박사학위논문
Ph.D. Dissertation

상호작용 기반의 AI 정렬을 위한 언어 요소 분해

Disentangling Language for Interactive AI Alignment

2026

김태수 (金太洙 Kim, Tae Soo)

한국과학기술원

Korea Advanced Institute of Science and Technology

박사학위논문

상호작용 기반의 AI 정렬을 위한 언어 요소 분해

2026

김태수

한국과학기술원

전산학부

상호작용 기반의 AI 정렬을 위한 언어 요소 분해

김 태 수

위 논문은 한국과학기술원 박사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2025년 12월 03일

심사위원장	김 주 호	(인)
심 사 위 원	이 탁 연	(인)
심 사 위 원	이 기 민	(인)
심 사 위 원	Gagan Bansal	(인)
심 사 위 원	Haijun Xia	(인)

Disentangling Language for Interactive AI Alignment

Tae Soo Kim

Advisor: Juho Kim

A dissertation submitted to the faculty of
Korea Advanced Institute of Science and Technology in
partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Computer Science

Daejeon, Korea
December 22, 2025

Approved by

Juho Kim
Associate Professor of Computer Science

The study was conducted in accordance with Code of Research Ethics[1].

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of Korea Advanced Institute of Science and Technology, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my thesis contains honest conclusions based on my own careful research under the guidance of my advisor.

DCS 김태수. 상호작용 기반의 AI 정렬을 위한 언어 요소 분해. 전산학부 . 2026년. 155+viii 쪽. 지도교수: 김주호. (영문 논문)
Tae Soo Kim. Disentangling Language for Interactive AI Alignment. School of Computing . 2026. 155+viii pages. Advisor: Juho Kim. (Text in English)

Abstract

Current AI systems allow people to express high-level intents in natural language, yet text-centric interfaces force users to manipulate low-level text and parse long text outputs to achieve their goals. This mismatch creates gulfs of execution and evaluation: users struggle both to iteratively refine and express their ill-defined intents, and to assess whether non-deterministic AI outputs satisfy their complex, multi-faceted objectives. In this thesis, I propose text disentanglement as a conceptual approach for closing these gulfs by disentangling text into interactable components that encapsulate the high-level concepts that users cognitively consider when performing their tasks.

On the execution side, I develop interfaces that disentangle user intents into atomic and combinable building blocks and into palettes of alternative interpretations. These ideas are instantiated in the "Cells, Generators, and Lenses" design framework for composing LLM-powered writing workflows, and Stylette, which lets novices adapt the style of websites by selecting from AI-generated interpretations of their natural language requests. On the evaluation side, I introduce EvalLM, which summarizes model outputs into multi-dimensional scores along user-defined criteria, and Evalet, which further surfaces semantically meaningful fragments from outputs that influence their quality according to these dimensions. To understand and extend the underlying capabilities of AI models, I present Cupid, a benchmark for disentangling user preferences from multi-session interaction logs, and DesignLLM, a training framework that teaches models to natively disentangle user messages into diverse response variants that capture distinct interpretations.

Across these contributions, user studies and technical evaluations show that text disentanglement helps people generate and iterate on intents more flexibly, examine more alternatives, and make sense of model outputs more effectively. Together, this thesis demonstrates that disentangling text into interactable components can scaffold, accelerate, and enrich interactive alignment with AI models.

Keywords Human-Computer Interaction, Natural Language Processing, Human-AI Interaction, Interactive Alignment, Interactive Evaluation

Contents

List of Tables

List of Figures

Chapter 1. Introduction

1.1 Background

State-of-the-art AI models have shifted the paradigm for interactions between users and computers. Instead of translating their high-level intents into low-level operations that a computer should perform, users can now simply state their intents directly to the computer [247]. Through advancements in natural language (NL) understanding and instruction following capabilities, recent models can perform tasks from user's high-level, NL inputs. For example, models can generate music [3, 66], images [236, 282], and even videos [44, 407] from user's descriptions. More advanced models, like Large Language Models (LLMs), possess general purpose capabilities that allow users to perform ever more complex and unique tasks, from writing full research papers [221] to developing interactive applications [253]. This new paradigm for human-computer interaction expands the space of possibilities of what users can achieve through computers [358].

While users can now reach greater possibilities by simply expressing their intents to an AI model, the nature of these intents and these AI models introduces new challenges in the interaction process. Specifically, while the user's intents are **ill-defined and complex**, the AI models are **black box and non-deterministic**. These qualities carve new *gulfs of execution and evaluation* [138].

- **Gulf of Execution**: As user's goals are **ill-defined** [306], they may be unsure about their goals and needs at the start of a task—leading to them not being able to accurately describe their intents to the AI model [318, 332]. Furthermore, as the AI model is **black box** but highly susceptible to how inputs are formatted [223, 389], users can struggle to foresee how they should ideally express or phrase this intent to the model [166] Thus, the user can fail to know what actions (i.e., what to input or say) to take in order to achieve their goals—*gulf of execution*.

- **Gulf of Evaluation**: Most AI models are **non-deterministic**, meaning that they can provide drastically different outputs even for the same input. As a result, users cannot anticipate what the model will produce for a given input [285]. Adding further challenge, as users' intents are frequently **complex** (i.e., consist of multiple interrelated objectives), they may expend significant cognitive effort assessing the output against these various objectives [168]. Due to these reasons, users can struggle to verify that their actions led to desired consequences—*the gulf of evaluation.*

Due to these gulfs, the users can only align the AI model (i.e., ensure it produces desired outcomes [332]) by interacting with the model through iterative loops of execution and evaluation—providing inputs and inspecting outputs. To allow users to reach the space of possibilities afforded by these AI models, interfaces should be designed to support this process of ***interactive alignment***. However, **text** has become the *de facto* interface between users and AI models: the user inputs their intents into a text box and the model's outputs are presented in a text box. While this is particularly true for LLMs, interfaces for Text-to-Image (T2I) and Text-to-Video models also predominantly rely on text as the main input interface. **Text**, however, introduces *friction* to interactive alignment:

- **Execution**: As users have complex intents that encompass multiple objectives, they must construct a single piece of text (e.g., a message) that encodes all of these objectives—requiring mental and physical effort. More challenging than expressing the initial intent, however, is iterating on that intent that has now been encoded into text. For example, imagine a user has written: *"write me a visionary, sci-fi-like research proposal about*

personalized AI tutors with references to recent literature". If the user's intent changes (e.g., add/remove, emphasize/de-emphasize objectives), the user has to edit the text to encode these changes (e.g., delete, add, rephrase) or encode new text that expresses these changes (e.g., "make it more realistic, a bit less visionary"). Despite the user cognitively operating on these high-level objectives, these text-based interfaces force the user to perform multiple low-level text operations to encode each of these high-level operations.

- **Evaluation** For a given input, AI models frequently return complete and extensive text artifact (e.g., a multi-paragraph research proposal). To understand if an output satisfies their intents, the user must inspect and parse through the whole text. Specifically, the user must interpret what are the characteristics and attributes contained in the text, and assess how these align or misalign with their intents. In the example where the user requests a visionary research proposal with references to recent literature, the user may read through the generated proposal, while checking each single citation and verify that only literature in the last 5 years were cited. The user may also notice that the proposed idea involves modifications to existing research, which is not as visionary as intended. Through this process, the user is essentially decoding high-level characteristics from the text that are related to their intents (e.g., citation recency, technical novelty)—requiring significant cognitive effort.

The key limitation of these text-centric interfaces is that the user is cognitively operating at a higher level of abstraction (e.g., objectives in inputs, attributes in outputs). However, these interfaces require users to perform low-level operations on the text (e.g., add, delete, parse, inspect). To interactively align the AI model, the user must continuously encode their high-level objectives through low-level text manipulations, and then decode out high-level qualities from outputs by parsing and interpreting the low-level text. This abstraction gap leads to significant cognitive and manual effort.

1.2 Disentangling Text for Interactive AI Alignment

To address these challenges in interacting with state-of-the-art AI models, this thesis proposes the concept of ***text disentanglement***. Specifically, this thesis proposes: If the text encodes the high-level intents and attributes that the user considers and cognitively operates on, what if we *disentangled* these intents and attributes out of the text and presented them as interactive components to the user? Instead of operating on the low-level text, the user could then directly operate and interact with components that represent the high-level abstractions that they are already cognitively operating on. This disentanglement proposes benefits to users in both the **execution** and **evaluation** phases of the interaction process.

1.2.1 Execution

Disentangling Intents into Building Blocks When interacting with an LLM, users may incorporate multiple intents into a single request. As described before, to iterate on each of these separate intents, users must either edit the request's text or send additional requests that express changes to this intent—an effortful process. To address this, this thesis proposes that the users' input intents can be disentangled or *decomposed* into individual building blocks. Then, each of these blocks can represent a distinct atomic intent, each block can be modified independently, and the blocks can be composed into diverse composite intents (e.g., remove a single atomic intent, create variations of only one atomic intent). This concept was instantiated in the *Cells, Generators, and Lenses* design framework [166], a set of guidelines that describe how to design LLM-powered interfaces that allow users to interact with the AI model through *composable objects*. Specifically, this framework introduces three types

of objects: *Cells*, represents text fragments, *Generators*, represent a set of model configurations, and *Lenses*, represent output sets. The framework describes how users can create multiple variations of each object, compose them together into diverse configurations, and experiment with these configurations in parallel. To showcase the framework's generalizability, three interfaces for different writing tasks (e.g., story writing, copy writing, email composing) were designed using the framework. Furthermore, a comparative user study (N=18) demonstrated that the disentangled objects helped users to more easily generate and experiment with the AI model.

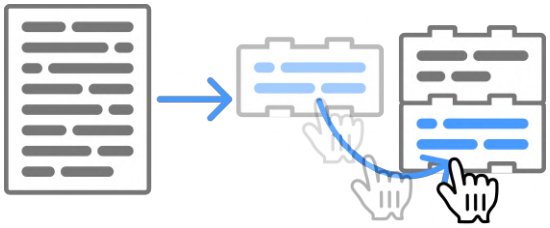

Figure 1.1: The *Cells, Generators, and Lenses* framework proposed how to disentangle input text into more atomic and composable objects.

Disentangling Intents into a Palette of Interpretations The *Cells, Generators, and Lenses* framework assumes that each block represents a single intent. However, due to the ill-defined and ambiguous nature of users' intents, each block can actually be interpreted into multiple, diverse intents. For example, "a sci-fi-like proposal" can be interpreted into multiple distinct intents: "style of sci-fi authors", "narrative structure", "speculative technology", etc. When the user is a novice in the task, their intents can become even more vague as they lack the expertise to know what is needed in their task and they also lack the knowledge about the precise terminology to use [100]. If the AI simply returns what it considers the best interpretation, this can cause problems: users may be forced to retry if the intent was misunderstand and it prevents users from discovering alternative options that they were unaware of. As a method to address this, Stylette [163] is a browser extension that allows novices to edit the code of any website by simply expressing their intents, where this intent is disentangled into its diverse interpretations. Then, instead of only returning the operations related to a single interpretation, the system returns a *palette* of operations for all of the different interpretations—allowing the user to explore, test, and apply these flexibly. A comparative study (N=40) showed that Stylette lowered the learning curve for the novice participants, helping them perform changes to the website code 35% faster than when using a baseline.

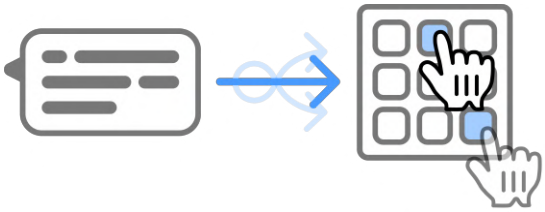

Figure 1.2: Stylette disentangles users' expressed intents into its diverse interpretations, creating a palette of operations from all these interpretations that users can use to directly perform the intended operations.

1.2.2 Evaluation

Disentangling Outputs into Dimension Scores As LLMs frequently return complete and extensive outputs, users need to expend significant cognitive effort in parsing the output according to the specific dimensions that are related to their intents (e.g., recency of references, how visionary the proposal is). Only by interpreting the

output according to each dimension can the user determine whether the output satisfied their intents and, if not, what should be improved. If the user is dealing with multiple outputs (e.g., they are creating an LLM application and testing it with diverse inputs), this requires a prohibitive level of cognitive effort. Instead of requiring users to do this interpretation themselves, if each output could be automatically disentangled according to each of these intent-relevant dimensions, then users can easily gain an understanding of how well each output satisfies their intents. EvalLM supports this by leveraging LLMs to evaluate outputs from another LLM according to user-defined criteria—standards relevant to the user's overall intents. For each output, the system returns scores for each criterion, summarizing the output according to each of these dimensions. A comparative study (N=12) showed that EvalLM, when compared to manual evaluation, helped participants compose more diverse criteria, examine twice as many outputs, and reach satisfactory prompts with 59% fewer revisions.

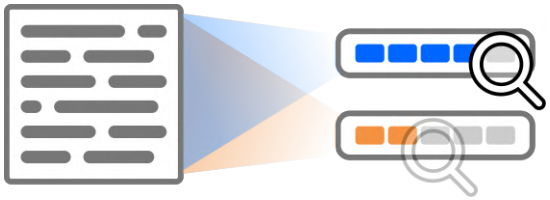

Figure 1.3: EvalLM disentangles text outputs by automatically evaluating and scoring them according to each user-defined criterion.

Disentangling Outputs into Functional Fragments While EvalLM effectively summarizes each output according to dimensions of interest for the user, the system returns a numeric score for each output on each dimension. These scores, however, are an opaque summary: users cannot directly interpret how the text was composed and how these different components led to that score. Thus, users must still parse the actual outputs to concretely understand where and how the outputs failed or succeeded at satisfying their intents. To address this, Evalet disentangles outputs, not simply into scores for each dimension, but by identifying and surfacing the actual components in the outputs that led to these scores. For example, a generated proposal that may have received a score of 3 out of 5 regarding how "visionary" it is. Evalet could surface the following components from the output: "proposes improvement on existing technology", "applies existing methods", "proposed benefits are minimal", etc. A user study (N=10) found that Evalet helped participants identify 48% more actionable issues in the LLM outputs.

