Cells, Generators, and Lenses: Design Framework for Object-Oriented Interaction with Large Language Models

Tae Soo Kim
taesoo.kim@kaist.ac.kr
School of Computing, KAIST
Daejeon, Republic of Korea

Yoonjoo Lee
yoonjoo.lee@kaist.ac.kr
School of Computing, KAIST
Daejeon, Republic of Korea

Minsuk Chang*
minsukchang@google.com
Google Research
Seattle, Washington, United States

Juho Kim
juhokim@kaist.ac.kr
School of Computing, KAIST
Daejeon, Republic of Korea

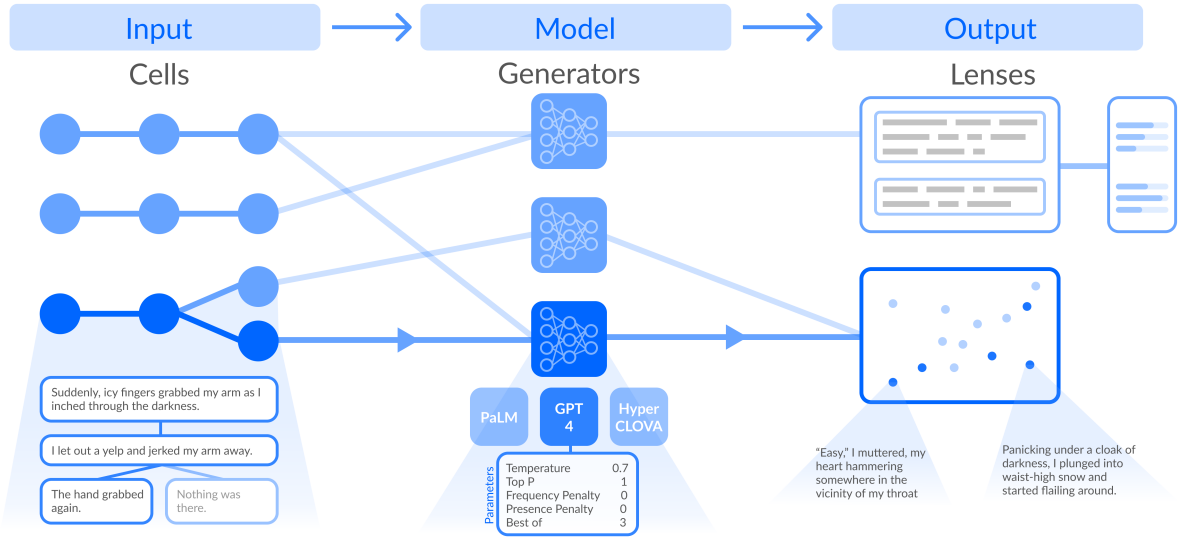

Figure 1: *Cells, generators, and lenses* is a design framework for object-oriented interaction with large language models (LLMs). Input units, model instances, and output spaces are represented as interactive objects: cells, generators, and lenses, respectively. By integrating these objects in their designs, designers can create interfaces that support users to flexibly create, modify, and link these objects to iterate and experiment with diverse configurations for the generative process of LLMs.

ABSTRACT

Large Language Models (LLMs) have become the backbone of numerous writing interfaces with the goal of supporting end-users across diverse writing tasks. While LLMs reduce the effort of manual writing, end-users may need to experiment and iterate with various generation configurations (e.g., inputs and model parameters) until results meet their goals. However, these interfaces are not designed for experimentation and iteration, and can restrict how end-users track, compare, and combine configurations. In this work, we present *"cells, generators, and lenses"*, a framework to designing interfaces that support interactive objects that embody configuration components (i.e., input, model, output). Interface designers can apply our framework to produce interfaces that enable end-users to create variations of these objects, combine and recombine them into new configurations, and compare them in parallel to efficiently iterate and experiment with LLMs. To showcase how our framework generalizes to diverse writing tasks, we redesigned three different interfaces—story writing, copywriting, and email composing—and, to demonstrate its effectiveness in supporting end-users, we conducted a comparative study (N=18) where participants used our interactive objects to generate and experiment more. Finally, we investigate the usability of the framework through a workshop with designers (N=3) where we observed that our framework served as both bootstrapping and inspiration in the design process.

*Minsuk conducted this work while at NAVER AI Lab.

UIST '23, October 29–November 01, 2023, San Francisco, CA, USA

ACM ISBN 979-8-4007-0132-0/23/10…$15.00
https://doi.org/10.1145/3586183.3606833

CCS CONCEPTS

• **Human-centered computing** → **Interactive systems and tools**; **Interaction design theory, concepts and paradigms**; *Empirical studies in HCI*; • **Computing methodologies** → **Natural language generation**.

KEYWORDS

Generative Models; Large Language Models; Reification; Writing-Support Tool

ACM Reference Format:

Tae Soo Kim, Yoonjoo Lee, Minsuk Chang, and Juho Kim. 2023. Cells, Generators, and Lenses: Design Framework for Object-Oriented Interaction with Large Language Models. In *The 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23), October 29–November 01, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3586183.3606833

1 INTRODUCTION

Large language models (LLMs)—e.g., ChatGPT [70], GPT-4 [71], PaLM [14], LLaMa [89]—have enabled users to write without actually writing. Users can delegate the manual effort of producing text to these models to increase productivity [11], inspire new ideas [12], or quickly "sketch" passages [16]. Besides shouldering the effort of producing text, LLMs can also enhance auxiliary processes such as editing [25, 52], feedback exchange [43, 91], or reflection [20]. However, when using these models, users are faced with a new task: manually configuring the model's generative process to produce desired outputs. Namely, users need to compose the *inputs* to the model [41, 84] (e.g., prompt engineering [102]) and adjust the model's *parameters* [53] (e.g., increase the *temperature* to generate more out-of-distribution text). Furthermore, users may want to configure how they view and explore the generated outputs based on their goals and tasks [18, 64]—e.g., using a list to carefully read specific edits or a spatial visualization to quickly compare the similarity between rough drafts. Thus, to accomplish their writing goals with LLMs, users need to configure the whole generation process—input, model, and output.

However, due to the black box and non-deterministic nature of LLMs, users can struggle to interpret why the models generated certain outputs and how one can "correct" them [78]. Users may then need to repeatedly experiment with generation configurations to understand their effect [15, 61, 87]—expending significant effort. In creative tasks, iteration (i.e., repeatedly developing an idea) and experimentation (i.e., enumerating and testing diverse ideas) are integral to understanding and exploring the design space [19, 80]. Thus, beyond the goal of understanding the models, users need to iterate and experiment with generation configurations to open up the vast space of writing alternatives that they can produce.

Despite the user needs for iterating and experimenting, LLM-powered writing interfaces largely adhere to the design of conventional text editors. These interfaces typically provide end-users with only one text area for inputs, which is frequently shared with the output, and one control panel to configure global settings for the parameters [1, 69, 85, 109]. To test different inputs and parameters in this type of interface, the user has to try each configuration one-by-one while overwriting previous configurations. As configurations are overwritten, the end-user cannot store previous configurations (i.e., versioning) to return to them if future iterations do not result in satisfying outputs [97], which introduces friction and hinders experimentation. Furthermore, these interfaces do not allow end-users to prototype configurations in parallel [24], which can create hurdles for comparing the effects of different configurations or combining aspects of the configurations for further iteration and experimentation. These limitations call for interfaces to move away from the designs of conventional text editors, and move towards a new paradigm that focuses on facilitating end-users' configuration of LLMs' behavior.

In this work, we introduce a design framework for interfaces that support object-oriented interaction with LLMs through *cells, generators, and lenses* (Fig. 1). Unlike existing interfaces where end-users interact with *one* input area, parameter setting, and output space, our framework proposes how interfaces can reify [8] the generation components so end-users can compose configurations by interacting with persistent, multiplicable, and composable objects. Within this framework, each object becomes its own configuration sandbox where end-users can experiment and iterate with changes, without affecting other configurations that they have created. Furthermore, our framework describes how interfaces can support end-users to flexibly assemble and reassemble these objects into diverse concurrent configurations—supporting parallel prototyping [24] and mix-and-match between configurations. Interface designers can use the framework to create interfaces that support their end-users' iteration and experimentation in their target writing tasks.

To demonstrate the value of our framework, we evaluate it according to three dimensions: *generalizability* (can it be applied to diverse writing tasks?), *effectiveness* (can it support end-users' iteration and experimentation?), and *usability* (can designers use and apply our framework?). First, to demonstrate how our framework can *generalize*, we applied it to design three interfaces that support diverse tasks: (1) story writing, (2) copywriting, and (3) email composing. Second, to evaluate *effectiveness*, we conducted a controlled study (N=18) where we investigated how end-users' iteration and experimentation is affected by the ability to create and compose multiple configuration objects. We observed that, when using our framework-based interface, participants were encouraged to generate more outputs, experiment with more inputs, and use generated outputs more substantially in their final writing. Finally, to demonstrate the *usability* of our framework, we conducted a workshop where we invited designers (N=3) and asked them to re-design existing writing interfaces with our framework. We found that the framework bootstrapped designers by illustrating concrete ways to design interactions for iteration and experimentation, and inspired them to reify other aspects of their interfaces to further support end-users.

Our framework aims to guide the design of a new line of interfaces that enable object-oriented interaction with LLMs to support end-users' iteration and experimentation with generation configurations. To bootstrap the design and development of interfaces based on our framework, we release our interface components for cells, objects, and lenses as an open-source ReactJS library: https://github.com/kixlab/llm-ui-objects.

2 BACKGROUND AND RELATED WORK

To exhibit the broader design space and the underlying shared themes, we survey interfaces for a broad range of generative models. Then, as our work focuses on LLMs, we review work on interfaces for these models. Finally, we review work on human-AI writing tools.

2.1 Interactions with Generative Models

To help users to better leverage the potential of generative models, a significant amount of research has investigated how users wish to use these models. This body of work demonstrated that the "ideal" form to use these models changes with the type of user, their goals, and the task. For example, several studies demonstrated that it is integral for the user to lead the model [61, 68], while others demonstrated that there are benefits in the model taking the lead [18, 37]. Beyond who leads, research has identified several other trade-offs: producing more generations increases exploration but decreases efficiency [11], and more unexpected generations can provide inspiration but can also seem less useful [18, 53, 108]. Due to these user-dependent factors, prior work [31, 35, 41] argues that how generative models are used should adapt according to users' changing goals. This highlights the need to support user-driven configuration in interfaces for generative models.

Existing interfaces for generative models have focused on facilitating the configuration of each component of the generation process: input, model, and output. On input, GANSliders [22] and GANSpace [40] support goal-driven input customization for GANs through visual feedforward sliders and semantic controls, respectively. Also, to facilitate input iteration, Opal [59] and 3DALL-E [60] modularize user inputs into keywords and provide keyword suggestions to facilitate composition of inputs for image generation. Regarding model parameters, Louie et al. [61] and Zhou et al. [118] provide more interpretable parameters to help users control generated outputs. More recently, TaleBrush [16] supports more seamless control over parameters by allowing users to sketch how the parameter should change or "flow" through a story. Finally, on outputs, DreamLens [64] allow users to navigate hundreds of generated 3D models through multiple, customizable views (e.g., gallery view, attribute grid view). While these approaches support configuration of each component in the generative process, no work has investigated how to support configuration of component combinations. Our work proposes a framework that guides the design of interfaces that can facilitate inter-component customization and can be extended to support these prior approaches for intra-component customization.

2.2 Interfaces for Large Language Models

Given a prompt as input (i.e., an instruction that may contain examples of expected outcomes), LLMs can generate text that follows this prompt. and even perform previously unseen tasks through zero and few-shot learning. Due to the opportunities presented by these capabilities, HCI researchers have designed an assortment of interfaces that leverage LLMs to support a variety of user tasks beyond writing, such as information seeking and consumption [5, 92], learning [55, 57, 63], and prototyping [49, 74, 76]. However, as the effectiveness of LLMs is significantly affected by minor variations in the input prompts, researchers have also proposed interfaces to facilitate the task of creating these inputs (i.e., prompt design or engineering). PromptMaker [46] and BotDesigner [112] allow users to create prompt templates and test them with different inputs. Expanding on these ideas, PromptIDE [83] allows the user to automatically create and evaluate prompt variations, and ScatterShot [101] supports an iterative loop of evaluating prompts and identifying effective examples to add to the prompts. Beyond singular prompts, AI Chains [102] and PromptChainer [100] allow users to iterate on multi-step prompts using interactive chains of prompts.

