

LLM-for-X: Application-agnostic Integration of Large Language Models to Support Personal Writing Workflows

Lukas Teufelberger, Xintong Liu, Zhipeng Li, Max Moebus, and Christian Holz

Department of Computer Science

ETH Zürich, Switzerland

firstname.lastname@inf.ethz.ch

ABSTRACT

To enhance productivity and to streamline workflows, there is a growing trend to embed large language model (LLM) functionality into applications, from browser-based web apps to native apps that run on personal computers. Here, we introduce *LLM-for-X*, a system-wide shortcut layer that seamlessly augments any application with LLM services through a lightweight popup dialog. Our native layer seamlessly connects front-end applications to popular LLM backends, such as ChatGPT and Gemini, using their uniform chat front-ends as the programming interface or their custom API calls. We demonstrate the benefits of *LLM-for-X* across a wide variety of applications, including Microsoft Office, VSCode, and Adobe Acrobat as well as popular web apps such as Overleaf. In our evaluation, we compared *LLM-for-X* with ChatGPT’s web interface in a series of tasks, showing that our approach can provide users with quick, efficient, and easy-to-use LLM assistance without context switching to support writing and reading tasks that is agnostic of the specific application.

1 INTRODUCTION

Large language models (LLMs) are becoming a common part of authoring processes, such as writing [23, 37, 53, 65], editing [50], and question answering [36, 66]. Users across different fields now employ LLMs to enhance productivity [2, 11] and creativity [19]. LLM-based co-authoring and co-editing thus mark a shift in how people approach content creation [8, 46, 54] and how they perceive individual agency over the process [7, 25, 33].

The dominant mode of interaction with LLMs is chat interfaces in the form of question & answer dialogs. Chat assistants offered by the major providers of LLM backend services, such as OpenAI ChatGPT [47], Google Gemini [18], or Anthropic Claude [5] mimic human conversations to make interactions with LLMs natural. Via chat queries, users can retrieve information, conduct research, and solve their problems [22, 42]. These capabilities have made chat-based LLMs a promising alternative to conventional search interfaces [51] to support users throughout task executions.

Beyond question & answer interaction, users frequently copy & paste content into and from these chat interfaces to effectively execute a variety of tasks inside *other* applications. This workflow uses the clipboard as the key interface to transfer textual information back and forth across tabs, windows, and applications. The latest generation of LLMs also supports receiving images as input [64] and generating images as output [9], which augments the chat interface with a rich modality for interaction.

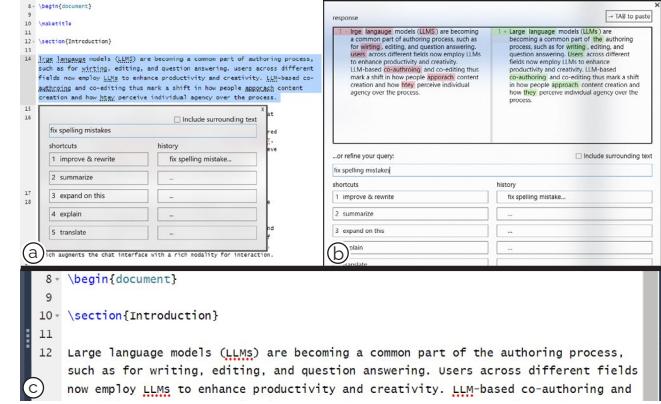


Figure 1: (a) *LLM-for-X* allows users to select text inside native and web apps and (b) execute predefined LLM commands or enter a custom query to (c) directly insert the response into the app—without the need for context switching or invoking copy & paste to transfer content between apps.

Video figure: [youtube.com/watch?v=fDDMaWobjVY](https://www.youtube.com/watch?v=fDDMaWobjVY)

Following the success, many developers have incorporated LLM services into their products, from smaller apps (e.g., Grammarly [27], personal AI Pi by Inflection [32], My AI by Snapchat [55]) to large software suites (e.g., Copilot in Microsoft Office [43], Google Workspace [26], Amazon Lex [4]), often charging (separately) for this added functionality. As a consequence, users must join multiple subscriptions, one for each app to enjoy LLM-based services—despite the large overlap of their capabilities and in the face of free general purpose use [49]. For this reason, copy & paste may remain the dominant interface for users to bridge LLM assistance and the apps they use to complete tasks in a desktop environment.

We introduce *LLM-for-X*, a user interface technique that provides interaction with LLM backends to users within any frontend app. As shown in Figure 1, *LLM-for-X* requires no copy & paste or switching windows to LLM frontends, and it allows users to almost directly operate on text inside an app using a light-weight on-demand popup UI. Our system service connects text selections and user queries to an LLM by either emulating user input to a chat interface (e.g., an existing subscription for ChatGPT) and seamlessly transfers responses back into the app. Alternatively, *LLM-for-X* connects to LLM backend APIs to retrieve responses. In both cases, our approach minimizes the effort to interface with LLMs, allowing users to focus on a task within an app without context switching.

1.1 Walk-through and interaction design

LLM-for-X is designed for efficient keyboard use, supporting the writing and editing process of text across apps with shortcuts. Figure 1 shows a typical app scenario. The user selects text inside an app, here Overleaf running in the web browser, and triggers LLM-for-X via keyboard shortcut ($\text{Alt} + 1$). A prompt menu appears with several options for quick actions and associated prompts that can be triggered using the number keys. Alternatively, the user may enter a custom LLM query. Here, the user types “fix spelling mistakes”, which sends the prompt and the text selection to an LLM chat in the background. LLM-for-X then previews the LLM response in the menu, which the user inserts into Overleaf by pressing TAB. This closes the menu and completes the interaction flow.

Below, we describe more interaction scenarios. We also refer the reader to our video figure for additional demonstrations: <https://youtube.com/watch?v=fDDMaWobjVY>.

Iterating on LLM responses. Before inserting the LLM response, LLM-for-X allows users to refine the prompt to iterate on the generated responses (Figure 2a). Here, the dialog shows the original text and the LLM response side-by-side, highlighting changes using a diff view. Users may alter the prompt, resubmit the query on the selected text, and see the updated LLM response in the preview area. Our tool thereby maintains the chat session in the background, such that query refinements preserve previous output in the LLM chat interface for context.

Insert vs. replace text in the app. While text replacement may be the most common action, LLM-for-X additionally supports appending the LLM response below the text selection when the user holds a modifier key SHIFT before pressing TAB for insertion (Figure 2b). This functionality can be useful when the purpose of the selected text is to act as specific context to inform the completion of the task. Examples include coding tasks where instructions or descriptions such as those in comments above a function signature should be left intact.