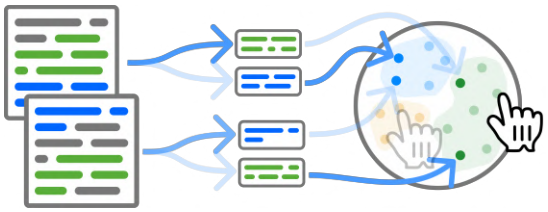

Figure 1.4: Evalet disentangles text outputs by extracting and surfacing components that are relevant to the user's intents.

1.3 Benchmarking and Training Models on Text Disentanglement

Several of the approaches introduced in this thesis are dependent on an AI model to automatically disentangle the relevant text artifacts. Thus, the success of text disentanglement for interactive alignment is dependent on

the AI model's capability to disentangle the text. Through various technical evaluations in each chapter, this thesis demonstrates that AI models are capable of effectively disentangling text with relatively simple artifacts (e.g., single user message, multi-paragraph outputs). However, to support more interaction scenarios, these AI models must possess the capabilities to disentangle more complex text (e.g., multi-session multi-turn chat dialogues, multi-page paper). CUPID [167] assesses whether LLMs can disentangle user's contextual and personal preferences from multi-session interaction histories. If the models possess this capability, they can support personalization by applying these disentangled preferences in future interactions.

Finally, this thesis presents preliminary results on DESIGNLLM, a novel training framework that teaches LLMs to interact with users through text disentanglement, intrinsically. Instead of relying on additional layers to disentangle the text post-hoc, which introduces latency in interaction, this framework trains models to disentangle user messages into multiple interpretations—similar to Stylette—to produce responses composed of multiple prototype artifacts. Each prototype embodies distinct attributes that allow users to explore the space of possibilities and more efficiently realize their intent when interacting with the model—similar to how *Cells, Generators, and Lenses* helps users explore by experimenting with diverse inputs. An evaluation with simulated users reveals that DESIGNLLM enhances intent formation and satisfaction, when compared to the base model and other LLMs trained for collaborative behaviors.

1.4 Contributions

This thesis makes two primary technical contributions: (1) computational methods for disentangling text into constituent components or attributes, and (2) interaction techniques that leverage these components to support execution-evaluation loops during human-AI interaction. The computational methods are validated through experiments that assess their accuracy and robustness, and the interaction techniques are assessed through evaluation studies that measure effects on users' behaviors, the quality of their resulting artifacts, and their qualitative experiences. Together, this thesis demonstrates how disentangling text can serve to scaffold, accelerate, and enhance users' interaction with AI models—enabling them to reach the diverse possibilities that are afforded by these models but were previously challenging to reach.

Furthermore, this thesis proposes a comprehensive suite of contributions that spans the complete end-to-end infrastructure—from conceptual frameworks to computational methods to interactive systems—necessary to support text disentanglement as a practical approach for human-AI interaction. Specifically, this thesis contributes: (1) **design frameworks** that guide interface designers to move towards disentangled interfaces; (2) **standalone systems** that implement and demonstrate interaction techniques built on disentangled text to support iterative refinement of user intents and granular evaluation of model outputs; (3) **extensions or plugins** that incorporate text disentanglement capabilities into existing systems or interfaces; (4) **computational pipelines** that automatically decompose and disentangle complex text into meaningful components; (5) **AI models** that are tuned to intrinsically support interaction through text disentanglement, and (6) **benchmarks** designed to rigorously evaluate the robustness and accuracy of these disentanglement methods.

Thesis statement: Disentangling text into interactable components supports interactive alignment of AI models by facilitating input iteration and output assessment.

1.5 Thesis Overview

- **Chapter 2** reviews prior work across four areas: (1) generative AI models and natural language interaction paradigms, (2) formulating intents through prompt design and domain-specific input methods, (3) assessing

quality through evaluation systems and sensemaking at scale, and (4) closing the loop through learning from human preferences and interaction histories.

- **Chapter 3** presents *Cells, Generators, and Lenses*, a framework for designing LLM-powered interfaces where text inputs and outputs are disentangled into interactive objects that are persistent, multiplicable, and composable to support intent iteration.
- **Chapter 4** introduces *Stylette*, a system that enables novices to manipulate the styling of website by speaking their intents, which are then disentangled into diverse interpretations and corresponding operations that users can directly apply and combine.
- **Chapter 5** describes EVALLM, a system that allows users to define their own criteria and then automatically evaluate multiple LLM outputs on these criteria—disentangling each output into a set of scores for each criterion.
- **Chapter 6** presents EVALET, a system that disentangles LLM outputs into fragment-level functions—i.e., semantically significant components or blocks of each output—and supports exploration and comparison of these functions across outputs.
- **Chapter 7** introduces the CUPID benchmark that evaluates whether LLMs can disentangle users' contextual preferences from prior user-LLM interactions and apply these in new contexts.
- **Chapter 8** introduces findings on DESIGNLLM, a general training framework for LLMs that can interact with users through text disentanglement.
- **Chapter 9** summarizes lessons from the proposed approaches for text disentanglement, and delineates the main design dimensions, considerations, and limitations of this approach to guide future researchers and practitioners in supporting human-AI interaction.

Chapter 2. Related Work

This chapter reviews prior work that lays the foundation to this thesis. In particular, the chapter reviews: generative AI models and interaction with these models, supporting iteration and experimentation with AI models, assessing the quality of AI model outputs, and closing the loop through learning and personalization.

2.1 Generative AI Models

Understanding the capabilities of generative AI models and the paradigms through which users interact with them provides the foundation for supporting interactive alignment.

2.1.1 Diversity of Generative Models

The increased advancement, availability, and diversity of generative models have enabled humans to leverage artificial intelligence (AI) to create a variety of artifacts. For example, researchers have developed models that can generate music [135, 134, 307], sketches [126, 208, 87], graphic designs [388], and 3D models [24]. More recently, extending on the GAN [114] and DCGAN [273] architectures, diffusion-based models [275, 281, 236, 289] have pushed the boundary of image generation by producing realistic and high-resolution outputs from text instructions. For language, the advent of massive-scale transformer-based [339] architectures—i.e., Large Language Models (LLMs) [45, 203, 310, 37, 334, 59, 335, 293, 252]—has enabled the generation of diverse types of text with previously unparalleled fluency and coherency.

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 [341, 23], learning [195, 193, 230], and prototyping [268, 262, 163]. These diverse generative models can produce thousands of outputs based on the users' relatively simple inputs. With the rising number of generative models and with their outputs starting to see more real-life use—evidenced by AI art [51] and commercial AI-powered writing tools [140, 181, 144, 362]—there is an increased need for designing interfaces that can better help end-users to fully leverage the generative power of these models [124, 118].

2.1.2 Natural Language as Interaction Paradigm

Novices struggle to translate high-level goals into tool operations due to the vocabulary problem [100]—the language used by the user and the tool do not match. Thus, empowering users to be able to design by simply stating their high-level goals has been a long-standing goal for HCI researchers. To support this, researchers have proposed diverse techniques and systems that can process users' natural language expressions and translate this into tool operations. For example, Query-Feature Graphs (QF-Graphs) [94] and CommandSpace [2] jointly modeled natural language descriptions with feature names in design applications (e.g., GIMP and Photoshop) to help users identify features based on their needs. Other systems support the use of natural language concepts to search for design references or components—images [93], graphic designs [152], or 3D models [55]. Beyond searching, several systems generate artifacts (e.g., images [186] or icons [403]) based on the semantic meaning of words, or facilitate editing of existing artifacts by decomposing natural language expressions into operations [373, 188].

In particular, substantial effort has been dedicated to bridge natural language and complex programming languages [237] to lower the barriers to programming. For instance, researchers have used semantic parsers [271] and bimodal models [10] to map natural language to code. Such techniques enabled systems that allow novice coders to quickly search for code snippets [283, 294], and non-coders to code small programs by demonstrating and describing tasks [200, 229]. Beyond mapping, a line of work has also developed techniques that take natural language as input and generate code—e.g., Python [385, 209], Bash commands [207], SQL queries [411], or API calls [355].

Recent advancements in natural language processing (NLP), and especially in Large Language Models (e.g., GPT-3 [45]), have led to performance boosts in natural language understanding. For example, by only providing a few natural language sentences, users can generate fully interactive video games with these models [392]. While these advances in natural language interaction have enabled users to express high-level intents directly to computers, as discussed in Chapter 1, this paradigm also introduces new gulfs of execution and evaluation when users' intents are ill-defined and AI models are black-box and non-deterministic.

2.1.3 Interaction Challenges and User Needs

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 [219, 248], while others demonstrated that there are benefits in the model taking the lead [64, 123]. Beyond who leads, research has identified several other trade-offs: producing more generations increases exploration but decreases efficiency [46], and more unexpected generations can provide inspiration but can also seem less useful [190, 64, 380]. Due to these user-dependent factors, prior work [108, 118, 133] argues that how generative models are used should adapt according to users' changing goals. This highlights the need to facilitating users' interaction with these models, while also empowering user to adapt how they interact with these models.

2.2 Formulating Intents: From Goals to Inputs

Translating user goals into effective inputs requires interfaces that bridge the gap between high-level intents and the specific formats expected by AI models.

2.2.1 Prompt Design and Engineering

Although LLMs can execute complex tasks for users without users having to collect data or train these models, designing satisfactory prompts can be an arduous task [213]. The specific format, phrasing, content, examples, or even the order of examples used in prompts can significantly affect performance [223, 210, 287**?** , 213]. However, as the space of possible natural language instructions is near infinite, users need to test as many possibilities as possible to construct successful prompts [389, 211].

To help users identify effective prompts, researchers have proposed various tools that facilitate prompt design. 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 [148] and BotDesigner [390] allow users to create prompt templates and test them with different inputs. Expanding on these ideas, another body of work [317, 366, 240, 166] supports users to create variations of model inputs to test and compare model performance on these inputs. Beyond singular prompts, AI Chains [370],

PromptChainer [367], and ChainForge [20, 396] allow users to iterate on multi-step prompts using interactive chains of prompts. Similarly, to facilitate iteration with text-to-image models, Opal [214] and 3DALL-E [215] modularize user inputs into keywords and provide keyword suggestions to facilitate composition of inputs for image generation. This decomposition approach supports iteration with the generative models by facilitating testing and experimentation of diverse input combinations, inspiring the disentanglement techniques proposed in this thesis.

2.2.2 Configuration and Parameter Control

Beyond inputs, several interfaces for generative models have also explored how to facilitate iteration and experimentation with both inputs and model parameters or configurations. GANSliders [73] and GANSpace [128] support users to customize the generative process of GANs by manipulating underlying parameters through visual feedforward sliders and semantic controls. Louie et al. [219] and Zhou et al. [415] provide more interpretable parameters to help users control generated outputs. Furthermore, TaleBrush [61] supports more seamless control over parameters by allowing users to sketch how the parameter should change or "flow" through a story.

2.2.3 Domain-Specific Input Methods

Beyond general-purpose prompt engineering, researchers have developed specialized interfaces for specific domains and tasks.

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 [61, 64, 386], screenplay writing [239], poetry [111], and argumentative writing [397, 340]. 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 [190, 64, 308] with several of these allowing writers to control the type of continuations that are generated by providing additional instructions [72, 386] or through visual sketches [61]. Other tools support auxiliary processes during writing such as ideation [109, 266, 108, 402], revision [82, 194], and reflection [340, 71]. This thesis builds on these prior literature by proposing a novel approach, *text disentanglement*, to support user-LLM interaction in diverse tasks, including writing tasks.

Web Design and Manipulation

As web interfaces are visual representations of HTML and CSS code, various tools have been designed to facilitate the process of modifying the code to produce desired visual changes. For example, openHTML [264] provides an educational environment which shows HTML code, CSS code, and a website preview side-by-side. Other systems [52, 278, 204] allow users to inspect the code behind pages to understand the connection between code and visuals. Beyond inspection, Chickenfoot [40] allows end-users to write simple scripts to modify components, and, more recently, Spacewalker [408] leverages genetic algorithms and crowdsourcing to generate design alternatives. These tools, however, were designed with developers or learners in mind, and require the user to understand and interact with the code—a task impractical for end-users with limited knowledge.

To make manipulation more practical, a separate line of research allows users to modify a website's visuals by directly interacting with the visuals. CrowdAdapt [246], CrowdUI [256], and XDBrowser [245] allow users to modify the positioning of web components through direct manipulation (e.g., drag-and-drop). Aimed at designers who have limited coding knowledge, Poirot [330] and CoCapture [57] provide designer-specific widgets to support

design editing and animation authoring, respectively, directly on websites. These tools, however, still require the user to expend time and effort deciding between and performing various possible editing operations. Example-based systems [179, 189, 91] aim to reduce this mental and manual effort by allowing users to copy the styles of other websites. This thesis proposes how this process can be simplified further: by disentangling users' intents into diverse editing operations, interfaces can support users to explore and realize their needed editing operations in web design.

2.3 Assessing Quality and Making Sense

Understanding whether AI outputs satisfy user intents requires methods that go beyond simple metrics to provide interpretable, actionable insights into output quality and model behavior.

2.3.1 Natural Language Generation Evaluation

Natural language generation (NLG) is the family of NLP tasks where the goal is to generate text that satisfies a communicative goal (e.g., summarize a document) while possessing several desired qualities (e.g., fluent, coherent) [106, 244, 117, 86]. While recent years have brought significant progress in NLG, especially due to LLMs, a constant obstruction to progress has been the difficulty of evaluating these tasks [156, 80, 378]. Unlike classification tasks where performance is measured by comparing a prediction to a ground-truth label, generation tasks are *ill-posed*—i.e., multiple dissimilar outputs can be equally valid. While researchers have proposed automatic metrics that compare outputs to several ground-truth references (e.g., *BLEU* [260], *ROUGE* [206]), the space of valid outputs in open-ended generative tasks can be overwhelmingly vast, making it nearly impossible to create sufficiently comprehensive references sets. Thus, human evaluations where annotators rate or rank generated text have become the golden standard [106]. However, the cost and effort involved in recruiting human annotators can make this type of evaluation prohibitive during early development stages. As an alternative, recent work [199, 406, 99, 217, 384, 58] has employed LLMs to simulate annotators and automatically evaluate outputs on their overall quality or a pre-defined set of criteria—demonstrating agreement with human evaluations on par with the level of agreement between human evaluators [406].

