code development, indicating an absence of dedicated tooling support for the tutorial authoring process. One participant indicated that any tools used by tutorial authors are primarily designed with developers in mind and often pose a steep learning curve that tutorial authors may find challenging. However, authors adapt to the developers' workflows mainly to avail themselves of developers' assistance when difficulties arise.

Writers create artifacts such as text, code, and screenshots and ensure their accuracy through testing, all while facing challenges like complex software configurations and learning curves of unfamiliar technologies and tools.

4.4 Meeting Quality Standards

Consistent with prior research [27], tutorial authors include textual explanations and narratives around the artifacts to facilitate the readers' understanding. While creating a tutorial, the writers focus on qualities like readability, conciseness, specificity, and a coherent structure to produce high-quality content that meets the informational needs of the readers. Writers modify the artifacts to improve their clarity and readability within the tutorial (e.g., $P7$ noted*"... I usually make some edits [to] make it more human-readable, pretty, and easily digestible. [I] break up commands into multiple lines and stuff like that"*). Writers complement the tutorial content with illustrative examples, screenshots, or external resources (e.g., URLs for sources and background information) to ensure the completeness of the tutorial. When any external resource is to be integrated into the tutorial, it is not merely referenced (e.g., by providing URLs for the readers to browse the information themselves) or copied and pasted. Instead, writers put effort into explaining the information and its context to the readers with the goal of minimizing the need for

readers to look for information elsewhere for clarification. Writers invest significant effort in structuring the tutorial to effectively deliver information, aiming to gradually build the reader's confidence to explore and independently navigate the software and avoid overwhelming or confusing them. One strategy involves crafting a series of short, focused tutorials. Each tutorial revolves around the same software or concepts and provides easily digestible knowledge. Over the course of the series, the scope of the tutorials gradually increases. The sequence of tutorials often begins with simpler 'Hello World' examples to introduce the software, with later tutorials progressively exploring more complex concepts. Writers attempt to align the organization of information with the progression of the reader's understanding, often leveraging tools like Confluence to structure and present content effectively.

Writers face several challenges while enhancing reader understanding, ranging from making decisions about including and positioning artifacts within the tutorial, establishing the necessary context for clarity, and tackling mental blocks. When writing tutorials, writers need to undertake the challenging task of anticipating the optimal number of artifacts that can facilitate learning for readers at different skill levels while providing the necessary context. Finding the right balance between context, clarity, and conciseness often requires multiple revisions and external feedback, making it a time-consuming process. Writers often encounter mental obstacles, such as writer's block and can have difficulties translating their ideas into text. To overcome such obstacles, writers often use tools that assist with paraphrasing or restructuring the content. Another strategy to streamline the writing process involves using simple writing templates, often drawn from writers' previous experiences with writing tutorials.

Upon the completion of writing, writers perform verification of the complete tutorial and seek feedback about the written content. Writers are aware that potential errors might arise while adding textual explanations and narratives and often execute and verify the content to ensure accuracy. Feedback is sought from both development teams and the end audience, enabling writers to assess the tutorial's effectiveness and make necessary adjustments. In some cases, writers must also comply with legal requirements for technical documentation, such as the translation of tutorials into different languages. Tutorials are usually composed in English and then translated into other languages as

needed.

> Writers aim towards fostering the reader's understanding and confidence to explore the software by strategically structuring the content, integrating relevant artifacts and their explanations, and continuously revising the content for clarity, completeness and other quality attributes.

4.5 Maintaining the published tutorials

A unique aspect of tutorial authoring is the continuous effort to maintain the accuracy and relevance of the tutorials. The need for upkeep can arise driven by two reasons: technical updates and feedback from developers or readers. The participants acknowledged that maintaining the tutorial could be more challenging than initially writing it (P1, P6). The maintenance process can sometimes become overwhelming when the writers manage several tutorials (P6), necessitating a thoughtful approach to tutorial writing.

Writers follow several strategies for tutorial maintenance, including relying on developers, writing a completely new tutorial in the face of major changes, and carefully determining the scope when initially crafting the tutorial, keeping potential maintenance in mind. Participants reported that they often rely on developers to alert them when there are changes to the code base and the tutorials are needed to be updated (P1, P3, P4, P7). Often in the case of minor maintenance, the responsibility is shouldered by the developers (P3, P4). With significant technological changes, as in machine learning, where the technical landscape changes frequently, writers often prefer to write a new tutorial rather than edit the existing one (P4, P5). A proactive view towards maintenance is that of developing tutorials that are robust enough to accommodate minor changes to the system so that extensive changes are not necessary (P6, P7). The robustness can be achieved by developing specific tutorials with short scopes to keep the maintenance minimal and manageable (P7).

> Writers design robust tutorials, seek developer assistance, and sometimes opt for completely new tutorials when facing substantial changes in order to maintain tutorial accuracy and relevancy amidst an evolving technology.

4.5 Maintaining the published tutorials

In this section, we described the five aspects of tutorial writing elicited from the study. We find that technical tutorial writing is influenced heavily by the diverse needs of its target audience, and assumptions regarding readers' expertise can inadvertently neglect their learning needs. The research phase for writers extends beyond information collection to a deeper contextual understanding, integrating formal guidelines in the form of official documentation and real-world insights through public channels like YouTube. Direct access to developers provides valuable and exclusive insights into software design intentions, and tutorials serve as a way to disseminate this information. Authors design tutorials to guide their audience progressively, emphasizing both technical depth and user accessibility. However, the need for post-publication maintenance prompted by the dynamic nature of the software significantly influences tutorial structures. This necessitates the structuring of tutorials as distinct, focused units to facilitate updates or replacements. Building on this understanding of tutorial writing, we further discuss our findings on the possibility of integration of LLMs in this domain and the potential implications of this integration.

Chapter 5

Interactions with LLM

In this section, we discuss our observation on how the participants seeked support from the LLM for their existing tutorial authoring workflows, and the challenges they faced in the process. The emerging themes covered four aspects of the interaction, i.e., integration of LLM into the exiting authoring processes, quality assessment of LLM-generated content, mental models of the users and finally usability issues posed by the model and the interface.

5.1 Integrating LLMs into Existing Aspects of Tutorial Writing

The introduction of LLMs to assist in tutorial authoring offers authors the potential to augment and enrich existing aspects discussed in the previous section. Participants actively calibrated their existing workflows in order to optimize their outcomes using the capabilities of the LLM.

Leveraging LLMs for tutorial writing poses a learning curve to the participants.
At the beginning of the hands-on exploration phase, the authors perceived LLMs to complement their traditional tutorial authoring practices (outlined in Section 4). Participants initially opted to use the LLM for the tasks that they perceived the model might do well. For example, after deliberating on the capability of the LLM, $P4$ wanted to generate code *"Let's say build a new NextJS application with a user login form" ($P4$)* and perceived the generated output as *"This is pretty trivial stuff because you could go to*

NextJS documentation and [get] the getting started [or] hello world kind of example ... At least I've got some text to start"" and proceeded to refine the content. From this, it's clear that while LLMs can aid in the writing process, they don't replace the need for human intervention and refinement. As participants explored and familiarised themselves with the capabilities of the LLM, they increasingly recognized its potential to enhance writing aspects. By querying the model about various technologies or soliciting its input on potentially challenging aspects, participants found themselves with a valuable tool to quickly expand upon their insights. For example, $P4$ researched and compared different technologies and anticipated the challenging aspects that the readers would need assistance in to include in the tutorial (e.g., *"... So [software A] is the backup tool, which is less interesting. I think the interesting part would be adding [software B] for user login" ($P4$)*). Next, they used this information to prompt the model ("Add [Software B] for user login to the above NextJS applications.") to generate the code example.

Participants often revise their initial content-related objectives based on evolving written content. Integrating LLMs lets the authors generate preliminary content drafts, which speeds up their content drafting workflows. For example, $P7$ initiated a web-based tutorial as *"create an AWS account and configure command line credentials for it"* and later evolved the tutorial into a more encompassing *"Let's go maybe one step further and ... let's create an S3 bucket for hosting a static website"*. Such an approach not only saves time but also brings aspects they might have previously overlooked to light (e.g., *"Clean up ... it even added clean up [section in tutorial] ... Nice" ($P7$)*). Sometimes, different aspects of the tutorial only surfaced when refining the content resulting from the previous rounds of interaction. For example, after reading the model output, $P7$ suggested certain edits to ensure necessary granularity in the content *"... there's multiple ways to install an OS ... my goal here is to document how to do it on Windows and it's gonna be easier for Desktop because I'm on Windows right now" ($P7$)*. As participants interacted more with the LLM, they found that the content it generated often served as a first draft. This draft was then refined with the authors' expertise. This human-LLM iterative collaboration process, ensured content that was both technically accurate and contextually relevant.

Modifying the Existing Authoring Strategies to Leverage Capabilities of LLM.

5.1 Integrating LLMs into Existing Aspects of Tutorial Writing

As familiarity with LLMs grew, authors began calibrating their traditional workflows to make the most of the LLM's capabilities. They grasped the model's limitations yet also identified the strategies to make LLMs more helpful. By breaking down complex tasks into simpler queries, they could extract more precise responses. For example, $P6$ tries to extract precise explanations to complex code using LLM by first breaking it down, *"Can I ask the model to break the function down into parts? ... like in the components?" (a)*nd later ask for explanations of individual components.

However, their desire to maintain their unique writing style was a driving force behind their efforts to align the LLM's outputs with their workflows. For instance, $P7$ sought to narrow down the model-generated content for specificity by editing the content generated by the LLM *"Everything that I've considered valid here, I'll retain and then I'll guide [the model] in a slightly different direction"*. In efforts to steer the LLM, participants devised multiple strategies, often inspired by their traditional content drafting approach without the LLM. By *providing an overarching structure of the target tutorial* or through *editing and reformulating prompts* and *including topic-specific keywords in the prompts*, participants could guide the LLM's output to resonate more closely with their own style and the intricacies of their chosen topics. For example, $P5$ laid out titles of sections for the tutorial since they *"want to start with an introduction. I would probably input the title and see [the model's] response"*. Their subsequent step involved populating these sections by crafting specific prompts. The participant further elaborated, *"To avoid bias, I removed 'NLTK' [from the context window], prompting it to explore 'GloVe'. When I excluded 'GloVe' and added the word 'choosing', it began suggesting alternatives. It eventually provided three sensible options" ($P5$).*

Participants began to place more trust in the LLM's ability to produce relevant output as they interacted with the LLM, as indicated by their increased delegation of content production. However, a significant part of this trust was rooted in their confidence to refine and improve the LLM's suggestions. $P7$ demonstrated this by delegating preliminary content generation to the model, *"I'm not sure if Python is the right way to install AWS, so I'll just let [the model] decide that for me"*, later revising the content to add specific details the model overlooked, *"Seems like [the model] didn't pick up on the fact that there's a specific [installer] link for Windows, so I'll put that in"*). Such interactions

indicate that while participants find the LLM valuable and are keen to integrate it into their workflows, they maintain the desire to oversee the final content, ensuring it aligns with their standards and style.

5.2 Quality Assessment of LLM-generated Content

Participants maintain clear quality criteria for LLM-generated tutorial content and assess it based on external resources and their own expertise.

Participants' quality concerns are primarily towards the accuracy and coherence of the generated content.

All participants emphasized the accuracy of the tutorial, which even reflected in their choice of topic for the tutorial (e.g., *"if I were to write it from scratch, I would want to have the runtime for running [the steps] ... This is a Windows [OS] and I'm gonna have to pull some new command line stuff so I'm thinking maybe I can do something more web-based" ($P7$)*). Participants actively fact-checked the generated content and devoted a significant part of the time allocated for hands-on exploration to ensure the correctness of the generated content. Even if the participants were unable to perform the planned verification steps due to time constraints, they still expressed an intention to verify (e.g., *"I'd have to go put this into an IDE, figure out each step, is that actually right? And double check it" ($P4$)*), indicating the importance of accuracy in the tutorial writing task. Participants were invested in the user's perception of the tutorial and wanted the content to be specific and coherent (e.g., *"I was hoping that [the generated content] would be a bit more specific here when it says run the installer. I mean it's pretty obvious for the end user, but I like to make [it] impossible to do the wrong thing kind of thing" ($P7$)*). Participants expect agency from the model to meet their writing quality (e.g., *"My ideal world would be that it goes to the above tutorial and edits the steps required ... maybe the other thing which may be more realistic based on just seeing how this works, I would think it would add below ... or like update the code and then it changes some lines" ($P4$)* on being asked about their expectation with a prompt).

5.2 Quality Assessment of LLM-generated Content

Participants verify the accuracy and coherence of the content by comparing it against their domain knowledge and existing documentation.

While participants drew upon their domain expertise or general knowledge to verify the generated content in certain instances, the prevailing approach was either executing the steps or corroborating the information against existing documentation. Tasks like translation (e.g., *"'getting started' doesn't translate to à propos de départ... [it] doesn't mean anything" ($P1$)*) were evaluated based on the participant's knowledge of the language. Participants with extensive experience in software development were able to leverage their knowledge to identify discrepancies in the output (e.g., $P7$ identifies an issue in the content about AWS access keys, *"I think there's some missing steps here ... Here's the thing, because we just created the account these [AWS] access keys will not exist at this point"*). When domain knowledge is insufficient, participants choose strategies such as cross-referencing with existing documentation, internet search or testing by execution. Minor details of generated content (e.g., URLs) were checked for authenticity and correctness by a browser search or against existing documentation. Aspects that involved complex reasoning (e.g., code snippets or steps for creating an AWS account as explored by $P7$, *"let's actually go ahead and test all of this"*) were tested through manual execution. Participants sometimes performed a combination of these strategies, depending on the generated information.