Additional context to the LLM. Our tool is able to, if selected by the user, provide additional context to the LLM query by including the textual content surrounding the selection. The additional context is flagged as context in LLM-for-X’s prompt to the LLM’s chat interface to ensure that the response does not directly include part of this augmentation.

In any case, LLM-for-X extracts the current app’s name and the window title as context for prompt customization. We also use the window title to decide between resuming a previous LLM chat session or starting a new session.

Direct in-place response. When the user holds the modifier SHIFT while submitting a query, the prompt menu will close immediately and our system service will insert the LLM response directly in the foreground app by simulating key presses—letter by letter and in real-time as produced by the chat interface (Figure 2c). Because this insertion appears to the foreground app as entered by the user, the LLM response can be handled as such and, for example, be undone using the app’s regular ‘undo’ function.

Querying information without inserting responses. Finally, LLM-for-X can also be used to simply obtain additional information. Selecting

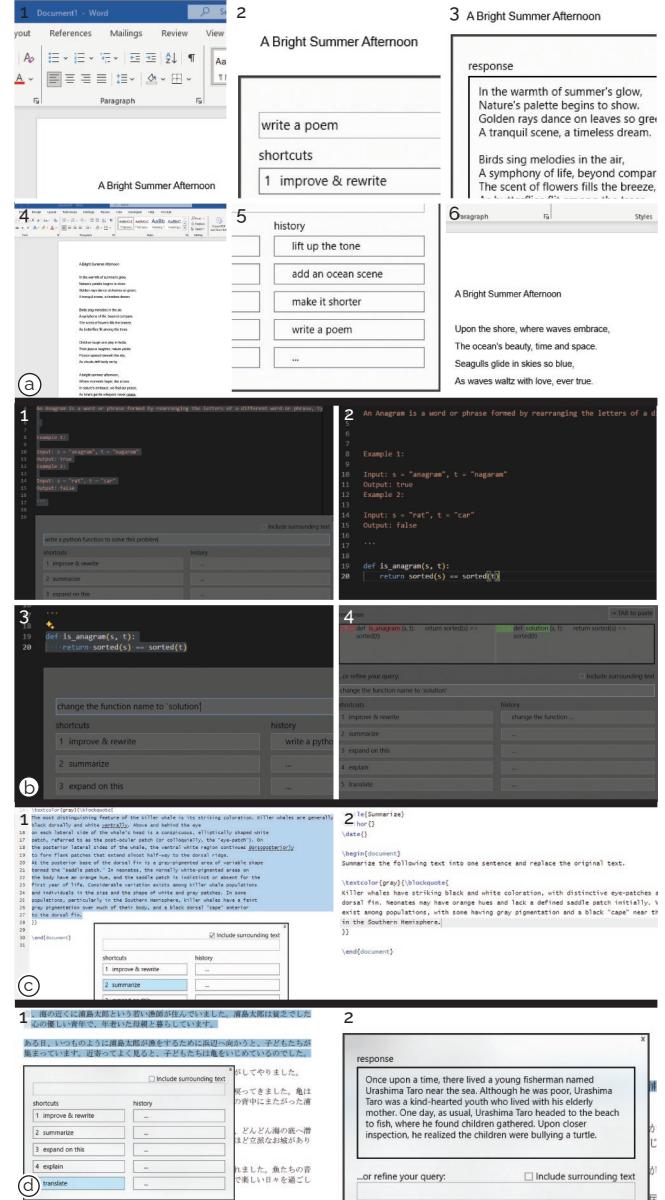


Figure 2: LLM-for-X walk-through. (a) Iterating on LLM responses, (b) pasting responses as ‘insert below’ vs. ‘replacing’ with diff view, (c) direct in-place pasting without preview, and (d) selecting and querying for information retrieval.

the relevant text, pressing $\text{Alt} + 1$, and hitting 4 (for ‘explain’) or 5 (for ‘translate’) will produce the LLM’s chat response in the menu’s preview area as before. This functionality can be useful for contextual information, such as in a reading app or PDF viewer when users may require context (Figure 2d). Another example is the use of the tool for simultaneous translation (Figure 2d) and explanation on foreign language (news) websites.

Since the app in focus may not support text entry, such as in the case of the PDF viewer, LLM-for-X recognizes this and hides the TAB button. Users may still refine the query for further details.

1.2 Design objectives

We primarily designed LLM-for-X to allow for quick interaction with LLM backends without the need for context (or app) switching as shown in Figure 3. While our interaction design is optimized for keyboard use (e.g., shortcuts to summon via $Alt + 1$, to select menu items via $1..5$, TAB for pasting, and modifier keys), all options are equally accessible via mouse use, too. Inside native apps, LLM-for-X’s prompt menu can be invoked using an icon in the system tray, whereas a toolbar icon triggers it in the browser. The menu stays open and in front until the user presses ESC or clicks the button in the top right. Upon pressing TAB, LLM-for-X inserts or replaces the selection at the location of the cursor. Our menu will remain active even if users click a different position in the text editor, such that they can determine where to paste the LLM response before pressing TAB or ESC to close the menu.

The visual design of LLM-for-X’s popup is lightweight to minimize distraction from the foreground app. LLM-for-X’s UI has three main elements: query input field (which is in focus when the menu appears), suggested actions (chosen based on popular use-cases of ChatGPT [60], accessible via shortcuts), and the preview output field for LLM responses, where changes are color-highlighted. The latter can be navigated with Page Up/Page Down or the scrollbar, while typing will modify the query prompt.

1.3 Contributions

Our research makes two main contributions:

- LLM-for-X, an operating system-wide shortcut interaction technique that allows prompting LLM backends from within any native app or web app, optionally based on a text selection for manipulation. Our technique is implemented through an OS service and can alternatively interface with a browser extension to facilitate LLM communication or an LLM API backend directly, either offered by popular LLM providers such as OpenAI ChatGPT and Google Gemini. Our technique thus bridges their text-based functionality with text input inside *any* app.
- a user study with 14 participants to compare their performance during writing, editing, and coding tasks using LLM-for-X and ChatGPT. Participants were significantly faster in editing text and reported higher usability (using SUS [10]), as LLM-for-X allowed them to stay within one app without context switching.

2 RELATED WORK

LLM-for-X is related to virtual assistants, LLM-based support tools, and shortcut interaction.

2.1 Virtual Assistants

A virtual assistant (VA) is a software agent that carries out a variety of tasks or services in response to user input, such as commands, questions, and verbal prompts [62]. Gero et al. categorize the research of intelligent writing assistants into task, user, technology, interaction, and ecosystems [23]. LLM-for-X falls under the interaction aspect, specifically emphasizing the hybrid interaction metaphor, localized user interface, and explicit user control.