While our work and these approaches hold the same goal of supporting iteration with LLMs, they focused only on the input component of the generative process—e.g., users cannot test model parameters in PromptIDE [83]. Additionally, all of these existing approaches propose point solutions to be used by developers and ML practitioners during prompt engineering. On the other hand, our work proposes general guidelines for interfaces that are designed to support end-users' iteration on all three generation component during writing tasks.

2.3 Human-AI Writing Support Tools

Advancements in natural language processing (NLP) has enabled the creation of tools that can support a diverse array of writing tasks and processes. For example, researchers have proposed human-AI writing tools for story writing [16, 18, 109], screenplay writing [66], poetry [33], and argumentative writing [91, 115]. Each of these tools leverages AI to support different aspects of the writing process. A large portion of these approaches provide automatic continuations to enhance writers' productivity [18, 53, 82] with several of these allowing writers to control the type of continuations that are generated by providing additional instructions [21, 109] or through visual sketches [16]. Other tools support auxiliary processes during writing such as ideation [31, 32, 75, 116], revision [25, 56], and reflection [20, 91]. Our work builds on these prior literature by proposing a general design framework for human-LLM writing tools that can be applied to a variety of writing tasks with the goal of supporting the processes of iteration and experimentation.

3 CELLS, GENERATORS, AND LENSES

To design writing interfaces that support iteration and experimentation with LLMs, this work proposes a design framework that conceptualizes the components of generation configurations as interactive objects. The framework describes how to design interfaces that encapsulate these objects and the interactions that can be supported on and between these objects. Prior work has demonstrated how elevating task elements (e.g., visual attributes [105], spatial selections [106], text passages [39]) into interactive objects can simplify and facilitate users' workflows [7, 17, 103, 107]. By supporting object-oriented interaction, interfaces can allow end-users to maintain task elements, which would have previously been transient, as persistent objects that can be reused and composed into new combinations [104]. Further, with persistent and composable objects, end-users can combine these into multiple parallel prototypes, which could prevent fixation and encourage experimentation [44].

Additionally, if end-users are able to create and maintain alternatives in parallel, they can readily compare these alternatives to solve problems [30] and understand the task in greater depth [9].

Inspired by the advantages of *reifying* task elements into *reusable* objects [8], in this work, we investigate how to reify the components of generation configurations into persistent, multiplicable, and composable objects. Specifically, we first conducted a systematic literature survey of prior work on interactions with LLMs to identify user needs, challenges, and design insights. Furthermore, we also review work on other types of generative models as HCI researchers have investigated diverse models prior to LLMs and these share various similarities. Then, based on this survey, we propose a design framework for interactive objects for LLMs that consists of *cells, generators, and lenses*. These three objects respectively represent the input, model, and output—the main configuration components for LLMs. By employing our design framework, designers can create interfaces that allow end-users to (1) create and maintain multiple variations of these generative components as objects, and (2) link them to each other to assemble and re-assemble various configurations in parallel.

3.1 Cells

Cells (Fig. 2) are object representations of discrete input units—i.e., fragments of text. For example, a cell can represent a sentence, a phrase, or a word. With respect to LLM prompts, a cell can also represent an instruction line (e.g., a specification) or an example of the expected behavior. Our decomposition of input into cells is analogous to how computational notebooks decompose code into cells, which has been shown to be effective in supporting flexible customization and testing [58, 96]. Below, we describe and justify two interactions that designers should support regarding cells, *create* and *assemble*.

3.1.1 Create. Users can create new cells to populate with differing inputs, or copy existing cells and edit them into various versions of the same input. With existing interfaces, users have to overwrite previous inputs when testing new variations as they are frequently only presented with a single input area, usually a text box [1, 69, 109]. In contrast, by interacting with cells, users can create and maintain various generation inputs that they can experiment with—allowing them to more easily answer their own "what if" questions about the effect of inputs on outputs [56, 84, 86]. For example, if a user is composing a poem generating prompt that specifies requirements such as topic and form, they can create alternative cells for each requirement (e.g., one cell for "haiku" and one for "sonnet"). In certain tasks, the generated output in one interaction turn can become part of the input for the next turn (e.g., generating continuations that are added to a story). In these cases, the generations themselves can become cells and allow the user to test how different generated outputs affect future generations (e.g., generating alternative storylines). As multiple text fragments can occupy significant screen space, designers can provide mechanisms to "minimize" cells in their interfaces. However, to prevent occluding the content of cells and hindering users' access [8], cell minimization should be designed such that it hints at the content by, for example, only decreasing font size or summarizing the text into keywords.

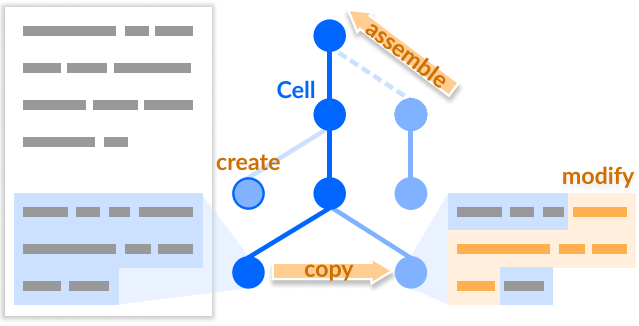

Figure 2: *Cells* are object representations of input units (e.g., sentences). To create variations of inputs, users can create, copy, modify, and assemble cells. Cells that have been assembled together are shown connected by blue edges.

3.1.2 Assemble. Our framework suggests that interfaces should split the input text into interactive cells as prior generative interfaces showed that partitioning can facilitate iteration on inputs [4, 10, 94, 102]. As units, cells can then be assembled together into generation inputs (e.g., sentences into an essay and requirements into a prompt) which allows users to quickly assemble input variations [6, 59] or to mix-and-match the variants [101, 111]. Finally, disentangling inputs also helps users to "lock" portions of the input and experiment with the effects of each portion separately—encouraging systematic testing [112]. This portion-wise experimentation can allow users to interactively align generations with their intentions [2, 117] and to more intuitively gain an understanding of the models [78, 110]. Designers can provide different forms of interaction for cell assembly based on the task and how cells are used. For example, the user could assemble cells by dropping them into a container, by selecting cells from multiple parallel configurations to mix-and-match, or by drag-and-dropping between cells to create the links. As cells can represent different types of text fragments (e.g., line, phrase), interfaces should have pre-defined rules that dictate how these text fragments are concatenated once cells are assembled. For example, an interface that represents lines as cells should assemble them by concatenating the text with line breaks, and one that represents phrases as cells should concatenate them with spaces.

3.2 Generators

Generators (Fig. 3) are object representations of model settings (i.e., type of LLM and parameter values).

3.2.1 Create. With *generators*, the user can create several model instances and separately modify each one (i.e., choose a different model and/or parameters) to experiment with their effects. Prior work on various types of generative models has revealed that different models and parameters can satisfy different user needs as they can produce different results [38, 48, 62]. Regarding writing and LLMs, Lee et al. [53] found that different parameter settings can fulfill different writing goals (e.g., diverging vs converging),

and Chung et al. [15, 16] argued that interfaces should facilitate parameter adjustment to support writers' control over the generative process. But, in existing interfaces, users cannot modify the underlying parameters [83, 109] or can only customize one global set of parameters [1, 33, 34, 61, 69, 81]. In comparison, designing interfaces with generators can allow users to simultaneously maintain several model instances, where each addresses a different sub-task or need [41]. Additionally, as the effect of configuration changes cannot be fully predicted due to the black box and non-deterministic nature of LLMs, users need to iteratively test different configurations and, as they iterate, may want to return previously tested configuration [97]. By reifying model configurations into generators, users are able to use these objects to maintain previously promising configurations—i.e., versioning. Similar to cells, generators should also be designed such that they visually hint at their parameter settings and also support efficient access to edit these parameters.

3.2.2 Link. Generators can be freely linked to different cells or cell ensembles to produce outputs. These links can be many-to-many to help the user test various combinations of inputs and model parameters. For example, users can link the same cell to multiple generators to compare the effects of different parameter settings, or link multiple cells to one generator to experiment with a range of inputs. Designers can create interfaces that make the linking process explicit (i.e., the user drag-and-drops between cells and generators to create links) or implicit (i.e., the user selects cells to use as input and then clicks on a generator to use its configurations to generate).

3.2.3 Track. As objects, generators can also keep track of their individual history of parameter changes, generated outputs, and linked cells used as input. To support this history, interfaces can log all of the generation events (i.e., input text, parameter settings, and an array of generated outputs) for each generator. With this tracking, users can examine and explore their iterative process to carry out sensemaking on how parameters affected the resulting generated outputs [3].

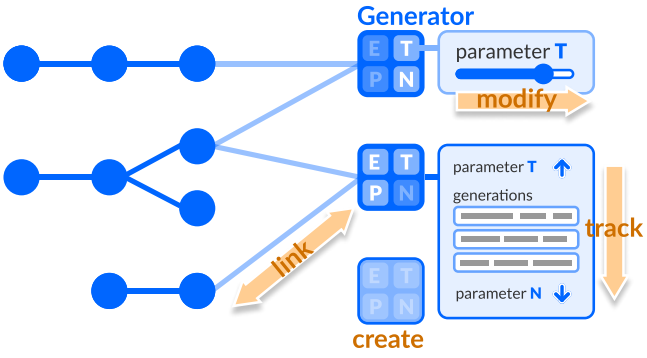

Figure 3: *Generators* are object representations of generative model instances (i.e., the type of model and its parameters). Users can create generators, modify their parameters, and then link them to one or multiple cells to generate outputs. Generators display their parameter settings in their faces (represented as letters in the diagram). Additionally, generators can maintain their own history to help users track how parameters have increased and decreased, and what generations resulted from these changes.

3.3 Lenses

Lenses (Fig. 4) are object representations of spaces that represent and visualize the generations produced. For example, these can include a list [18], a gallery [88], or a real-time confidence visualization [65].

3.3.1 Link. By linking generators to lenses, users can represent the generation outputs in diverse ways. Effective representations of the generations support the exploration of the generations and sensemaking of the models' capabilities [28, 95, 114, 118]. However, as revealed by work on human-AI writing interfaces, the "most effective" representation can be dependent on two dimensions. First, the user's needs: when brainstorming storylines, for example, visually representing the generated story arc can help with sensemaking [16] but, when choosing the next sentence for a story, a list of generations allows the user to concretely compare them [31, 109]. Second, the generation amount (i.e., length or number of alternatives): as they write, users may want to see longer or more alternatives and, due to the increased reading cost, interfaces need to provide user's with different ways to parse and examine these [11]. Thus, our framework suggests interfaces to include a variety of lenses to help users customize how they visualize and explore generations (e.g., linking a generator to a suitable lens, switching between lenses, or comparing generators by linking them to one lens).

3.3.2 Assemble. Lenses can also be assembled together to view the same generation outputs through multiple representations [64, 82]. For example, a list lens and a sentiment scatterplot lens (i.e., predicts sentiment of text) could be joined to allow the user to explore both the content and sentiment of generated text. As users consider various characteristics or metrics when making sense of generation outputs [23], allowing them to assemble lenses can support more comprehensive sensemaking.

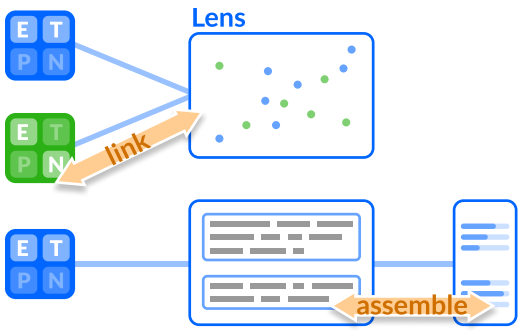

Figure 4: *Lenses* are object representations of output spaces that represent and visualize generations from linked generators (e.g., a list or a 2D grid). Lenses can be assembled together to visualize the same generations in multiple ways.