Participants chose to manually edit the LLM-generated content to achieve better accuracy and clarity.

While participants acknowledged the usefulness of LLM-generated content, they expressed a need for refinement. A perceived inability of the model and usability issues posed by the tool to achieve better accuracy and clarity made them resort to manually editing the generated content (e.g., $P7$ mentions *"I would just remove extra information [from the generated content]. Yeah, that's my style"* and proceeds to edit manually). However, improving the tool's usability and explicitly indicating the LLM capabilities can improve its adoption. This was evident by $P7$'s use of the Insertion feature of the Playground to add specific content based on the interviewer's input instead of manually editing the content, *"Oh yeah, that's a good thought there" ($P7$)*, leading to a satisfactory

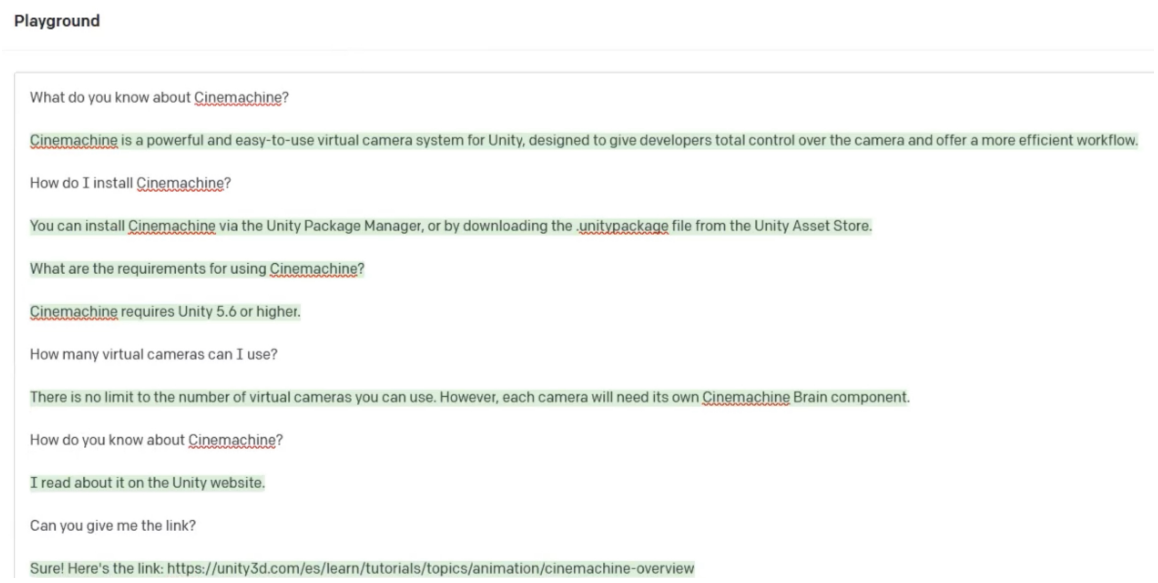


Figure 5.1: Participants often tried to get the source of the generated content from the LLM. $P6$ interestingly used the text editor interface in the Playground to chat with the model and identify the source of the generated tutorial. This study was performed prior to the release of ChatGPT.

interaction *"it's repeated number one, but other than that it's perfect"*.

5.3 Mental Model of LLMs

We observe that the participants' behavior is strongly influenced by their mental model of LLMs. Throughout the interaction, they revise their mental model to reason the capacity of LLMs depending on the immediate outcome of their explorations.

Existing experiences and mental models of the participants dictate the approach and expectation towards the LLM interaction and can affect their overall experience.
Participants based their interactions with the LLMs on their existing mental models of LLMs and tutorial writing. The concept of using LLMs for tutorial writing was novel to all the participants. Two participants ($P1$, $P4$) had used LLMs in other settings before the study and the rest of the participants were brief introduced about LLMs and OpenAI playground at the beginning of the hands-on exploration phase. Despite being brief, those

information helped the participants to understand what the LLMs can do (e.g., *"just repeating whatever worked successfully before, as you showed me" ($P6$)*). Participants form expectations of LLMs from their existing tutorial writing experience. For example, $P6$ mentioned *"if I find something curious or unexpected or unintuitive, then I'll ask the developer about it"* and started to explore the capability of the LLM to find information about the software in a conversational way, as when asking the developer.

The existing mental models define the expectations and shaped their outlook toward LLM usage. For example, $P1$ mentions *"I'm a very tech-positive thinker... I don't think it's gonna immediately replace everything we do, but I would love to see where AI-assisted writing could help our writers do twice as much work faster"* during the hands-on exploration phase. In contrast, $P1$ mentions *"I've seen that one example on the Internet about securities issues ... when you would ask it about API keys and it will give you someone's API keys"* leading to a prompt *"What happens if you tell the playground to give you AWS keys?"* and critical feedback on the generated output *"If it could start with not giving out like personal details about people I don't know when I'm trying to complete things that would be a great start"*. These statements indicate that the participants are influenced by their existing mental models which can shape their expectations, approach and outlook towards LLMs. Familiarity and knowledge about the working of the LLM and experience in the software documentation and writing process can help the participants start off on a better footing in terms of leveraging LLMs for tutorial writing.

Participants form strategies to evaluate the capabilities of LLM to connect to other software and generate up-to-date information about various technology topics and explore the interaction techniques that can get them the best outcome.

Participants usually perform additional technical setup prior to writing and often try to identify the capabilities of LLMs to support this exercise (e.g., *"We need an Airtable base with some real data ... in order to make that ... I don't know if you can make OpenAI [Playground]..., it doesn't know how to go off to other apps, right?" ($P4$)*). During the actual task of writing, participants expect the model to have up-to-date and wide technology coverage in the content it produces and therefore explored if the model satisfies these requirements (e.g., $P4$ reflects on evaluating the model by prompting it to

"tell me about the latest updates to Redis and see which version [the model] tells me about because that kind of gives you the decay of how updated [the model] is"). They expressed an intention to understand the model's technical boundaries and what are the sources of the information (e.g., *"Is the model working from the information they got from the help content like the documentation or from the source code?" ($P6$)*).

Participants also explored different prompts and parameters and tried to optimize them to get a better output. Participants dedicated significant effort to determining elements of a good prompt and often talked about "phras[ing] this for the machine" ($P4$) or choosing the right parameters. (e.g., $P4$ mentions *"I guess I can up [maximum token length parameter] and see what happens"* since they expected the generated content to take more than 256 characters in generated length). Participants also often tried to clarify, *"How short can the prompt be?" ($P6$)* or *"Is it kind of similar to Google execution ...?" ($P3$)* to understand how they should approach prompting. Participants tried to understand the connections between their prompts and responses (e.g., *"I want to see if I can be more specific about the prompt and see what the model's reaction will be" ($P6$)*). They often compared subsequent outputs to understand the difference in the prompting but are unable to understand what caused the change (e.g., *"I've modified two things in this latest query, so it's a little different than before. I'm not sure what made it different" ($P6$)*). Participants had polarized experiences with the LLM. On one hand, they expressed frustration when the model deviated from the prompt completely (e.g., *"No, no, no, no. I don't know what it understood, but it's not that. That's not what I am trying" ($P2$)*), while on the other they welcomed the model generating additional but relevant content (e.g., *"Now [LLM] is creating an error page. I don't know if I need an error page, but you know it's kind of cool." ($P4$)*

Participant revise their mental models on the model's capabilities and interaction strategies based on the relevance of generated content.

Participants constantly reasoned about the capabilities of the LLM and calibrated their interaction strategies with the LLM and the tool based on the relevance and accuracy of the content generated by the model. The reasoning was grounded in the pre-existing beliefs about how the LLM operates. For example, $P4$ hypothesizes about the working of

LLM *"It isn't really contextually aware. It's just pulling, you know, text and trying to figure out what text makes sense around that text"*. The hypotheses were sometimes incorrect (e.g., *"It needs ML [as a keyword in the prompt]. It doesn't work for any other thing" ($P5$)*), which eventually led to concerns about the model's potential to assist in tutorial writing.

To verify their hypotheses, participants actively engaged with the LLM or sought external references (e.g., $P6$ refers to the software's hosted documentation with *"a curiosity to see how it's actually listed [on the website]. Is the [generated] text verbatim?"*). They adjusted and refined their hypotheses when the generated content did not align with their expectations. For example, when the model did not generate any content about the software they mentioned in the prompt $P4$ thought *"I think for [software], it's even more niche, it's not a well-known tool. So the problem is it's probably not much content to pull from ... It probably just ignored [software] as a tool"*. Inconsistencies in LLM outputs introduced additional confusion about its capability and therfore its utility. These shifts in understanding underscore the need for clearer guidelines on LLM capabilities. Participants' concerns and misunderstandings could hinder the broader adoption and utility of LLMs in tutorial writing.

5.4 Usability Concerns

Through out the interaction with LLM, we observe that poor usability significantly affects tool usage, mostly due to the formation of participants' incorrect mental models about model capability, sometimes eventually leading participants to abandon LLMs.

Participants experience challenges in understanding the input context to the LLM, leading to a misunderstanding of the model capabilities.

OpenAI Playground provides a single textbox in the Completion mode, with the entire content in the textbox being used as a context for future output generation. The dual usage of this textbox leads to confusion among participants about the ways to interact with the model as well as due to the generated output straying away from the prompts. While some

5.4 Usability Concerns

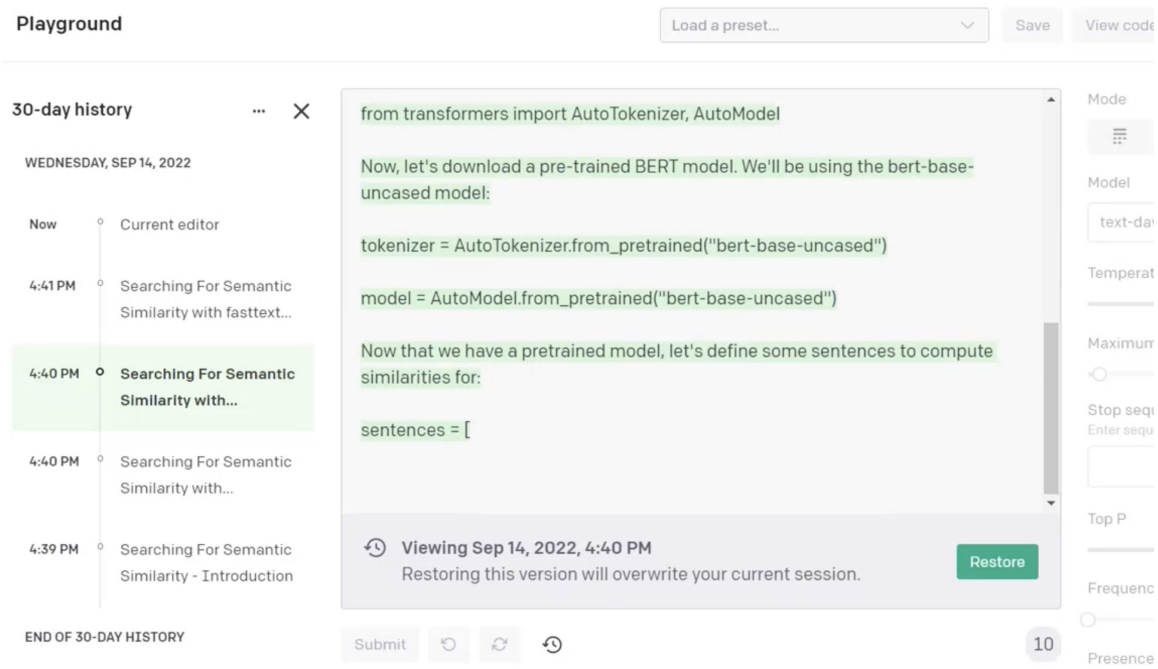


Figure 5.2: Participants refer to their previous interactions with the LLMs to calibrate their understanding of prompting.

participants eventually understand the reason behind the deviation (e.g., *"... and there's nothing related to deployment [in the generated content] because it's biased by the multitude of input before deployment" ($P5$)*) and form strategies like removing and adding necessary content in the textbox to prune the context, these are not intuitive to everyone and often does not scale to the actual task of tutorial writing (e.g., *"But if you want to write a blog in continuation, how can you not have the whole next thing? As in, is it expected to completely remove [content every time] to let this tool do the job?" ($P5$)*, described in figure 5.3). Participants often do not discover such nuanced interaction strategies and eventually declare the tool unusable for writing tutorials.

Participants struggle with prompting the model and choosing the right parameters for better output.

Participants spend significant time understanding how the model behaves and crafting the right prompt that can get them a relevant output from the model. This activity diverts substantial hours from their core writing tasks, causing experienced users to question the actual value derived from using the model (e.g., *"it takes this art form to get it to actually [produce relevant output] ... I have to think about what am I actually getting it to do and I have to be really smart. When I am a developer who's done this for many years it would be faster for me to just like do it myself because it's kind of dumb" ($P4$)*). The tool lacks clear signifiers or mappings [41] to illustrate the effect of prompts or the model parameters causing the participants to often speculate on the model's likely response and the right parameters needed (e.g., *"I'm sure it's not gonna be able to create all that in just 256 characters. I guess I can up this and see what happens" ($P4$)*). Several participants observed the generation stopping mid-sentence due to a tool-imposed limitation on the token length *"OK, so I ran out of tokens there" ($P6$)* and resorted to strategies like clicking on the 'Submit' button again to continue generation, *"I guess if I hit submit, is it gonna keep going or what? What would it do?" ($P4$)*. However, this wasn't apparent to many users and resulted in them raising usability requirements like *"fall short, but at least complete a sentence" ($P5$)*.