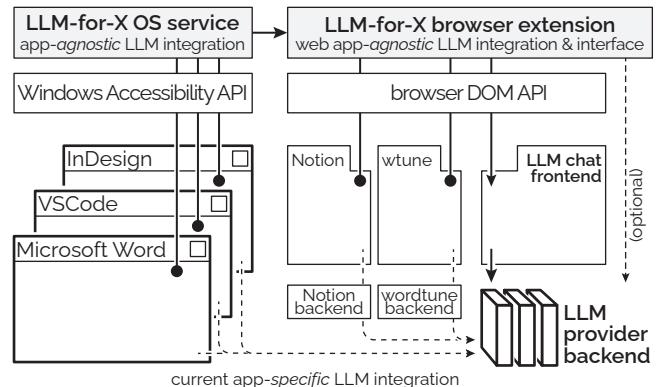


Figure 3: LLM-for-X offers a system-wide shortcut from any web or native application to various text-based LLM backend.

Intelligent Personal Assistants. With the integration of machine learning and advanced speech recognition technologies into personal devices, the prevalence of intelligent personal assistants (IPAs) highlights the effectiveness of AI-powered personal assistants in everyday tasks [17]. These software agents are present in wearables, smartphones, and desktops [17], such as Siri [6], Alexa [3], and Cortana [44]. However, most users rely on IPAs for only basic tasks, such as playing music and scheduling appointments [15, 21].

IPAs in the Age of LLMs. Researchers envision the next generation of IPAs to be powered by LLMs and to become more prevalent as omnipresent personal companions [20]. Early prototypes for LLM-based IPAs on smartphones include GPTVoiceTasker [59] and GPT-Droid [40]. Windows Copilot [43] is among the first commercially-available personal companions for desktop environments.

2.2 LLM-based Tools

With the rapid expansion of open-source NLP research, there are many studies on specialized bots and apps for specific use cases such as domain-specific or task-orientated text generation [48, 56]. Popular uses for LLMs include text generation, information retrieval, and problem solving [56, 60].

Writing. Co-writing systems such as Wordcraft [65] show that users make use of AI assistants throughout many stages in the writing process [54] and tools such as TexGPT from Writeful [63] even offer platforms-specific AI assistants, e.g., for Overleaf. More advanced writing assistants focus on providing new perspectives for reflection and inspiration in addition to grammar correction and text polishing [16, 19].

Information Retrieval. Learning and browsing online could also benefit from assistance. Complex search tasks require similarly complex search prompts [30]. The success of complex search tasks is then heavily dependent on the quality of the search prompt [61]. GPT models also possess suitable translation capabilities [31] and can be useful in educational settings [28, 35].

Coding. AI assistants are also popular for aiding software engineering tasks as so-called copilots or companions [52]. The advancements of LLMs fine-tuned on publicly available code facilitate more

efficient coding with AI assistants (e.g., Codex [12]). Previous studies showed that while Github Copilot and ChatGPT-assisted coding does not outperform human-written solutions, users acknowledge their ability to handle fundamental tasks [1, 58].

2.3 Efficient interaction and shortcuts

When trying to harness the full power of LLMs and AI assistants, the bottleneck during interaction with generative AI models is speed and output quality [45, 57]. Since many LLM commands are invoked through menus, keyboards can provide preferable and faster access over mouse use [34] via shortcuts.

Further, using AI assistants to everyday tasks should disrupt the underlying tasks as little as possible [67]. Across a broad range of professions including primary care [13] and software engineering [38], such interruptions otherwise significantly impair productivity and quality of work, even those as small as context switching. For software engineers, context switching strongly affects perceived productivity and mood [41]. AI assistants should thus be seamlessly integrated into everyday tasks to minimize interruptions and context switches.

3 LLM-FOR-X IMPLEMENTATION

Figure 3 shows an overview of the architecture of our implementation, including LLM-for-X’s operating system (OS)-level background service, which produces its prompt menu UI, its browser extension, and its optional direct API use. LLM-for-X’s prompt menu is the main interactive and user-facing element, configured by our background service that monitors for short-cut activation. This service also interfaces with the content inside native apps and retrieves surrounding content for context. For interfacing with web apps, our system includes a browser extension for deep access to web pages. In both cases, our background service retrieves selected text and surrounding content and replaces or inserts selections with the responses from the LLM. Our background service also interfaces with the web-based frontend of LLM services via our browser extension and operates them by simulating a user’s copy&paste interaction or alternatively directly queries backends via their APIs.

We implemented LLM-for-X to run on Windows 10 and up. It supports Google Chrome or Microsoft Edge through its extension API. If support for native apps is not needed, LLM-for-X can also operate solely based on its browser extension (including an in-browser short-cut listener and a browser-based menu). This makes it compatible with all platforms that fully support Chrome or Edge and extensions without requiring a native component.

3.1 OS-level background service

LLM-for-X runs as a background app to listen for triggers of the prompt menu across applications and web browser tabs. We implemented the service in C# using .net APIs, registering a system-level keyboard hook to watch for global keyboard shortcuts. Once our service detects a shortcut, our native app interface extracts application details and text selection to prepare the prompt.

3.2 Native app interface (via accessibility API)

Access to text contents within an app is available via UI Automation API (UIA) on Windows, which is part of the Microsoft Accessibility

API. Our service has a registry key in Windows set to be able to communicate with the browser. From the currently focused element, we obtain the window title and Process ID (PID), the latter we use to capture the current app’s name. Both window title and app name we use for context.

To extract selected text, we start with the focused element and browse the subtree of elements for a UI element with a UIA’s `TextPattern`. Once the property is available, we discover the content for the prompt body via UIA’s `GetSelection`.

LLM-for-X can also obtain surrounding text for additional context as mentioned above. We reuse the identifier of the UI element with the `TextPattern` and obtain the entire body of the text. While we use the selected text as main input into the prompt, the rest we add to the prompt for context explicitly declared as such.

Paste LLM responses. Finally, to insert the responses from the chat interface, we emulate direct user input into the control. LLM-for-X’s backend first copies the response to the clipboard and then invokes ‘Paste’ with `SendKeys.SendWait` to simulate the paste operation. The benefit of proceeding this way is that input gets entered into the app’s undo/redo stack. In case the text is not to be replaced but inserted below, we simulate a ‘Cursor Right’ key press beforehand.

3.2.1 Fallback: if the API fails. While the UI Automation API supports most apps on Windows, we have found some not to reveal text content or selections. In case LLM-for-X discovers no existence of `TextPattern`, our service makes use of the clipboard and injects a copy shortcut command (CTRL + C).