4 APPLYING THE FRAMEWORK

To illustrate how our framework can *generalize* to diverse writing tasks, we applied it to design and develop three interfaces for different tasks: story writing, copywriting, and email composing. Specifically, we exhibit how cells, generators, and lenses can be adapted into interface design to support end-users iteration and experimentation in the context of specific tasks. We designed these interfaces based on existing ones that support the same tasks to demonstrate how the interaction changes when our framework is applied. To gain a preliminary understanding about how end-users could use our interfaces to iterate and experiment, we invited two participants to use each interface (total N=6, 3 female, 3 male), where all participants had previous writing experiences and at least a basic understanding of machine learning (ML). We describe these preliminary observations after describing each interface. Furthermore, through the development of these interfaces, we modularized UI components for cells, generators, and lenses and have packaged these into an open-source ReactJS library[1]. In contrast to frameworks such as LangChain [13] that facilitate the development of backends for LLM applications, we hope that this library can foster wider adoption of our framework by facilitating the development of frontends for LLM-powered interfaces.

4.1 Copywriting Interface

Our copywriting interface (Fig. 5) allows end-users to create advertisements from a couple of product specifications. The interface was designed based on copywriting tools [18, 45, 99] that provide forms where end-users specify requirements for the desired advertisement (e.g., tone, audience) and an LLM then attempts to generate it. By offloading the effort of writing to the model, end-users can use these tools to produce various alternatives for their advertisements. However, as end-users can only interact with a single form, every edit overwrites previous specifications and can hinder end-users' abilities to recall previous attempts or to iterate on these attempts by combine them.

To address these issues, we applied our framework to design a new copywriting interface. In this interface, end-users can create and maintain specification alternatives as cells, and then assemble these into new combinations (Fig. 5a). To allow end-users to test various model parameters, the interface allows them to create multiple generators, each with its own parameter settings (Fig. 5b). Finally, to help end-users navigate the advertisements they generated, the interface provides two lenses that are assembled together to allow navigation based on content, similarity, sentiment, or emotion (Fig. 5c).

4.1.1 Composing the Specifications. In our copywriting interface, end-users compose specifications for their desired advertisement (e.g., product description, tone, keywords) by creating and editing cells in the prompt area (Fig. 5a). In each line, the user can specify a different specification type (e.g., "tone"), and then create multiple variations for that specification by adding or copying cells in that line (e.g., "informative, friendly" or "comical, humorous"). By selecting a cell for each line or not selecting any for some lines, the user can mix-and-match different specification sets to use as inputs when generating.

[1] https://github.com/kixlab/llm-ui-objects

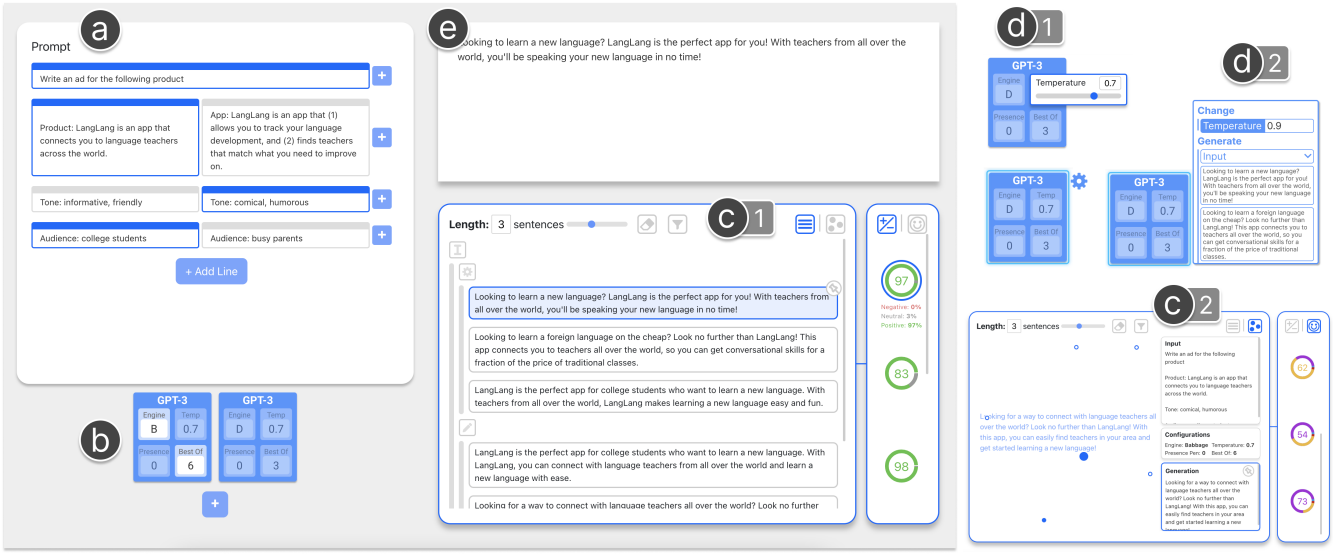

Figure 5: The copywriting interfaces allows end-users to provide a set of specifications to generate advertisements by creating and editing cells (a). In the generator tray (b), end-users can create multiple generators and modify their parameters (d-1). Then, by clicking on a generator, end-users can generate advertisements that are presented in a list (c-1) or a 2D space (c-2), and rated according to their predicted emotion or sentiment. To look back on how the parameters of each generator where changed and what outputs it generated, end-users can also browse through the history of each generator (d-2).

4.1.2 Generator Tray. Under the prompt area, the interface shows the tray of generators where end-users can create, copy, and maintain multiple settings for the model's parameters (Fig. 5b). Each generator presents four parameters that the user can modify: *engine* (i.e., the model type and version), *temperature* (i.e., the degree to which out-of-distribution tokens are generated), *presence penalty* (i.e., penalty on the probability of generating tokens that have already been generated), and *best of* (i.e., number of candidate outputs to generate from which the model returns the "best").[2] Generators show the current value for each parameter in its face and users can change these value by clicking on a parameter to reveal a control panel (e.g., dropdown menu, slider). By clicking on a generator, the user can start generating outputs with that generator's parameters and the currently selected specifications as the input, which are concatenated with line breaks. To look back on how parameter changes may have affected the outputs, end-users can open the history panel to browse through the generator's individual log of parameter changes and generated outputs (Fig. 5d-2).

4.1.3 Lenses: List, Space, and Rating. Initially, end-users are presented with the list lens (Fig. 5c-1, left) which presents generations as a list of text entries that are grouped at two levels: the input that was used, and the parameters that were used. This two-level grouping can help end-users distinguish where they made changes to the generation configurations and compare generations across groups. To explore generations based on their similarities and differences, end-users can toggle the space lens (Fig. 5c-2, left) where outputs are presented as dots in a 2D space where closer dots represent more semantically similar outputs. Next to the toggleable list-space lens, end-users can view a different representation of the outputs through the rating lens (Fig. 5c-1 and Fig. 5c-2, right). The rating lens provides a high-level impression of the generated advertisements based on their emotion (i.e., joy, sadness, anger, optimism) or sentiment (i.e., positive, negative, neutral)—the user can toggle between these two options. If the end-user finds a generated advertisement that they like, they can click on it to copy it to the text editor (Fig. 5e) where they can then edit and combine it with other generations.

4.1.4 Use Cases. For the copywriting interface, the two participants were asked to write advertisements for two imaginary products: an AI-based language teaching service, and a super-insulated tumbler. To generate fragments that had the "tone" that they desired, participants created various specifications in individual cells: the portion of the advertisement that should be generated (P1, P2), the target audience (P2), or adding generated fragments as examples (P1). Further, participants experimented with various specification sets by copying, modifying, and assembling different cells. By experimenting with the cells and generators, participants were also able to better understand the effect of inputs and parameters on the generations. For example, at the beginning of the session, P4 tried only experimenting with one cell that had *"keyword"* as the prefix, and mentioned how this allowed her to learn how that specific cell affected the output. Also, both participants iterated by alternating their experimentation between cells and generators: experimenting with generators until outputs appeared adequate, reusing that generator with different cells until outputs defied expectations, and then debugging by testing different generators again—resembling more systematic testing [112]. Regarding the lenses, both participants used the rating lens to quickly compare generations from diverse configurations, and then used the list lens to identify specific phrases that they liked.

[2]Among the parameters provided by the OpenAI API, these were identified to be the most useful parameters by end-users in our preliminary studies.

4.2 Email Composing Interface

During email composition, as the end-user frequently has a concrete idea of *what* to write, the LLM can instead help with *how* to write it (e.g., changing the tone, paraphrasing). To support this, existing LLM-powered interfaces for emails [27, 52] provide designated *"brushes"* that allow end-users to select text and perform specific generative functions on the selected text [47]. However, these existing interfaces do not allow end-users to design their own LLM-powered brushes to satisfy their personal needs.

To enable greater customization, we applied our framework to envision an email composing interface (Fig. 6) that packages cells, generators, and lenses into brushes—allowing the user to design their own reusable LLM-powered brushes by iterating with these objects (Fig. 6b). For each brush, end-users can assemble cells to specify the brush's purpose (Fig. 6c), set multiple generators, and choose their preferred lens to show the generated outputs (Fig. 6e). We considered that, compared to copywriters, email writers would be more task-driven and less inclined to explore various cell-generator-lens permutations. Thus, we limited linking to one-to-many where each brush can only house one cell ensemble and one lens, but multiple generators. This design limits end-users to consider one input alternative at a time but, with a single click of a brush, they can simultaneously test multiple generators and compare them in one visual space.

4.2.1 LLM-Powered Brushes. To the right of the text editor (Fig. 6a), the user can create and manage the LLM-powered brushes (Fig. 6b). To use a brush, the user simply clicks on the corresponding button or, if a brush performs actions on user-selected text (e.g., paraphrase a chosen phrase), the user should first select text in the editor. To modify the configurations behind a brush, the user can hover over a brush and click on the right arrow to display its configurations.

4.2.2 Composing the Prompt. Similarly to the copywriting system, end-users configures the input for the brush by using cells to represent individual specifications (Fig. 6c). A difference is that this interface provides two additional types of cells: *"selection"* and *"whole text"*. When the input is assembled, a "selection" cell is transformed into an input line by concatenating the user-written prefix with text that the user selects in the editor, and a "whole text" cell concatenates the prefix with all of the text in the editor.

4.2.3 Generator Set. Each brush can hold several generators (Fig. 6d), which function like those in the copywriting interface. When a brush is used, the interface generates outputs with the parameters of each contained generator—with the same specification cells as input.

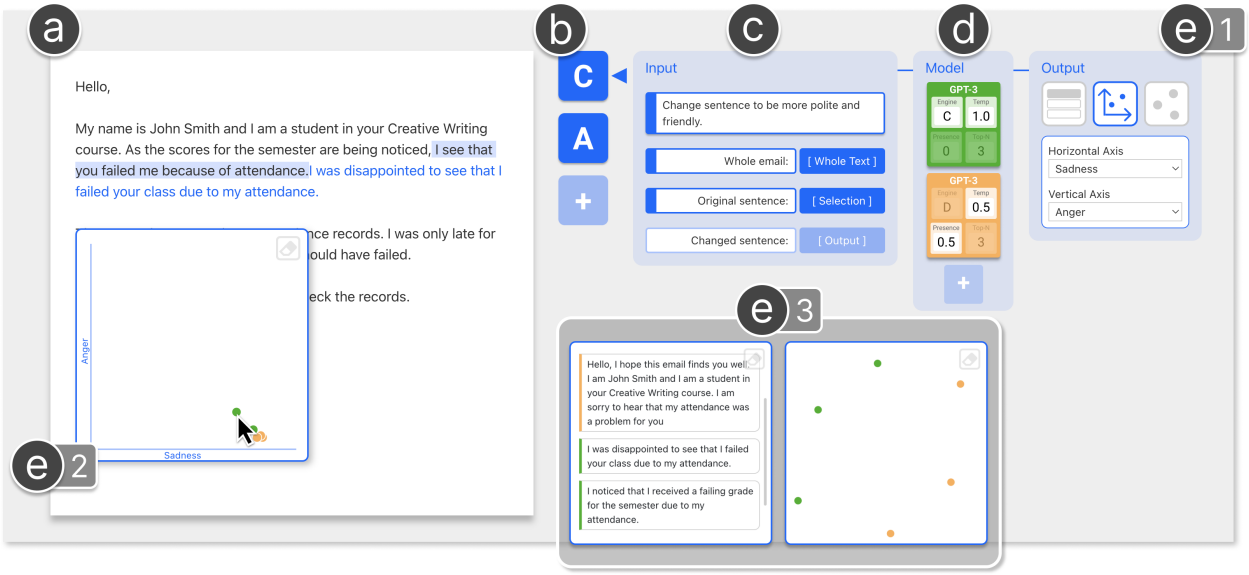

Figure 6: In the email composing interface, end-users can write an email in the text editor (a), and create dedicated LLM-powered brushes that can be configured to perform specific generative functions (b). For each of these brushes, the user composes an instruction prompt with cells (c), sets multiple generators (d), and selects between the list, space (e-3), or plot lenses (e-2) to present the outputs. When the user clicks on a brush, the model runs according to the designed configuration, generates outputs, and displays these in a hovering lens.