Discoverability of the features and capabilities of the model is restricted due to poor

5.4 Usability Concerns

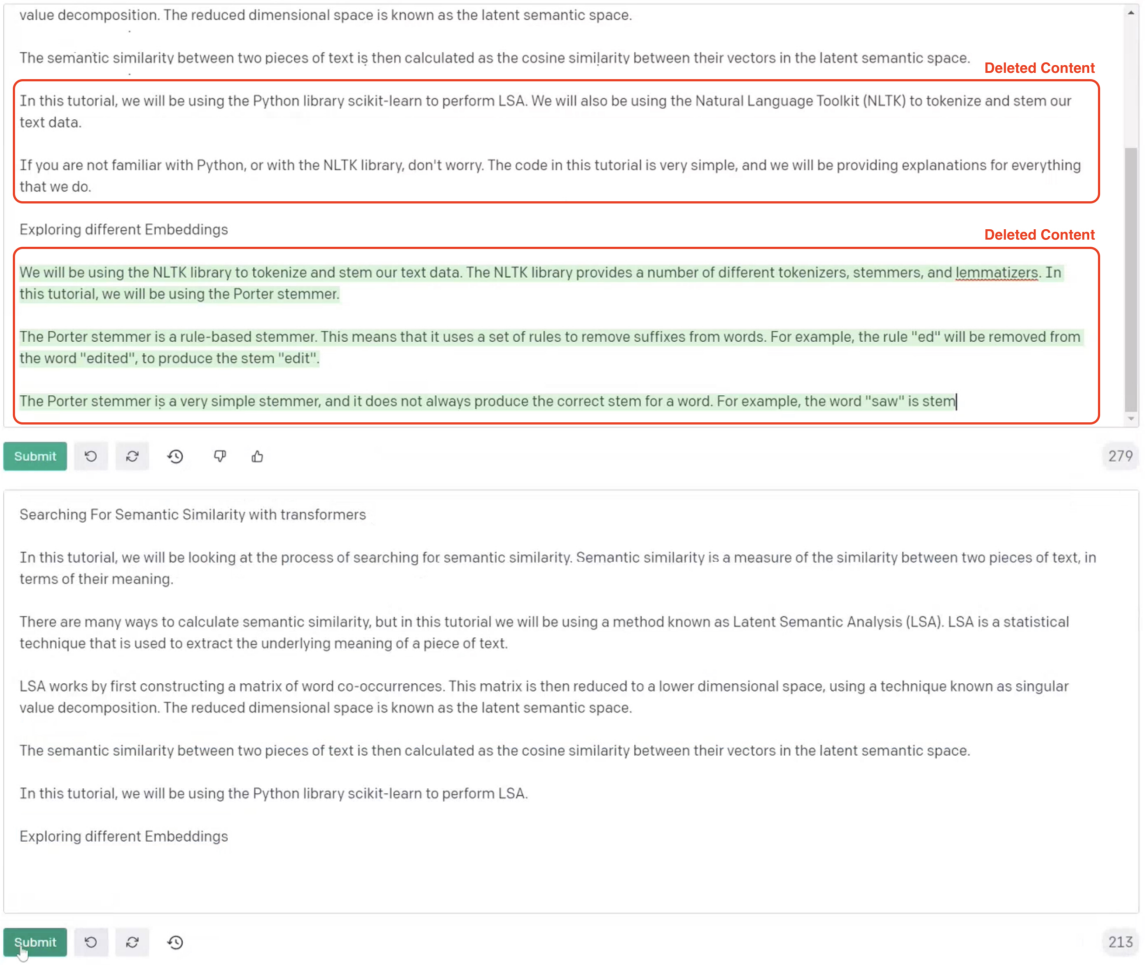


Figure 5.3: $P5$ resorted to the strategy of removing the generated content in order to align the LLM to generate appropriate content, however, expressed frustration with having to do so every time.

usability of the tool.

Participants found success in utilizing distinct tool features for specific writing use cases. For example, the Instruct mode was employed for high-level tasks, and the Insert/Edit mode was used for refining generated content. However, using the features interchangeably during the writing process was tedious due to the necessity of manually copying and pasting the target content between the modes. In the Insert mode, participants ran into issues such as managing multiple inserts. Moreover, the intended use cases for these features were not immediately apparent, with users often requiring the interviewer to clue them in, indicating the restricted discoverability of LLM features due to poor tool usability. This poses a problem for adoption, since while those who could successfully navigate the tool frequently expressed appreciation for the model's capabilities, participants who could not dismissed the tool's utility for writing tutorials, noting *"I think there's too many issues for it to be worth working on it. I have no idea [how to make it usable]" (*$P1$*).*

In this section, we identified and described the way participants leveraged LLMs for tutorial authoring and the several challenges and potentials that we uncovered in the process. Based on our observations, participants employed LLMs to expedite tutorial authoring, enhancing both pace and content quality. However, they consistently highlighted the important role of human oversight in order to ensure accuracy and authenticity. Participants' prior experiences dictate their optimism or skepticism towards LLMs, highlighting the need for transparency and proper orientation to align user expectations. While LLMs possess extensive capabilities, they sometimes fall short in capturing individual writing nuances. Further, navigating the LLM interface and optimizing prompts became evident pain points, signifying a need for more intuitive design and user guidance. Some participants encountered difficulties in discovering all the tool's features, implying that both technical strength and user-centric design are crucial for broader adoption. Drawing from these observations, we will discuss the design recommendations for optimizing the LLM-assisted tutorial authoring process in the subsequent section.

Chapter 6

Tool Design

Based on our formative study, we conceptualize a set of design guidelines and a preliminary prototype that realizes them. A summary of the guidelines and corresponding conceptual design elements is provided in Table 6.1.

6.1 Design Guidelines

The design guidelines are derived from the insights of the formative study which are aimed at understanding the current workflow of tutorial writing and the potential of optimizing it the use of LLMs. In particular, the interview phase sheds light on the complexities and strategies employed by the authors in the tutorial writing process. Two key findings from the interview phase are the mechanisms of information sharing among the processes and their influence on the evolution of tutorial content over time. Subsequently, we observed how authors directly interact with LLMs for tutorial authoring during the hands-on exploration stage and elicit their experience and perspective during post-study interview phase. These interactions uncovered a range of interface usability issues and emphasized the need for robust verification and editing capabilities, considering the authors' focus on fact-checking and maintaining content accuracy. In this section, we categorize our findings into three central themes, and discuss the corresponding design guidelines for potential LLM-assisted tutorial writing tools.

6.1.1 Supporting Authoring Processes and Evolving Content

Tutorial authoring involves various aspects and workflows, as outlined in Section 4. Authors use the data they have collected during one tutorial authoring workflow in

6.1 Design Guidelines

Design Guidelines	Evidence	Potential Design Actions
DG❶: Writers should be able to manage information dependencies during authoring processes while accommodating evolving content. *(Section 6.1.1)*	4.2, 4.3, 4.5, 5.1	✓ Develop a dedicated writing interface to facilitate tutorial writing processes. ✓ Facilitate the availability of pre-researched information during writing and for prompting. ✓ Implement content traceability and provenance mechanisms to support enhanced editing and tutorial evolution.
DG❷: Writers should be facilitated with enhanced control during prompt creation and clear visibility into variations in LLM responses to tackle usability issues. *(Section 6.1.2)*	4.3, 5.3, 5.4	✓ Provide writers with control to select relevant portions from pre-researched information and in-session writing while prompting. ✓ Clearly visualize the differences between subsequent LLM prompt-output pairs.
DG❸: Writers should be equipped with verification and editing features to ensure the improved quality and accuracy of content generated by LLMs. *(Section 6.1.3)*	4.3, 4.4, 5.1, 5.2	✓ Provide mechanisms for accurate verification of LLM-generated artifacts both individually and within the overall tutorial context. ✓ Provide the capability to switch seamlessly between granular and holistic editing and prompting strategies.

Table 6.1: Design guidelines distilled from the formative study.

another, demonstrating a close dependency between the workflows. To adequately support the task-specific writing workflows and facilitate seamless information sharing among them, there is a need for **dedicated tutorial writing interface**, supplemented by robust **information sharing mechanisms**.

As indicated by our findings in Section 4.3, the authors prioritize the accuracy of the tutorial content and invest significant effort to verify the precision of the artifacts and tutorials. Researching existing tutorial topics and developing accurate artifacts are a part of the proactive validation that the writers perform to ensure the written content is relevant and accurate. The benefit of taking the developed artifacts and writer's research into account while generating suggestions is twofold: first, it helps in better alignment of

LLM-generated output due to the input prompt (in our context, the prompts are limited to the user instances of prompts and not the system prompts. While the OpenAI API allows system prompts,[1] the OpenAI platform interface limits the access to user instances) being supervised and carefully curated, and second, it boosts the writer's perceived sense of control over LLM generation, which has been shown to improve the sense of authorship and trust in the generated content. Moreover, having researched content available within the system context reduces any cognitive load resulting from having to fact-check the content externally. Therefore, there is a need for the system to **incorporate the researched information and artifacts in the writing context** and to provide users with the control to **include existing writer-developed content into the prompt**.

Writers typically edit tutorials in two situations: when they receive feedback from stakeholders (e.g., developers) about the written content and during tutorial maintenance activities. Feedback is often specific, targeting certain sections of the tutorial where the content may be incorrect or where the quality of the content does not meet desired standards. Similarly, tutorial maintenance often involves dealing with inaccuracies (e.g., changes to specific included code snippets) and modifying specific tutorial parts. In both cases, writers often need to maintain multiple versions of the tutorial, which they either reuse or reference during editing, thereby establishing the need for **maintaining traceability and provenance** of the content. Traceability helps in understanding where and what feedback led to changes in the tutorial, capturing the reason for modifying the content. Provenance records the tutorial's evolution from its inception to its current form and answers how the content was changed. The advantages of traceability and provenance extend to the use of LLM-assisted tools. Traceability allows writers to link the changes suggested by the LLM to the specific feedback received and ensure that the updates accurately address the identified issues. In addition, provenance provides an understanding of how the tutorial content has evolved with LLM assistance and allows a clearer overview of the editing process. Implementing traceability and provenance in LLM-assisted tools enables writers to make informed decisions during tutorial editing.

[1] Shown in the example: https://platform.openai.com/docs/guides/fine-tuning/example-format, the prompts with the role of system are system prompts, and the prompts with the role of user are user instances.

6.1.2 Providing Control and Visibility to address Usability Issues

The hands-on exploration phase of the user study highlights two categories of usability issues in LLM-based systems. The first category concerns issues originating from the unpredictability of the model, often leading to a perceived lack of control over the output and the resulting user confusion. The second category of issues arises from limitations in the tool's user interface, which prevent writers from exercising the desired granularity of control over the model. Moreover, these interface-related issues can amplify the challenges of the model's unpredictability. This was evident in the study when the writers encountered a critical usability issue due to the dual usage of the text field to prompt and write while using the OpenAI playground, discussed in Section 5.4. This issue caused the writers to fail to distinguish between the prompts and the tutorial text, often resulting in the users devising elaborate strategies to prompt the model. Unfortunately, the developed strategies were mostly unsuccessful, leading to dissatisfaction with the model's inability to produce meaningful results consistently.

Users encounter reproducibility issues when interacting with LLMs, as even slightly different prompts yield significantly different outputs. Such variability in the model's responses can obstruct users' ability to devise strategies for optimal output, resulting in a perceived lack of control over the system. Although known techniques like setting the temperature parameter to zero can reduce the model's sensitivity and ensure low randomness in output, they do not address users' inherent need to be in control and understand if their instructions are being followed. Given that users are primarily concerned with the output and a high-level understanding of the model's working rather than detailed knowledge, a simple and passive approach of **visualizing differences between several prompt-output pairs** can benefit users in constructing their mental models of the LLMs working.

Tutorial authoring involves iteratively introducing precise information to specific sections of the tutorial while maintaining its overall coherence and accuracy. Some parts of the previously written tutorial text must be included in the prompt context to ensure coherence when using an LLM. When addressing a minute but critical detail, including the entire tutorial in the prompt might not be effective, as evidenced in the hands-on

exploration of the OpenAI Playground, where minor details can cause the LLM to stray away from the topic. Therefore, writers should have the **capability to select specific sections of the content to include in the prompt context**. It is also necessary that the interface provides a clear distinction between the text that forms part of the prompt context and the text that does not, thereby providing clear visibility on what the model is considering as the prompt for a specific output.