2.3.2 Interactive Evaluation Systems

Evaluation is fundamental to the development of machine learning models for real-world applications. Beyond assessing performance on a single metric, practitioners (e.g., developers, researchers, engineers) assess more fine-grained model behaviors to identify flaws and potential improvements [277]. Traditionally, Machine Learning (ML) models are evaluated on benchmarks with automated metrics, where performance is aggregated into a single statistic or score. However, this provides limited signal into how the model behaves, what its flaws are, and what are the specific areas for improvement [277, 393, 242].

Fine-Grained Evaluation of ML Models

To support fine-grained assessments, prior work has introduced various systems that allow practitioners to interactively evaluate outputs. For example, *Zeno* [48], the *What-If Tool* [360], and *Errudite* [368] help practitioners to identify *slices* or subsets of data that may reveal distinct model failures. Beyond testing on existing input data, *Polyjuice* [369] and *AdaTest* [276] allow practitioners to generate potentially challenging input data to test a model's behavior and iteratively evaluate models by creating challenging input data and testing how the models behave

on these cases. To aid practitioners in resolving issues after models are deployed, *Angler* [279] combines online and offline data to help practitioners prioritize performance issues, and *Deblinder* [47] allows practitioners to collect and analyze model failure reports from crowdworkers. Furthermore, researchers have proposed various tools [48, 360, 368, 279, 309] that help practitioners to unpack evaluations by identifying *slices* or subsets of data, and testing models on these to identify specific flaws or limitations.

Tailored Evaluation of LLMs

The general-purpose capabilities of LLMs have enabled novel AI-based applications, but have also increased the difficulty in verifying that these models perform as intended. Specifically, as these models are applied to new tasks and contexts, there are no benchmarks or metrics to automate evaluation [168] and, as their input and output space is near infinite, the models have to be tested with numerous and diverse samples [389, 211]. More recently, the success of *LLM-as-a-Judge* [406] (i.e., LLMs evaluating other LLMs) has led to several systems [168, 151, 22, 299] that employ LLM-based evaluators to support interactive evaluation on diverse aspects or criteria. By assessing outputs on multiple criteria, these approaches offer a multi-dimensional view of model performance. However, as they only provide holistic scores and overall justifications, practitioners must manually review the outputs and justifications to identify specific strengths and weaknesses and validate evaluations [168, 105]—requiring effort that is impractical at scale. Inspired by these approaches, we incorporate LLM-powered simulated evaluators to support interactive evaluation of LLM prompts on user-defined criteria and investigate how users interact with these evaluations to refine prompts. This thesis proposes how LLM outputs can be disentangled based on users' intents or goals to facilitate interpretation and evaluation of LLMs according to their unique tasks and contexts.

2.3.3 Sensemaking and Comparison at Scale

Substantial work has explored how to support large-scale sensemaking of text artifacts to identify patterns and distill insights [185, 350, 112, 383, 158, 258, 272, 259]. Recent work has started to investigate how to support sensemaking over LLM outputs [320, 150] to facilitate exploration and analysis of diverse outputs. For example, *Luminate* [319] guides LLMs to generate outputs according to key dimensions and then visualizes these outputs according to these dimensions, helping writers explore the space of plausible outputs. Gero et al. [110] explored various designs and algorithms (e.g., unique words, exact matches) to support comparison of LLM outputs and help users form mental models of how the LLM behaves. *Policy Projector* [184] *"maps"* LLM input-output pairs into a 2D space to help users explore common groups of outputs, classify these groups, and define policies or rules on the model's behaviors based on these classified groups. This significant body of work on interactive evaluation and sensemaking highlights the need for more granular analysis of model performance. This thesis extends these approaches by proposing text disentanglement as a method to evaluate and make sense of outputs at a more fine-grained level—analyzing semantically meaningful components or attributes within outputs rather than treating outputs as monolithic artifacts.

2.4 Closing the Loop: Learning and Personalization

To continuously improve alignment, AI systems must learn from interactions and adapt to individual user preferences across different contexts.

2.4.1 Alignment with Human Values and Preferences

To reduce harms and increase helpfulness, research has explored how to align LLMs with human values [29, 28] by training them on *general* preferences [29, 28, 257, 69, 159]. Recent work has taken a more personalized view to alignment, considering individual preferences and values [313, 337]. For example, Kirk et al. [172] collected a preference dataset with detailed user information (e.g., demographics, behavior attributes). Other work explored prompting LLMs with user profiles [381, 345], interaction logs [26], or user-written artifacts [291, 290]. Alternatively, Jang et al. [142] and Lee et al. [192] fine-tuned LLMs on decomposed preferences to produce diverse responses for users. More recently, PREFEVAL [404] evaluates whether LLMs can identify user's global preferences from long-context conversations.

2.4.2 Learning from Interaction Histories

LLMs possess the capability to interact with human users [349]. User-LLM interactions can organically reveal details about user intents and preferences [170, 302], enabling models to align themselves with users. Recent work explores how to extract feedback from user-LLM interactions [78, 364], and how to enhance LLMs' *memory* to leverage user knowledge from interaction histories [412, 347, 413, 363, 157]. Other work focused on training LLMs to capture richer details from user interactions by clarifying intents or eliciting information [270, 197, 15]. This work on personalization highlights the importance of understanding and applying user preferences across contexts—a capability that benefits from disentangling contextual preferences from interaction histories.

Chapter 3. Cells, Generators, and Lenses: Design Framework for Object-Oriented Interaction with Large Language Models

This chapter presents the first example of text disentanglement during the execution phase of interactive alignment. *Cells, Generators, and Lenses* is a design framework that proposes how interfaces can be designed to decompose users' text inputs into interactive objects that can be independently manipulated and composed together. This chapter has adapted, updated, and rewritten content from a paper at UIST 2023 [166]. All uses of "we", "our" and "us" in this chapter refers to coauthors of the aforementioned paper.

3.1 Motivation & Contributions

Large language models (LLMs)—e.g., ChatGPT [251], GPT-4 [252], PaLM [59], LLaMa [335]—have enabled users to write without actually writing. Users can delegate the manual effort of producing text to these models to increase productivity [46], inspire new ideas [50], or quickly "sketch" passages [61]. Besides shouldering the effort of producing text, LLMs can also enhance auxiliary processes such as editing [181, 82], feedback exchange [140, 340], or reflection [71]. 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 [133, 321] (e.g., prompt engineering [370]) and adjust the model's *parameters* [190] (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 [64, 231]—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.

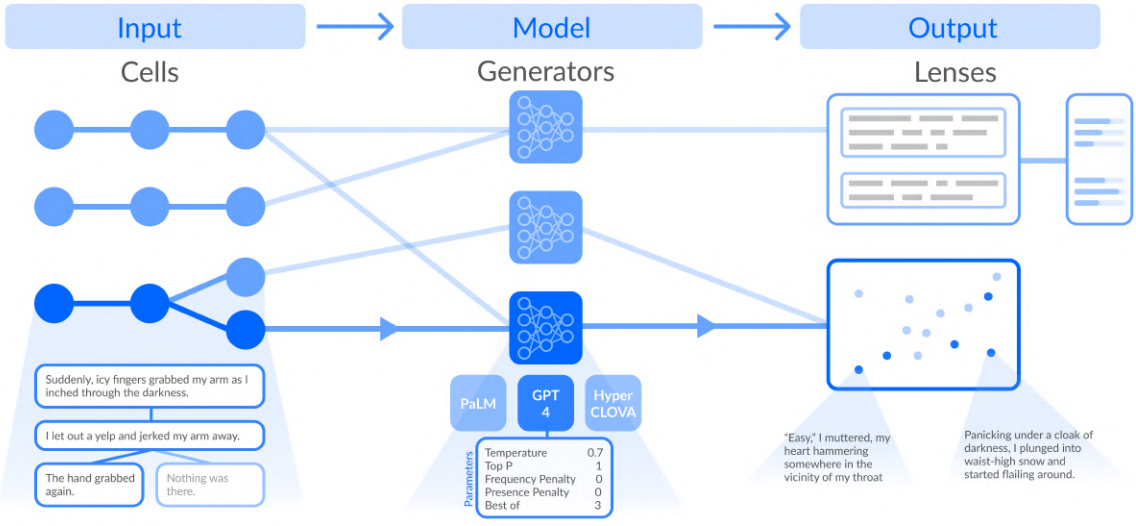

Figure 3.1: *Cells, generators, and lenses* is a design framework for object-oriented interaction with large language models (LLMs). Input units, model instances, and output spaces are represented as interactive objects: cells, generators, and lenses, respectively. By integrating these objects in their 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.

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 [285]. Users may then need to repeatedly experiment with generation configurations to understand their effect [219, 60, 325]—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 [300, 68]. 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 [386, 249, 7, 322]. 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 [359], which introduces friction and hinders experimentation. Furthermore, these interfaces do not allow end-users to prototype configurations in parallel [81], 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 chapter, we introduce a design framework for interfaces that support object-oriented interaction with LLMs through *cells, generators, and lenses* (Fig. 3.1). Unlike existing interfaces where end-users interact with *one* input area, parameter setting, and output space, our framework proposes how interfaces can reify [34] 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 [81] 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 released our interface components for cells, generators, and lenses as an open-source ReactJS library: `https://github.com/kixlab/llm-ui-objects`.

3.2 Cells, Generators, and Lenses

To design writing interfaces that support iteration and experimentation with LLMs, this chapter 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 [375], spatial selections [376], text passages [127]) into interactive objects can simplify and facilitate users' workflows [33, 62, 377, 373]. 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 [374]. Further, with persistent and composable objects, end-users can combine these into multiple parallel prototypes, which could prevent fixation and encourage experimentation [143]. Additionally, if end-users are able to create and maintain alternatives in parallel, they can readily compare these alternatives to solve problems [107] and understand the task in greater depth [42].

Inspired by the advantages of *reifying* task elements into *reusable* objects [34], in this chapter, 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.

3.2.1 Cells

Cells (Fig. 3.2) 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 [357, 201]. Below, we describe and justify two interactions that designers should support regarding cells, *create* and *assemble*.

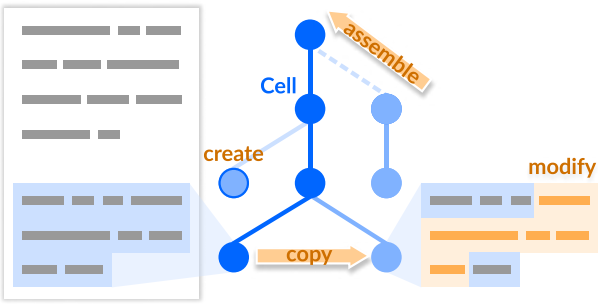

Figure 3.2: *Cells* are object representations of input units (e.g., sentences). To create variations of inputs, users can create, copy, modify, and assemble cells. Cells that have been assembled together are shown connected by blue edges.

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 [386, 249, 7]. 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 [321, 324, 194]. 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 [34], 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.

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 [13, 43, 352, 370]. 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 [32, 214] or to mix-and-match the variants [388, 366]. Finally, disentangling inputs also helps users to "lock" portions of the input and experiment with the effects of each portion separately—encouraging systematic testing [390]. This portion-wise experimentation can allow users to interactively align generations with their intentions [8, 403] and to more intuitively gain an understanding of the models [387, 285]. 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

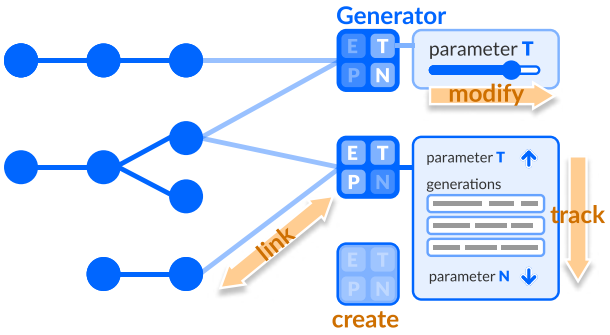

Figure 3.3: *Generators* are object representations of generative model instances (i.e., the type of model and its parameters). Users can create generators, modify their parameters, and then link them to one or multiple cells to generate outputs. Generators display their parameter settings in their faces (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.

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.

3.2.2 Generators

Generators (Fig. 3.3) are object representations of model settings (i.e., type of LLM and parameter values).

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 [220, 125, 153]. Regarding writing and LLMs, Lee et al. [190] found that different parameter settings can fulfill different writing goals (e.g., diverging vs converging), and Chung et al. [61, 60] 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 [386, 316] or can only customize one global set of parameters [219, 249, 7, 303, 111, 113]. In comparison, designing interfaces with generators can allow users to simultaneously maintain several model instances, where each addresses a different sub-task or need [133]. 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 [359]. 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.

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).

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 [11].

3.2.3 Lenses

Lenses (Fig. 3.4) are object representations of spaces that represent and visualize the generations produced. For example, these can include a list [64], a gallery [327], or a real-time confidence visualization [232].

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 [98, 356, 395, 415]. 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 [61] but, when choosing the next sentence for a story, a list of generations allows the user to concretely compare them [386, 108]. 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 [46]. 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).

Assemble

Lenses can also be assembled together to view the same generation outputs through multiple representations [231, 308]. 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 [74], allowing them to assemble lenses can support more comprehensive sensemaking.

3.3 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.

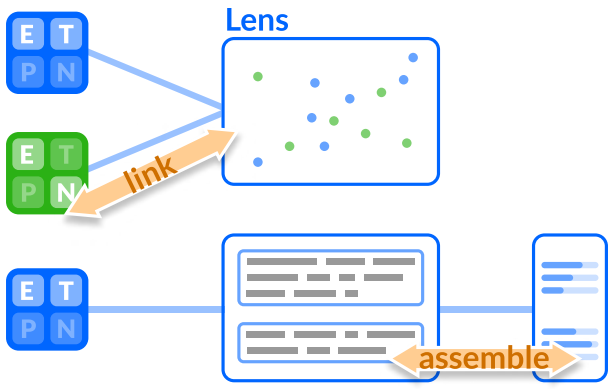

Figure 3.4: *Lenses* are object representations of output spaces that represent and visualize generations from linked generators (e.g., a list or a 2D grid). Lenses can be assembled together to visualize the same generations in multiple ways.