4.2.4 Lens: List, Space, and Graph. After an LLM-powered brush is clicked, the generated outputs are shown in a lens that hovers over the text editor. For each brush, the user can choose between three lenses: list, space, and plot. The plot lens (Fig. 6e-2) presents generations as dots in a scatter plot where the axis represents the output's score for a sentiment or emotion class. By setting what class to use for each axis, the user can choose their preferred "metrics" to explore the outputs with (Fig. 6e-1).

4.2.5 Use Cases. For the email writing interface, the two participants were instructed to write an email to a professor apologizing for not attending lectures, but asking for a passing grade in the course. Both participants mostly designed LLM-powered brushes that refined sentences or phrases in their emails (e.g., *"change the text to be more persuasive"* or *"change text to be more professional"*). For each brush, both participants created multiple generators as it allowed them to quickly identify the parameters that worked best for that brush's function—supporting efficient testing and evaluation. Compared to the other participant, one participant tested larger sets of generators simultaneously and was able to more quickly pinpoint what component of the pipeline he needed to iterate on.

4.3 Story Writing Interface

LLMs have also been employed for story writing. Various interfaces [1, 43, 69, 109] provide end-users with a text editor where they can write a story and then use an LLM to generate continuations for their story. While these interfaces can help end-users to quickly develop one plotline, they can struggle to manage and iterate on multiple plotlines in parallel as they would have to juggle alternatives in and out of the single editor. However, as identified by Dow et al. [24], parallel prototyping can prevent fixation and lead to higher quality and creative outputs.

By applying our framework, we introduce a story writing interface (Fig. 7) that partitions stories into cells to enable writers to assemble these into branching and parallel plotlines (Fig. 7b). In this interface, end-users can experiment with and comapre configurations by linking cells to multiple generators (Fig. 7c) and linking multiple generators to the same lens (Fig. 7d-3).

4.3.1 Branching and Parallel Paths of Cells. Next to the text editor, the user can view the tree representation of cells (Fig. 7b). Each cell represents a sentence and it is presented only with a keyword extracted from its content to prevent the screen from becoming excessively busy. Paths down the tree represent ensembles of sentences or, in other words, separate plotlines that the end-user has created. To view and edit the text of a plotline in the editor (Fig. 7a), end-users can click on a cell in the tree to ensemble all of the down the tree up to the selected cell. End-users can add more cells to create more branching paths by typing continuations in the editor, copying cells, or connecting cells to one another.

4.3.2 Linking Paths to Generators. The user can also extend a path by generating continuations. For this, end-users can drag-and-drop link a cell and one or more generators by drag-and-drop (Fig. 7c). When the user clicks on a generator, it will take the text in the path down to the linked cell as input to generate outputs.

4.3.3 Lenses: List, Space, and Peek. The user can choose between three lenses in this interface: list (Fig. 7d-1), space (Fig. 7d-2), or

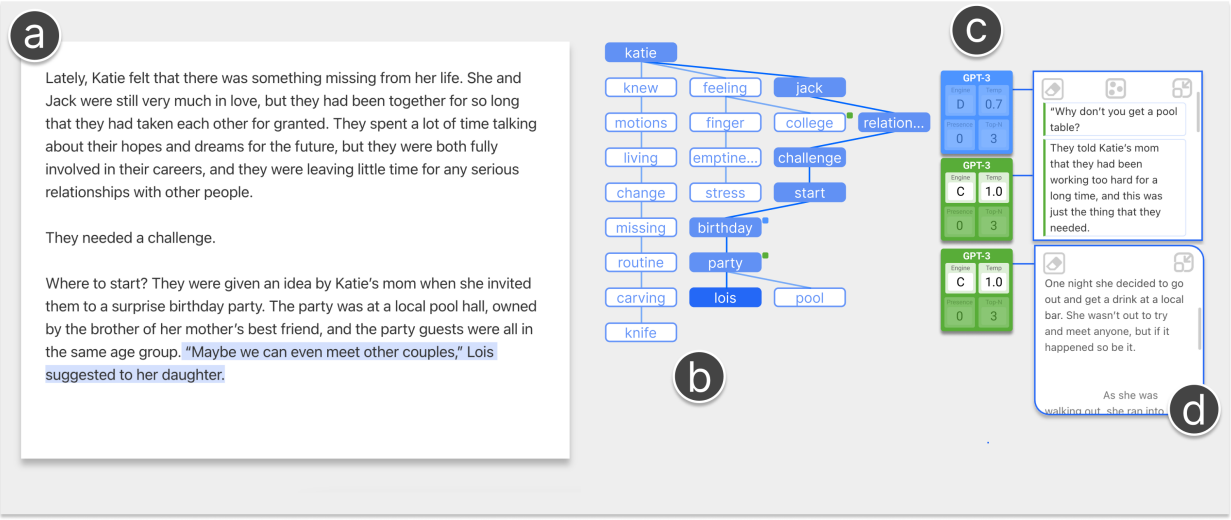

Figure 7: With the story writing interface, end-users can explore multiple, alternative plotlines. The user can create multiple plotlines by creating branching cells in a tree representation (b). Each cell contains a story sentence and is represented as a block enclosing a keyword extracted from the sentence. The most opaque block is the currently selected cell and it is shown highlighted in the editor. The user can create multiple generators (c), and then drag-and-drop between cells and generators to link them. Then, by clicking on a generator, the user can generate continuations to the linked cell which are then displayed in three types of lenses: list lens, space lens, or peek lens (d).

peek lens (Fig. 7d-3). The peek lens automatically extends the story from the linked cell by periodically generating a new sentence—until the user clicks on the generator again to stop it. When the end-user clicks on a generated output in any lens, it is added as a new cell that branches out of the cell that was used as input.

4.3.4 Use Cases. For the story writing interface, the two participants were asked to write a story based on a starting sentence. Through the objects, the participants were able to explore multiple plotlines. P6 generated multiple branches from the initial story prompt which they then developed individually with the same generator. Similarly, P5 created two branches, but developed each with a different generator to simultaneously test possible plotlines and model parameters. In terms of lenses, both participants used the peek lens to develop plots that they found interesting but did not know how to progress, and used the list and space lens to build on plots more deliberately. Both participants initially expanded horizontally—developing multiple plotlines with various generators—and mentioned how this helped them quickly identify both promising configurations and plotlines.

4.4 Implementation

We implemented the three interfaces using HTML, CSS, JavaScript, and ReactJS. The text editors were built using the SlateJS library[3]. To request and post-process the generations, we built a backend server with Flask that obtained generations through the OpenAI API[4]. The HuggingFace Transformers library [98] was used to process the generations for sentence similarity [29], sentiment [77], and emotion [67]. The KeyBERT technique [36] was used to extract keywords for the story writing interface.

[3]https://docs.slatejs.org/
[4]https://openai.com/api/

5 EVALUATION

To investigate the *effectiveness* of *cells, generators and lenses* in supporting end-users' iteration and experimentation, we conducted a between-subjects study where we compared our copywriting interface against a baseline that only provides one modifiable configuration. For this study, we focused on copywriting as a task, as we expected that this task would require substantial experimentation and iteration: the writer needs to effectively transmit a message with a limited set of sentences. In this study, participants were asked to write two advertisements, back-to-back, based on the provided product descriptions. In this study, we posed the following three research questions:

- RQ1. Can *cells, generators, and lenses* promote users' experimentation and iteration with various generation configurations?
- RQ2. How are users' generative processes affected by the presence of *cells, generators, and lenses*?
- RQ3. How do *cells, generators, and lenses* affect users' perceptions about the generative model and their final outcomes?

5.1 Participants and Apparatus

We recruited 18 participants (3 female, 15 male, age M=21.7 and SD=1.9), all of whom reported being comfortable with English reading and writing, and interest in creative writing. All of the participants had prior experiences with AI-based writing support tools

(e.g., grammar checkers, autocomplete), but had no experiences writing with LLMs.[5] Participants were randomly divided into the treatment group, which used our copywriting interface, and the control, which used a baseline. Although similar to our interface, the baseline (Figure 8) limited end-users to one input alternative per line, one set of parameter settings, and only a list representation of outputs. This baseline resembles existing interfaces for LLMs where end-users can only work with one configuration and must continuously overwrite it to iterate and experiment.

During the task, participants were asked to write advertisements for two products from a crowd-founding site[6]: a plant-based jerky, and a portable air conditioner. Participants were provided with the descriptions available on the funding pages of each product. To provide participants with a starting point and help them understand how they could prompt the model, we also provided them with a basic advertisement generation prompt (see Supplementary Materials).

5.2 Study Procedure

The study took place face-to-face while strictly following COVID-19 guidelines. Participants were provided with a laptop that had their assigned interface open. After reading and signing the informed consent form, participants first received a short explanation of the generative model they would be interacting with in the study, and a short tutorial on how to use their assigned interface. After the walkthrough, participants received the description of the first product. The order in which participants saw each product was also counter-balanced. Participants were then given 15 minutes to read the product description and use the given interface to write an advertisement that was two to five sentences long. After the allocated time, participants completed a short survey. After the survey, participants proceeded to the second product: they used the interface to write an advertisement for 15 minutes and completed the same survey. After the second advertisement and survey, a short interview was conducted where participants were asked about their experience.

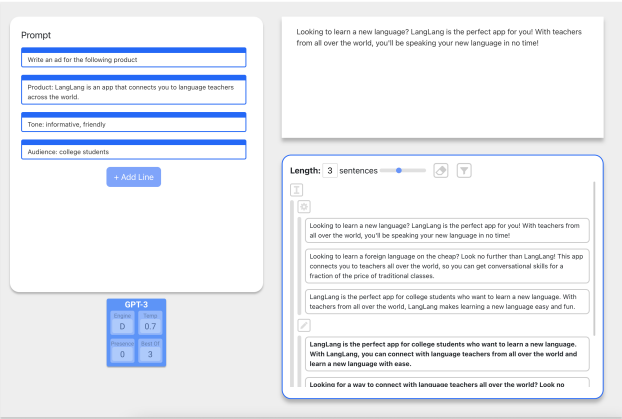

Figure 8: The baseline interface resembles our copywriting interface, but presents the users with only one cell per instruction line, one generator, and one lens.

[5]The study was conducted in early 2022 before the widespread adoption LLMs due to models like ChatGPT.

[6]https://kickstarter.com

5.3 Measures

For measures, we collected participants' survey responses after each advertisement. Similar to prior work [53, 54, 109], we asked participants to rate their agreement on a 7-point Likert scale (1: Strongly Disagree, 7: Strongly Agree) with the following statements:

- Helpful: *"I found the AI helpful."*
- Ease: *"I found it easy to write the advertisement."*
- Experiment: *"I felt that I experimented with various ideas and generated alternatives."*
- Iterate: *"I felt that I iterated various times on ideas and the generation process."*
- Pride: *"I'm proud of the final advertisement."*
- Unique: *"The advertisement I wrote feels unique."*

Additionally, we measured quantitative metrics related to participants' generation processes. For each advertisement written by participants, we measured the number of times that they generated, and the number of different unique inputs and unique parameter settings that were used to generate. Additionally, to measure the degree to which participants accepted the models' generations, we calculated the similarity between their final advertisements and the generated suggestions. While prior work [53] evaluated acceptance of AI generations by measuring the proportion of generations that were selected, in our study, we saw that participants frequently used fragments from generations without explicitly copying them with the interface. Thus, we calculated the BLEU score [73]—a measure used to evaluate the similarity between a piece of text and references—between participants' final advertisements and the generations they received. Additionally, to evaluate whether participants saw diverse or similar generations, we measured the Self-BLEU score [119], a metric frequently used to measure the diversity of generated outputs. The Self-BLEU score calculates the average BLEU score between each generation and all other generations.