6.1.3 Facilitating Verification and Editing Capabilities to ensure Quality and Accuracy

Tutorials are designed to deliver accurate and high-quality content to facilitate the reader's learning process. To maintain high standards in their writing, writers engage actively with the generated content through careful verification and editing processes. Given the potential unpredictability of LLM outputs, these processes are especially crucial when utilizing them for writing. By investing time in verification and editing, writers can ensure that the output fulfills strict accuracy standards and aligns with their quality guidelines involving factors such as clarity, readability, coherence and stylistic consistency.in

Verification of tutorial content involves careful inspection and validation of LLM-generated artifacts, such as URLs and code snippets. Writers use different verification techniques depending on the type of artifact. For instance, verifying a URL requires confirming its online existence and ensuring that the information contained within the linked web resource is relevant to the tutorial. Moreover, it is necessary to validate the content synthesized from the web resource for accuracy and relevance before including it in the tutorial. Tutorial development workflows involve testing the code artifacts for accuracy when they are created. During the writing process, the complete code is broken down into logical snippets, then supplemented with writer-provided explanations to avoid overwhelming the readers and gradually build their knowledge. However, LLMs can be used to combine these processes and let the writers directly generate the code snippets. In such scenarios, it is necessary to confirm the accuracy of individual code snippets and ensure the complete code, composed of all snippets, is logically sound and functional. The system can facilitate both URL and code validation use cases by providing **mechanisms to ensure accuracy in an individual capacity, as**

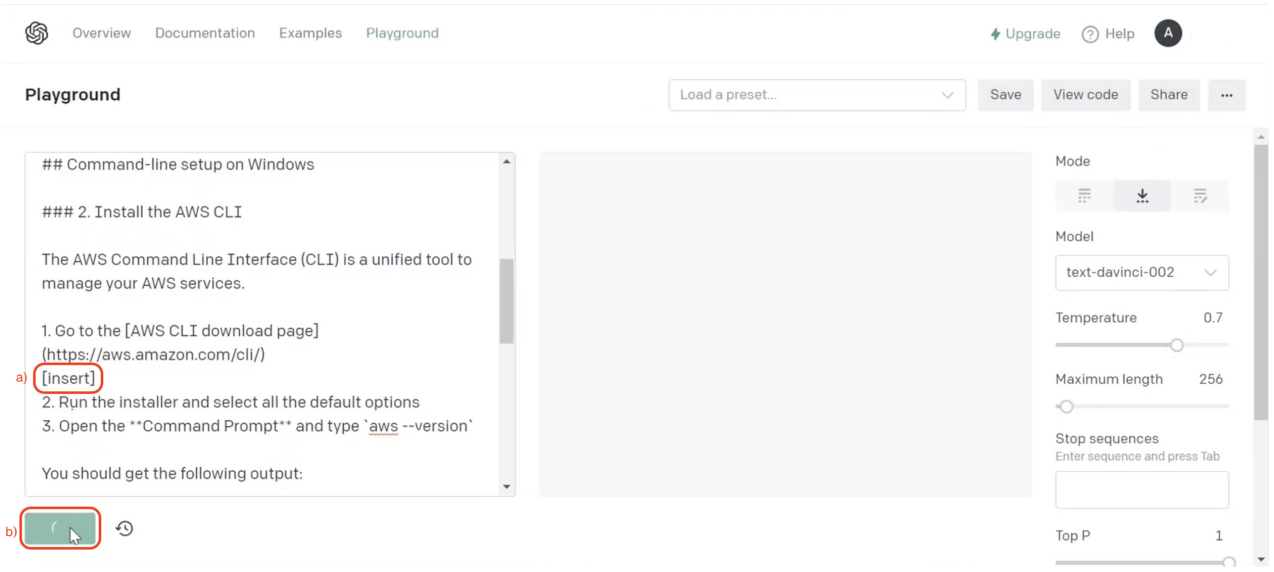
well as in the context of the overall tutorial.

The writers edit the tutorial content to meet their self-imposed rigorous quality standards. The editing strategies can range from precise and granular, such as improving sentence or word level readability, or holistic and comprehensive such as adjusting the overall tone of the tutorial or performing a grammar check over the entire content. Even when using LLM for editing, writers apply both types of editing strategies, resorting to manual edits when they consider LLMs too complex for specific tasks (discussed in Section 5.2). Evidence that writers adopted both editing strategies is demonstrated during the interview with $P7$, shown in figure 6.1. While discussing the potential use cases for LLMs, $P4$ mentions *"A client asked me to write a similar article to what I've already written but in Java. Maybe I could use this to help me bootstrap that and get it started and get a basic version going, and then I can fill in the gaps"*. To account for these editing strategies, the system must offer a **flexible interface that allows seamless switching between granular and comprehensive editing** and **allow prompting at a granular and holistic level**, allowing writers to leverage LLMs across varying contexts.

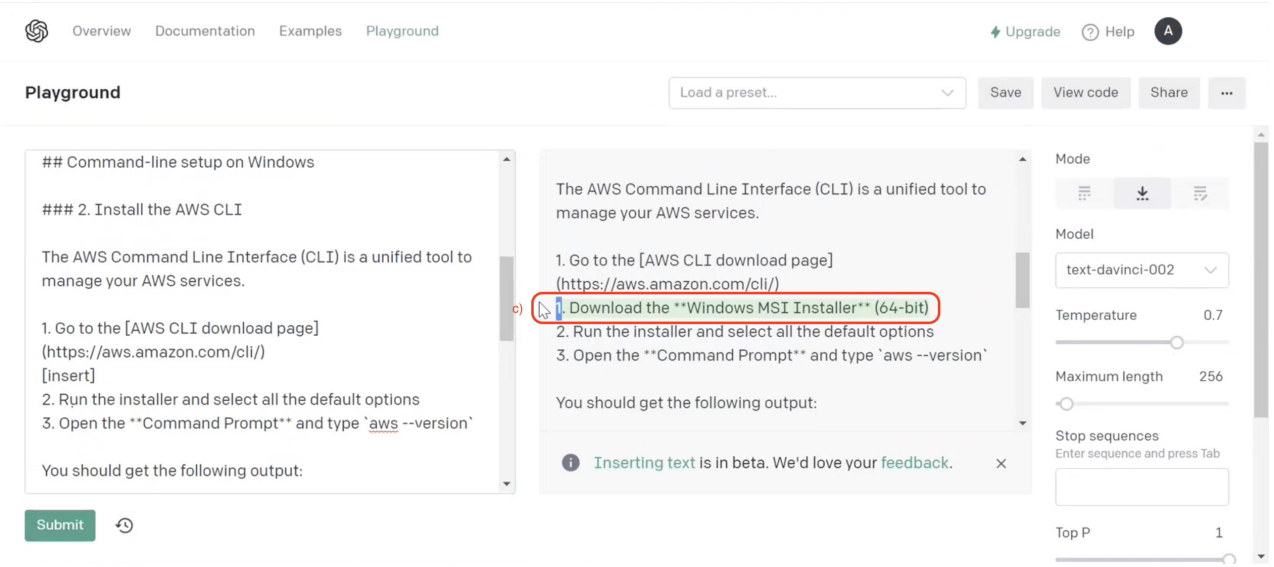
6.2 Conceptual Design

We aim to realize the design guidelines outlined in the previous section in a conceptual design. The design guidelines are aimed at enhancing the overall LLM-assisted writing experience. Writers engaged in authoring processes should have the capability to manage evolving content while addressing information dependencies. To this end, the proposed guidelines suggest the development of a dedicated writing interface tailored for tutorial composition. The interface ensures that the writers can access researched information seamlessly during both the writing and prompting phases. Further, the system should enable mechanisms for content traceability and provenance, enhancing the editing process and accommodating the evolution of tutorials, both during the authoring process and later during maintenance. In addition, writers need enhanced control during prompt creation and should have clear visibility into variations in LLM responses to address usability issues. In order to achieve this, the interface should provide writers with the ability to select relevant sections from their pre-researched materials and in-session writings when prompting the LLM. It's also crucial to clearly visualize the differences between

6.2 Conceptual Design



(a) Two actions performed by the participant while prompting the LLM: a) Participant provides an 'insert' token with an intention directing the LLM to elaborate the steps. b) Next, they submit the prompt.



(b) Actions performed once the LLM generates content: c) Participant verifies the step generated by the LLM and notices that LLM mentions the generated step as the first point.

Figure 6.1: Sequence of steps performed by the user ($P7$) to perform an edit using an LLM during the tutorial writing task.

subsequent LLM prompt-output pairs. Finally, to ensure the quality and accuracy of LLM-generated content, writers should have access to verification and editing features. This can be facilitated by mechanisms that allow for precise verification of LLM outputs, both as individual pieces and within the broader tutorial context. A significant aspect of this is the capacity to switch between granular and holistic editing and prompting strategies effortlessly. We develop four design elements that work in tandem to achieve these interactions. A low-fidelity prototype of the design is shown in 6.2.

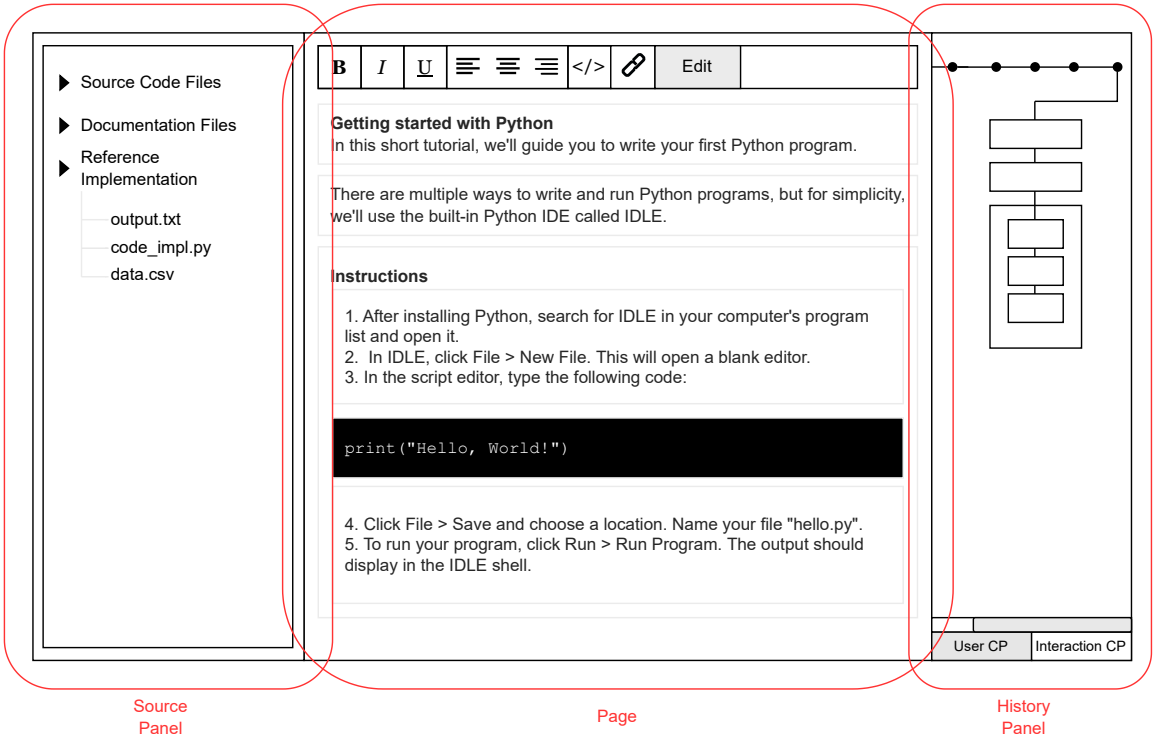


Figure 6.2: A sketch of the prototype. The prototype highlights the *Page*, *Source Panel* and the *History Panel* elements. In the *Edit* mode, authors can manually edit the content without having to invoke LLMs.

6.2.1 Design Elements

The design consists of four design elements aimed at enhancing user interaction with the LLM while integrating LLMs into their existing writing processes. These elements work

in tandem to address the three design guidelines outlined in Section 6.1.

Page. Page is the system's central area, providing affordances for two aspects of LLM-assisted writing, first, the manual writing of the tutorials and second, interaction with the LLMs. The Page has two modes, *edit mode* and *interact mode*. This separation allows for better usability by distinguishing the manual authoring process and LLM-assisted authoring. In the edit mode, the page is a black text area with a toolbar offering common formatting capabilities provided at the top of the page to help writers write and style their content. The Interact mode provides features to create LLM Blocks and use them to interact with an LLM. The LLM Blocks can be structured and reorganized as needed, allowing writers to organize blocks according to their narrative flow or structural needs. Any restructuring of the content is maintained across both the edit and interact modes.

The page offers two visualization strategies to illustrate the relationship between prompts and the corresponding outputs. The first strategy highlights the LLM Blocks selected by the user as the prompt and the resulting output generated by the LLM. The second strategy, similar to diff mode, compares the current prompt-output pair with a selected pair from the history panel. These visualization strategies are designed to enhance the user's comprehension of the impact of prompts on the LLM's output.

LLM Blocks. LLM Blocks are UI elements that allow the writers to engage with LLM for content generation. Each block is essentially an interactive text area for manually writing and editing text. The blocks are visible and interactive only when the Page is set to *interact mode*. By default, all the content in the edit mode of the Page is considered to be in a single LLM Block. The block can be further divided into multiple blocks in order to be able to address specific parts of the content using a button provided on each block. The button splits the block into two based on where the cursor is located. The blocks can then be dragged around and rearranged, allowing for better control of the writer to organize the content. On returning to the *edit mode*, the blocks are highlighted so that authors know how they are structured, but it is not possible to interact with them.