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 [53] 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.

3.3.1 Copywriting Interface

Our copywriting interface (Fig. 3.5) allows end-users to create advertisements from a couple of product specifications. The interface was designed based on copywriting tools [64, 362, 144] 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 combining 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. 3.5a). To allow end-users to test various model parameters, the interface allows them to create multiple generators, each with its own parameter settings (Fig. 3.5b). 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. 3.5c).

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. 3.5a). In each line, the user can

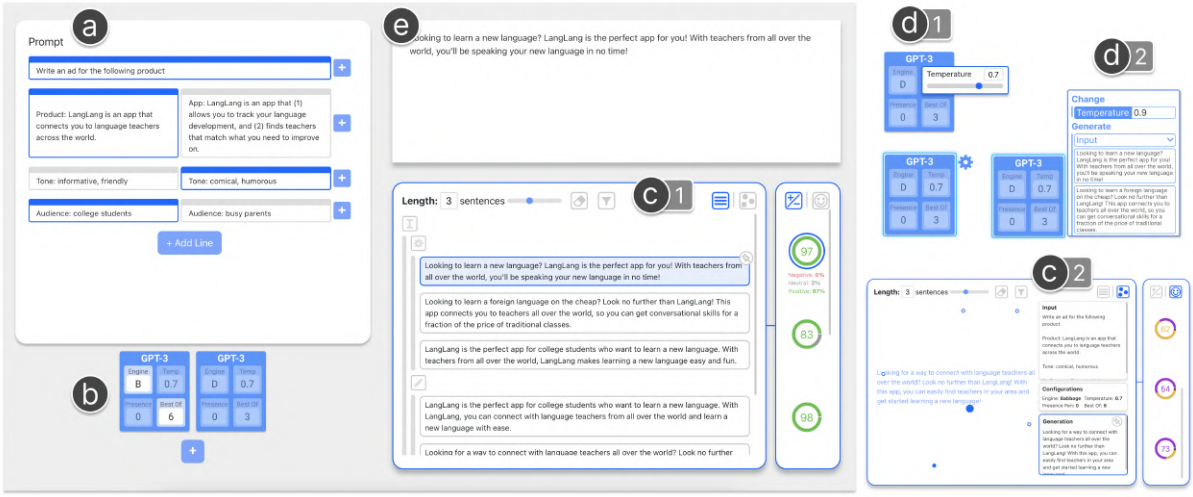

Figure 3.5: The copywriting interfaces allows end-users to provide a set of specifications to generate advertisements by creating and editing cells (a). In the generator tray (b), end-users can create multiple generators and modify their parameters (d-1). Then, by clicking on a generator, end-users can 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 were changed and what outputs it generated, end-users can also browse through the history of each generator (d-2).

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

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.

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. 3.5b). 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. 3.5d-2).

Lenses: List, Space, and Rating

Initially, end-users are presented with the list lens (Fig. 3.5c-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. 3.5c-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. 3.5c-1 and Fig. 3.5c-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. 3.5e) where they can then edit and combine it with other generations.

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

[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.

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 [390]. 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.

3.3.2 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 [92, 181] provide designated *"brushes"* that allow end-users to select text and perform specific generative functions on the selected text [149]. 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. 3.6) 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. 3.6b). For each brush, end-users can assemble cells to specify the brush's purpose (Fig. 3.6c), set multiple generators, and choose their preferred lens to show the generated outputs (Fig. 3.6e). 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.

LLM-Powered Brushes

To the right of the text editor (Fig. 3.6a), the user can create and manage the LLM-powered brushes (Fig. 3.6b). To use a brush, the user simply clicks on the corresponding button or, if a brush performs actions on user-selected

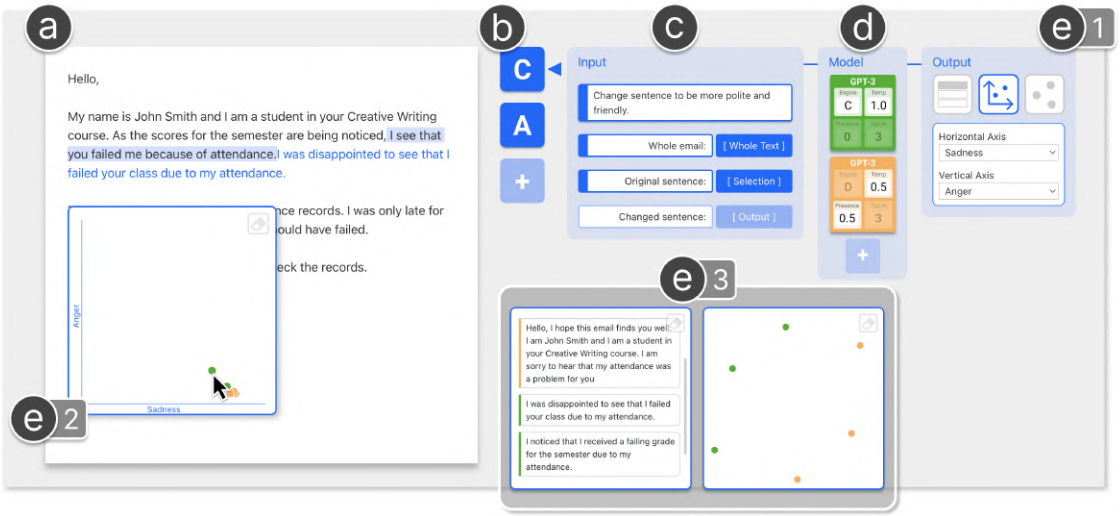

Figure 3.6: In the email composing interface, end-users can write an email in the text editor (a), and create dedicated LLM-powered brushes that can be configured to perform specific generative functions (b). For each of these brushes, the user composes an instruction prompt with cells (c), sets multiple 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.

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.

Composing the Prompt

Similarly to the copywriting system, end-users configures the input for the brush by using cells to represent individual specifications (Fig. 3.6c). 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.

Generator Set

Each brush can hold several generators (Fig. 3.6d), 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.

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. 3.6e-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. 3.6e-1).

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.

3.3.3 Story Writing Interface

LLMs have also been employed for story writing. Various interfaces [140, 386, 7, 249] 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. [81], parallel prototyping can prevent fixation and lead to higher quality and creative outputs.

By applying our framework, we introduce a story writing interface (Fig. 3.7) that partitions stories into cells to enable writers to assemble these into branching and parallel plotlines (Fig. 3.7b). In this interface, end-users can experiment with and compare configurations by linking cells to multiple generators (Fig. 3.7c) and linking multiple generators to the same lens (Fig. 3.7d-3).

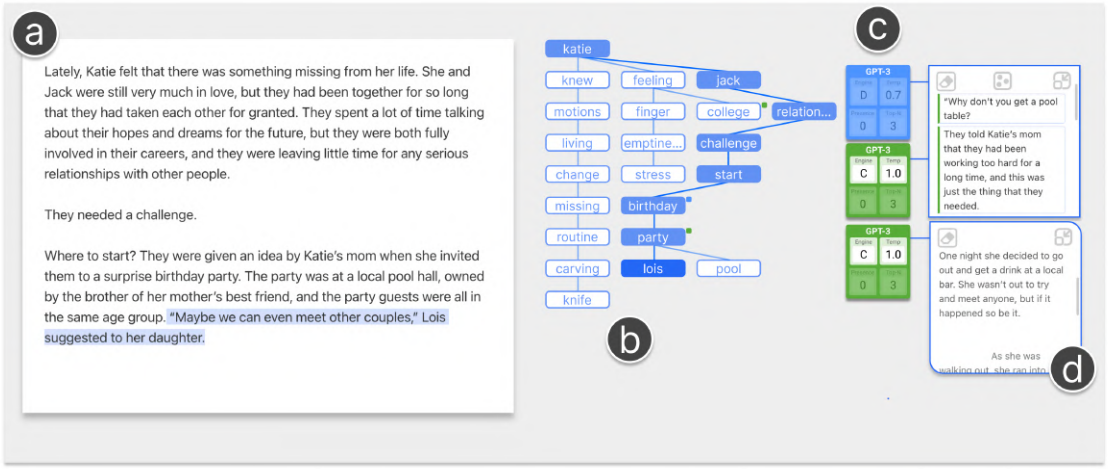

Figure 3.7: With the story writing interface, end-users can explore multiple, alternative plotlines. The user can create multiple plotlines by creating branching cells in a tree representation (b). Each cell contains a story sentence and is represented as a block enclosing a keyword extracted from the 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).

Branching and Parallel Paths of Cells

Next to the text editor, the user can view the tree representation of cells (Fig. 3.7b). 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. 3.7a), 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.

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. 3.7c). 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.

Lenses: List, Space, and Peek

The user can choose between three lenses in this interface: list (Fig. 3.7d-1), space (Fig. 3.7d-2), or peek lens (Fig. 3.7d-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.

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.

3.3.4 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 [361] was used to process the generations for sentence similarity [103], sentiment [284], and emotion [241]. The KeyBERT technique [121] was used to extract keywords for the story writing interface.

3.4 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?

3.4.1 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 3.8) 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-funding site[6]: a plant-based jerky, and a portable air conditioner. Participants were provided with the descriptions available on the

[3] https://docs.slatejs.org/
[4] https://openai.com/api/
[5] The study was conducted in early 2022 before the widespread adoption LLMs due to models like ChatGPT.
[6] https://kickstarter.com

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).

3.4.2 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.

3.4.3 Measures

For measures, we collected participants' survey responses after each advertisement. Similar to prior work [191, 386, 190], we asked participants to rate their agreement on a 7-point Likert scale (1: Strongly Disagree, 7: Strongly Agree) with the following statements:

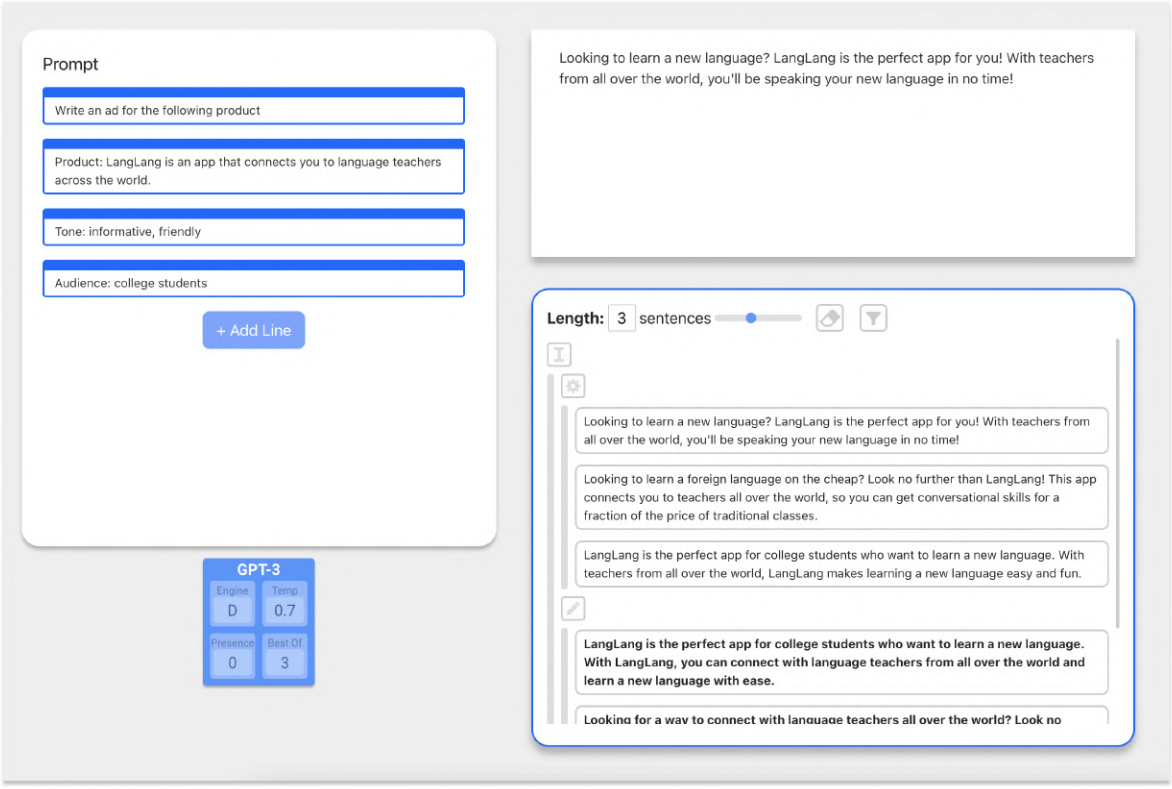

Figure 3.8: The baseline interface resembles our copywriting interface, but presents the users with only one cell per instruction line, one generator, and one lens.

- 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 [190] 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 [260]—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 [416], 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.

3.4.4 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 statistical 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).

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 [81].

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 [174].

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. [390], this challenge of identifying how to modify or add to LLM inputs (i.e., prompts) resembles the *selection* barriers in end-user programming [174].

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

	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 3.1: 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.

used *cells, generators, and lenses* to explore a wider portion of the generation space.

Subjective Perceptions

According to the survey results (Table 3.1), 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. 3.1). 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 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.

3.5 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.

3.5.1 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.

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 [338], screenplay [239], or essay [71]. 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.

3.5.2 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).

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 designing or

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

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 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.

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 [370], 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.

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.

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.

3.6 Discussion

In this chapter, 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.

3.6.1 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 [214], image examples for image-to-image (ITI) models [286], or music bars for music generation [219, 98]. 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 [133, 352] and storyboarding [394, 303]). 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).

3.6.2 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.

3.6.3 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 [169]. 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.

3.6.4 Further Development of the Framework

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

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 [89] 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 [251, 257].

Generators: Explainability of Generative Model Parameters

In our study, participants felt that changing 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 [61, 397, 348, 136].

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.

3.6.5 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. Finally, 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.