5.4 Results

Overall, our results demonstrated that our copywriting interface encouraged participants to generate more, with more diverse inputs, and, as a consequence, make greater use of the generated outputs in their final advertisements. These findings suggest that supporting object-oriented interaction for generation configuration, which is enabled by applying our framework, could support iteration and experimentation with LLMs. However, the benefits of the framework might have been moderated by the difficulties in modifying the individual configuration components. For the statistic analysis of measures, we conducted a Shapiro-Wilk test to determine if the data was parametric (noted with "P") or non-parametric (noted with "NP"). Then, we compared conditions with an independent T-test (if the data was parametric) and a Mann-Whitney U test (if non-parametric).

5.4.1 Generate More and With More Inputs. Our results indicate that treatment participants generated more and with a greater variety of inputs. Participants in the treatment condition generated significantly more times (M=9.78, SD=3.07) than those in the control condition (M=6.33, SD=3.77, p=0.006). Additionally, participants in the treatment condition generated with a significantly higher number of unique inputs (M=5.89, SD=1.66) than those in the control condition (M=3.06, SD=2.17, p<0.000).

As creating multiple cells allowed treatment participants to maintain various alternative inputs, they were less inclined to fixation and more prone to experimentation. We observed in the study that both treatment and control participants dedicated significant time to "set up" their inputs at the start of the task. Before generating for the first time, participants carefully read the product descriptions and thoroughly edited the template input provided. However, after set up, most control participants only made a minimal number of edits to their initial input. For example, P10C (participant 10, Control condition) mentioned, *"I didn't really change the text in the [input] much. I felt that I had set all my desired instructions and didn't think about adding new instructions because I thought it would be enough with [what I had]."* In contrast, treatment participants mentioned that the ability to create multiple cells encouraged them to experiment with different inputs, even if they were not confident that it would yield better results. P7T (Treatment condition) mentioned, *"I set multiple options for each line because I didn't know specifically what I wanted so I could have various sets of options to [experiment with]."* Similarly, P9T mentioned how having multiple cells encouraged him to test more inputs beyond his initial one him: *"[I would] have multiple [cells] and then just toss words in. If you relied too much on your first [set of cells], it wouldn't be efficient."* This comment suggests that supporting the creation of multiple objects in the interface allowed participants to perform parallel prototyping [24].

5.4.2 Barriers to Modifying Inputs and Parameters. Despite treatment participants testing more inputs with multiple *cells*, the possibility of creating multiple *generators* did not significantly increase their experimentation with parameters. We observed no significant difference in the number of unique parameter settings used by treatment participants (M=4.17, SD=2.93) and by control participants (M=3.04, SD=2.50, p=0.936). While some treatment participants mentioned how using multiple generators helped them experiment and *"find the best combo"* of parameters (P1T), the majority felt that it was challenging to test parameters as it was difficult to predict the effect of changes. Specifically, participants mentioned that the function of the parameters was "not explicit" (P6C), and that deciding between continuous values made changes feel "arbitrary" (P9T). This difficulty in predicting and evaluating the effect of parameters is analogous to the *understanding* and *information* barriers in end-user programming [51].

Similarly, although treatment participants did experiment with more inputs, modifying inputs was also not a simple task. While the effect of input changes was easier to predict and distinguish, participants mentioned that deciding on how to change the input could be challenging. For example, P8C mentioned, *"I didn't know what [input] could have done what I wanted to do here."* Unlike model parameters that had well-defined sets of values (e.g., a numerical value within a given range), participants could not think of how to change the inputs. Due to this, several participants expressed how they would want the interface to provide keywords (P2C) or suggest prompts (P12C) on how to change the input. As discussed by Zamfirescu-Pereira et al., this challenge of identifying how to modify or add to LLM inputs (i.e., prompts) resembles the *selection* barriers in end-user programming [51].

5.4.3 Higher Adoption of Generated Outputs. Analysis of participants' generations and their final advertisements revealed that treatment participants made greater use of the LLM's generations in their writing process. The BLEU scores showed that the advertisements of treatment participants were significantly more similar to their generations (M=0.884, SD=0.125) when compared to the similarity between the advertisements and generations of control participants (M=0.768, SD=0.196, p=0.045). This result suggests that, due to how treatment participants generated and experimented more with *cells, generators, and lenses*, they were able to produce outputs that they were more satisfied with and more willing to incorporate into their final writing. The self-BLEU score for the similarity between received generations revealed no significant differences between the generations of treatment participants (M=0.712, SD=0.119) and those of control participants (M=0.680, SD=0.139, p=0.438). This indicates that the two groups of participants saw generations that were similarly diverse. However, considering that treatment participants generated more and thus saw more generated outputs, this could also suggest that treatment participants had a larger pool of different generations available to use—the pool size was bigger but with the similar degree of diversity. Thus, this implies that treatment participants used *cells, generators, and lenses* to explore a wider portion of the generation space.

5.4.4 Subjective Perceptions. According to the survey results (Table 1), both groups indicated positive perceptions regarding the LLM's helpfulness and their final advertisements (i.e., pride and uniqueness)—with no significant differences between the two conditions. For most participants, the study was their first experience interacting with an LLM and they were surprised by the high quality of the results. For example, P1T mentioned how some of the generated outputs were *"perfect"* and P10C described how the model was *"really good"* compared to other models they had tested before. The novelty of the model and its performance led to positive ratings from most participants.

The survey results also revealed that treatment participants rated the task to be relatively easy (i.e., mean rating close to 6) and had no significant difference with control participants ("Ease" in Tab. 1). Considering that treatment participants generated more, tested more inputs, and also had to interact with more cells, generators and lenses, these results suggest that our framework might have reduced effort in other sub-tasks (e.g., remembering previous inputs). This finding is further supported by considering how, as treatment participants generated more, they also had to read through more outputs, which was effortful and time-consuming. Although participants did use the various lenses to browse through the outputs, they ultimately resorted to using the list lens to read each generation to ensure that they did not overlook any promising outputs. P11T said, *"At some point, I generated 10 or more sentences and [...] it took a long time to read through them."* Also, P7T noted how it would be useful

	Helpful	Ease	Experiment	Iterate	Proud	Unique
Treatment	5.889 (0.936)	5.944 (0.911)	4.778 (1.133)	5.222 (1.356)	5.722 (0.650)	4.500 (1.118)
Control	6.222 (0.711)	6.000 (0.667)	5.333 (1.155)	5.111 (1.197)	5.278 (1.044)	4.056 (1.353)
***p*-value**	0.307	0.960	0.082	0.755	0.195	0.304

Table 1: Mean and standard deviation (in parentheses) of participants' subjective ratings across conditions. Ratings showed no significant differences between the perceptions of treatment and control participants.

if the lenses surfaces specific aspects of the outputs that could be improved. Considering how all of the participants checked outputs for desirable characteristics (e.g., keywords, creative staring lines), these results indicate a need for lenses that can efficiently represent different user-desired characteristics.

Surprisingly, despite quantitative measures indicating that treatment participants generated more and with more unique inputs, there was no significant difference in their perceptions on their amount of experimentation and iteration. Several participants in both conditions mentioned how the generative model would frequently return similar outputs. For example, P17T mentioned that *"most of the time the AI was generating similar text"*. Due to the aforementioned barriers in modifying inputs and model parameters, participants were frequently unable to control the model to generate more diverse outputs. Considering that participants in the treatment condition generated more frequently and experimented with more inputs, it is possible that they developed higher expectations that the model would return diverse outputs. However, as the model did not fully match their expectations, this resulted in them perceiving that they experimented less than they actually did.

6 DESIGN WORKSHOP

Finally, we evaluate our framework in terms of *usability*: can interface designers, beyond ourselves, effectively use and apply our framework to design writing interfaces that support end-users' iteration and experimentation? For this purpose and gain expert critiques about our framework, we conducted a workshop where we invited interface designers to follow our framework and design writing interfaces that support object-oriented interaction.

6.1 Participants

We focused our recruitment on graduate students in the field of HCI with prior experience designing interfaces and using LLMs for writing. Through recruitment posts in online communities of a technical university, we recruited three designers (1 female, 2 male, age M=27.7 and SD=4.7). All participants were graduate students in industrial design (2 M.S., 1 Ph.D.) and currently conducting research in HCI. The participants provided samples of previous designs, which included interfaces for ride-sharing, social communities, and journaling. Participants' previous experiences with LLM ranged from use for composing emails or translating to actually developing LLM-powered interfaces.

6.1.1 Study Procedure. Participants were first provided with an introduction to the workshop, including a brief reminder about LLMs and an explanation of our design framework—i.e., its motivation, the interactive objects and their possible interactions, and the three interfaces we designed. After the introduction, participants were asked to choose the AI-powered writing interface that they would re-design. Participants were provided with three interface options, where each supported a different writing task and writing process: poetry [90], screenplay [66], or essay [20]. Then, we provided participants with a link to a Figma[7] document that contained screenshots of the interface to re-design, a summarized explanation of our design framework, and pre-made design components for cells, generators, and lenses. Participants were then asked to design a new interface by adapting the basic workflow and features supported in the chosen reference interface, but by applying our framework to integrate *cells, generators, and lenses*. We decided on this type of re-design task as our goal was to see if designers could apply our framework to incorporate the objects into interface designs, instead of observing whether they could ideate new designs from scratch. After designing for one hour, we concluded with a group interview where each participant described their design, how they used the framework, and their experiences of applying the framework.

6.2 Findings

Overall, designers could apply our framework to design LLM-powered writing interfaces with the goal of supporting end-users' iteration and experimentation. We observed that our framework not only bootstrapped participants' design process by supplying the interactive objects as design materials, but also encouraged them to consider how they could further modularize their envisioned end-users' generative process to support iteration and experimentation. In this section, we describe how each participant integrated and designed each of our framework's objects, and the reasoning behind their designs. Finally, we present participants' overall perception of our framework. We denote the designers as DP (designer for the poetry writing interface), DS (for screenplay), and DE (for essay).

6.2.1 Cells. Designers integrated cells into their designs in diverse ways to support iteration and experimentation. Specifically, all designers envisioned interfaces where end-users could create multiple versions of inputs and assemble them differently each time they generated. Designers reflected that this would help end-users to *"create diverse variations and compare the generated results"* (DS). Beyond this, designers also envisioned novel ways for desinging or using cells. For example, DP envisioned that the LLM could help users *"think of inputs"* by brainstorming and generating them for the end-users to use as cells. Additionally, DS designed an interaction where end-users could select individual sentences from cells to use the selection as temporary cells to help end-users *"experiment*

[7] https://www.figma.com/

with diverse inputs at the sentence-level". Finally, DE mentioned that, in essay writing, *"what content to include is more defined and that, instead, the writer needs help to assemble this content."* Thus, his interface *"extended cells"* to contain drafts of paragraphs, rather than shorter text-like phrases.

6.2.2 Generators. In terms of generators, all participants produced designs that supported multiple generators with their differences in how they envisioned that generators would be linked to cells. DS's design resembled our storywriting interface where end-users link individual cells to generators, while DE's design resembled our email composing interface where end-users create multiple generators and all of them are used when generating. Unlike our designs, DP envisioned that end-users would chain configurations [102], where generated outputs from one configuration step would used as inputs for the next. Thus, as he expected that *"[the effect of each step] would be affected by each generator's configurations"*, he designed the interface to allow end-users to create multiple generators at each generation step.

6.2.3 Lenses. All of the participants produced designs where lenses were assembled to provide end-users with various views of the output. Both DP and DS designed interfaces that enabled end-users to use multiple assembled lenses to *"explore"* (DP) and *"evaluate"* (DS) generated outputs. To provide further control, DE's essay writing interface was envisioned to provide two views for generated outputs: a draft view, which displayed text, and an analysis view, where end-users could customize how they analysed generations by *"adding [lenses] that they prefer"* from a set of pre-defined lenses.