Likewise, to retrieve the surrounding context if so configured, we simulate ‘Select All’ (CTRL + A) and another copy shortcut. Just before, we simulate pressing ‘Cursor Right’ and ‘Space’ to enter an operation to the undo stack. After selecting all and copying the text, we undo the last command (space) to set the cursor position to the bottom of the selection. We cannot recover the text selection state, however.

3.3 Web app interface (via browser extension)

If a web browser is in the foreground, LLM-for-X’s browser extension connects to our backend service upon detecting the trigger shortcut. The extension then interfaces with the foreground web page to extract the selected text (if any), textual content, and later handle text replacement or insertion.

Through Javascript and the DOM API, LLM-for-X’s browser extension obtains the selected element in the DOM and extracts the selected text inside. Our extension then finds the first common ancestor and takes its `parentNode` to extract the selection’s context.

Insert LLM responses. LLM-for-X’s extension again uses the current selection and appends `textNodes` to the end. This reliably works for both editable UI elements (e.g., `textarea` and `input`) but also any other elements (e.g., `div` with `contenteditable` set).

If responses cannot be pasted. Finally, our extension determines whether the element that contains the selected text is not editable, such that an LLM response could not be pasted. In this case, LLM-for-X’s prompt menu does not show the TAB button. We conclude this from the element’s tag name (`input` or `textarea`) and from whether it is set to read-only, disabled, or if it is `contenteditable`.

3.4 Native service-to-extension interface

For communication between our browser extension and native service, we send commands via the Native Messaging API [14]. Our browser extension thereby starts the native app and uses the standard input/output interface for message exchange.

3.5 LLM interface (via browser extension)

LLM-for-X currently supports ChatGPT, Mistral, and Gemini. Because they all follow the same interaction paradigm using a chat interface, our browser extension emulates user input when a query is submitted from the prompt menu, extracts the response from the LLM web UI, and transfers it back to the prompt menu.

3.5.1 Communicating with LLM chat interface pages. Upon opening an LLM provider's chat interface, our extension detects the chat window, the query input field, and the response element. The extension searches for the input element using a combination of tag names and element paths. It then inserts the query by either setting the `textContent` or `value` of the element with dispatching a few events first to simulate user interaction. The query is submitted by injecting a 'return' or 'click' event.

Afterwards, our extension repeatedly polls the DOM in short intervals to detect changes in the output element. We obtain the text from the `innerText` property of the output element. Whenever a change in content is detected, our extension forwards the complete content to our OS-level service and into the prompt menu's preview field. Once no changes have been detected for a predefined duration, it will signal to our background service that the response is complete. We opted for this procedure, because chat interfaces perform various operations on the returned raw response throughout, such as replacing symbols.

LLM chat interfaces and changes. LLM-for-X has default tag names and element paths configured for the currently supported chat interfaces of ChatGPT, Gemini, and Mistral. In the case that a chat interface changes, our extension first uses heuristics to rediscover the elements (looking for the `tagnames`, `id`, and an overlap in `classes` added, or the same path in the DOM tree). If this fails, our extension reports a problem and provide a tool to select the input box and the output element manually (similar to Chrome's 'Select an element to inspect' function) to update our settings. Because all parameters and URLs are stored in the extension settings, users could also add further LLM chat interfaces in the future.

Browser handling to optimize for responsiveness. When a user triggers LLM-for-X's prompt menu, our browser extension opens the LLM's web page in a browser background window. If the user has interacted with the current foreground app with this title before using LLM-for-X, our browser extension loads the corresponding previous chat, such that the user can continue prompting within the context of the previous interaction sequence. This accounts for loading times while the user still sees and operates LLM-for-X's prompt menu, ensuring that upon submitting a query our tool does not add to the response latency.

3.5.2 Prompt padding. For submitting the prompt entered in LLM-for-X's prompt menu (or that associated with a menu item), we augment the text selection as follows:

```

1 Your task is to answer the following query from the user. Do not express
2 approval or your own judgment of the query. Just respond with a clear
3 answer. If prompted for code, just output the code, no explanation, just
4 one response of code and nothing else.
5
6 User query: %USER_QUERY%
7
8 The user's query refers to this specific text:
9 %SELECTED_TEXT%
10
11 The user issued the query while working with %APP_NAME% and the document
12 %WINDOW_TITLE%.
13
14 The user has provided additional context for their query. Do not directly
15 quote this context, but use it to formulate a response.
16
17 Context: %TEXT_CONTEXT%

```

3.6 Direct LLM interface via API (optional)

As an alternative to browser-based LLM access, LLM-for-X also supports direct interaction with LLM backends via their APIs. While this incurs query-specific charges for each use of LLM-for-X, it affords more freedom over query types while still bringing LLM support to many apps. Responses are also faster than those from the browser-based interface without variable-speed delays.

4 USER STUDY

The purpose of our study was to evaluate the efficacy of LLM-for-X in the context of authoring, reading, and coding tasks. In a controlled user study, we compared participants' performance solving these tasks using one of two *Interfaces*, LLM-FOR-X and CHATGPT (version 3.5 through the web interface).

4.1 Tasks

Participants completed three tasks during the study. Each task had two instantiations that were similar in difficulty and format but varied in specific content.

4.1.1 Writing. The writing task comprised three common subtasks for intelligent writing assistants: summarizing, editing, and text composing [16, 19, 24]. For summarizing, participants summarized a paragraph from an academic paper draft in Overleaf. For editing, they rewrote a narrative paragraph in Microsoft Word in a different style. For composing, they wrote an email in Microsoft Outlook given a use-case: an invitation to an after-work celebration and an out-of-office announcement for a personal vacation.

4.1.2 Reading. For the purpose of answering a question, participants read a one page long folk story in a foreign language that they were not familiar with in Acrobat Reader. Their task was to answer a question about the text in English.

4.1.3 Coding. Participants completed a simple coding task using Python in VSCode similar to previous studies [1, 58]. The task involved extracting information from a CSV file, aggregating dataframes, and writing to a text file as output.

4.2 Procedure

The study started with a general introduction to the study context and explained the tasks to complete. Participants received an introduction to and demonstration of CHATGPT and LLM-FOR-X, and they were allowed to train in both for up to 20 minutes, using any native or web apps they wanted.

Participants then moved on to the main part of the study, for which they received written instructions. Participants completed all three tasks using the same condition, took a break, and repeated the tasks in the second instantiation with the other condition. The conditions LLM-FOR-X and CHATGPT were counterbalanced across participants. During each (sub)task, a timer on the screen displayed the remaining time to participants.