Chapter 4. Stylette: Styling the Web with Natural Language

This chapter presents the second example of text disentanglement during the execution phase of interactive alignment. Stylette is a natural language-based tool that allows novices to edit the styling of any website by simply speaking their intents. The tool disentangles the different interpretations of the user's intent to produce a palette of CSS operations that the user can test, apply, and combine. This chapter has adapted, updated, and rewritten content from a paper at CHI 2022 [163]. All uses of "we", "our" and "us" in this chapter refers to coauthors of the aforementioned paper.

4.1 Motivation & Contributions

The web is inherently malleable. Websites are rendered out of documents—HTML, CSS, and JavaScript code—which are transmitted to the user's browser and, thus, can be readily accessed and modified on the user side. This malleability allows users to improve their experiences on the web by personalizing pages [245], self-repairing existing issues [246], or even enhancing pages with additional features [139, 400, 326]. In addition, by sculpting others' creations, users can create their own new web pages [52, 278, 204]. The appeal of this malleability has led to the Greasemonkey [120] and Tampermonkey [36] plugins, which manage user scripts for these types of modifications, to collectively amass more than 10 million users. However, although such plugins allow users to install modifications designed by others, designing their own personal modifications may be out of reach for general end-users. To edit a web page's visual design or style, for example, users must be able to edit the underlying HTML and CSS files, but this requires an understanding of the code's language and structure. Thus, without the necessary expertise, most users are unable to mold websites into their own design.

To make the web more malleable for everyone, various end-user programming tools [246, 330, 182] have been designed to allow users with no expertise to directly manipulate a web page's visual design—abstracting away the underlying code. While these approaches allow users to focus on the visual representation, they require the user to manually perform several low-level operations (e.g., scrubbing on a color picker, typing in values) which can

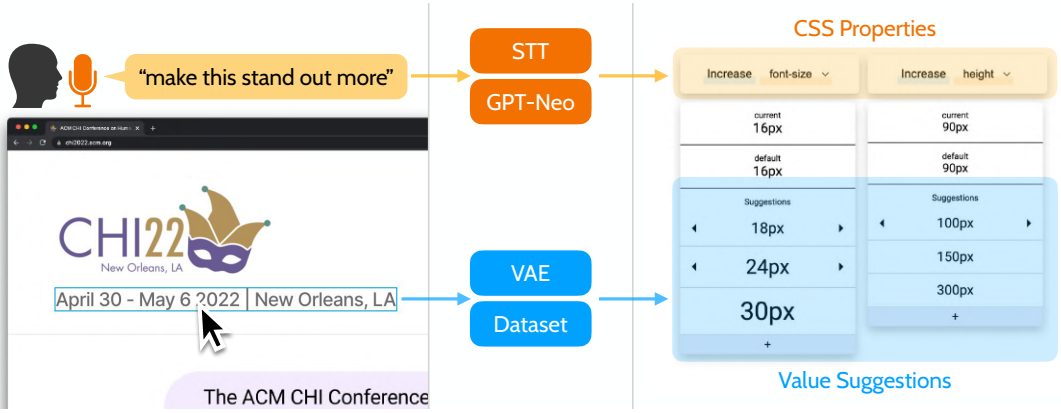

Figure 4.1: *Stylette* enables end-users to change the style of websites they visit by clicking on components and saying a desired change in natural language. A computational pipeline (1) transcribes the request and predicts plausible CSS properties with a large language model, and (2) encodes the clicked component using a convolutional neural network to identify and extract styling values from similar components in our large-scale dataset. These outputs are then presented in a *palette* that the user can use to iteratively change the component's style.

be tedious and effortful. Additionally, users must be able to decompose their high-level goals into the low-level operations supported by these tools—a task that inexperienced users frequently struggle with in other design-related tasks [188, 2]. Thus, to be able to easily transform a web page's design according to their goals, users require another level of abstraction.

Natural language interfaces allow users to perform complex, compound operations by simply saying or writing their intentions. The promise of this form of interaction has led to the development of various general-purpose voice assistants—e.g., Apple's Siri, Google Assistant, or Amazon's Alexa. In addition, task-specific natural language interfaces have also been designed to help inexperienced users perform complex tasks such as photo editing [188] or data visualization [102]. Similarly, if users could simply say what change they want to see, they could easily manipulate a web page without thinking about the underlying code or the low-level operations.

To investigate what language users would use when changing the style of a web page and how they would expect such changes to be presented, we conducted novice-expert sessions (N=8). In these sessions, novices used their voice to request changes on a web page's visual design and the expert, a developer, would then directly perform the changes using an in-browser developer tool. Our findings revealed that novices were frequently vague in their requests: omitting specific details (e.g., what color for the background), or using abstract terms that could not be clearly mapped to specific changes (e.g., "modern" or "vivid"). In addition to being vague due to inexperience, novices were also purposefully ambiguous as they wanted to explore the design space by seeing the expert's changes. Thus, novices expected the expert to make assumptions and provide a set of alternative changes that they could test and further iterate on.

Based on these findings, we designed Stylette, a natural language-based interface that assumes the user's intentions to provide a *palette* of web design properties and values. Stylette allows the user to modify a web component by clicking on it, and then saying or typing their desired change (e.g., "increase the size" or "make this cleaner"). Based on the user's input, the system provides a toolbox that contains (1) a set of CSS properties that could be changed to satisfy the request, and, (2) for each property, a set of alternative values to explore and sample. The user can then simply change the component by applying the different property values found in the toolbox. To generate these toolboxes, we designed a computational pipeline that processes and combines the two input modalities, natural language and clicks. Specifically, a GPT-Neo-based architecture predicts suitable CSS properties from the natural language request, and a variational autoencoder (VAE) model encodes the clicked-on component to extract the values of similar components from our dataset of 1.7 million components.

To evaluate Stylette, we conducted a between-subjects study (N=40) in which participants performed a design recreation task and an open-ended design task with either our system or DevTools, the Chrome Browser's developer tool. Our study revealed that Stylette helped participants perform styling changes 35% faster and with a higher success rate—80% of Stylette participants successfully recreated a design within the allowed time while only 35% succeeded with DevTools. Additionally, our system led participants to experiment with and familiarize themselves with a more diverse set of properties. As participants acquired more knowledge with Stylette, however, natural language interaction limited their productivity as they could not apply this knowledge to directly make changes themselves. These insights suggest a need for a hybrid approach: natural language interaction to initially support quick familiarization with a tool, and then gradually phasing in more direct interaction methods.

This chapter presents the following contributions:

1. Stylette: A novel system that allows users to change the design of websites by using natural language to express their goal, and then iterating with the set of alternatives presented by the system.

2. A computational pipeline that combines NLP and CV techniques to process a natural language request and a web component into a set of plausible CSS property and value changes.

3. Findings from a between-subjects study that reveals how natural language support can help novices familiarize with and perform a previously unknown design/coding task.

4.2 Formative Study

We conducted a formative study to investigate how novices would change the design of websites and how they would naturally request such changes. In this study, participants freely browsed through a website and requested styling changes by speaking aloud. One of the researchers, with several years of development experience, acted as an expert and made these changes on-the-go.

4.2.1 Participants

We invited 8 participants (5 female, 3 male), all of whom had no background in web development. Each participant sat alongside the expert or, if participating remotely, shared their screen through a video conferencing tool[1]. To reduce the time participants spent familiarizing themselves with a website and to prompt more realistic requests, participants chose a website they frequently visit for the study. Most participants chose either our university's web portal or its learning management system.

4.2.2 Study Procedure

During the study, participants were asked to examine the website, and request any styling changes that they want or could improve their future experiences on the site. On their own computer, the expert used the Chrome Browser's DevTools[2] to directly edit the CSS code. The expert would then share the edits and, if participants were not satisfied, they could ask for further edits. After around 30 minutes of editing, the participants were then asked a couple of questions about their experience. Sessions lasted a maximum of 40 minutes and participants made 8.38 requests on average.

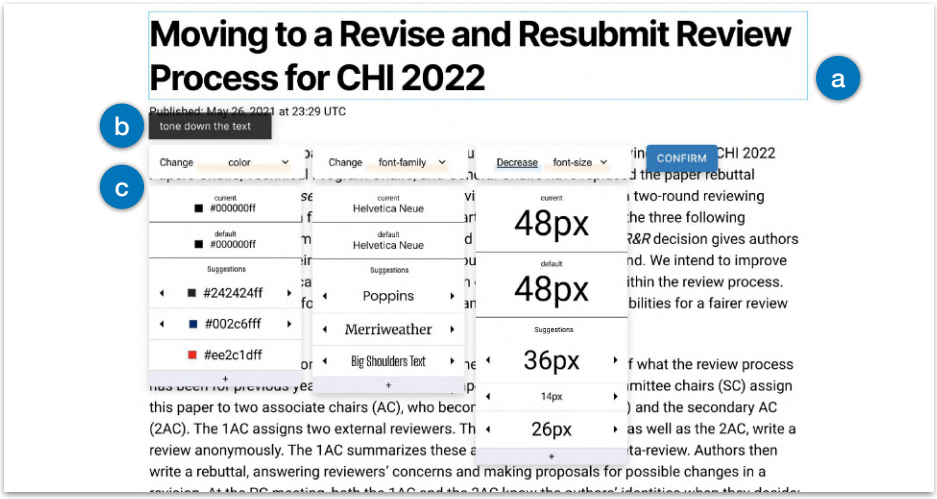

Figure 4.2: *Stylette* is shown overlaid on a website. When activated, the system shows a blue border (a) over components the user has hovered-on or clicked. After the user selects a component and records a request, Stylette transcribes the request (b) and displays a *palette* that contains CSS properties and values.

[1] https://zoom.us

[2] https://developer.chrome.com/docs/devtools/

4.2.3 Requests were Vague and Abstract

Despite being able to concretely specify which web component they wanted to edit, participants struggled to concretely explain how it should be changed. Participants generally relied on vague phrases (e.g., "more readable" or "emphasize this") or abstract terms (e.g., "modern", "vivid" or "dull") that did not immediately reveal what visual aspect of the component should be changed or how. Even if they were specific about which aspect to change, participants would also tend to be vague about the value to set for that aspect. For instance, a participant said "more transparent" without specifying how much more transparent.

We observed that the behavior of our participants was beyond not knowing the names of CSS properties—like the vocabulary problem observed in other tasks [100]. Participants also struggled to specify the visual aspects of the web components even without using the actual property names. For example, a participant requested a text component to be highlighted but, when asked if the text should be bolder or colored differently, they were unable to provide a definite answer. Participants explained that their hesitation was either because (1) they were unsure about which aspect to change, or (2) they could decide on an aspect but were not confident that it would "look good".

4.2.4 Assumptions Over Questions

To concretize the participants' vague requests, the expert asked questions to prompt further details. For example, when a participant asked to make a component "less tacky", the expert asked about what made it appear "tacky". While participants recognized how these questions helped them iteratively decompose their goals, they found this back-and-forth to be tedious. As participants were unsure about the details, they did not want to dedicate the mental effort to ponder about the details and, instead, expected the expert to assume the details for them. They mentioned that it would be easier to distinguish what they liked or disliked if the expert made these assumptions and presented a visual result. Additionally, instead of one outcome for each request, participants wanted various options for the same request in order to explore the design space.

4.2.5 Natural Language is Not a Panacea

For most participants, the use of voice or natural language was a major positive aspect about interacting with the expert. Participants mentioned how it was "comfortable" to use natural language to simply explain what they wanted to change. However, while they felt that natural language helped to get the editing process started, participants desired more direct control when iterating on edits. Specifically, when deciding on a value for a property, they felt frustrated about having to test different values by turn-taking with the expert. Instead, participants wanted to be presented with widgets that allowed them to test alternative values by themselves.

Based on the study insights, we derive the following design goals:

- DG1: Interpret vague requests to present plausible changes.
- DG2: Provide multiple alternative properties and values that could satisfy one request.
- DG3: Allow users to directly iterate on the details for a change.

4.3 Stylette

Based on our design goals, we present Stylette (Fig. 4.2), a system that enables end-users to change the visual design of any website by simply clicking on a component and saying what change they want to see. The system interprets the user's request through an NLP pipeline trained on vague language (DG1) to present a *palette* that

consists of multiple CSS properties (DG2) that could be changed to satisfy the request. To iteratively edit each property, the user can directly adjust values and experiment with various suggestions extracted from a large-scale dataset (DG2, DG3). Stylette is implemented as a Chrome Extension and, using a method similar to Tanner et al.'s [330], it saves the user's changes in the browser's memory so that they persist when the user returns to the page.

4.3.1 User Scenario

To illustrate how Stylette can be used, we follow Sofia, a sociologist preparing for a paper submission to CHI 2022. While Sofia frequently visits the conference's website to check for submission details or recent news, she feels that the design can make it challenging to look for and read the contained information. As she has no web development experience, she decides to use Stylette to make some styling changes to the website.

Selecting a Component

In the frontpage of the CHI website, Sofia feels that the header text is overemphasized and prevents several news posts from being seen in one glance. To start editing, she clicks on the Stylette icon on her extension toolbar. Now, she can select components to edit so she clicks on the first header in the page (Fig. 4.2a).

Making a Verbal Request

Once the component is selected, Stylette overlays a transcript box on the website, prompting Sofia to say her request. To do so, she holds down the **Ctrl** key and says: *"tone down the text"* (Fig. 4.2b). After releasing the **Ctrl** key and a short processing period, the transcript box now displays a transcript of what Sofia said. In case the transcription is wrong, Sofia can correct it by typing directly on the box and pressing **Enter** to process the corrected

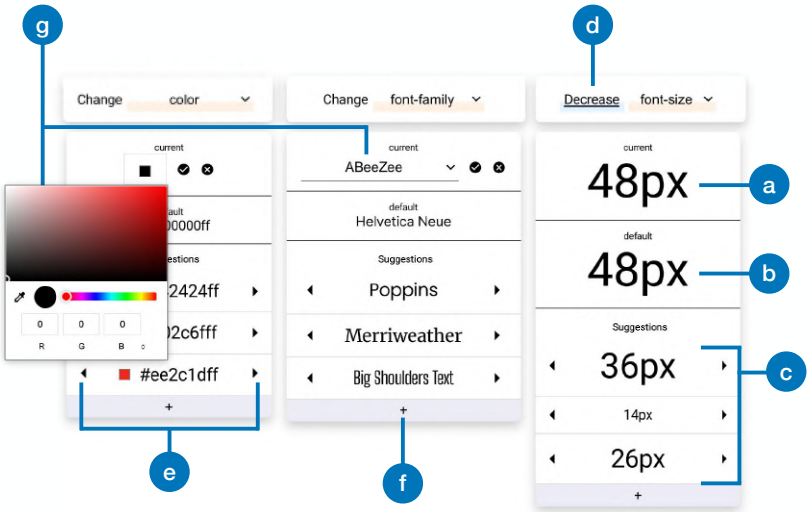

Figure 4.3: For each property, the *palette* presents the current value (a), the default or original value before any changes (b), and a list of suggested values (c). For numerical values, the palette presents suggested values that are either larger or smaller than the current value based on the system's prediction (d). To see other similar suggestions, the user can click on the arrows next to a suggested value (e). To see different suggestions, the user can click on the "+" button (f). The user can also click on the current value to reveal widgets to manually set values (g): input box for numerical properties (e.g., *font-size*), drop-down menu for nominal properties (e.g., *font-family*), or color picker for colors.

transcript. In addition, a *palette* is now presented, showing three different CSS properties that Sofia can edit to satisfy her needs (Fig. 4.2c): she can make the text smaller with *font-size*, change it to a slimmer *font-family*, or apply a lighter *color*.