6.2.4 Framework as Design Material and Inspiration. As illustrated by their application of cells, generators, and lenses, designers were able to apply our framework to design LLM-powered writing interfaces with the intention of supporting iteration and experimentation. Regarding the framework, designers noted how them, by thinking in terms of the framework, they were able to *"determine how [they could] facilitate end-users use of LLMs in the interfaces they design"* (DS). Particularly, participants felt that the cells and generators were *"concrete"* and *"detailed"* (DE)—serving as actionable design materials. However, perceptions regarding the lenses were mixed. Both DS and DE mentioned that *"it was not clear how the [different] lenses could be used [...] and where they could be used"* (DS) due to how there is greater *"freedom"* on how they can be designed (DE). Thus, they both mentioned that they required more time to think about how to incorporate lenses into their designs. In contrast, DP mentioned how *"how having less concrete guidance on how to design lenses granted designers with more flexibility"* about what type of views to support and how. These comments indicate that, while more concrete examples for lenses should be incorporated into the framework, a more abstract description can allow designers to adapt lenses for their intended tasks.

Beyond facilitating the integration of cells, generators, and lenses into their designs, participants noted how the framework inspired them to consider how they could further modularize the generative process in their interfaces. DE mentioned how the framework *"inspired"* him to design his essay writing interface to enable end-users to maintain multiple versions of their input and generated drafts as individual "tabs". In his poem writing interface, DP incorporated an intermittent step through which end-users test their cells and generators by generating partial outputs in a sandbox—helping them identify fruitful configurations before proceeding to generate the full poem. Similarly, while the existing screenplay writing interface had end-users sequentially generate elements of a screenplay (e.g., title, characters, setting), DS modularized the generation of these elements so her end-users could iterate and experiment on each of these elements individually. Thus, these findings suggest that, by encouraging an object-oriented perspective, our framework could encourage designers to envision further affordances and interactions that support iteration and experimentation with LLMs.

7 DISCUSSION

In this work, we present *cells, generators, and lenses*, a design framework for supporting object-oriented interaction with LLMs. We propose that designers can integrate object-oriented interaction in their interfaces to mitigate end-users' challenges in interacting with black box and non-deterministic LLMs, and support iteration and experimentation during writing.

We comprehensively evaluate our framework according to three dimensions: *generalizability*, *effectiveness*, and *usability*. For generalizability, we applied our framework to re-design three existing writing interfaces to illustrate how the framework can be applied in diverse tasks but also how its application needs to be adapted for each task. For effectiveness, we conducted a comparative lab study and observed that end-users could use the interactive objects to create, combine, and compare diverse generation configurations. This in turn encouraged them to generate more, experiment more, and make greater use of generations in their writing—suggesting higher satisfaction with the generations they produced. Finally, for usability, a workshop with designers revealed that our framework bootstrapped the design process by providing concrete means through which designers could facilitate end-users' configuration of LLMs. Furthermore, the framework inspired designers to consider how they could further support iteration and experimentation in their designs, beyond incorporating cells, generators, and lenses.

7.1 Generalizability of the Framework

Instead of contributing one interface that implements cells, generators, and lenses, we provide a general framework to enable designers to integrate these objects in diverse interfaces and tailor them according to the specific writing task. We showcased this generalizability through the three interfaces we designed and the three interfaces that participants produced in the designer workshop. These designs encompassed the diversity of writing tasks across various dimensions: short to long artifacts, goal-focused to open-ended, phased vs dynamic writing processes. As applying our framework can produce interfaces that help end-users to experiment with LLM configurations and sense-make on their behavior, we believe that our framework can be beneficial to most writing tasks where LLMs are beneficial. However, we also observed that our framework can be most valuable for open-ended writing tasks as the experimentation with LLM configurations has the added

benefit of helping end-users explore the design space of text artifacts. Thus, applying the framework to writing interfaces for more open-ended tasks can provide end-users with dual benefits.

Although our work focused on LLMs and writing tasks, we found that various needs and challenges about LLMs were shared with other generative models. Future work could investigate how to extend the concept of object-oriented interaction to a wider range of generative models, tasks, and types of artifacts. For example, cells can represent text keywords for text-to-image (TTI) models [59], image examples for image-to-image (ITI) models [79], or music bars for music generation [28, 61]. Additionally, by modularizing model types into generators, one interface can seamlessly interweave multiple classes of generative models—a necessity in more complex creative tasks (e.g., songwriting [41, 94] and storyboarding [81, 113]). Finally, interfaces should provide a variety of lenses that are designed specifically for the type of media that is generated by the models (e.g., gallery for images, sequence visualizations for music).

7.2 Cells, Generators, and Lenses as Design Materials

In our workshop, designers mentioned how the cells, generators, and lenses served as design materials to help them construct interfaces for LLMs. Beyond guiding the design of interfaces, we also present an open-source ReactJS library with the goal of facilitating the development of these interfaces and widen the adoption of our framework. We hope that, with this package, developers can readily build or integrate these components into LLM-powered writing interfaces to support object-oriented interaction. Beyond scaffolding designers and developers, we can also envision a future interface that enable end-users, themselves, to construct writing interfaces through cells, generators, and lenses. By circumventing the need for designers or developers, this could enable end-users to personalize writing interfaces to their own specific needs and challenges.

7.3 Potential of Object-Orientation: Analyze and Extend

Beyond supporting iteration and experimentation, we believe that the abstraction of generation components into objects can have further implications for the design of LLM-based interfaces. Specifically, we believe that *cells, generators, and lenses* can be used as an analytical framework to examine existing LLM-powered interfaces by identifying their differences in how they design UI components for each generation component, and distilling high-level design themes or patterns. Furthermore, by encouraging designers to view LLM-powered interfaces in terms of objects, our framework can motivate designers to transition from point solutions to more extensible interfaces. If interfaces are modularized into objects (and shared as open-source), designers can readily adopt and integrate object designs from other interfaces or, like inheritance in object-oriented programming, extend and improve on existing object designs. Designers can also grant this customizability and extensibility directly to end-users. Instead of deciding on what objects to incorporate in their interfaces, future designers can create flexible and customizable interfaces that provide end-users with a collection of modular objects. Then, end-users themselves could select, combine, and arrange these objects into personalized writing environments [50]. As illustrated, we believe our work reveals new possibilities for LLM-based interfaces and that future work can further explore this potential of object-based abstraction for LLMs.

7.4 Further Development of the Framework

While we observed the benefits of cells, generators, and lenses, the design of these objects can be further refined.

7.4.1 Cells: Suggesting Augmentations. During our study, participants often struggled to identify how to modify cells as they did not know what language they could use. To help users take more advantage of cells, future designers can take inspiration from data augmentation techniques [26] to automatically suggest various input alternatives that end-users can use and combine. Further, future work could also explore how to leverage end-users' input iterations as organic data to train an augmentation suggestion model—similar to how human feedback has been used to finetune LLMs [70, 72].

7.4.2 Generators: Explainability of Generative Model Parameters. In our study, participants felt that chanigng model parameters was easy, but predicting the effect of changes was challenging. As these parameters were designed based on the technical generation process of LLMs, they fail to be user-centric as their function and values can mismatch with users' mental models. For example, it is unclear what decreasing "temperature" by 0.1 would do. Future research could investigate and explore the design space of model parameters that coincide with human mental models and, thus, are more user-centric [16, 42, 93, 115].

7.4.3 Lenses: Balancing Integrity and Efficiency. While study participants noted how lenses afforded different ways to explore generations, all of them would eventually default to using the list lens to read every output to check if they possessed their desired characteristics. However, this meant that significant time was dedicated to reading and participants had fewer opportunities to explore more outputs. These findings suggest that lenses should balance integrity and efficiency: accurately represent user-desired characteristics in outputs, but also minimize the effort needed to check them. Future work could investigate this trade-off to explore the design space and identify effective designs for lenses.

7.5 Limitations

Our work has several limitations which we address in this section. First, in our user study, we did not evaluate the quality of participants' final writing as we focused on measuring iteration and experimentation, and measuring quality can be subjective and task dependent. Future work should investigate the effect of our framework on writing quality in specific tasks. Second, our workshop task involved re-designing existing interfaces as we focused on investigating whether designers could apply our framework when they know what task and process they want to support, However, future work is needed to investigate how our framework influences designers' processes when they are in the ideation stage. Fianlly, as our work focuses on guiding designers, we did not investigate how our framework can influence the development stage of interfaces. We release our component library as open-source and future work could investigate how developers apply this library.

8 CONCLUSION

This work introduces a design framework for object-oriented interaction of large language models. This framework reifies the input, model, and output components of generation configurations into interactive objects: *cells, generators, and lenses*. To portray how these objects be incorporated into interfaces to support various writing tasks, we apply the framework to design three writing-support systems for copywriting, email composing, and story writing. Then, through a comparative lab study, we evaluated the effect of applying our framework and observed that an interface designed with our framework could encourage more experimentation and greater use of generated outputs in writing. Finally, a design workshop revealed that designers could successfully follow the framework to construct interfaces that support iteration and experimentation for various writing tasks. Beyond these benefits, we believe that the framework's modularization of configuration components offers a flexible and extensible way to design and develop generative interfaces—enabling developers to easily share new cells, generators, and lenses and the rapid creation of new interfaces by combining these.

ACKNOWLEDGMENTS

This work was supported by KAIST-NAVER Hypercreative AI Center. The authors would like to thank our participants for their positive engagement during the studies. Additionally, we thank the reviewers as their feedback helped us improve our paper.

REFERENCES

[1] AI21. 2021. *AI21 Studio.* Retrieved March 28, 2022 from https://www.ai21.com/studio

[2] Emre Aksan, Fabrizio Pece, and Otmar Hilliges. 2018. DeepWriting: Making Digital Ink Editable via Deep Generative Modeling. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3173574.3173779

[3] Saleema Amershi, Max Chickering, Steven Drucker, Bongshin Lee, Patrice Simard, and Jina Suh. 2015. ModelTracker: Redesigning Performance Analysis Tools for Machine Learning. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI 2015)* (proceedings of the conference on human factors in computing systems (chi 2015) ed.). ACM - Association for Computing Machinery. https://www.microsoft.com/en-us/research/publication/modeltracker-redesigning-performance-analysis-tools-for-machine-learning/

[4] Fraser Anderson, Tovi Grossman, and George Fitzmaurice. 2017. Trigger-Action-Circuits: Leveraging Generative Design to Enable Novices to Design and Build Circuitry. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 331–342. https://doi.org/10.1145/3126594.3126637

[5] Tal August, Lucy Lu Wang, Jonathan Bragg, Marti A Hearst, Andrew Head, and Kyle Lo. 2022. Paper plain: Making medical research papers approachable to healthcare consumers with natural language processing. *arXiv preprint arXiv:2203.00130* (2022).

[6] David Bau, Hendrik Strobelt, William Peebles, Jonas Wulff, Bolei Zhou, Jun-Yan Zhu, and Antonio Torralba. 2019. Semantic Photo Manipulation with a Generative Image Prior. *ACM Trans. Graph.* 38, 4, Article 59 (jul 2019), 11 pages. https://doi.org/10.1145/3306346.3323023

[7] Michel Beaudouin-Lafon. 2000. Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (The Hague, The Netherlands) *(CHI '00)*. Association for Computing Machinery, New York, NY, USA, 446–453. https://doi.org/10.1145/332040.332473

[8] Michel Beaudouin-Lafon and Wendy E. Mackay. 2000. Reification, Polymorphism and Reuse: Three Principles for Designing Visual Interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (Palermo, Italy) *(AVI '00)*. Association for Computing Machinery, New York, NY, USA, 102–109. https://doi.org/10.1145/345513.345267

[9] Lera Boroditsky. 2007. Comparison and the development of knowledge. *Cognition* 102, 1 (2007), 118–128.

[10] Matthew Brehmer, Robert Kosara, and Carmen Hull. 2022. Generative Design Inspiration for Glyphs with Diatoms. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2022), 389–399. https://doi.org/10.1109/TVCG.2021.3114792

[11] Daniel Buschek, Martin Zürn, and Malin Eiband. 2021. The Impact of Multiple Parallel Phrase Suggestions on Email Input and Composition Behaviour of Native and Non-Native English Writers. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 732, 13 pages. https://doi.org/10.1145/3411764.3445372

[12] Alex Calderwood, Vivian Qiu, Katy Ilonka Gero, and Lydia B Chilton. 2020. How Novelists Use Generative Language Models: An Exploratory User Study.. In *HAI-GEN Workshop at IUI 2020*.