LLM Blocks facilitate granular editing and prompting. Each LLM Block includes a checkbox which allows the writer to explicitly indicate whether its content should be

included in the prompt to the LLM, offering a greater degree of control over the AI's output. The interaction with the LLM within these blocks is through *generation* and *instruct* modes. In generation mode, LLM uses the existing text within the block as a prompt to generate additional text. The instruct mode activates a dedicated textbox attached to the text area of the block, allowing the writer to provide explicit instructions to the LLM. The LLM's response replaces the content in the text area.

LLM Blocks also have the ability to be grouped together which provides for holistic editing and prompting strategies. Multiple blocks can be selected and grouped to create a larger unified LLM Block. The larger LLM Block has the same functionalities as a single LLM Block entity, the only difference being the larger block's content is the aggregated content of the constituent blocks. This feature mimics the traditional writing process, where the writers group the content into sections and subsections. The grouping is reversible, therefore allowing for a seamless transition between granular and holistic editing and prompting.

Source Panel. The source panel facilitates the use of previously researched materials by the authors both for their personal reference and as contextual input to the LLM. The panel allows the writers to import code and documentation files through upload or URL, serving as a centralized location for writers to access and refer to their research materials as they are crafting their content. Writers can also use the imported files in their prompt context, when they are prompting the LLM. This is enabled through the use of checkboxes, similar to those on the LLM Blocks that allow the writers to select the files they want to include in the context when prompting the model. This mechanism ensures information sharing to align LLM output to the researched tutorial context.

History Panel. The History Panel tracks the evolution of a writer's work by recording the state of LLM Blocks at specific checkpoints. These checkpoints are either automatically created during LLM interactions or manually saved by the user. The automatically created checkpoints are developed with the intention of allowing the writer to compare the effectiveness of the prompts and develop a better mental model about prompting. The user saved checkpoints is for the users to maintain versions of their writing. While all user-saved checkpoints are preserved for future reference, only the five most recent interaction checkpoints are retained to provide users with broad yet manageable set

6.2 Conceptual Design

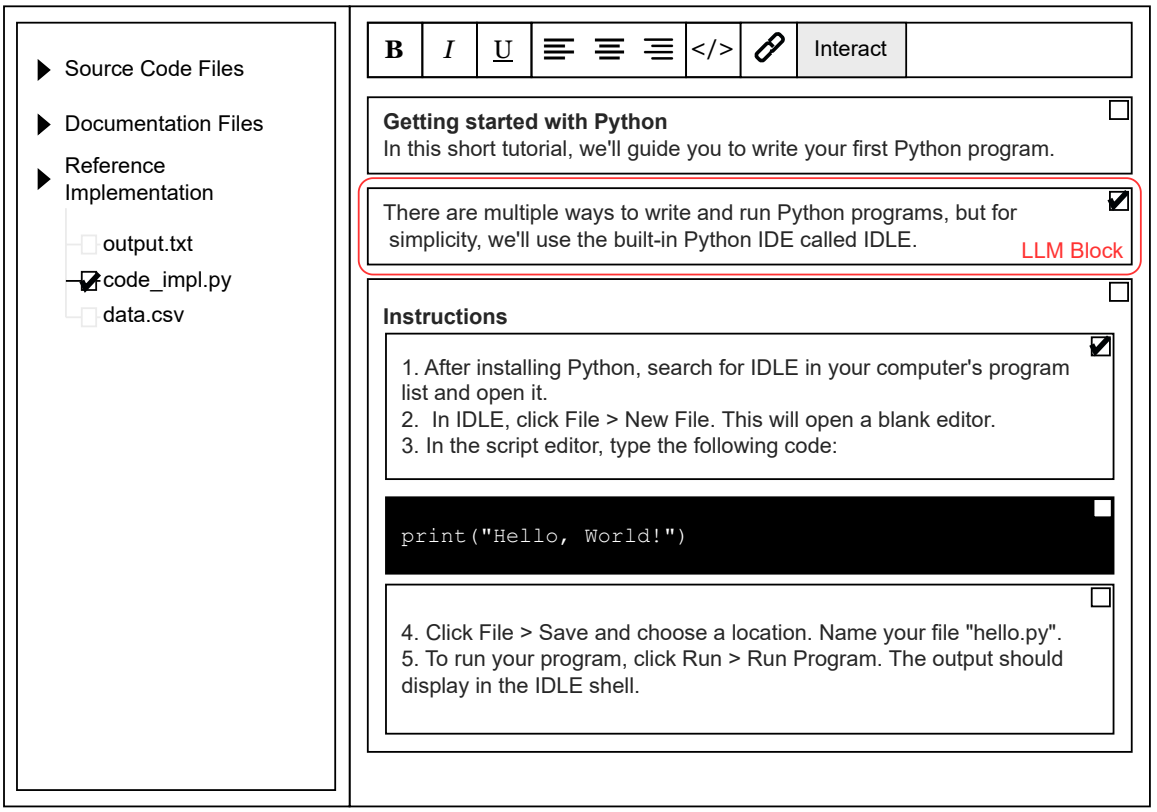


Figure 6.3: Sketch of the Interact mode. The LLM Blocks are highlighted in the Interact mode, indicating that they can be interacted with along with a checkbox that appears on each block. Clicking on the checkbox adds the content of the blocks to the prompt request.

records.

The panel has a horizontally scrollable layout displaying LLM Blocks as individual rectangular nodes in a linear and vertical manner. The hierarchical relationship between individual and grouped blocks is indicated with a larger node enclosing smaller nodes. The panel has a user checkpoint view showing the user-saved checkpoints and an interaction checkpoint view showing automatically created checkpoints both with identical horizontally scrollable layout. The user checkpoint view lets the writers compare the state of current page with state of previously saved user checkpoint. When a checkpoint is selected, differences between the checkpoint and the current state are visually indicated on the panel, guiding writers to the exact sections in the article where changes have occurred. The changes are shown in detail on the Page with green and red

highlights respectively showing added and deleted content. On the other hand, the interaction checkpoint view focuses on tracking the changes in the content resulting from the LLM interactions between the most recent saved checkpoint and the current state. Selecting a specific checkpoint in this view allows the writer to compare the prompt, chosen LLM blocks, and the corresponding output. Any manual edits the writer has made will be indicated in yellow.

Along with comparing state of the content, writers can also revert to a previous state. It's important to note that reverting leads to a 'hard revert', implying that all subsequent checkpoints are lost.

Chapter 7

Discussion

7.1 Potential of LLMs in Tutorial Writing Workflows

In this section, we synthesize and discuss the findings from the hands-on exploration phase outlined in Chapter 4. The needs of the target audience significantly influence the depth and style of the tutorial content. Tutorials designed for a wider audience require greater detail and substantial effort from the participants. Despite their best intentions, the authors make assumptions about the expertise of their readers, as reflected in $P4$'s statement: "[I] might be less worried about who exactly the audience is, it might just be people like me". Such assumptions overlook the distinct learning needs or backgrounds of varied readers. However, these oversights are not deliberate and often happen because authors cannot directly connect with end-users, prompting authors to turn to external sources or lean on their own experiences. Past research emphasizes the value and has provided features to integrate audience feedback into the tutorials [33, 17]. However, a potential challenge in this scenario is when the readers struggle to articulate their needs and communicate with the authors.

The research phase in tutorial writing is not merely for information collection but for a deeper contextual understanding of the topics. Authors draw formal guidelines from official sources like structured professional documentation as well as practical insights from real-world users from YouTube and Reddit. This indicates that the authors try to take diverse perspectives into account, and a thorough understanding isn't built purely from formal sources but also from the practical experiences and challenges of real-world users. A unique source of information for the authors is access to the developers. Direct interactions with developers offer authors the opportunity to find information that is not

available in public domains, such as the design intentions behind specific software decisions. Tutorials play a crucial role in bridging this gap in information, offering a complete view of the topic. However, engaging with developers presents challenges. Developers may unintentionally assume that the authors share their foundational knowledge of the software. For tutorial writers, especially those new to technology, this can make their job difficult since they must not only grasp new concepts fast but also translate them into digestible content for their readers. Additionally, coordinating with developers, especially during peak work phases, is challenging. This tension can sometimes result in delayed or less comprehensive tutorial content. While technology makes collaboration more accessible (e.g., videoconferencing software like Zoom), such flexibility might pave the way for a more interactive tutorial authoring process, with authors and developers refining content as software developments occur.

Artifacts, such as reference source code implementations, screenshots, and GitHub repository links, significantly enrich the tutorials. This is most evident in programming-focused tutorials where the reference source code is not a mere addition but is a central component around which content is structured. Such integration not only brings clarity to complex concepts, but also demonstrates their application in real-world scenarios. Moreover, the tangible nature of reference source code addresses the diverse needs of readers and serves as a marker for authors indicating areas that might benefit from expanded explanations. Development of the reference source code is a meticulous process. Authors frequently reference multiple sources, collaborate with developers, and iteratively refine their implementation to ensure its accuracy. Through these refinements, they are able to identify potential user challenges enabling them to proactively embed solutions within the tutorial. This establishes tutorial authoring as an active problem-solving exercise and highlights the role of tutorials as more than just knowledge dissemination tools.

The creation of high-quality tutorials is an intricate process. Authors prioritize both technical accuracy and a user-friendly presentation, optimizing readability and conciseness to produce easily digestible content. Tutorials are structured to mirror a learner's journey, introducing basic concepts and progressively delving deeper. This scaffolding approach ensures that learners establish a solid foundation before they move

to complex topics. By segmenting the tutorial into focused sections, the authors cater to varied learning paces, providing flexibility in content consumption. Furthermore, while incorporating external resources, the authors synthesize and embed them contextually to foster a holistic understanding. The aim is to reduce the readers' need to seek external information by ensuring the tutorial is as self-contained as possible. Drafting an ideal tutorial involves balancing several quality standards, including content density, conciseness, completeness, context, clarity, and correctness. This meticulous balancing act helps authors create an engaging learning experience, ensuring a seamless educational journey where potential reader questions or confusions are anticipated and addressed. However, even with expertise, writers sometimes encounter challenges in articulating ideas, shedding light on the fact that conveying information effectively is a nuanced art.

After publishing a tutorial, continuous maintenance is necessary due to the software's dynamic nature and feedback from stakeholders, such as developers and end users. This ongoing need for updates underlines the lasting value of a tutorial beyond its publication date. In particular, tutorials demand more frequent updates than other forms of writing. Our study indicates that the maintenance phase is often more challenging than the initial creation, especially for authors managing multiple tutorials. The rapidly evolving tech landscape means that even if a tutorial is accurate at the outset, changes in technology may necessitate updates. This could be a potential reason for the preference for text-based tutorials over video, given the ease of updating the former. Collaboration between writers and developers is essential. Developers, being more attuned to software changes, inform writers of necessary updates. Occasionally, developers might handle minor content updates themselves, ensuring accuracy. This suggests that, while the initial tutorial creation requires extensive expertise and effort, subsequent maintenance can be more straightforward. In domains like machine learning, where rapid changes are common, it's sometimes more efficient to draft a new tutorial rather than update an existing one. This mirrors software development practices where a complete rewrite can be more effective than patching existing code. To streamline maintenance, writers often structure tutorials to be concise and specific. Each tutorial, then, functions as a distinct unit that's simpler to update or replace, benefiting both authors and readers. This approach resonates with the single responsibility design principle in software engineering, which ensures a class has

just one function, making it easier to maintain. Similarly, a focused tutorial addresses one topic, simplifying both maintenances for writers and comprehension for readers.

7.2 Addressing Needs and Challenges in Leveraging LLMs for Tutorial Authoring

In this section, we synthesize and discuss the findings from the hands-on exploration phase outlined in Chapter 5.

LLMs offer the potential to aid the tutorial authoring process, augmenting and enriching the existing aspects of writing and speeding up the process. However, human intervention remains crucial. This outlines the idea that advanced models such as LLMs are tools rather than replacements. During early interactions, authors leverage the LLM on tasks they felt it might handle well, such as generating code snippets. As they familiarized themselves with the model, they realized its broader potential, such as querying it on various technologies, generating drafts and outlines which they could later refine therefore saving time and often finding out overlooked aspects of content. However, there is an evident learning curve as authors move from initial experimentation to more sophisticated and nuanced use of LLMs. This phase of acclimatization to the tool is crucial for optimal results. An intriguing observation is how the use of LLMs can lead to the evolution of content. For instance, a simple AWS setup tutorial could evolve into a more complex S3 bucket creation for hosting, indicating the model's role in expanding the scope based on iterative feedback. The collaboration process became iterative, with the LLM's content serving as a draft, refined later by the authors. This ensures that the final output is technically accurate and contextually relevant. A significant portion of the provided text focuses on how authors shape the output of the LLM to fit their unique styles and needs. This suggests that while LLMs are powerful, their generic outputs may often require tailoring for specific purposes. While authors grew more trusting of the LLM's capabilities, their desire to maintain unique writing styles and ensure content accuracy meant they retained control over the final output. There's an interesting tension between the growing trust in LLMs and the persistent need for control. Even as authors delegate more tasks to the LLM, they are aware of its limitations and continuously iterate upon its

outputs.