Iterating with the *palette*

Under each property, the *palette* presents a list of values: the current value for the property (Fig. 4.3a), the default or original value (Fig. 4.3b), and a set of value suggestions (Fig. 4.3c). For properties with numerical values, like *font-size*, the system also interprets whether the user wants to increase or decrease the current value (Fig. 4.3d) and provides suggestions accordingly. As Sofia hovers over the suggested values for *font-size*, Sofia can see how the header would look with that *font-size*. After finding one she feels satisfied with, she clicks on it to apply that change. If Sofia actually wanted to increase the *font-size* and the system gave an incorrect prediction, she could click on "Decrease" next to the property name to switch it to "Increase" and the suggestions would change accordingly. If she wanted to change another property similar to *font-size*, she could also click on the property name to see a drop-down of other properties with similar names (e.g., *font-style*, *font-weight*).

After setting the *font-size*, Sofia also notices the *color* property. As she feels that this could also be toned down a bit, she clicks on the lighter black color ("#242424ff") in the suggestions. After seeing this change, she feels that the header's color should be even lighter, so she clicks on the arrows next to that suggestion (Fig. 4.3e) to see other similar suggestions. Going through the carousel, she finds a color that she likes so she clicks on it. If she is unsatisfied with the suggestions, she can see other different suggestions by clicking on the "+" at the bottom (Fig. 4.3f), or manually set her own value by clicking on the current value to expose manual change widgets (Fig. 4.3g).

4.3.2 Pipeline

To support the interaction presented in the scenario, we present a computational pipeline that processes the two input modalities, voice and click, to generate the *palettes* (Fig. 4.4). For voice, the audio is recorded and automatically transcribed. For clicks, a screenshot of the component selected by the user is automatically captured. These inputs are then processed separately by the computational pipeline.

Processing Natural Language

Our pipeline's NLP module takes the transcribed request, and predicts relevant CSS properties and the direction of the change (e.g., increase, decrease, or neither). For this purpose, we employ the 2.7 billion parameter version of the GPT-Neo model [38], an open-source implementation of OpenAI's GPT-3 model [45]. With a well-crafted prompt and a small number of examples, these models have been shown to achieve high performance on previously unseen tasks. However, hand-crafting prompts can be a time-consuming and very imprecise process—small alterations can lead to significant differences in performance.

Architecture: As an alternative, we implement the P-tuning technique [216] that automatically searches for a prompt with high performance. In this technique, prompts are composed by concatenating *pseudo-tokens* to the natural language input and training the embeddings for these pseudo-tokens. In our pipeline, we use 12 pseudo-tokens. For training, we template the prompt as [$P_{1:4}$,R,$P_{5:8}$,C,$P_{9:12}$,D], where $p_{1:12}$ are the pseudo-tokens, R is the natural language request, C is the CSS properties separated by commas, and D is the change direction (i.e., "increase", "decrease", or "none"). During inference, we template the prompt as shown in Figure 4.4 ("Concatenate"): [$P_{1:4}$,R,$P_{5:8}$]. This templated prompt is passed as input to the model and the model's output is controlled to generate at least three CSS properties—to provide multiple alternatives to users ("Input" and

Model	CSS Property Prediction				Direction
	Acc.	Pre.	Rec.	F1	Acc.
P-tuning	0.557	0.670	0.761	0.653	0.819
Hand-crafted	0.509	0.623	0.648	0.585	0.413

Table 4.1: With trained P-tuning, the GPT-Neo model achieved higher performance when predicting CSS properties and change directions, when compared to using a hand-crafted prompt as input.

"Generate" in Fig. 4.4). Then, the generated CSS properties and the remaining pseudo-tokens are concatenated to the initial prompt, and this result is passed to the model again to generate the change direction.

Dataset: Another merit of the P-tuning technique is that it only requires a small amount of data for training. To train our pseudo-tokens, we created a small-scale dataset consisting of 300 triplets of (1) vague natural language requests, (2) CSS property sets, and (3) change directions. As a first step in creating this dataset, we requested 29 web developers to each write three hypothetical vague requests that a user could ask when wanting to change a website's design. Then, each developer looked at the requests written by another person and wrote CSS properties to change and the direction for the change that could satisfy each request. We removed requests that were too specific (e.g., included property names), and added requests from our formative study and system's pilot studies. The CSS properties in this initial data are the ones supported in our system (Table 4.2). We then expanded the dataset by automatically augmenting the initial data with synonym/antonym replacement [401, 224], and/or backtranslation [297]—one of the authors checked and corrected the augmentations. Finally, as performance can deteriorate significantly due to class imbalance [405], we ensured that each CSS property appeared in at least 10% of the requests—the representation of CSS properties in the dataset is shown in Table 4.2. After augmentation and balancing, we finalized our dataset of 300 triplets.

Training: In the training process, we used 200 triplets for training and validation (80%-20% split), and reserved 100 for testing. The pseudo-tokens were trained on the generative loss from the GPT-Neo model with the

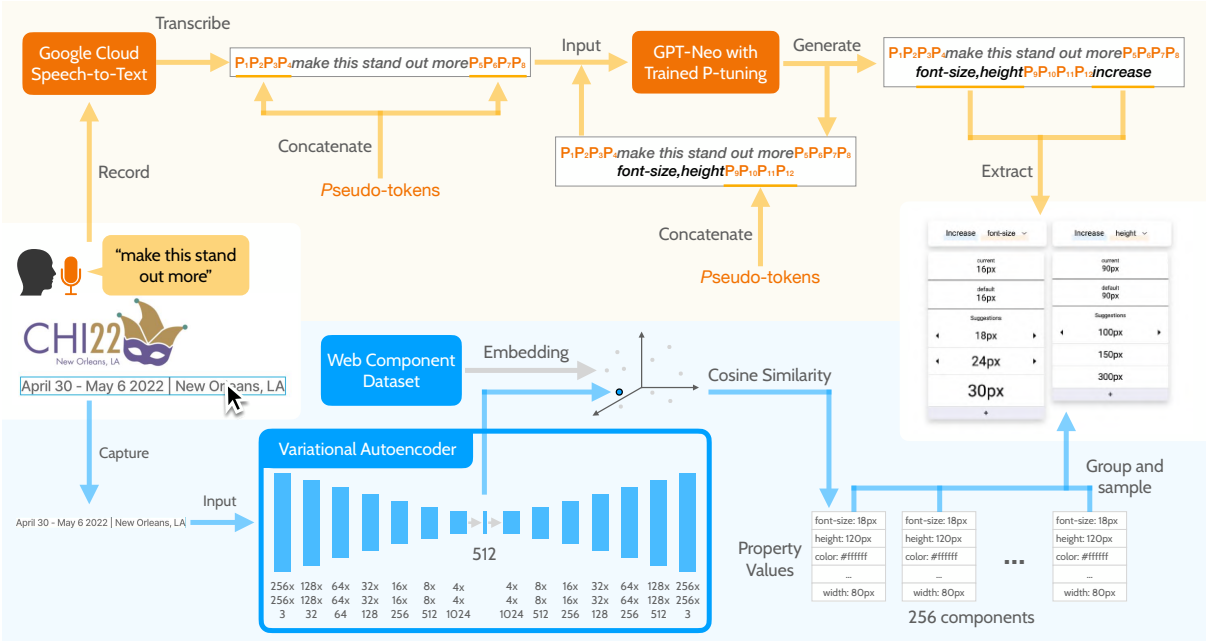

Figure 4.4: Our computational pipeline integrates a natural language processing (NLP) module (top, orange) and a computer vision (CV) module (bottom, blue). The diagram illustrates the pipeline at inference time—processing user's input of natural language and clicks to generate a set of CSS property alternatives and value suggestions.

Adam optimizer, until early stopping on the validation loss. We used an initial learning rate of 0.0001, batch size of 8, weight decay value of 3e-7, and gradient clipping value of 5. When compared to the model with our best hand-crafted prompt, GPT-Neo with trained P-tuning achieved a higher F1-score when predicting CSS properties and higher accuracy when predicting change direction (Table 4.1). Additionally, the recall with P-tuning exceeds 75% which suggests that, for the average request, the model will likely return most of the properties that the user might need.

Processing Web Components

As our formative study revealed, users can struggle when deciding on a value for a change (e.g., what color for the background) and may benefit from seeing various alternatives. The components in other websites can be a rich source for these alternatives. However, as the style of a component depends on what that component represents (e.g., the *font-size* for a header vs that for a paragraph), selecting random components would not lead to sensible and useful alternatives. Thus, it would be more beneficial to identify components in other websites that are similar to the one the user wants to change. Similarity could be measured by calculating property differences and aggregating these into one measure, but, as properties differ in the scale and type of values, this requires the difference and aggregation calculations to be carefully formulated.

Architecture: As an alternative, we leverage a variational autoencoder (VAE) model [171] ("Variational Autoencoder" in Fig. 4.4) to automatically learn a concise representation of the visual features of components. In our pipeline, we use this VAE model to encode the screenshot image of a component into a 512-dimensional vector. Through cosine similarity, this vector is then compared to the vector representations of all the components in our large-scale dataset to identify 256 similar components and retrieve their property values ("Cosine Similarity" in Fig. 4.4). To provide coarse diversity but also more fine-grained alternatives, the *palette* presents suggested values in two levels: (1) different values as separate rows, and (2) similar values as a carousel in the same row. To support this, the pipeline groups the retrieved values according to specific rules ("Grouping Method" in Table 4.2) and each group represents a suggestion row. Then, a maximum of 10 values are randomly sampled for each group and these alternatives are presented through the carousel. For color-related properties and the *font-family* property, users in the pilot studies wanted more diverse values so, for these properties, we populate other suggestion groups by retrieving the values from random components in the dataset.

Dataset: Although datasets for mobile UI components [56] or for whole web pages [178] are available, there are none for individual web components. Thus, to train the VAE model, we constructed our own dataset. We first compiled a list of websites from various sources: the S&P500, the Webby Awards [25], and the Open PageRank dataset [77]. We removed any websites that (1) could not be accessed, (2) had very similar URLs, or (3) had less than 16 components. This led to a final list of 7,565 websites. For each website's main page, we used a crawler to capture each component's CSS properties and screenshot image. After removing components that were less than 10 pixels wide or tall, the final dataset consisted of 1,761,161 components.

Training: The VAE model is composed of six convolutional layers for encoding, one linear layer as a bottleneck, and six transposed convolutional layers for decoding. The dimensions of the outputs at each layer are shown in Figure 4.4 ("Variational Autoencoder"). During training, the image of a component is encoded into a vector using the encoding and bottleneck layers, and then this vector is passed through the decoding layers to recreate the image. The model is trained to maximize the evidence lower bound (ELBO) value between the original image and the recreated image. We trained our model for 3 epochs with an Adam optimizer, using a learning rate of 0.0001 and batch size of 256.

CSS Property	Grouping Method	Percent
height	Interval binning (N=20)	11.7%
width	Interval binning (N=20)	11.7%
margin	Interval binning (N=10)	11.3%
padding	Interval binning (N=10)	12.9%
color	K-means clustering (N=6)	12.9%
background-color	K-means clustering (N=6)	12.5%
opacity	Interval binning (N=2)	10.4%
font-size	Interval binning (N=10)	13.3%
font-family	Google Fonts categories (N=5)	13.8%
font-style	Nominal value	11.3%
font-weight	Interval binning (N=10)	11.7%
text-align	Nominal value	11.3%
text-decoration	Nominal value	11.3%
border-width	Interval binning (N=10)	11.3%
border-color	K-means clustering (N=6)	11.3%
border-radius	Interval binning (N=10)	11.3%

Table 4.2: The CSS properties supported by Stylette. The table presents the representation of each property in the natural language request dataset as a percentage. Each row also shows how values for a property are grouped when suggested to the user: interval binning into N equally-spaced intervals, K-means clustering with the elbow method, based on the categories from Google Fonts, and no grouping for properties with nominal values.

4.3.3 Implementation

We implemented the interface of Stylette as a Chrome Extension, using JavaScript, HTML, and CSS. For the backend, we used a Node.js server to pre-process requests from the interface and transcribe the audio with the Google Cloud Speech-to-Text API[3]. To serve the computational pipeline, we used a Flask server running with DeepSpeed[4].

4.4 Evaluation

We conducted a between-subjects study where we compared Stylette against the Chrome Browser's DevTools, a tool widely available for general end-users to edit websites with. The study was composed of (1) a well-defined task of redesigning a website to look like a given outcome, and (2) an open-ended task of styling a website to follow the design direction of provided references. We designed these two tasks to investigate how Stylette helped participants style components when (1) they have a clear idea about how it should change, or (2) they only have a vague sense of direction. Specifically, we pose the following research questions:

- RQ1. How does Stylette help novice users find the CSS properties required to perform desired styling changes?
- RQ2. Can Stylette encourage novices to perform a greater number of changes and use more diverse CSS properties?
- RQ3. How does novices' usage of Stylette affect their self-confidence regarding their own web designing abilities?