[13] Harrison Chase. 2023. *Welcome to LangChain — LangChain 0.0.132.* Retrieved April 5, 2023 from https://python.langchain.com/en/latest/index.html

[14] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. PaLM: Scaling Language Modeling with Pathways. *arXiv preprint arXiv:2204.02311* (2022).

[15] John Joon Young Chung, Minsuk Chang, and Eytan Adar. 2022. Gestural Inputs as Control Interaction for Generative Human-AI Co-Creation. (2022). https://hai-gen.github.io/2022/papers/paper-HAIGEN-ChungJohn.pdf

[16] John Joon Young Chung, Wooseok Kim, Kang Min Yoo, Hwaran Lee, Eytan Adar, and Minsuk Chang. 2022. TaleBrush: Sketching Stories with Generative Pretrained Language Models. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*. Association for Computing Machinery, New York, NY, USA.

[17] Marianela Ciolfi Felice, Nolwenn Maudet, Wendy E. Mackay, and Michel Beaudouin-Lafon. 2016. Beyond Snapping: Persistent, Tweakable Alignment and Distribution with StickyLines. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 133–144. https://doi.org/10.1145/2984511.2984577

[18] Elizabeth Clark, Anne Spencer Ross, Chenhao Tan, Yangfeng Ji, and Noah A. Smith. 2018. Creative Writing with a Machine in the Loop: Case Studies on Slogans and Stories. In *23rd International Conference on Intelligent User Interfaces* (Tokyo, Japan) *(IUI '18)*. Association for Computing Machinery, New York, NY, USA, 329–340. https://doi.org/10.1145/3172944.3172983

[19] Nigel Cross. 1982. Designerly ways of knowing. *Design studies* 3, 4 (1982), 221–227.

[20] Hai Dang, Karim Benharrak, Florian Lehmann, and Daniel Buschek. 2022. Beyond Text Generation: Supporting Writers with Continuous Automatic Text Summaries. *arXiv preprint arXiv:2208.09323* (2022).

[21] Hai Dang, Sven Goller, Florian Lehmann, and Daniel Buschek. 2023. Choice Over Control: How Users Write with Large Language Models using Diegetic and Non-Diegetic Prompting. *arXiv preprint arXiv:2303.03199* (2023).

[22] Hai Dang, Lukas Mecke, and Daniel Buschek. 2022. GANSlider: How Users Control Generative Models for Images using Multiple Sliders with and without Feedforward Information. *CoRR* abs/2202.00965 (2022). arXiv:2202.00965 https://arxiv.org/abs/2202.00965

[23] Nicholas Davis, Chih-PIn Hsiao, Kunwar Yashraj Singh, Lisa Li, and Brian Magerko. 2016. Empirically Studying Participatory Sense-Making in Abstract Drawing with a Co-Creative Cognitive Agent. In *Proceedings of the 21st International Conference on Intelligent User Interfaces* (Sonoma, California, USA) *(IUI '16)*. Association for Computing Machinery, New York, NY, USA, 196–207. https://doi.org/10.1145/2856767.2856795

[24] Steven P Dow, Alana Glassco, Jonathan Kass, Melissa Schwarz, Daniel L Schwartz, and Scott R Klemmer. 2010. Parallel prototyping leads to better design results, more divergence, and increased self-efficacy. *ACM Transactions on Computer-Human Interaction (TOCHI)* 17, 4 (2010), 1–24.

[25] Wanyu Du, Zae Myung Kim, Vipul Raheja, Dhruv Kumar, and Dongyeop Kang. 2022. Read, revise, repeat: A system demonstration for human-in-the-loop iterative text revision. *arXiv preprint arXiv:2204.03685* (2022).

[26] Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for NLP. *arXiv preprint arXiv:2105.03075* (2021).

[27] Flowrite. 2022. *Flowrite - Supercharge your daily communication.* Retrieved April 4, 2022 from https://www.flowrite.com/

[28] Emma Frid, Celso Gomes, and Zeyu Jin. 2020. *Music Creation by Example.* Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376514

[29] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*.

[30] Dedre Gentner and Arthur B Markman. 1997. Structure mapping in analogy and similarity. *American psychologist* 52, 1 (1997), 45.

[31] Katy Ilonka Gero and Lydia B. Chilton. 2019. Metaphoria: An Algorithmic Companion for Metaphor Creation. In *Proceedings of the 2019 CHI Conference*

on Human Factors in Computing Systems (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300526

[32] Katy Ilonka Gero, Vivian Liu, and Lydia Chilton. 2022. Sparks: Inspiration for science writing using language models. In *Designing Interactive Systems Conference*. 1002–1019.

[33] Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. 2017. Hafez: an interactive poetry generation system. In *Proceedings of ACL 2017, System Demonstrations*. 43–48.

[34] Seraphina Goldfarb-Tarrant, Haining Feng, and Nanyun Peng. 2019. Plan, Write, and Revise: an Interactive System for Open-Domain Story Generation. *CoRR* abs/1904.02357 (2019). arXiv:1904.02357 http://arxiv.org/abs/1904.02357

[35] Imke Grabe, Miguel González-Duque, Sebastian Risi, and Jichen Zhu. 2022. Towards A Framework for Human-AI Interaction Patterns in Co-Creative GAN Applications. (2022). https://hai-gen.github.io/2022/papers/paper-HAIGEN-GrabeImke.pdf

[36] Maarten Grootendorst. 2020. KeyBERT: Minimal keyword extraction with BERT. https://doi.org/10.5281/zenodo.4461265

[37] Matthew Guzdial, Nicholas Liao, Jonathan Chen, Shao-Yu Chen, Shukan Shah, Vishwa Shah, Joshua Reno, Gillian Smith, and Mark O. Riedl. 2019. Friend, Collaborator, Student, Manager: How Design of an AI-Driven Game Level Editor Affects Creators. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300854

[38] Matthew Guzdial and Mark Riedl. 2019. An Interaction Framework for Studying Co-Creative AI. *CoRR* abs/1903.09709 (2019). arXiv:1903.09709 http://arxiv.org/abs/1903.09709

[39] Han L. Han, Junhang Yu, Raphael Bournet, Alexandre Ciorascu, Wendy E. Mackay, and Michel Beaudouin-Lafon. 2022. Passages: Interacting with Text Across Documents. 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 338, 17 pages. https://doi.org/10.1145/3491102.3502052

[40] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. 2020. Ganspace: Discovering interpretable gan controls. *Advances in Neural Information Processing Systems* 33 (2020), 9841–9850.

[41] Cheng-Zhi Anna Huang, Hendrik Vincent Koops, Ed Newton-Rex, Monica Dinculescu, and Carrie J. Cai. 2020. AI Song Contest: Human-AI Co-Creation in Songwriting. *CoRR* abs/2010.05388 (2020). arXiv:2010.05388 https://arxiv.org/abs/2010.05388

[42] Lianghua Huang, Di Chen, Yu Liu, Yujun Shen, Deli Zhao, and Jingren Zhou. 2023. Composer: Creative and controllable image synthesis with composable conditions. *arXiv preprint arXiv:2302.09778* (2023).

[43] Human++ Inc. 2022. *Sudowrite*. Retrieved April 2, 2022 from https://www.sudowrite.com/

[44] David G Jansson and Steven M Smith. 1991. Design fixation. *Design studies* 12, 1 (1991), 3–11.

[45] Jasper.ai. 2022. *Jasper (Formerly Jarvis) - #1 AI Writing Assistant*. Retrieved April 2, 2022 from https://www.jasper.ai/

[46] Ellen Jiang, Kristen Olson, Edwin Toh, Alejandra Molina, Aaron Donsbach, Michael Terry, and Carrie J Cai. 2022. PromptMaker: Prompt-based Prototyping with Large Language Models. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*. 1–8.

[47] Ellen Jiang, Edwin Toh, Alejandra Molina, Aaron Donsbach, Carrie J Cai, and Michael Terry. 2021. GenLine and GenForm: Two Tools for Interacting with Generative Language Models in a Code Editor. In *The Adjunct Publication of the 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 145–147. https://doi.org/10.1145/3474349.3480209

[48] Pegah Karimi, Jeba Rezwana, Safat Siddiqui, Mary Lou Maher, and Nasrin Dehbozorgi. 2020. Creative Sketching Partner: An Analysis of Human-AI Co-Creativity. In *Proceedings of the 25th International Conference on Intelligent User Interfaces* (Cagliari, Italy) *(IUI '20)*. Association for Computing Machinery, New York, NY, USA, 221–230. https://doi.org/10.1145/3377325.3377522

[49] Tae Soo Kim, DaEun Choi, Yoonseo Choi, and Juho Kim. 2022. Stylette: Styling the Web with Natural Language. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 1–17.

[50] Tae Soo Kim, Arghya Sarkar, Yoonjoo Lee, Minsuk Chang, and Juho Kim. 2023. LMCanvas: Object-Oriented Interaction to Personalize Large Language Model-Powered Writing Environments. arXiv:2303.15125 [cs.HC]

[51] Amy J Ko, Brad A Myers, and Htet Htet Aung. 2004. Six learning barriers in end-user programming systems. In *2004 IEEE Symposium on Visual Languages-Human Centric Computing*. IEEE, 199–206.

[52] AI21 Labs. 2021. *Wordtune*. Retrieved April 2, 2022 from https://www.wordtune.com/hero

[53] Mina Lee, Percy Liang, and Qian Yang. 2022. CoAuthor: Designing a Human-AI Collaborative Writing Dataset for Exploring Language Model Capabilities. *CoRR* abs/2201.06796 (2022). arXiv:2201.06796 https://arxiv.org/abs/2201.06796

[54] Mina Lee, Megha Srivastava, Amelia Hardy, John Thickstun, Esin Durmus, Ashwin Paranjape, Ines Gerard-Ursin, Xiang Lisa Li, Faisal Ladhak, Frieda Rong, et al. 2022. Evaluating Human-Language Model Interaction. *arXiv preprint arXiv:2212.09746* (2022).

[55] Yoonjoo Lee, John Joon Young Chung, Tae Soo Kim, Jean Y Song, and Juho Kim. 2022. Promptiverse: scalable generation of scaffolding prompts through human-AI hybrid knowledge graph annotation. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 1–18.

[56] Yoonjoo Lee, Tae Soo Kim, Minsuk Chang, and Juho Kim. 2022. Interactive Children's Story Rewriting Through Parent-Children Interaction. In *Proceedings of the First Workshop on Intelligent and Interactive Writing Assistants (In2Writing 2022)*. 62–71.

[57] Yoonjoo Lee, Tae Soo Kim, Sungdong Kim, Yohan Yun, and Juho Kim. 2023. DAPIE: Interactive Step-by-Step Explanatory Dialogues to Answer Children's Why and How Qestions. (2023).

[58] Xingjun Li, Yuanxin Wang, Hong Wang, Yang Wang, and Jian Zhao. 2021. NBSearch: Semantic Search and Visual Exploration of Computational Notebooks. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–14.

[59] Vivian Liu, Han Qiao, and Lydia Chilton. 2022. Opal: Multimodal Image Generation for News Illustration. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–17.

[60] Vivian Liu, Jo Vermeulen, George Fitzmaurice, and Justin Matejka. 2022. 3DALL-E: Integrating Text-to-Image AI in 3D Design Workflows. *arXiv preprint arXiv:2210.11603* (2022).

[61] Ryan Louie, Andy Coenen, Cheng Zhi Huang, Michael Terry, and Carrie J. Cai. 2020. *Novice-AI Music Co-Creation via AI-Steering Tools for Deep Generative Models*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376739

[62] Ryan Louie, Jesse Engel, and Cheng-Zhi Anna Huang. 2022. Expressive Communication: Evaluating Developments in Generative Models and Steering Interfaces for Music Creation. In *27th International Conference on Intelligent User Interfaces* (Helsinki, Finland) *(IUI '22)*. Association for Computing Machinery, New York, NY, USA, 405–417. https://doi.org/10.1145/3490099.3511159

[63] Julia M Markel, Steven G Opferman, James A Landay, and Chris Piech. 2023. GPTeach: Interactive TA Training with GPT-based Students. https://doi.org/10.1145/3573051.3593393

[64] Justin Matejka, Michael Glueck, Erin Bradner, Ali Hashemi, Tovi Grossman, and George Fitzmaurice. 2018. *Dream Lens: Exploration and Visualization of Large-Scale Generative Design Datasets*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173943

[65] Jon McCormack, Toby Gifford, Patrick Hutchings, Maria Teresa Llano Rodriguez, Matthew Yee-King, and Mark d'Inverno. 2019. In a Silent Way: Communication Between AI and Improvising Musicians Beyond Sound. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3290605.3300268

[66] Piotr Mirowski, Kory W Mathewson, Jaylen Pittman, and Richard Evans. 2022. Co-writing screenplays and theatre scripts with language models: An evaluation by industry professionals. *arXiv preprint arXiv:2209.14958* (2022).