Participants prioritize the accuracy of LLM-generated content, as evident in their choice of tutorial topics and the extensive time they dedicate to fact-checking. This emphasis suggests that while participants may appreciate the efficiency and breadth of content generation through LLMs, they remain skeptical about it. Tutorials are instructional, and any inaccuracies can impede the learning process or cause issues for the user. This insight reinforces that while automation tools like LLMs can assist in tutorial writing, human oversight remains indispensable to ensure the highest quality of instruction. To verify the generated content, participants leverage their domain expertise and also cross-reference the content against existing documentation or execute the provided steps. This corroborative approach indicates a deep-rooted trust in established knowledge sources and their own expertise over the LLM's outputs. The utilization of domain knowledge and external verification methods shows the skepticism participants have towards machine-generated content. This skepticism indicates an understanding that automated systems, regardless of their sophistication, can make mistakes or overlook nuances. The diverse verification methods, whether it's cross-referencing, execution, or leveraging personal expertise, further elucidate the participants' commitment to ensuring that the generated content is both correct and applicable. Manual editing emerges as an important step in the content generation process. This observation suggests a gap in the LLM's ability to discern the intricacies of the author's style and the specific requirements of the tutorial's content. However, it's also worth noting that manual editing isn't solely a function of LLM's shortcomings. Every author has a unique style, and even if the LLM's content is technically accurate, it might still require adjustments to fit their desired tone, style, or depth.

Participants' prior experiences and understanding shaped their expectations and interactions with LLMs. Some were optimistic about the potential of AI-assisted writing, while others were more cautious or skeptical. This underlines the importance of introductory training or orientation. Participants' prior biases can significantly impact their approach, making them either overly optimistic or overly critical. Addressing this with thorough initial training can set realistic expectations and better harness LLMs' potential. Participants were keen on understanding LLM's scope, its up-to-date

knowledge, and its connectivity with other software. They were actively trying to gauge how recent the LLM's knowledge was, and from where it sourced its information. This suggests a need for transparency from the LLM's developers about its training data, its update frequency, and its limitations. Users seem to want a clear understanding of what they are working with, and this clarity can boost confidence in the tool. Participants invested significant effort in refining prompts and optimizing parameters to elicit desired responses from the LLM. However, they often faced confusion about the connection between their prompts and the LLM's responses. The effort expended on prompt optimization indicates that LLMs may have a steep learning curve for optimal usage. A more user-friendly interface, feedback mechanisms, or guidelines on effective prompting could streamline this process and enhance user satisfaction. Participants actively revised their understanding of LLM capabilities based on its outputs. They formed hypotheses, sometimes incorrectly, which led to concerns about the tool's potential utility in tutorial writing. This continuous calibration of understanding underscores the dynamic nature of LLM interactions. Misunderstandings can affect user confidence and trust in the tool. Clear communication regarding LLM capabilities, perhaps via documentation or interactive guides, can help align user expectations and reality. Participants sought external references to verify the LLM's outputs and expressed concerns when they noticed inconsistencies or when content didn't match their expectations. Trust is a fundamental aspect of technology adoption. The concerns raised by participants emphasize the need for more consistent LLM outputs. Perhaps introducing a feature that cites sources or offers a confidence level for given information could alleviate some of these concerns.

Participants' experiences with the LLMs were tainted by poor usability. This was evident from the fact that incorrect mental models about the LLM's capabilities were formed, which sometimes led to participants abandoning the tool altogether. The participants' struggles underscore the importance of intuitive tool design. A tool meant to aid tutorial writing or any similar task should seamlessly integrate into the workflow. Here, users were sidetracked trying to understand the tool instead of focusing on their primary writing tasks. OpenAI Playground's dual use of the textbox (both for input and context) confused users. Participants found it hard to navigate the tool's usability when they wanted the model to consider previous text, which is often necessary for continuity in

tutorial writing. While some users found workarounds, they were neither intuitive nor scalable. While having a textbox that captures all interactions might provide continuity, it's clear that this design can confuse users. They need to be explicitly made aware of how the context influences the model's output. Potential solutions could involve segregating prompts from the context or providing clearer visual cues. Crafting an effective prompt became an art, distracting participants from their main writing task. Lack of clarity on how model parameters (e.g., token length) influence the output led to misunderstandings and suboptimal outputs. The LLMs' capabilities require a learning curve. Users should be given guidance on crafting prompts, understanding token limitations, and utilizing the tool's features optimally. This might come in the form of tutorials, tooltips, or inline examples. The tool's poor usability hindered participants from understanding and effectively using its distinct features, such as Instruct and Insert/Edit modes. Moreover, manual efforts like copying and pasting between modes made it tedious. For users to appreciate and leverage the full power of a tool, its features need to be easily discoverable, which can be achieved through an intuitive interface. Participants' reliance on the interviewer for guidance indicates a clear need for better onboard instructions or more intuitive design. Features like copying content from one mode to another or managing multiple inserts should be streamlined. A long term adoption of LLM for any task hinges on user experience. It's evident that while some participants who navigated the usability challenges saw the potential of LLMs in tutorial writing, others were put off. The broader adoption of LLMs in such settings will largely depend on how these usability concerns are addressed. Addressing these issues can turn the tool from a potential hindrance into a powerful asset for writers. In essence, while the underlying model and its capabilities are powerful and promising, the interface and interaction design play a pivotal role in its successful adoption. Addressing usability issues is not just about making the tool easier to use but also about ensuring users form the right mental models about its capabilities.

7.3 Framework for Interaction with LLM

In this section, we present and discuss a framework for human-LLM interaction design, informed by insights elicited from the hands-on exploration phase of our user study. The framework groups the various intricate processes users employ when engaging with an

LLM, from the initial formulation of goals, prompting the model to achieve these objectives, to the verification of generated content and the eventual refinement of the goals and interaction strategies based on the outputs. This sequence of user actions and reflections can be conceptualized as distinct stages of interaction with the LLM. As a result, we have termed this the **'Stages of Interaction with LLMs'** or the **SIL framework**. Our objective of the SIL framework is to provide comprehensive guidelines, aiding model developers and tool designers in crafting LLM-based tools that are both efficient and optimized for user productivity.

7.3.1 Four stages of the Framework

Formulation Stage

The Formulation Stage is the initial phase in which users define their intentions and expectations when approaching the LLM. During this stage, users solidify their objectives, develop initial expectations based on previous knowledge or perceptions, and actively gauge the LLM's capabilities and limitations. This stage essentially lays a foundation for all subsequent interactions with the LLM.

The motivation for interacting with the LLM might arise from an immediate need for information, intention to generate content towards a broader goal, exploration of possible ways to elicit optical content from the LLM or just curiosity towards the LLM. As users navigate this stage, they become more concrete about their objectives, ranging from seeking straightforward answers to constructing intricate documents and corresponding interactions, ranging from exploratory attempts to precise and task-specific queries. Previous interactions or secondhand knowledge about the LLM significantly shape users' initial expectations. It is evident that misaligned expectations can lead to dissatisfaction, as observed during the exploration phase. On one end of the spectrum, optimistic users ambitiously probe the LLM's potential, while on the other end, cautious users narrow down their interactions seeking specific outcomes. These diverse approaches underscore the importance of aligning user expectations with the LLM's actual capabilities. The observed unpredictability in the model's outputs and the formation of inaccurate mental models, often exacerbated by tool usability issues, further complicate this alignment.

Ultimately, the Formulation Stage plays a pivotal role, setting the stage for users' subsequent engagements with the LLM.

Articulation Stage

The Articulation Stage is characterized by users' efforts to effectively communicate their intentions to the LLM. Within this phase, participants focus on formulating and refining prompts and ensuring that they capture their intention accurately in a query. The stage encapsulates challenges tied to the interface design, the comprehension of model parameters, and the overall clarity of user inputs. The success of the interaction heavily relies on how well users can articulate their needs and how effectively the system's design can aid in this articulation process.

The process of forming a prompt that captures the user's intention and ensures it is interpreted by the LLM accurately is a nuanced and difficult task. This could be due to several reasons. Users might not always know the best way to phrase their queries, potentially resulting in ambiguous or misdirected prompts. Their perceptions of what the LLM can or cannot do can skew the prompts toward either oversimplification or excessive ambition. Moreover, when the objectives are not clear in the user's mind, formulating a concise prompt becomes even more complex. Moreover, complex tasks often might necessitate elaborate prompts or even multiple iterations. However, users might either be unaware of the need to iterate or might be reluctant to engage in such a repetitive process. Therefore, users need to be guided during the articulation process, either by external training resources or through the interface assisting them during the interaction process to optimally leverage the LLM's capability.

The usability of the interface plays a crucial role in the articulation stage. The interface bridges the gap between the user's intentions and the computational logic of the LLM. A well-designed interface can help the user interact with the model. and additionally guide and inform the user. Conversely, if an interface is not intuitively designed or lacks clarity, users might find it difficult to interact with the LLM, leading to user frustration, misinterpretation of user prompts by the LLM and ineffective outcomes. Therefore, it is important that the interface actively facilitates the interactions between the user and the LLM. The exploration phase clearly shows the consequence of poor usability

of an LLM interface. The first example is the dual-purpose design of the text box in the OpenAI playground. The use of the textbox to both prompt the model and edit the content leads to confusion among the users. They are unsure where to input their prompts and how to differentiate their prompts from the model's output. Notably, many users were unaware that the entire textbox content acted as the prompt until informed by the interviewer. This situation demonstrates that interface design should focus on clarity and ease of use instead of adding potentially confusing features. The second observation revolves around the ambiguity in the model's parameters. A specific point of confusion was the token length, which, if exceeded, truncates the generated content. Some users, unaware of this restriction, presumed the abruptly shortened content as a model flaw. This underscores the need for interfaces to offer more transparent and intuitive parameter management. Lastly, while the users found value in using both the interaction modes with the model, such as the completion or edit mode, the interface's provision to do so was not intuitive. Effective interfaces should match users' task flow, allowing easy changes between tasks. Struggling with such basic functionalities can occupy a user's cognitive load and shift their attention from generating content to troubleshooting.

Observation

The Observation Stage emphasizes the engagement of the user with the LLM output. Users do not just consume the output but rather carefully evaluate it. While accuracy is one of the prominent factors for evaluation, users also ensure the reliability and relevance of the generated output in relation to their initial prompts and the current context of their writing.

Interaction with a machine needs to be precise, both in terms of accuracy and alignment with the user's intent. Especially when we consider LLMs, which pose significant difficulty with prompting and are traditionally prone to hallucinations, it is necessary to evaluate the generated content. The accuracy of the content can be verified through the user's own domain expertise, cross-referencing with reputable external sources, or in the case of actionable content like tutorials, testing the content's real-world applicability through executing it. Beyond factual accuracy, there is also a need to ensure alignment with user intentions. A potential check is to compare the generated content against the user's original prompt to ensure that the output does not stray from the user's

request. Another important aspect is the tone and style of the content. Specifically, in the context of writing support, users want the generated information delivered in a tone consistent with their personal authorship style to make the content feel more authentic and relatable. Furthermore, the generated output needs to be consistent with the existing content from the previous interaction iterations. As users engage with the LLM over time, they are not just focused on isolated outputs but rather looking at how the content evolves and stays connected to their overarching goals. Each new interaction should build upon and enhance the content but not deviate or become disjointed from the context.

Revision

The Revision Stage is the final step in the user-LLM interaction cycle. In this stage, users develop and refine hypotheses about the functioning of the LLM based on a comparison of the LLM's outputs to their initial prompts and expectations. Acceptances or minor adjustments to the LLM's outputs suggest alignment of the output with users' expectations, while revisions indicate a need for refining their understanding of the LLM's capabilities and subsequent interactions.

The revision stage centers on reflecting on the interaction strategies from the Articulation stage, the LLM's corresponding responses, and the evaluation conducted during the Observation stage. The objective is to identify and comprehend the root causes behind any output discrepancies. If deviations arise from user prompts, those are revised for subsequent sessions. When the LLM's outputs are frequently accepted or require minimal revisions, it indicates a growing trust in the LLM's capabilities. This can lead to expansion in the breadth of tasks users delegate to the LLM, moving beyond simple requests to seeking assistance with complex content creation and editing.

The expansion in the tasks delegated is influenced by three primary factors: the updated perception of the LLM's capabilities, enhanced knowledge of interaction strategies, and the evolving content produced by the LLM. The first two factors correspond to the mental models of the users and can have long-term implications for the usage of LLM. The final factor has to do with the task that the user set out and indicates the success of the short-term goal. In the initial stages, interactions tend to be broad or general, reflective of users tentatively testing the LLM's fundamental capabilities. As they

witness the machine's reactions to diverse prompts, they gain insights into its strengths, weaknesses, and nuances. These insights, combined with an updated understanding of effective interaction strategies, drive the users to refine their approach. This adaptability highlights the feedback cycle between human goals and corresponding machine output. As users become more aware of the capabilities of the LLM, their trust in the system increases leading to them delegating more complex tasks to the LLM. Interactions that begin out of curiosity soon evolve from basic questions to deeper insights. Users rely on the LLM for more advanced content and complex tasks. This shift in task complexity is due to users adjusting their goals based on a clearer understanding of the LLM's abilities, better ways to interact with it, and the quality of the content it provides.

Conversely, when users consistently need to make extensive revisions to the LLM's outputs, it signals a disconnect between the expectations of the users and the capabilities of the LLM. Continuous revisions might lead to users doubting the clarity of their instructions and the LLM's capability to comprehend and deliver. This can lead to users eventually stop relying on the LLM. They might limit its use to less critical tasks or, in more pronounced cases, consider abandoning it due to perceived inefficiencies.