After completing all tasks using one interface, participants filled out a questionnaire that included SUS [10] and NASA TLX [29]. When participants had finished both conditions, they had the option to provide qualitative feedback about both conditions.

The study ran on a Windows 10 computer with a full keyboard, mouse, and a 27" display. An experimenter monitored participants' performance, answered questions, and ensured correct task outcomes. All participants completed the study in less than an hour.

4.3 Participants

We recruited 14 participants from various departments within our institution (5 female, 9 male, ages 22–37 years). All participants had prior experience with Python. More than half of them reported that they use ChatGPT and other LLM-based tools (e.g., Copilot) multiple times a day. Coding is the most popular reported task for LLM use, and 12 participants that they routinely use ChatGPT for coding. 9 mentioned using it for writing, editing, and conducting research. 7 reported using ChatGPT for drafting emails and messages. Participants received a small gratuity for their time after the study.

4.4 Measurements

We recorded the following metrics to assess participants' performance and impressions during the study: (sub)task completion time, time-stamped logs during task completion, Likert-scale questionnaire responses, and qualitative comments.

4.4.1 Completion time. The timer started once participants had completed reading the instructions of each task and had asked questions if they had any. The timer stopped only when the experimenter had verified the correct completion of the task. For writing, the text needed to be complete with no placeholders remaining (as CHATGPT often includes them). For reading and question answering, the answer needed to be correct, otherwise participants could try again. For coding, their code needed to produce the correct output, for which we provided a verification script that they ran on the command line. We measure all completion times in seconds.

4.4.2 System Usability Scale. We assessed the usability of LLM-FOR-X and CHATGPT for daily desktop-based tasks through the system usability scale (SUS) [10]. In particular, we scored the outcome of the SUS using the following standard formula [39]: $2.5 * (20 + \sum_{i=1,3,5,7,9} SUS_i - \sum_{i=2,4,6,8,10} SUS_i)$, where SUS_i denotes the i^{th} question in the SUS for $i \in 1, \dots, 10$.

4.4.3 Task Load Index. We used the NASA task load index (NASA TLX) questionnaire [29] to assess participants' perceived workload while using LLM interfaces. It consists on six sub-scales on mental demand, physical demand, temporal demand, performance, effort, and frustration level. To increase the convenience for study participants, we adapted each sub-scale to a 5-point Likert item ranging

from 'very low' to 'very high'. We then calculated the load index via the following formula, which mimics the standard aggregation of the NASA TLX [29]: $TLX = \frac{25}{6} \sum_{i=1, \dots, 6} (TLX_i - 1)$, where TLX_i is the i^{th} sub-scale of our adapted NASA TLX questionnaire.

4.4.4 Questionnaire. In addition to SUS and NASA TLX, participants provided their experience in using LLM-based support for the tasks (following Adamson and Bägerfeldt's study [1]). A 5-point Likert scale assessed the condition's perceived impact on their efficiency and productivity, how often they needed to edit the output content to meet task requirements, and the follow-up work needed to reformat the output to produce the final result.

4.4.5 Feedback. After the study, participants could report any feedback they had on either interface. Participants commented on the task design, UI design, and the experience of solving the tasks using LLM-FOR-X and CHATGPT. After their initial comments, we followed up with questions as reported in previous work [1].

4.5 Results

4.5.1 Completion time. Figure 4 shows all completion times. We could not find a statistically significant difference of total completion time across all tasks (LLM-FOR-X: $M = 382.21$, $SD = 127.09$; CHATGPT: $M = 436.07$, $SD = 167.19$; $Z = 80.5$, $p = 0.43$). We did find a main effect of *Interface* on *Completion Time* during the editing task ($Z = -2.18$, $p < 0.05$), where participants completed the task significantly faster using LLM-FOR-X ($M = 31.71$, $SD = 19.04$) than when using CHATGPT ($M = 51.14$, $SD = 32.50$). We did not observe any other significant differences.

4.5.2 System Usability Scale. We found a significant effect of *Interface* on *System Usability* across all tasks ($Z = 2.64$, $p < 0.01$). Post-hoc tests showed that participants rated LLM-FOR-X ($M = 62.54$, $SD = 9.2$) significantly higher than CHATGPT ($M = 51.68$, $SD = 10.94$) (Figure 5).

4.5.3 Task Load Index. We did not find a statistically significant overall difference for NASA TLX ($Z = -1.03$, $p = 0.30$). Participants on average scored CHATGPT with 22.32 ($SD = 14.58$) and LLM-FOR-X with 17.26 ($SD = 11.77$). However, participants rated LLM-FOR-X significantly lower on the effort-related sub-scale of the NASA TLX for ease of use ($M = 1.64$, $SD = 0.49$) compared to CHATGPT ($M = 2.27$, $SD = 1.03$). We found no other significant difference between sub-scales of the NASA TLX.

4.5.4 Questionnaire. We could not find a statistically significant difference in the questionnaire answers. Participants' reported answers for efficiency were LLM-FOR-X: $M = 4.64$ ($SD = 0.63$) vs. CHATGPT: $M = 4.50$ ($SD = 0.65$). For perceived quality of their work, they rated LLM-FOR-X with $M = 4.43$ ($SD = 0.65$) and CHATGPT with $M = 4.21$ ($SD = 0.89$). On perceived difficulty, they rated LLM-FOR-X with $M = 1.71$ ($SD = 0.61$) compared to CHATGPT ($M = 1.57$, $SD = 0.65$). For number of needed modifications they needed to perform, they rated LLM-FOR-X with $M = 2.43$ ($SD = 0.51$) and CHATGPT with $M = 3.07$ ($SD = 1.21$) and for "less follow-up work was required" they scored LLM-FOR-X: $M = 2.21$ ($SD = 0.80$) vs. CHATGPT: $M = 2.64$ ($SD = 1.28$).

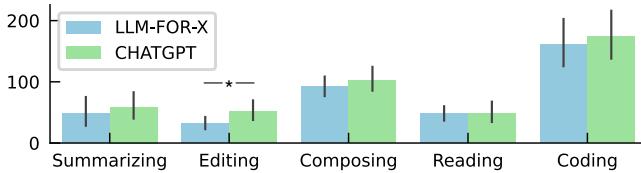


Figure 4: Effect of Interface on task completion time [sec].