[3] https://cloud.google.com/speech-to-text
[4] https://www.deepspeed.ai/

4.4.1 Participants and Apparatus

We recruited 40 participants (11 female, 29 male; age M=21.5 and SD=3.05) who all reported having no previous experience with web design or coding (no knowledge of HTML and CSS). We also verified that participants were relatively fluent in spoken English to reduce frustration due to the performance of speech-to-text technologies. Participants were divided into two equally-sized groups and each group was assigned to use either Stylette or Chrome DevTools. As six participants mentioned having other prior design experiences and this could affect performance (e.g., the term "padding" is used in other design tasks), they were also equally split into each condition. To simulate a realistic setting, participants who used Chrome DevTools were also allowed to freely use search engines to find resources and information. The study lasted a maximum of 90 minutes and participants were compensated with 30,000 KRW (approximately 26 USD).

4.4.2 Study Procedure

The study took place face-to-face, strictly following the COVID-19 guidelines: participants had to wear masks and plastic gloves, and their temperature was checked before sessions. Each participant was provided with a computer with a Chrome browser installed and their assigned tool, Stylette or DevTools, already opened. After reading and signing the informed consent form, participants were first provided with a brief walkthrough of their assigned tool and were then allowed to test the tool for a total of 5 minutes. After this, participants completed a short pre-task survey.

After the survey, participants started Task 1. Participants were tasked with using their assigned tool to redesign our institute's "About" web page[5] to look as close as possible to a provided final design. This final design was provided as a before-after image with circling around components to change and labels showing how many properties to change for each component. Natural language explanations of the changes were not provided to prevent biasing the language used by Stylette participants. The task involved changing 14 different components and

Tag	Properties to Change	Success Range
h2	font-size (FSz)	80px - 120px
p	font-weight (FW)	700 - 900
span	background-color (BgC) color (C)	(0.0, 0.0, 0.6) - (0.2, 0.2, 1.0) (1.0, 1.0, 1.0) - (0.8, 0.8, 0.8)
video	border-radius (BR)	60px - 100px
button	border-width (BW) border-color (BC)	6px - 10px (0.8, 0.4, 0.0) - (1.0, 0.8, 0.2)
div	text-align (TA)	"center"
h2	font-style (FSt)	"italic" or "oblique"
button	padding (P)	30px - 60px
img	width (W)	600px - 800px
h3	font-family (F)	Any in "cursive" category
h3	text-decoration (TD)	"underline"
div	margin (M)	80px - 120px
img	height (H)	350px - 450px
img	opacity (O)	0.3 - 0.7

Table 4.3: List of the components that participants had to change during Task 1, in the order that they had to be changed. For each component, the table shows its tag type and the properties that had to be changed. For the properties, the list also shows their abbreviated names (which are used hereafter), and the range of values that were accepted as successful changes (color values shown as RGB triplets).

[5] https://www.kaist.ac.kr/en/html/kaist/01.html

all of the 16 CSS properties supported by our system (Table 4.3). Participants were asked to change the components in the order that they appeared in the website, but were allowed to skip challenging components and come back to them later. A researcher verified that a component had been successfully changed once the values of the correct properties were within the accepted success range ("Success Range" in Table 4.3). Participants had 30 minutes to successfully change all the components. After the task, participants completed a short survey.

After Task 1, participants started Task 2 after a 5-minute break. To ensure that all participants started Task 2 with the same amount of knowledge, those that did not complete Task 1 were first shown how to perform the changes that they did not complete. The aim of Task 2 was to investigate how participants used their assigned tool when only provided with a vague direction for changes and allowed greater flexibility. Participants were tasked with changing a given website such that it followed the design direction of four reference websites (Fig. 4.5). These references were chosen as they shared a similar modern aesthetic, but also differed in how their content was structured. Participants were given 25 minutes for this task. After this task, participants completed a short survey. Finally, a short interview was conducted asking participants about their experiences during both tasks.

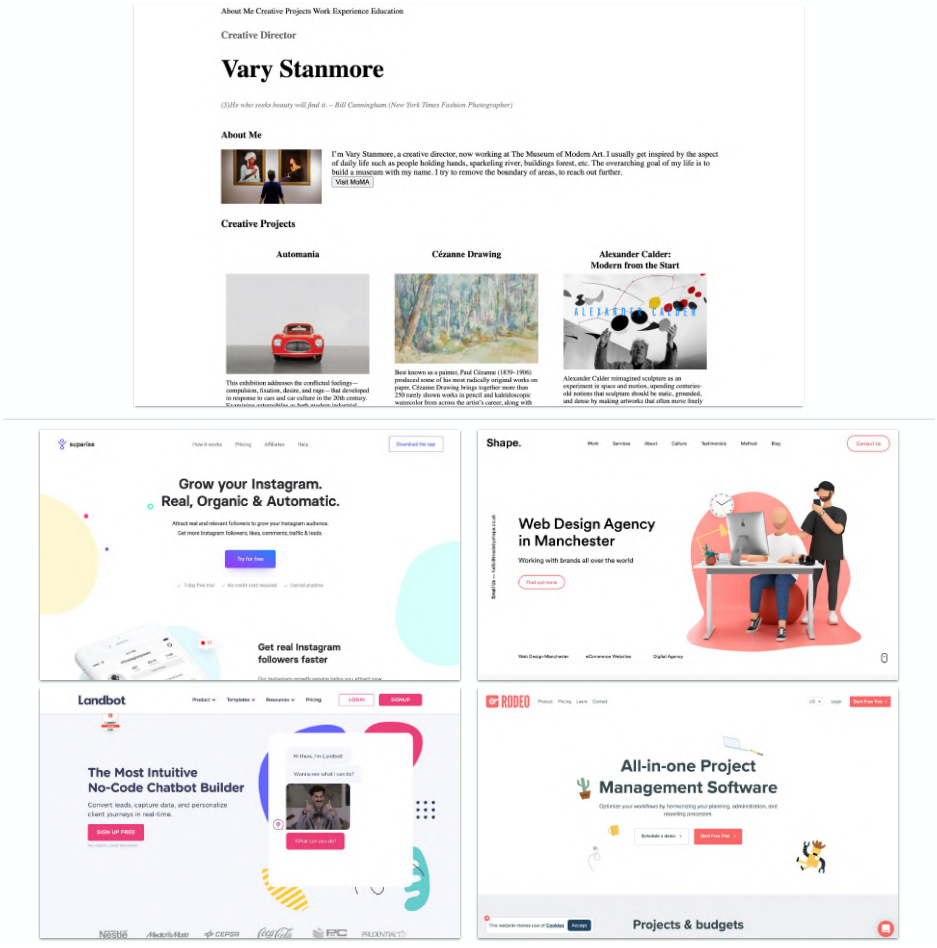

Figure 4.5: (Top) The website that participants styled in Task 2 mimics the portfolio of a creative director for a museum. The website only has basic styling to encourage participants to be creative and make many changes. (Bottom) The four reference websites that were provided during the task: Suparise (https://suparise.com), MadeByShape (https://madebyshape.co.uk), Landbot (https://landbot.io), and Rodeo (https://getrodeo.io).

4.4.3 Measures

We collected responses to the pre-survey and the post-surveys after each task. All of the surveys contained four questions asking participants to rate, on a 7-point Likert scale, their self-confidence with respect to their ability to (1) perform a website design changing task, (2) plan design changes, (3) iterate on changes, and (4) use the given tool. We averaged the responses to these questions to derive one score for self-confidence. The two post-surveys included the six questions from the NASA-TLX questionnaire [129] to measure participants' perceived workload.

We also quantitatively measured task-related metrics. For Task 1, we measured the time taken to successfully change each component and to complete the whole task. We hypothesized that Stylette would help participants find properties faster, and therefore complete changes in less time. Although all participants had no previous web design experiences and those with other design experiences were equally divided into each condition, individual design interest and skill could still affect the quality of the final designs in Task 2. Due to this reason, we did not rate these designs and, instead, we measured how many property changes were made in total. We hypothesized that Stylette participants would make more changes as they could explore diverse properties and values. A value close to 0 indicates equal usage, spread across various properties, and one close to 1 indicates unequal usage, few properties used excessively.

For qualitative data, we analyzed participants' responses during the short interviews to understand their perceptions of the given tool and how they leveraged it for their purposes. We also iteratively coded the requests used by Stylette participants in Task 2 to classify them according to the vagueness of their content and language.

4.5 Results

Our results demonstrated that Stylette helped participants perform styling changes faster and with greater success in Task 1, but it did not enhance productivity in Task 2. For the statistical analysis of each measure, we first conducted a Shapiro-Wilk test to determine if the data was parametric (noted with "P") or non-parametric (noted with "NP"). When comparing between conditions, we used an independent t-test (if parametric) and a Mann-Whitney U test (if non-parametric). When comparing between tasks within the same condition, we used a

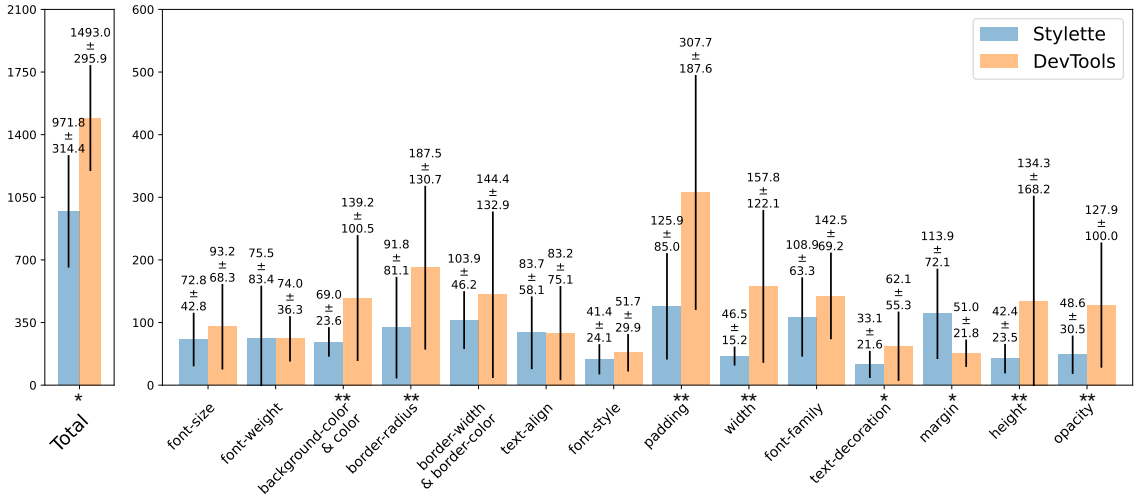

Figure 4.6: The average time taken for participants to successfully change each component using Stylette or DevTools. Each component is represented with the abbreviated names of the properties changed (Table 4.3). For each property, the figure shows if the difference in time taken for each condition was statistically significant (*: p <.05, **: p < .01).

Task	Condition	Mental	Physical	Temporal	Effort	Performance	Frustration
1	**Stylette**	3.90 (1.41)	2.55 (1.57)	3.45 (1.73)	3.15 (1.79)	5.45 (1.10)	3.00 (1.52)
	DevTools	4.35 (1.42)	1.65 (0.93)	4.50 (0.95)	4.00 (1.45)	4.85 (1.53)	2.25 (1.21)
	p	0.14	**0.02**	**0.01**	**0.03**	0.11	**0.05**
2	**Stylette**	4.75 (1.29)	2.90 (1.71)	4.35 (1.66)	4.25 (1.48)	4.05 (1.43)	3.55 (1.47)
	DevTools	4.90 (1.21)	2.00 (1.45)	4.55 (1.50)	4.55 (0.83)	4.45 (1.39)	2.65 (1.50)
	p	0.34	**0.03**	0.69	0.23	0.24	**0.03**

Table 4.4: For Task 1, participants' average ratings on the perceived workload questions (NASA-TLX) showed that temporal demand and effort were significantly lower with Stylette, but physical demand and frustration were significantly higher. For Task 2, physical demand and frustration were still rated significantly higher with Stylette, but temporal demand and effort no longer differed significantly.

paired t-test (if parametric) and a Wilcoxon signed-rank test (if non-parametric).

4.5.1 Task 1: Well-Defined Task

To answer RQ1, we analyzed participants' performance in Task 1 (i.e., time taken and success rate to perform the given changes). We additionally measured participants' perceived workload.

Performance

Overall, participants using Stylette significantly outperformed DevTools participants in this task (Fig. 4.6). While only 7 out of 20 DevTools participants completed all changes, 16 out of 20 Stylette participants completed the task. Additionally, when comparing only those who completed the task, Stylette participants (M=971.8s, SD=314.4s) completed the task in 35% less time than those that used DevTools (M=1493.0s, SD=295.9s, t=-3.72, p=0.001, P). Comparing the time taken to successfully change each component revealed that DevTools participants struggled significantly with specific properties (e.g., *border-radius* (BR) and *padding* (P) in Fig. 4.6). These struggles generally involved two scenarios: (1) vague search queries led to unhelpful results, or (2) the name of a CSS property did not immediately reveal its visual function.

To illustrate the first scenario, several participants tried queries like *"enlarge the border in CSS"* when searching for the *padding* property, but this only returned results for *border-width*—the search engine took them "too literally" (D2, D7, D9, D14). In other cases, participants' vague queries returned search results for more advanced changes beyond their needs. For example, to adjust the *height* or *width*, participants searched "resize image in CSS" but this returned results about "responsive images". In contrast, as our system was trained on vague requests and presents multiple properties for one request, Stylette participants had more success using similarly vague language—requesting *"enlarge the border"* to the system returned *padding* among the options.

The second scenario involved properties with names that could be unclear for novices, such as *border-radius* or *text-decoration*. In these situations, the properties were frequently found in the search results, but DevTools participants would overlook them as they could not immediately visualize the functions from the names or mismatched with their mental models. However, hovering over the suggested values allowed them to quickly use and test the functions of properties. S5 mentioned: *"By applying [the recommendations], I could understand what [visual] concept the [margin and padding] were related to"*.

Perceived Workload

In Task 1, responses to the NASA-TLX questions revealed that the effect of Stylette on perceived workload was mixed (Table 4.4). Stylette participants reported experiencing significantly less temporal demand (U=118.0, p=0.0118, NP) and effort (U=133.0, p=0.0336, NP) than those that used DevTools. DevTools participants felt significant time pressure due to the lengthy and effortful process of thinking about what to search, skimming through search results, and reading resources. In comparison, Stylette participants could simply say something and look through the three to five properties presented by the system.