7.3.2 Comparison with Norman's Seven Stages of Action

Our proposed framework draws inspiration from Norman's seven stages of action framework [41] and specifically addresses the various aspects that are specific to LLM interactions. Starting with the various stages, both models recognize the necessity of intent identification. The Formulation stage of our SIL framework and Norman's Goal Formation and Plan stages discuss the user's intent. The Seven Stages of Action framework identify the progression from goal identification to planning its achievement. Similarly, in the Formulation stage, users not only identify but also implicitly navigate how to leverage the LLM based on previous knowledge and system capabilities. The Articulation stage in the LLM framework mirrors Specify and Perform stages, focusing on translating intentions into actionable commands and executing these. However, the Specify and the Perform stages, while implying interface interactions, are unclear when distinguishing between actions like specifying and performing in LLM contexts. From our user study, we see that clarity is, in fact, necessary. The interface and its usability play a

crucial role in translating the intentions of the user to the LLM. The explicit emphasis on the interface in the Articulation stage highlights the importance of its design in determining the outcome of LLM interactions. Similarly, in the context of an LLM-generated output, Perceive and Interpret are analogous. Therefore we reduce these two stages into a single Observation stage. Finally, while the Compare stage is similar to our Revision stage, we emphasize the user-LLM collaboration dynamics, which affects the evolving user trust and future interactions, which is unexplored in the Seven Stages of Action framework. Therefore a four-stage framework segregates the interaction better than the Seven Stages of Action framework, at least in the context of LLMs.

A second distinction is the implications for design targeted by the two frameworks. The Seven Stages of Action framework outlines three design dimensions – Visceral (comprising Perform and Perceive), Behavioral (comprising Specify and Interpret) and Reflective (comprising Plan and Compare). The visceral dimension corresponds to the immediate reaction to how something looks, sounds, or feels. It's the first impression we get from a product based on its appearance and sensory feedback. The behavioural dimension corresponds to the overall experience of interacting with a product, such as ease of use. Finally, the reflective dimension corresponds to the broader implications of a product's value and significance. These design dimensions enable the designers to target better experiences, often through evoking unique emotions which result in positive user experiences. Conversely, our framework is rather focussed on ensuring better designs for assisting the users in understanding the capability of the LLM (in the *Formulation* stage), improving the usability of the interface in prompting (in the *Articulation* stage), allowing for efficient evaluation and comparisons (in the *Observation* stage) and finally providing affordances for easy revision of the goals (in the *Revision* stage).

In essence, both frameworks aim to enhance human-system interactions. While Norman's Seven Stage of Action framework lays down broad principles, our framework is tailored to address unique issues of user-LLM collaboration and provide structured guidelines for model developers and tool designers to specifically develop efficient and productive LLM-based tools.

7.3.3 Illustrating the Framework through an example

To illustrate the application of our four-stage framework, we investigate a potential scenario[1] of using LLMs for creating software tutorials, inspired by the hands-on exploration stage of our user study.

In a typical interaction, the user begins by identifying a primary objective. Taking our study as an example, the goal is to craft a tutorial on plotting data with matplotlib. The user then segments this overarching aim into more actionable parts to navigate the interactions with Codex on the OpenAI playground. For example, the goal can be divided into producing relevant commands for library installation in different environments, generating and explaining code snippets, and improving the readability of the tutorial. Such segmentation demonstrates the intricacies of the Formulation Stage. Following this, with more clarity, the user prompts Codex. They might input a specific request, such as 'Provide a code snippet for a scatter plot using matplotlib, given Python list data points, and also explain the code.' The user's experience and familiarity with the topic play a crucial role for them to design a good prompt. Additionally, interface features that offer alternate prompt suggestions or enable prompt modifications can further enhance this Articulation stage. The Observation stage begins once Codex delivers its output. The user, depending on his expertise, analyzes the accuracy and relevance of the generated output. For instance, a user experienced with matplotlib would be more adept at spotting any unusual patterns or discrepancies in the provided code. If not, they might visit external resources to ensure the output is correct. This stage might also involve validating the output in real-world scenarios, like executing the returned code snippet in an Integrated Development Environment (IDE) or testing against predefined unit tests. Finally, in the Revision stage, the user reflects on both their interaction method and the responses from Codex. If the initial output isn't satisfactory, they will revise their approach, potentially rephrasing their query or adjusting their expectations based on the feedback. This iterative cycle, where the user revises their interactions based on the machine's output, underscores the continuous feedback loop that's central to our framework.

[1] This example is inspired by our position paper titled "Approach Intelligent Writing Assistants Usability with Seven Stages of Action"[4]

Through this scenario, we depict the unique challenges and considerations when collaborating with a language model, specifically in the domain of software tutorial generation.

7.4 Future Work

The design guidelines presented in Section 6.1 are motivated as a result of our user study. However, they need further validation to ensure that they are beneficial to users. The conceptual design outlined in Section 6.2 serves as a way to validate these guidelines. A future user study is currently planned to validate and refine the design before implementation, and a final summative study to observe the practical utility of the tool. Furthermore, while the SIL framework is discussed separately, the guidelines also correspond to the four stages outlined in the framework. We plan to expand on this and dedicate future user studies to validate the framework using the developed tool.

7.5 Limitations of our study

Identification of the various stages of interaction was made based on the hands-on exploration stage, which was conducted over a brief duration and was open-ended, which may not capture the full range of interactions that technical writers can have with the LLM. Moreover, these interactions might differ in a specialized tool for documentation writing rather than a general-purpose tool. However, we observed rich interactions even within this short period, which only underscores the need for extended studies with dedicated tasks to understand the interaction dynamics in a tutorial writing setting more comprehensively. Our method of grouping the interaction processes with the model is nuanced, particularly during the Goal Formation, Planning, and Compare stages. These stages are largely internal to the user and are difficult to study directly. While the read-aloud protocol helped capture these stages, as participants became accustomed to the tool, these stages seemed to blend together, making them harder to distinguish. After familiarization with the tool, some of the strategies and interactions become second nature to the participants, which complicates the identification of distinct interaction stages. This can also make it harder for designers to target specific stages and understand where users

may need assistance. Despite these limitations, our study provides a good lens for exploring user interaction with LLMs. Having focused writing tasks, longer study duration and specialized tools with necessary telemetry can facilitate the study to understand the human-LLM interactions.

Chapter 8

Conclusion

In our study, we investigated the existing workflows of the authors in tutorial writing and the potential benefits and challenges of leveraging LLMs. Our findings indicate that authors prioritize the needs of their readers, blending their personal experiences with thorough research of both official documentation and practical applications from diverse platforms. Many authors even collaborate with developers to enrich the information and ensure the tutorial is accurate and easy to understand. By focusing on specific subjects, authors make their tutorials more reader-friendly and easier to update as software changes. Once published, tutorials require continuous updates due to software-related changes, and structuring tutorials with a focused topic can simplify this maintenance while enhancing reader comprehension.

Our results further highlight the need for effective collaboration between humans and AI and underscore the need for designs to facilitate it. While LLMs can streamline the writing process, it is essential for authors to review and verify the content for its accuracy. In their early interactions with LLMs, authors often cross-check the model's outputs against trusted sources. Their past experiences and initial perceptions significantly shape how they use the LLM. Participants faced challenges in understanding the model's response to prompts and desired more user-friendly interfaces and clear guidelines. Poor usability of interfaces and unclear prompting mechanisms prevented the optimal use of LLMs, with some participants resorting to manual workarounds. For LLMs to be widely adopted in tutorial writing or similar tasks, it's crucial to address these usability issues, ensuring seamless integration into writers' workflows.

Given these insights, we present two primary recommendations. First, we distill and present several design guidelines essential for optimizing tutorial writing with LLMs

based on the interview phase and the hands-on exploration phase of the study. These emphasize the key factors in the tutorial authoring process, such as output accuracy, traceability, usability, author control, rigorous verification, and flexible editing. As LLMs contribute suggestions, they should leverage and align to the author's prior research, ensuring content remains both relevant and accurate. Offering traceability and a historical overview of tutorial changes are pivotal to improving the users' mental model. Similarly, the distinction between writers' prompts and LLM outputs should be clear to mitigate confusion. Moreover, flexibility is crucial, and authors must have the control to choose specific content sections when seeking LLM input. Rigorous verification tools, especially for LLM-generated links and code snippets, are necessary. Additionally, the editing should account for both granular and overarching content revisions. In essence, these guidelines bridge the technical capacities of LLMs with the specific needs of tutorial writers.

The second recommendation offers a broader perspective, suitable for general LLM tool design. We propose a framework that outlines the user's interaction with LLMs into four stages: *Formulation*, *Articulation*, *Observation*, and *Revision*. In the *Formulation* stage, users define objectives and establish initial expectations based on past experiences. The *Articulation* stage involves the clear expression of intentions formed to the LLM, with emphasis on effective prompt creation and usable interface. During the *Observation* stage, users examine LLM outputs for accuracy and alignment with their initial intentions, considering factors like context. In the *Revision* stage, users adjust their understanding of the LLM's abilities by comparing results to initial expectations and prompts. Positive interactions increase trust and encourage users to delegate more complex tasks to the LLM. In contrast, frequent revisions suggest a gap between expectations and LLM performance. The clarity and intuitiveness of the interface play a key role in user experiences, and continuous feedback loops between human intentions and LLM responses contribute to subsequent interactions.

In summary, we highlight the potential of LLMs in enhancing tutorial writing while emphasizing the challenges related to usability and trust. As AI continues to evolve, the integration of human expertise and machine capabilities will be pivotal in crafting comprehensive and precise content. The discussed design guidelines and framework form a crucial bridge between the capabilities of LLMs and the nuanced needs of authors.

Contributions

A preliminary version of the Stages of Interaction with LLMs framework detailed in Section 7.3 was submitted as a position paper [4] to the In2Writing workshop.[1] Major parts of this thesis will be reworked into a submission to a human-computer interaction conference.

The collaborators contributed to the framework and provided valuable inputs to the design of the user study.

[1] https://in2writing.glitch.me/papers2023.html

Bibliography

[1] Amershi, S., Weld, D., Vorvoreanu, M., Fourney, A., Nushi, B., Collisson, P., Suh, J., Iqbal, S., Bennett, P. N., Inkpen, K., Teevan, J., Kikin-Gil, R., and Horvitz, E. (2019). Guidelines for Human-AI Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 1–13, New York, NY, USA. Association for Computing Machinery.

[2] Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., Joseph, N., Kadavath, S., Kernion, J., Conerly, T., El-Showk, S., Elhage, N., Hatfield-Dodds, Z., Hernandez, D., Hume, T., Johnston, S., Kravec, S., Lovitt, L., Nanda, N., Olsson, C., Amodei, D., Brown, T., Clark, J., McCandlish, S., Olah, C., Mann, B., and Kaplan, J. (2022). Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. arXiv:2204.05862 [cs].

[3] Bhat, A., Agashe, S., Oberoi, P., Mohile, N., Jangir, R., and Joshi, A. (2023a). Interacting with Next-Phrase Suggestions: How Suggestion Systems Aid and Influence the Cognitive Processes of Writing. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*, IUI '23, pages 436–452, New York, NY, USA. Association for Computing Machinery.

[4] Bhat, A., Shrivastava, D., and Guo, J. L. C. (2023b). Approach Intelligent Writing Assistants Usability with Seven Stages of Action. arXiv:2304.02822 [cs].

[5] Biermann, O. C., Ma, N. F., and Yoon, D. (2022). From Tool to Companion: Storywriters Want AI Writers to Respect Their Personal Values and Writing Strategies. In *Proceedings of the 2022 ACM Designing Interactive Systems Conference*, DIS '22, pages 1209–1227, New York, NY, USA. Association for Computing Machinery.

[6] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G.,

Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language Models are Few-Shot Learners. arXiv:2005.14165 [cs].

[7] Buschek, D., Eiband, M., and Hussmann, H. (2022). How to Support Users in Understanding Intelligent Systems? An Analysis and Conceptual Framework of User Questions Considering User Mindsets, Involvement, and Knowledge Outcomes. *ACM Transactions on Interactive Intelligent Systems*, 12(4):29:1–29:27.

[8] Buschek, D., Mecke, L., Lehmann, F., and Dang, H. (2021). Nine Potential Pitfalls when Designing Human-AI Co-Creative Systems. arXiv:2104.00358 [cs].

[9] Calderwood, A., Qiu, V., Gero, K., and Chilton, L. B. (2020). How Novelists Use Generative Language Models: An Exploratory User Study.

[10] Chakrabarty, T., Padmakumar, V., and He, H. (2022). Help me write a Poem: Instruction Tuning as a Vehicle for Collaborative Poetry Writing. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6848–6863, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[11] Christiano, P., Leike, J., Brown, T. B., Martic, M., Legg, S., and Amodei, D. (2023). Deep reinforcement learning from human preferences. arXiv:1706.03741 [cs, stat].

[12] Chung, J. J. Y., Kim, W., Yoo, K. M., Lee, H., Adar, E., and Chang, M. (2022). TaleBrush: Sketching Stories with Generative Pretrained Language Models. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, pages 1–19, New York, NY, USA. Association for Computing Machinery.

[13] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs].

[14] Donahue, C. and Lillis, T. (2014). 4 Models of writing and text production. In *4 Models of writing and text production*, pages 55–78. De Gruyter Mouton.

[15] Dong, Q., Li, L., Dai, D., Zheng, C., Wu, Z., Chang, B., Sun, X., Xu, J., Li, L., and Sui, Z. (2023). A Survey on In-context Learning. arXiv:2301.00234 [cs].