4.5.5 Feedback. Participants expressed that they felt more efficient when conducting the tasks with LLM-FOR-X, because it eliminates the need for context switching. P1 said “*Because LLM-FOR-X is more integrated into the environment, I feel less the need to switch my focus compared to when I use CHATGPT.*” The shortcuts for menu initiation and insertion were also appreciated. For example, P7 said “*It is cool that no copy paste is required when using the tool.*”

However, some participants still prefer the CHATGPT interface for user-friendliness. P6 pointed out the personalization of CHATGPT saying “*CHATGPT talks in a more friendly manner and feels more like a personal assistant.*” P2 found the code generated by the CHATGPT web app easier to read.

5 DISCUSSION

Overall, the results of our evaluation were encouraging for our approach. While LLM-FOR-X achieved a significant difference in performance only for the editing task, interestingly with a 40% faster completion time on average, participants’ ratings on the SUS and for ease-of-use on TLX confirmed our original motivation. It is worth pointing out that all participants were at least well familiar with CHATGPT and that the majority of them used it frequently, such that we can assume that they were trained to the point of routine operation. Therefore, CHATGPT was a challenging baseline despite the training participants received for LLM-FOR-X.

One observation during the study was that participants’ use of CHATGPT varied based on personal preference. Some heavily relied on it, copying & pasting in large amounts of contents, while others use them more deliberately and sparingly. For example, when replacing the placeholders in the composed emails, more than half of the participants insisted and requested in a prompt that this be done by the LLM, while the minority just filled them out manually.

During or after the study, no participant commented on the choice of shortcuts or other design choices of LLM-FOR-X, except for the 2 who commented on the lack of a ‘personalized assistant experience’. Taken together, we interpret this as an encouraging sign that our design objectives for the UI and the interaction did not interfere with participants’ workflows. On the contrary, participants offered several recommendations how the UI could be extended to better support the tasks at hand. A frequent suggestion was that the menu buttons (1..5) should be generated based on the context of the selected text and application window.

6 LIMITATIONS AND FUTURE WORK

In-situ editing. Participants took time to get used to LLM-for-X’s UI, as LLM responses flowed directly into the foreground app upon pressing TAB. Despite the preview in our UI, they were used to seeing the formatted output in the context of previous responses

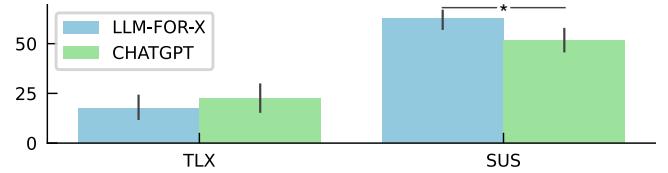


Figure 5: Effect of Interface on SUS and NASA TLX scores.

and using copy & paste to transfer (portions of) the response. Future implementations could include collapsible displays of history for previously generated responses, syntax highlighting, or the option to insert only portions of the previewed response.

LLM-for-X vs. apps with built-in LLM support. While our approach provides general-purpose text generation and editing across apps, those with built-in LLM assistance can offer better tailored support. For example, Github Copilot’s Codex is specifically integrated into programming environments (e.g., VSCode, JetBrains). Similarly, assistants in creativity apps can generate and manipulate app-specific objects (e.g., Adobe Illustrator or Premiere), where mere text responses are insufficient.

Context is limited to text. Our current implementation is limited to sourcing text from the surrounding context. Future work could additionally include screenshots to GPT-4, such as to capture a screenshot of the rendered draft when editing in Overleaf or capturing surrounding images in InDesign for suggestions.

Integration with other web-based query services. LLM-for-X’s focus is on LLM-based chat interactions, but our implementation can also interface with other web-based services, such as Google Translate and DeepL (see video), as query elements and response fields are equally well defined on the web page. Future implementations could explore more general media types (e.g., Wolfram Alpha).

7 CONCLUSION

We have presented LLM-for-X, a system-wide bridge between applications and LLM-based assistance for textual content. By pressing a shortcut, our system shows a lightweight overlay to query LLM chat systems with a text selection and allows the response to seamlessly transfer back into the app. In the background, our system interfaces with browser-based LLM chat interfaces to submit the query and process the response or it can query LLM backends directly through their APIs. LLM-for-X thus allows users to leverage LLM capabilities for text manipulation across virtually all apps, even if an app does not build in native LLM support.

In our comparison of LLM-for-X with ChatGPT, we found statistically significant differences only in its ability to support text editing more quickly as part of a rewrite task and its perceived usability (rated on SUS [10]). Given these results and the qualitative insights during the study as the basis for future developments, we believe that LLM-for-X is a promising option for supporting text manipulation tasks and supporting quality of work across applications, without investing in app-specific LLM subscriptions.