[67] Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 task 1: Affect in tweets. In *Proceedings of the 12th international workshop on semantic evaluation*. 1–17.

[68] Changhoon Oh, Jungwoo Song, Jinhan Choi, Seonghyeon Kim, Sungwoo Lee, and Bongwon Suh. 2018. I Lead, You Help but Only with Enough Details: Understanding User Experience of Co-Creation with Artificial Intelligence. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3174223

[69] OpenAI. 2021. *Playground - OpenAI API*. Retrieved March 28, 2022 from https://beta.openai.com/playground

[70] OpenAI. 2022. *Introducing ChatGPT*. Retrieved March 28, 2023 from https://openai.com/blog/chatgpt

[71] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[72] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* 35 (2022), 27730–27744.

[73] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 311–318.

[74] Joon Sung Park, Lindsay Popowski, Carrie Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2022. Social Simulacra: Creating Populated Prototypes for Social Computing Systems. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–18.

[75] Savvas Petridis, Nicholas Diakopoulos, Kevin Crowston, Mark Hansen, Keren Henderson, Stan Jastrzebski, Jeffrey V Nickerson, and Lydia B Chilton. 2023. AngleKindling: Supporting Journalistic Angle Ideation with Large Language

Models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 225, 16 pages. https://doi.org/10.1145/3544548.3580907

[76] Savvas Petridis, Michael Terry, and Carrie J Cai. 2023. PromptInfuser: Bringing User Interface Mock-ups to Life with Large Language Models. (2023).

[77] Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)*. 502–518.

[78] Andrew Ross, Nina Chen, Elisa Zhao Hang, Elena L. Glassman, and Finale Doshi-Velez. 2021. Evaluating the Interpretability of Generative Models by Interactive Reconstruction. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 80, 15 pages. https://doi.org/10.1145/3411764.3445296

[79] Mattias Rost and Sebastian Andreasson. 2023. Stable Walk: An interactive environment for exploring Stable Diffusion outputs. (2023).

[80] Mike Sharples. 1996. An account of writing as creative design. *The science of writing* (1996), 127–148.

[81] Yang Shi, Nan Cao, Xiaojuan Ma, Siji Chen, and Pei Liu. 2020. *EmoG: Supporting the Sketching of Emotional Expressions for Storyboarding*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376520

[82] Nikhil Singh, Guillermo Bernal, Daria Savchenko, and Elena L. Glassman. 2022. Where to Hide a Stolen Elephant: Leaps in Creative Writing with Multimodal Machine Intelligence. *ACM Trans. Comput.-Hum. Interact.* (feb 2022). https://doi.org/10.1145/3511599 Just Accepted.

[83] 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. https://doi.org/10.48550/ARXIV.2208.07852

[84] Jiao Sun, Q. Vera Liao, Michael Muller, Mayank Agarwal, Stephanie Houde, Kartik Talamadupula, and Justin D. Weisz. 2022. Investigating Explainability of Generative AI for Code through Scenario-Based Design. In *27th International Conference on Intelligent User Interfaces* (Helsinki, Finland) *(IUI '22)*. Association for Computing Machinery, New York, NY, USA, 212–228. https://doi.org/10.1145/3490099.3511119

[85] Simeng Sun, Wenlong Zhao, Varun Manjunatha, Rajiv Jain, Vlad I. Morariu, Franck Dernoncourt, Balaji Vasan Srinivasan, and Mohit Iyyer. 2021. IGA: An Intent-Guided Authoring Assistant. *CoRR* abs/2104.07000 (2021). arXiv:2104.07000 https://arxiv.org/abs/2104.07000

[86] Harini Suresh, Kathleen M Lewis, John Guttag, and Arvind Satyanarayan. 2022. Intuitively Assessing ML Model Reliability through Example-Based Explanations and Editing Model Inputs. In *27th International Conference on Intelligent User Interfaces* (Helsinki, Finland) *(IUI '22)*. Association for Computing Machinery, New York, NY, USA, 767–781. https://doi.org/10.1145/3490099.3511160

[87] Ben Swanson, Kory Mathewson, Ben Pietrzak, Sherol Chen, and Monica Dinalescu. 2021. Story Centaur: Large Language Model Few Shot Learning as a Creative Writing Tool. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Online, 244–256. https://doi.org/10.18653/v1/2021.eacl-demos.29

[88] Amanda Swearngin, Chenglong Wang, Alannah Oleson, James Fogarty, and Amy J. Ko. 2020. *Scout: Rapid Exploration of Interface Layout Alternatives through High-Level Design Constraints*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376593

[89] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[90] David Uthus, Maria Voitovich, and RJ Mical. 2021. Augmenting poetry composition with verse by verse. *arXiv preprint arXiv:2103.17205* (2021).

[91] Thiemo Wambsganss, Christina Niklaus, Matthias Cetto, Matthias Söllner, Siegfried Handschuh, and Jan Marco Leimeister. 2020. AL: An adaptive learning support system for argumentation skills. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–14.

[92] Bryan Wang, Gang Li, and Yang Li. 2022. Enabling Conversational Interaction with Mobile UI using Large Language Models. *arXiv preprint arXiv:2209.08655* (2022).

[93] Yunlong Wang, Shuyuan Shen, and Brian Y Lim. 2023. RePrompt: Automatic Prompt Editing to Refine AI-Generative Art Towards Precise Expressions. *arXiv preprint arXiv:2302.09466* (2023).

[94] Kento Watanabe, Yuichiroh Matsubayashi, Kentaro Inui, Tomoyasu Nakano, Satoru Fukayama, and Masataka Goto. 2017. LyriSys: An Interactive Support System for Writing Lyrics Based on Topic Transition. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces* (Limassol, Cyprus) *(IUI '17)*. Association for Computing Machinery, New York, NY, USA, 559–563. https://doi.org/10.1145/3025171.3025194

[95] Ariel Weingarten, Ben Lafreniere, George Fitzmaurice, and Tovi Grossman. 2019. DreamRooms: Prototyping Rooms in Collaboration with a Generative Process. In *Proceedings of Graphics Interface 2019* (Kingston, Ontario) *(GI 2019)*. Canadian Information Processing Society, 9 pages. https://doi.org/10.20380/GI2019.19

[96] Nathaniel Weinman, Steven M Drucker, Titus Barik, and Robert DeLine. 2021. Fork It: Supporting stateful alternatives in computational notebooks. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–12.

[97] Justin D Weisz, Michael Muller, Jessica He, and Stephanie Houde. 2023. Toward General Design Principles for Generative AI Applications. *arXiv preprint arXiv:2301.05578* (2023).

[98] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *CoRR* abs/1910.03771 (2019). arXiv:1910.03771 http://arxiv.org/abs/1910.03771

[99] Writesonic. 2022. *Writesonic - AI Writer, AI Copywriter & Writing Assistant*. Retrieved April 2, 2022 from https://writesonic.com/

[100] Tongshuang Wu, Ellen Jiang, Aaron Donsbach, Jeff Gray, Alejandra Molina, Michael Terry, and Carrie J Cai. 2022. PromptChainer: Chaining Large Language Model Prompts through Visual Programming. *arXiv preprint arXiv:2203.06566* (2022).

[101] Tongshuang Wu, Hua Shen, Daniel S Weld, Jeffrey Heer, and Marco Tulio Ribeiro. 2023. ScatterShot: Interactive In-context Example Curation for Text Transformation. *arXiv preprint arXiv:2302.07346* (2023).

[102] Tongshuang Wu, Michael Terry, and Carrie J. Cai. 2021. AI Chains: Transparent and Controllable Human-AI Interaction by Chaining Large Language Model Prompts. *CoRR* abs/2110.01691 (2021). arXiv:2110.01691 https://arxiv.org/abs/2110.01691

[103] Haijun Xia. 2020. *Crosspower: Bridging Graphics and Linguistics*. Association for Computing Machinery, New York, NY, USA, 722–734. https://doi.org/10.1145/3379337.3415845

[104] Haijun Xia. 2020. *Object-Orieinted Representation and Interaction: A Step Towards Cognitively Direct Interaction*. Ph. D. Dissertation. University of Toronto (Canada).

[105] Haijun Xia, Bruno Araujo, Tovi Grossman, and Daniel Wigdor. 2016. Object-Oriented Drawing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 4610–4621. https://doi.org/10.1145/2858036.2858075

[106] Haijun Xia, Bruno Araujo, and Daniel Wigdor. 2017. Collection Objects: Enabling Fluid Formation and Manipulation of Aggregate Selections. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 5592–5604. https://doi.org/10.1145/3025453.3025554

[107] Haijun Xia, Nathalie Henry Riche, Fanny Chevalier, Bruno De Araujo, and Daniel Wigdor. 2018. *DataInk: Direct and Creative Data-Oriented Drawing*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3173797

[108] Daijin Yang, Yanpeng Zhou, Zhiyuan Zhang, Toby Jia-Jun Li, and Ray LC. 2022. AI as an Active Writer: Interaction strategies with generated text in human-AI collaborative fiction writing. (2022). https://hai-gen.github.io/2022/papers/paper-HAIGEN-YangDaijin.pdf

[109] 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* (Helsinki, Finland) *(IUI '22)*. Association for Computing Machinery, New York, NY, USA, 841–852. https://doi.org/10.1145/3490099.3511105

[110] Nicola Zaltron, Luisa Zurlo, and Sebastian Risi. 2020. CG-GAN: An Interactive Evolutionary GAN-Based Approach for Facial Composite Generation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2544–2551. https://ojs.aaai.org/index.php/AAAI/article/view/5637

[111] Loutfouz Zaman, Wolfgang Stuerzlinger, Christian Neugebauer, Rob Woodbury, Maher Elkhaldi, Naghmi Shireen, and Michael Terry. 2015. GEM-NI: A System for Creating and Managing Alternatives In Generative Design. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 1201–1210. https://doi.org/10.1145/2702123.2702398

[112] J Zamfirescu-Pereira, Richmond Wong, Bjoern Hartmann, and Qian Yang. 2023. Why Johnny can't prompt: how non-AI experts try (and fail) to design LLM prompts. In *Proceedings of the 2023 CHI conference on human factors in computing systems (CHI'23)*.

[113] Chao Zhang, Cheng Yao, Jianhui Liu, Zili Zhou, Weilin Zhang, Lijuan Liu, Fangtian Ying, Yijun Zhao, and Guanyun Wang. 2021. *StoryDrawer: A Co-Creative Agent Supporting Children's Storytelling through Collaborative Drawing*.

Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3411763.3451785

[114] Enhao Zhang and Nikola Banovic. 2021. Method for Exploring Generative Adversarial Networks (GANs) via Automatically Generated Image Galleries. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 76, 15 pages. https://doi.org/10.1145/3411764.3445714

[115] Lvmin Zhang and Maneesh Agrawala. 2023. Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543* (2023).

[116] Zheng Zhang, Jie Gao, Ranjodh Singh Dhaliwal, and Toby Jia-Jun Li. 2023. VISAR: A Human-AI Argumentative Writing Assistant with Visual Programming and Rapid Draft Prototyping. *arXiv preprint arXiv:2304.07810* (2023).

[117] Nanxuan Zhao, Nam Wook Kim, Laura Mariah Herman, Hanspeter Pfister, Rynson W.H. Lau, Jose Echevarria, and Zoya Bylinskii. 2020. *ICONATE: Automatic Compound Icon Generation and Ideation.* Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376618

[118] Yijun Zhou, Yuki Koyama, Masataka Goto, and Takeo Igarashi. 2021. *Interactive Exploration-Exploitation Balancing for Generative Melody Composition.* Association for Computing Machinery, New York, NY, USA, 43–47. https://doi.org/10.1145/3397481.3450663

[119] Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st international ACM SIGIR conference on research & development in information retrieval*. 1097–1100.