[16] Du, W., Kim, Z. M., Raheja, V., Kumar, D., and Kang, D. (2022). Read, Revise, Repeat: A System Demonstration for Human-in-the-loop Iterative Text Revision. In *Proceedings of the First Workshop on Intelligent and Interactive Writing Assistants (In2Writing 2022)*, pages 96–108, Dublin, Ireland. Association for Computational Linguistics.

[17] Dubois, P., Dziubak, V., and Bunt, A. (2017). Tell Me More! Soliciting Reader Contributions to Software Tutorials. In *Proceedings of Graphics Interface 2017*, volume Edmonton, pages 8 pages, 632.60 KB. Canadian Human-Computer Communications Society / Société canadienne du dialogue humain-machine. Artwork Size: 8 pages, 632.60 KB ISSN: 0713-5424 Medium: application/pdf.

[18] Eiband, M., Buschek, D., and Hussmann, H. (2021). How to Support Users in Understanding Intelligent Systems? Structuring the Discussion. In *26th International Conference on Intelligent User Interfaces*, IUI '21, pages 120–132, New York, NY, USA. Association for Computing Machinery.

[19] Flower, L. and Hayes, J. R. (1981). A Cognitive Process Theory of Writing. *College Composition and Communication*, 32(4):365–387. Publisher: National Council of Teachers of English.

[20] Gero, K., Calderwood, A., Li, C., and Chilton, L. (2022a). A Design Space for Writing Support Tools Using a Cognitive Process Model of Writing. In *Proceedings of the First Workshop on Intelligent and Interactive Writing Assistants (In2Writing 2022)*, pages 11–24, Dublin, Ireland. Association for Computational Linguistics.

[21] Gero, K. I., Liu, V., and Chilton, L. (2022b). Sparks: Inspiration for Science Writing using Language Models. In *Proceedings of the 2022 ACM Designing Interactive Systems Conference*, DIS '22, pages 1002–1019, New York, NY, USA. Association for Computing Machinery.

[22] Gero, K. I., Long, T., and Chilton, L. B. (2023). Social Dynamics of AI Support in Creative Writing. In *Proceedings of the 2023 CHI Conference on Human Factors*

in Computing Systems, CHI '23, pages 1–15, New York, NY, USA. Association for Computing Machinery.

[23] Ghazvininejad, M., Shi, X., Priyadarshi, J., and Knight, K. (2017). Hafez: an Interactive Poetry Generation System. In *Proceedings of ACL 2017, System Demonstrations*, pages 43–48, Vancouver, Canada. Association for Computational Linguistics.

[24] Ginosar, S., De Pombo, L. F., Agrawala, M., and Hartmann, B. (2013). Authoring multi-stage code examples with editable code histories. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 485–494, St. Andrews Scotland, United Kingdom. ACM.

[25] Hayes, J. and Flower, L. (1981). Uncovering Cognitive Processes in Writing: An Introduction to Protocol Analysis.

[26] Hayes, J. R. (2012). Modeling and Remodeling Writing. *Written Communication*, 29(3):369–388. Publisher: SAGE Publications Inc.

[27] Head, A., Jiang, J., Smith, J., Hearst, M. A., and Hartmann, B. (2020). Composing Flexibly-Organized Step-by-Step Tutorials from Linked Source Code, Snippets, and Outputs. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, pages 1–12, New York, NY, USA. Association for Computing Machinery.

[28] Ippolito, D., Yuan, A., Coenen, A., and Burnam, S. (2022). Creative Writing with an AI-Powered Writing Assistant: Perspectives from Professional Writers. arXiv:2211.05030 [cs].

[29] Jameson, A. D. (2009). Understanding and Dealing With Usability Side Effects of Intelligent Processing. *AI Magazine*, 30(4):23–23. Number: 4.

[30] Kaddour, J., Harris, J., Mozes, M., Bradley, H., Raileanu, R., and McHardy, R. (2023). Challenges and Applications of Large Language Models. arXiv:2307.10169 [cs].

[31] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling Laws for Neural Language Models. arXiv:2001.08361 [cs, stat].

[32] Kreminski, M. and Martens, C. (2022). Unmet Creativity Support Needs in Computationally Supported Creative Writing. In *Proceedings of the First Workshop on Intelligent and Interactive Writing Assistants (In2Writing 2022)*, pages 74–82, Dublin, Ireland. Association for Computational Linguistics.

[33] Lafreniere, B., Grossman, T., and Fitzmaurice, G. (2013). Community enhanced tutorials: improving tutorials with multiple demonstrations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 1779–1788, New York, NY, USA. Association for Computing Machinery.

[34] Lee, Y., Kim, T. S., Chang, M., and Kim, J. (2022). Interactive Children's Story Rewriting Through Parent-Children Interaction. In *Proceedings of the First Workshop on Intelligent and Interactive Writing Assistants (In2Writing 2022)*, pages 62–71, Dublin, Ireland. Association for Computational Linguistics.

[35] Li, Y., He, J., Zhou, X., Zhang, Y., and Baldridge, J. (2020). Mapping Natural Language Instructions to Mobile UI Action Sequences. arXiv:2005.03776 [cs].

[36] Lieberman, H. (2009). User Interface Goals, AI Opportunities. *AI Magazine*, 30(4):16–16. Number: 4.

[37] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. (2023). Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Computing Surveys*, 55(9):195:1–195:35.

[38] Mirowski, P., Mathewson, K. W., Pittman, J., and Evans, R. (2023). Co-Writing Screenplays and Theatre Scripts with Language Models: Evaluation by Industry Professionals. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, pages 1–34, New York, NY, USA. Association for Computing Machinery.

[39] Mysore, A. and Guo, P. J. (2017). Torta: Generating Mixed-Media GUI and Command-Line App Tutorials Using Operating-System-Wide Activity Tracing. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, UIST '17, pages 703–714, New York, NY, USA. Association for Computing Machinery.

[40] Mysore, A. and Guo, P. J. (2018). Porta: Profiling Software Tutorials Using Operating-System-Wide Activity Tracing. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, pages 201–212, New York, NY, USA. Association for Computing Machinery.

[41] Norman, D. A. (2002). *The Design of Everyday Things*. Basic Books, Inc., USA.

[42] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. arXiv:1802.05365 [cs].

[Radford et al.] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving Language Understanding by Generative Pre-Training.

[44] Reynolds, L. and McDonell, K. (2021). Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI EA '21, pages 1–7, New York, NY, USA. Association for Computing Machinery.

[45] Roemmele, M. (2021). Inspiration through Observation: Demonstrating the Influence of Automatically Generated Text on Creative Writing. arXiv:2107.04007 [cs].

[46] Ross, S. I., Martinez, F., Houde, S., Muller, M., and Weisz, J. D. (2023). The Programmer's Assistant: Conversational Interaction with a Large Language Model for Software Development. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*, IUI '23, pages 491–514, New York, NY, USA. Association for Computing Machinery.

[47] Shen, Z., August, T., Siangliulue, P., Lo, K., Bragg, J., Hammerbacher, J., Downey,

D., Chang, J. C., and Sontag, D. (2023). Beyond Summarization: Designing AI Support for Real-World Expository Writing Tasks. arXiv:2304.02623 [cs].

[48] Shneiderman, B. (2020). Bridging the Gap Between Ethics and Practice: Guidelines for Reliable, Safe, and Trustworthy Human-centered AI Systems. *ACM Transactions on Interactive Intelligent Systems*, 10(4):26:1–26:31.

[49] Singh, N., Bernal, G., Savchenko, D., and Glassman, E. L. (2022). Where to Hide a Stolen Elephant: Leaps in Creative Writing with Multimodal Machine Intelligence. *ACM Transactions on Computer-Human Interaction*. Just Accepted.

[50] Sun, S., Zhao, W., Manjunatha, V., Jain, R., Morariu, V., Dernoncourt, F., Srinivasan, B. V., and Iyyer, M. (2021). IGA : An Intent-Guided Authoring Assistant. arXiv:2104.07000 [cs].

[51] Swain, J. (2018). *A Hybrid Approach to Thematic Analysis in Qualitative Research: Using a Practical Example*. SAGE Publications Ltd.

[52] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2023). Attention Is All You Need. arXiv:1706.03762 [cs].

[53] Wan, Q., Hu, S., Zhang, Y., Wang, P., Wen, B., and Lu, Z. (2023). "It Felt Like Having a Second Mind": Investigating Human-AI Co-creativity in Prewriting with Large Language Models. arXiv:2307.10811 [cs].

[54] Wang, A. Y., Head, A., Zhang, A. G., Oney, S., and Brooks, C. (2023). Colaroid: A Literate Programming Approach for Authoring Explorable Multi-Stage Tutorials. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, pages 1–22, New York, NY, USA. Association for Computing Machinery.

[55] Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. (2022). Emergent Abilities of Large Language Models. arXiv:2206.07682 [cs].

[56] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. (2023). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903 [cs].

[57] Weisz, J. D., Muller, M., He, J., and Houde, S. (2023). Toward General Design Principles for Generative AI Applications. arXiv:2301.05578 [cs].

[58] Weisz, J. D., Muller, M., Houde, S., Richards, J., Ross, S. I., Martinez, F., Agarwal, M., and Talamadupula, K. (2021). Perfection Not Required? Human-AI Partnerships in Code Translation. In *26th International Conference on Intelligent User Interfaces*, IUI '21, pages 402–412, New York, NY, USA. Association for Computing Machinery.

[59] Yang, Q., Steinfeld, A., Rosé, C., and Zimmerman, J. (2020). Re-examining Whether, Why, and How Human-AI Interaction Is Uniquely Difficult to Design. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, pages 1–13, New York, NY, USA. Association for Computing Machinery.

[60] Yildirim, N., Pushkarna, M., Goyal, N., Wattenberg, M., and Viégas, F. (2023). Investigating How Practitioners Use Human-AI Guidelines: A Case Study on the People + AI Guidebook. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, pages 1–13, New York, NY, USA. Association for Computing Machinery.

[61] Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.-Y., and Wen, J.-R. (2023). A Survey of Large Language Models. arXiv:2303.18223 [cs].

[62] Zhong, M., Li, G., Chi, P., and Li, Y. (2021). HelpViz: Automatic Generation of Contextual Visual Mobile Tutorials from Text-Based Instructions. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, UIST '21, pages 1144–1153, New York, NY, USA. Association for Computing Machinery.

Acronyms

LLM	Large Language Model
SIL	Stages of Interaction with LLMs
AWS	Amazon Web Services
GPT	Generative Pre-trained Transformer
BERT	Bidirectional Encoder Representations from Transformers

Appendix

8.1 Recruitment Texts

These advertisements were placed on Reddit and LinkedIn forums.

I'm a part of a research team from McGill University and Université de Montréal, recruiting participants for a study to understand how to design AI-driven tools to support tutorial authoring. We are interested in hearing about your experience with writing tutorials for software. Previous uses of existing AI driven tools such as GitHub Copilot/Codex, or similar tools are preferred but not required. The study will take around 60 minutes and will be a remote study over MS Teams. You will be compensated $20 CAD in the form of a gift card as a token of appreciation for your time. If you are interested in participating, please answer this short screener. Thank you!

8.2 Sample Demographic Survey & Screening

These questions were asked via a survey hosted on Microsoft Forms and sent to potential participants to determine if they meet the selection criteria outlined above.

* Required

1. Please enter your full name.* _____
2. Please enter your email address.* _____
3. How long have you been involved in professional software development?*
    - 1-3 years
    - < 5 years
    - 5-10 years

- 10-15 years
- > 15 years

4. Which programming language do you primarily use?*
   - Python
   - Java
   - JavaScript
   - C/C++
   - Other: _____
5. Please enter your current occupation.*
   - Software Developer
   - Technical Writer
   - UI/UX Designer
   - Program/Product Manager
   - Student
   - Other: _____
6. How often do you write/maintain software tutorials in the past three years? For the purpose of this study, we define a "tutorial" as a file/document that is used to provide information about a software or its features written for beginner, intermediate or advanced users and contains both descriptions in a natural language and code examples. A tutorial must contain more information than just an API specification. Other names for a tutorial can be "how-to" or "walkthrough".*
   - 2-3 times a week
   - Once every week
   - Once a month
   - Once in several months
   - I've never written a tutorial before
7. Can you please include a link to a tutorial that you have written and is publicly

available? _____

8. How many software tutorials have you written so far (an estimate)?* _____
9. How comfortable are you with English? You can assess your skill level based on the ILR Scale (https://en.wikipedia.org/wiki/ILR_scale).*
    - Native/Bilingual Proficiency
    - Full Professional Proficiency
    - Professional Working Proficiency
    - Limited Working Proficiency
    - Elementary Proficiency
10. Have you previously used any of the following AI-driven tools (or other similar ones) for any software engineering-related activities?*
    - VS Code IntelliSense (https://code.visualstudio.com/docs/editor/intellisense)
    - Codex (https://openai.com/blog/openai-codex/)
    - Docly by Codist (https://codist-ai.com/)
    - GPT-3 based tools (https://openai.com/blog/gpt-3-apps/)
    - Other: _____
    - I have not used such tools previously
11. Please provide the time zone you're in (For example, EST, PST, CDT).* _____
12. Please indicate a day between 20th August to 30th August when you'll be available for the study.* _____
13. Please indicate another day between 20th August to 30th August when you'll be available for the study.* _____
14. Do you have any suggestions/comments to add? _____