REFERENCES

- [1] Victor Adamson and Johan Bägerfeldt. 2023. Assessing the effectiveness of ChatGPT in generating Python code.
- [2] Valentina Alto. 2023. *Modern Generative AI with ChatGPT and OpenAI Models: Leverage the capabilities of OpenAI's LLM for productivity and innovation with GPT3 and GPT4*.
- [3] Amazon. 2024. What is Alexa. <https://developer.amazon.com/en-US/alexa> Accessed: 30-03-2024.
- [4] Amazon. 2024. what is Amazon Lex. <https://docs.aws.amazon.com/lex/latest/dg/what-is.html> Accessed: 30-03-2024.
- [5] anthropic. 2024. Introducing Claude. <https://www.anthropic.com/news/introducing-claude> Accessed: 30-03-2024.
- [6] Apple. 2024. Siri. <https://www.apple.com/siri/> Accessed: 30-03-2024.
- [7] Kenneth C Arnold, Krysta Chauncey, and Krzysztof Z Gajos. 2020. Predictive text encourages predictable writing. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*. 128–138.
- [8] Tamara Babaian, Barbara Grosz, and Stuart Shieber. 2002. A Writer’s Collaborative Assistant. *International Conference on Intelligent User Interfaces, Proceedings IUI (06 2002)*. <https://doi.org/10.1145/502716.502722>
- [9] James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. 2023. Improving image generation with better captions. *Computer Science*. <https://cdn.openai.com/papers/dall-e-3.pdf> 2, 3 (2023), 8.
- [10] John Brooke. 1995. SUS: A quick and dirty usability scale. *Usability Eval. Ind.* 18 (11 1995).
- [11] Erik Brynjolfsson, Danielle Li, and Lindsey R Raymond. 2023. *Generative AI at work*. Technical Report. National Bureau of Economic Research.
- [12] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* (2021).
- [13] Carey D Chisholm, Amanda M Dornfeld, David R Nelson, and William H Cordell. 2001. Work interrupted: a comparison of workplace interruptions in emergency departments and primary care offices. *Annals of emergency medicine* 38, 2 (2001), 146–151.
- [14] Chrome. 2023. Native messaging. <https://developer.chrome.com/docs/extensions/develop/concepts/native-messaging> Accessed: 2024-04-03.
- [15] Benjamin R Cowan, Nadia Pantidi, David Coyle, Kellie Morrissey, Peter Clarke, Sara Al-Shehri, David Earley, and Natasha Bandeira. 2017. "What can i help you with?" infrequent users' experiences of intelligent personal assistants. In *Proceedings of the 19th international conference on human-computer interaction with mobile devices and services*. 1–12.
- [16] Hai Dang, Karim Benharrak, Florian Lehmann, and Daniel Buschek. 2022. Beyond text generation: Supporting writers with continuous automatic text summaries. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–13.
- [17] Allan da Barcelos Silva, Marcio Miguel Gomes, Cristiano André da Costa, Rodrigo da Rosa Righi, Jorge Luis Victoria Barbosa, Gustavo Pessin, Geert De Doncker, and Gustavo Federizzi. 2020. Intelligent personal assistants: A systematic literature review. *Expert Systems with Applications* 147 (2020), 113193. <https://doi.org/10.1016/j.eswa.2020.113193>
- [18] Google DeepMind. 2024. Welcome to Gemini. <https://deepmind.google/technologies/gemini/#introduction> Accessed: 30-03-2024.
- [19] Giulia Di Fede, Davide Rocchesso, Steven P. Dow, and Salvatore Andolina. 2022. The Idea Machine: LLM-based Expansion, Rewriting, Combination, and Suggestion of Ideas. In *Proceedings of the 14th Conference on Creativity and Cognition (Venice, Italy) (C&C '22)*. 623–627. <https://doi.org/10.1145/3527927.3535197>
- [20] Xin Luna Dong, Seungwhan Moon, Yifan Ethan Xu, Kshitiz Malik, and Zhou Yu. 2023. Towards next-generation intelligent assistants leveraging llm techniques. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5792–5793.
- [21] Mateusz Dubiel, Martin Halvey, and Leif Azzopardi. 2018. A survey investigating usage of virtual personal assistants. *arXiv preprint arXiv:1807.04606* (2018).
- [22] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: Program-aided Language Models. In *Proceedings of the 40th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 202)*. Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 10764–10799. <https://proceedings.mlr.press/v202/gao23f.html>
- [23] Katy Gero, Alex Calderwood, Charlotte Li, and Lydia Chilton. 2022. 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)*. Ting-Hao 'Kenneth' Huang, Vipul Raheja, Dongyeop Kang, John Joon Young Chung, Daniel Gissin, Mina Lee, and Katy Ilonka Gero (Eds.). Association for Computational Linguistics, Dublin, Ireland, 11–24. <https://doi.org/10.18653/v1/2022.in2writing-1.2>
- [24] Frederic Gmeiner and Nur Yildirim. 2023. Dimensions for Designing LLM-based Writing Support. In *In2Writing Workshop at CHI*.
- [25] Frederica Gonçalves, Pedro Campos, Julian Hanna, and Simone Ashby. 2015. You're the Voice: Evaluating user interfaces for encouraging underserved youths to express themselves through creative writing. In *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition*. 63–72.
- [26] Google. 2024. Google Workspace. <https://workspace.google.com/> Accessed: 30-03-2024.
- [27] Grammarly. 2024. Grammarly AI Writing Tools. <https://www.grammarly.com/ai-writing-tools> Accessed: 30-03-2024.
- [28] Simone Grassini. 2023. Shaping the Future of Education: Exploring the Potential and Consequences of AI and ChatGPT in Educational Settings. *Education Sciences* 13, 7 (2023). <https://doi.org/10.3390/educsci13070692>
- [29] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Human Mental Workload*, Peter A. Hancock and Najmeddin Meshkati (Eds.). Advances in Psychology, Vol. 52. North-Holland, 139–183. [https://doi.org/10.1016/S0166-4115\(08\)62386-9](https://doi.org/10.1016/S0166-4115(08)62386-9)
- [30] Ahmed Hassan Awadallah, Ryon W White, Patrick Pantel, Susan T Dumais, and Yi-Min Wang. 2014. Supporting complex search tasks. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*. 829–838.
- [31] Amr Hendy, Mohamed Abdelrehim, Amr Sharaf, Vikas Raunak, Mohamed Gabr, Hitokazu Matsushita, Young Jin Kim, Mohamed Afify, and Hany Hassan Awadalla. 2023. How good are gpt models at machine translation? a comprehensive evaluation. *arXiv preprint arXiv:2302.09210* (2023).
- [32] Inflection. 2024. The new Inflection. <https://inflection.ai/the-new-inflection> Accessed: 30-03-2024.
- [33] Maurice Jakesch, Megan French, Xiao Ma, Jeffrey T Hancock, and Mor Naaman. 2019. AI-mediated communication: How the perception that profile text was written by AI affects trustworthiness. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13.
- [34] John Karat, James E. McDonald, and Matt Anderson. 1986. A comparison of menu selection techniques: touch panel, mouse and keyboard. *International Journal of Man-Machine Studies* 25, 1 (1986), 73–88. [https://doi.org/10.1016/S0020-7373\(86\)80034-7](https://doi.org/10.1016/S0020-7373(86)80034-7)
- [35] Enkelejda Kasneci, Kathrin Sessler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günemann, Eyke Hüllermeier, Stephan Krusche, Gitta Kutyniok, Tilman Michaeli, Claudia Nerdel, Jürgen Pfeffer, Oleksandra Poquet, Michael Sailer, Albrecht Schmidt, Tina Seidel, Matthias Stadler, Jochen Weller, Jochen Kuhn, and Gjergji Kasneci. 2023. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences* 103 (2023), 102274. <https://doi.org/10.1016/j.lindif.2023.102274>
- [36] Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large Language Models are Zero-Shot Reasoners. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 22199–22213. https://proceedings.neurips.cc/paper_files/paper/2022/file/8bb0d291acd4acf06ef12099c16f326-Paper-Conference.pdf
- [37] Mina Lee, Percy Liang, and Qian Yang. 2022. CoAuthor: Designing a Human-AI Collaborative Writing Dataset for Exploring Language Model Capabilities. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (New Orleans, LA, USA) (CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 388, 19 pages. <https://doi.org/10.1145/3491102.3502030>
- [38] Sophie Leroy, Aaron M Schmidt, and Nora Madjar. 2020. Interruptions and task transitions: Understanding their characteristics, processes, and consequences. *Academy of Management Annals* 14, 2 (2020), 661–694.
- [39] James R Lewis. 2018. The system usability scale: past, present, and future. *International Journal of Human-Computer Interaction* 34, 7 (2018), 577–590.
- [40] Zhe Liu, Chunyang Chen, Junjie Wang, Mengzhuo Chen, Boyu Wu, Xing Che, Dandan Wang, and Qing Wang. 2023. Chatting with gpt-3 for zero-shot human-like mobile automated gui testing. *arXiv preprint arXiv:2305.09434* (2023).
- [41] André N Meyer, Laura E Barton, Gail C Murphy, Thomas Zimmermann, and Thomas Fritz. 2017. The work life of developers: Activities, switches and perceived productivity. *IEEE Transactions on Software Engineering* 43, 12 (2017), 1178–1193.
- [42] Grégoire Mialon, Roberto Dessì, María Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Aslı Celikyilmaz, et al. 2023. Augmented language models: a survey. *arXiv preprint arXiv:2302.07842* (2023).
- [43] Microsoft. 2024. Copilot AI Features. <https://www.microsoft.com/en-us/windows/copilot-ai-features> Accessed: 30-03-2024.
- [44] Microsoft. 2024. What is Cortana. <https://support.microsoft.com/en-us/topic/what-is-cortana-953e648d-5668-e017-1341-726f7df0f825> Accessed: 30-03-2024.
- [45] Aditi Mishra, Utkarsh Soni, Anjana Arunkumar, Jinbin Huang, Bum Chul Kwon, and Chris Bryan. 2023. Promptaid: Prompt exploration, perturbation, testing and iteration using visual analytics for large language models. *arXiv preprint arXiv:2304.01964* (2023).
- [46] Eric Nichols, Leo Gao, and Randy Gomez. 2020. Collaborative storytelling with large-scale neural language models. In *Proceedings of the 13th ACM SIGGRAPH*

- Conference on Motion, Interaction and Games.* 1–10.
- [47] OpenAI. 2024. Introducing ChatGPT. <https://openai.com/blog/chatgpt> Accessed: 30-03-2024.
 - [48] OpenAI. 2024. Introducing GPTs. <https://openai.com/blog/introducing-gpts> Accessed: 30-03-2024.
 - [49] OpenAI. 2024. Start Using ChatGPT. <https://openai.com/blog/start-using-chatgpt-instantly> Accessed: 30-03-2024.
 - [50] Vishakh Padmakumar and He He. 2022. Machine-in-the-Loop Rewriting for Creative Image Captioning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (Eds.). Association for Computational Linguistics, Seattle, United States, 573–586. <https://doi.org/10.18653/v1/2022.naacl-main.42>
 - [51] Filip Radlinski and Nick Craswell. 2017. A theoretical framework for conversational search. In *Proceedings of the 2017 conference on conference human information interaction and retrieval*. 117–126.
 - [52] Steven I Ross, Fernanda Martinez, Stephanie Houde, Michael Muller, and Justin D Weisz. 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*. 491–514.
 - [53] Hanieh Shakeri, Carmen Neustaeder, and Steve DiPaola. 2021. SAGA: Collaborative Storytelling with GPT-3 (*CSCW ’21 Companion*). Association for Computing Machinery, New York, NY, USA, 163–166. <https://doi.org/10.1145/3462204.3481771>
 - [54] Nikhil Singh, Guillermo Bernal, Daria Savchenko, and Elena L Glassman. 2023. Where to hide a stolen elephant: Leaps in creative writing with multimodal machine intelligence. *ACM Transactions on Computer-Human Interaction* 30, 5 (2023), 1–57.
 - [55] Snapchat. 2024. What is My AI on Snapchat and how do I use it. <https://help.snapchat.com/hc/en-us/articles/13266788358932-What-is-My-AI-on-Snapchat-and-how-do-I-use-it> Accessed: 30-03-2024.
 - [56] Shahab Saqib Sohal, Faiza Farhat, Yassine Himeur, Mohammad Nadeem, Dag Øivind Madsen, Yashbir Singh, Shadi Atalla, and Wathiq Mansoor. 2023. Decoding ChatGPT: A taxonomy of existing research, current challenges, and possible future directions. *Journal of King Saud University - Computer and Information Sciences* 35, 8 (2023), 101675. <https://doi.org/10.1016/j.jksuci.2023.101675>
 - [57] Hendrik Strobelt, Albert Webson, Victor Sanh, Benjamin Hoover, Johanna Beyer, Hanspeter Pfister, and Alexander M Rush. 2022. Interactive and visual prompt engineering for ad-hoc task adaptation with large language models. *IEEE transactions on visualization and computer graphics* 29, 1 (2022), 1146–1156.
 - [58] Priyan Vaithilingam, Tianyi Zhang, and Elena L Glassman. 2022. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *Chi conference on human factors in computing systems extended abstracts*. 1–7.
 - [59] Minh Duc Vu, Han Wang, Zhuang Li, Jieshan Chen, Shengdong Zhao, Zhenchang Xing, and Chunyang Chen. 2024. GPTVoiceTasker: LLM-Powered Virtual Assistant for Smartphone. *arXiv preprint arXiv:2401.14268* (2024).
 - [60] Jiayin Wang, Weizhi Ma, Peijie Sun, Min Zhang, and Jian-Yun Nie. 2024. Understanding User Experience in Large Language Model Interactions. *arXiv:2401.08329 [cs.HC]*
 - [61] Ryen W White. 2023. Navigating complex search tasks with AI copilots. *arXiv preprint arXiv:2311.01235* (2023).
 - [62] Wikipedia. 2024. Virtual Assistants. https://en.wikipedia.org/wiki/Virtual_assistant Accessed: 30-03-2024.
 - [63] Writeful. 2024. TexGPT: Harness the power of ChatGPT in Overleaf. <https://blog.writefull.com/texgpt-harness-the-power-of-chatgpt-in-overleaf/> Accessed: 30-03-2024.
 - [64] Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. 2023. A survey on multimodal large language models. *arXiv preprint arXiv:2306.13549* (2023).
 - [65] Ann Yuan, Andy Coenen, Emily Reif, and Daphne Ippolito. 2022. Wordcraft: story writing with large language models. In *27th International Conference on Intelligent User Interfaces*. 841–852.
 - [66] Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2024. Toolqa: A dataset for llm question answering with external tools. *Advances in Neural Information Processing Systems* 36 (2024).
 - [67] Julie S Zide, Maura J Mills, Comila Shahani-Denning, and Carolyn Sweetapple. 2017. Work interruptions resiliency: toward an improved understanding of employee efficiency. *Journal of Organizational Effectiveness: People and Performance* 4, 1 (2017), 39–58.