However, Stylette participants also experienced significantly higher frustration compared to DevTools participants (U=139.5, p=0.0472, NP). According to participants, this frustration was partially attributed to the fact that the coupled AI algorithms (i.e., speech-to-text and property prediction) could both fail. For example, as they did not notice the transcription errors, several participants were confused when concrete requests (e.g., "underline text") did not return the correct properties. Other participants were overly preoccupied with the transcription and immediately corrected any errors—failing to notice that the system had already returned desired properties. When fixing errors, participants also had to alternate between modalities (i.e., voice, text, and clicks) which could explain why Stylette participants reported feeling a higher physical demand (U=127.0, p=0.0187, NP).

4.5.2 Task 2: Open-Ended Task

To answer RQ2, we evaluated participants' productivity in Task 2 (i.e., how many changes were made and whether varied properties were used). As in Task 1, we also analyzed perceived workload. Samples of the participants' final designs (Fig. 4.7) show their creativity and how they each focused on different aspects of the website.

Productivity

While participants in the Stylette condition (M=42.85, SD=12.18) made more property changes than those in the DevTools condition (M=39.30, SD=12.71), this difference was not statistically significant (t=0.901, p=0.3729, P). The lack of a statistical difference could be attributed to the benefits and drawbacks of each tool's interaction method. DevTools participants spent more time searching for information, but, once they had the required knowledge, they

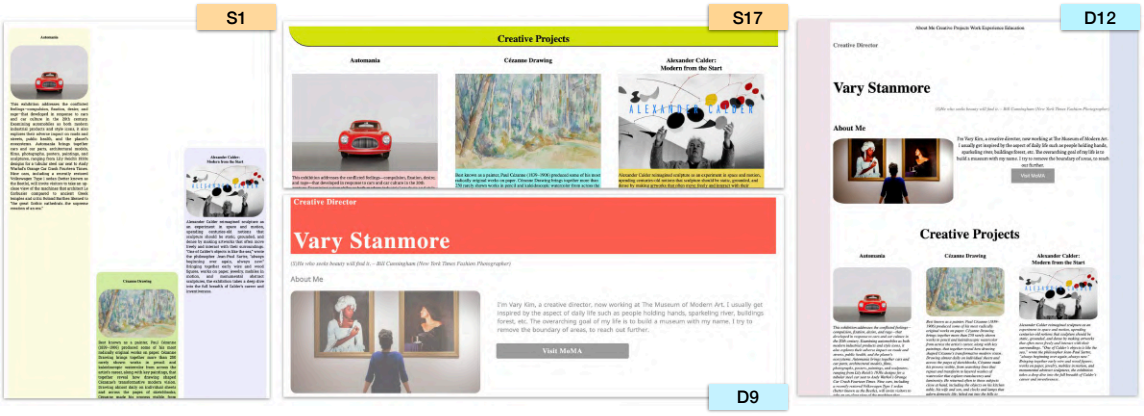

Figure 4.7: Sample of designs created by Task 2 participants. S1 used *padding* to spread content vertically such that each item would appear gradually as the user scrolls down the page. S17 serendipitously found the *border-width* property and used it to add a "shadow" to the container for the "Creative Projects" subheader. D9 used *opacity* in several components to lighten the web page's content. D12 increased the *border-width* and added *border-color* to add colored bars on the sides of the page.

could directly make changes. Stylette participants could use natural language to easily find properties, but, even if they already knew which property to change, they expended time waiting for the system to process requests and fixing any AI-related errors. Several participants (S5, S6, S7, S15) noted that, after learning the properties in Task 1, they wanted to directly change the properties in Task 2—without using natural language.

Additionally, as Stylette presents other options in the *palette*, participants appeared to spend additional time browsing through them. While this exploration could increase effort, it also appeared to encourage familiarization with a wider range of properties. The Gini index for property usage shows that Stylette participants tried various properties (M=0.292, SD=0.045) while DevTools participants mostly stuck with a few properties that they were accustomed to (M=0.325, SD=0.052, t=-2.169, p=0.0364, P). Beyond encouraging experimentation with more properties, in some cases, the system also led participants to serendipitously find alternative uses for known properties. S17 mentioned, *"Accidentally I just found [border-width] while trying to change the radius [so I changed it] and it shows a shadow effect that looks really, really good."* (design shown in Fig. 4.7).

Perceived Workload

Similar to the results of Task 1, participants in the Stylette condition reported experiencing higher physical demand (U=131.5, p=0.0278, NP) and frustration (U=129.0, p=0.0258, NP) than those in the DevTools condition (Table 4.4). Unlike Task 1, however, Stylette participants no longer reported feeling significantly less temporal demand or effort. It is plausible that, due to the open-ended nature of Task 2, Stylette participants now spent more time and effort exploring the design space through the alternatives presented by the system—Gini index results support this explanation.

Usage Patterns of Stylette

As Task 2 allowed for more flexible and natural use, we also analyzed participants' usage of Stylette during this task. Participants issued an average of 36.8 requests (max=58, min=18, SD=11.2) and the requests had an average length of 3.2 words (max=12, min=1, SD=1.2).

Type of Request	Description	Examples	Percentage	Q1	Q4
Property Specific (PS)	Specific property name expressed in request.	*"change background color"* *"align text in the center"*	48.1% (352)	46.6%	56.0%
Property Partial (PP)	Property name partially expressed in the request.	*"add border"* *"change the font"*	35.3% (258)	35.2%	34.7%
Property Vague (PV)	Property name not clearly apparent in the request.	*"make this bigger"* *"increase the spacing"*	11.5% (84)	11.9%	4.7%
Property Total	-	-	94.9% (694)	93.8%	95.3%
Value Specific (VS)	Specific value expressed in the request.	*"change to dark grey color"* *"increase font size to 14 px"*	11.4% (83)	14.0%	9.8%
Value Vague (VV)	Vague direction given for the value.	*"decrease the height"* *"make the edges rounder"*	20.0% (146)	25.9%	15.0%
Value Total	-	-	31.2% (229)	39.9%	24.9%
Abstract (A)	Request with abstract description of a change.	*"make it look more stylish"* *"make it more playful"*	3.3% (24)	3.6%	3.1%

Table 4.5: Coding of the participants' requests during Task 2. Requests can either mention both properties and values, only properties or only values, or be abstract. The percentage of requests for each category are shown. The table also shows the percentage for each category for the first quartile (Q1) and last quartile (Q4) of participants' requests.

Request	Type	Expected	Predicted				
"change the font family to Helvetica" (S7)	PS & VS	FF	**FF**	FSz	FSt	FW	
"increase padding" (S8)	PS & VV	P	BW	M	**P**	W	
"change text color" (S6)	PS	C	BgC	**C**	FSt	O	TD
"change the picture radius to 24" (S18)	PP & VS	BR	BC	**BR**	C	FSz	W
"increase the size" (S16)	PP & VV	FSz	BR	BW	H	P	W
"change borders" (S11)	PP	BW	BC	BR	**BW**	C	W
"make it go in the middle" (S15)	PV & VS	TA	H	M	P	W	
"add some spacing at the bottom" (S2)	PV & VV	P	BR	H	M	**P**	
"change the distance" (S5)	PV	M	BW	C	H	P	W
"make this modern" (S19)	A	FF	BW	H	M	P	W

Table 4.6: A sample of participants' requests in Task 2, ordered from most specific to most vague/abstract. For each property, the table shows the request type, the property expected by the user, and the properties predicted by the system.

Our categorization of these requests showed that, unlike our formative study results, the requests were frequently specific and became more specific and less vague towards the end of the task (Table 4.5). Participants' interviews revealed that this gradual specificity was due to various reasons. For one, the tool helped participants learn property names so they could now use them in requests (S5, S8, S13, S19). Others observed that the system was more accurate if they were more specific, so they adjusted their requests accordingly (S3, S4, S16, S20). A sample of participants' requests (Table 4.6) shows that the system was indeed more likely to predict users' expected properties if the requests included more specific information.

Like our formative study, however, around half of the requests were vague ("PP", "PV" and "A" in Table 4.5). Several vague requests were due to participants not remembering the name of a property, but they were able to quickly remember them by seeing Stylette's predicted properties. In other cases, vagueness was to deliberately get the system to act in a certain way. Several participants (S5, S6, S7, S14, S18, S19) mentioned using requests as "macros"—being vague (e.g., "change font") so the system returned several related properties that could be changed in one go. Others (S1, S2, S4, S8, S15) used vague requests to explore what other styling changes they could make.

Regarding the value suggestions, there were three particular uses: (1) as a "starting point", (2) as a "guideline", or (3) as a "shortcut". For the first type, participants (S4, S11, S14, S17, S20) picked a suggested value and then manually adjusted it more to their preference. Others (S2, S5, S9, S10, S18) used the suggestions as a guideline—hovering through values to mentally map numerical differences to visual differences. Finally, as similar values would be suggested for similar components, several participants (S1, S7, S15) looked for the same suggestion when editing multiple similar components—as a sort of "value shortcut".

4.5.3 Self-Confidence Across Tasks

To answer RQ3, we evaluated how self-confidence changed during the study by analyzing intra-condition differences in participants' responses (Fig. 4.8). Stylette participants' self-confidence increased significantly between the pre-survey (M=4.30, SD=1.12) and the end of Task 1 (M=5.13, SD=1.22, z=18.5, p=0.0035, NP). Participants felt satisfied about completing Task 1, and mentioned that it was easy to learn about and make changes using Stylette: *"It gave me the feeling of learning and becoming familiarized with web development terms."* (S12). Surprisingly, DevTools participants' self-confidence also increased significantly between the pre-survey (M=4.01, SD=1.26) and Task 1 (M=4.81, SD=1.25, z=34.5, p=0.0148, NP). Despite most of these participants not completing Task 1, they were satisfied with what they had accomplished as they expected that CSS code would be exceptionally challenging.

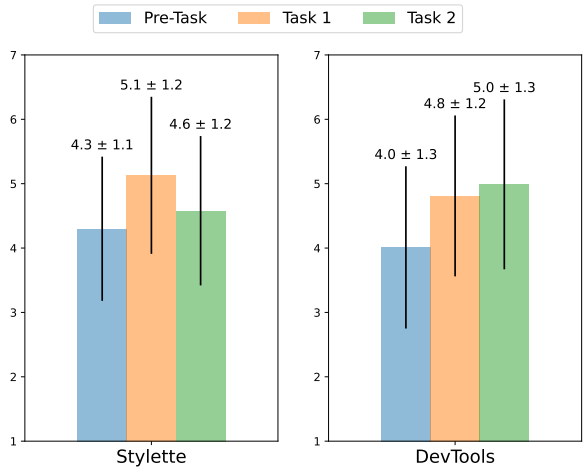

Figure 4.8: For both conditions, participants' reported self-confidence increased significantly between the pre-survey and the post-Task 1 survey. However, self-confidence decreased significantly for Stylette participants after Task 2, but did not change significantly for DevTools participants.

For similar reasons, DevTools participants' self-confidence increased between Task 1 (M=4.81, SD=1.25) and Task 2 (M=4.99, SD=1.32), although this was not statistically significant (t=0.540, p=0.595, P). These participants felt proud about their own effort and learning during the study: *"This is my first time handling [CSS] but I did this!"* (D14). In contrast, self-confidence for Stylette participants decreased significantly between Task 1 (M=5.13, SD=1.22) and Task 2 (M=4.58, SD=1.16, t=-3.204, p=0.0047, P). Unlike DevTools participants' self-reflective comments, Stylette participants' comments mostly focused on the tool. Some participants (S9, S16, S17, S20) mentioned how the system presented too many possibilities, making it difficult to decide on changes: *"It was hard [to choose] because the suggestions were all cute."* (S16). On the other hand, several participants (S4, S7, S11, S15) felt limited by the tool's possibilities—expecting the system to reveal new properties or support more complex changes (e.g., adding a "sparkle" animation).

4.6 Discussion

In this chapter, we propose Stylette, a system that allows users to easily edit a website's design through a suggested set of properties and values generated from natural language requests. Stylette can be generalized to a variety of applications: expanded with a community feature for users to share website modifications, implemented as an IDE plugin to support web developers' help-seeking, or integrated into tools for user feedback. In this section, we further elaborate on the potential of Stylette and suggest opportunities for future work.

4.6.1 Stylette as a Web Designing Springboard

In our study, Stylette allowed users with no prior knowledge to quickly perform desired styling changes on websites. Unlike search engines that can take the meaning of queries "literally", our system interpreted the vagueness behind users' requests to present more varied and suitable solutions. Stylette also allowed users to "learn-by-doing" by immediately testing the functions of properties by hovering on value suggestions—instead of having to skim through search results. As a side effect of interpreting vagueness, the system appeared to encourage creativity by presenting users with alternatives beyond their initial intentions. Together, these insights suggest that Stylette can support novices to explore and learn about CSS with continued usage.

The back-to-back tasks in our study provided a window into such continued usage of Stylette. We observed that users gradually developed knowledge about concrete CSS property names and values. For these now more knowledgeable users, the system still provided benefit: enabling the request of multiple properties for increased efficiency, supporting exploration of the design space, and helping users quickly remember forgotten information. However, we also observed that perceived effort could increase with continued usage and users' learning. This owed to the fact that, even after acquiring the knowledge to directly make changes by themselves, user still had to interact with the underlying, probabilistic AI—waiting for its processing and correcting any errors.

Thus, while Stylette is well-suited for novices to learn about CSS, its benefit may decrease with users' increasing knowledge due to the form of interaction. Elaborating on Amershi et al.'s guidelines [12], this suggests how human-AI interaction should be designed for over time use in the context of novice support systems. For future work, we propose an adaptive approach: initial natural language interaction to help users acquire knowledge about properties, and then gradually exposing direct manipulation widgets for properties that users have acquired knowledge about. Knowledge could be modeled by identifying previously used properties, repeated usage of a property, or the use of its name in voice requests.

To further overcome the frustration and physical demand observed in the study, future work could also investigate mechanisms to support discoverability of natural language input. Prior work [314, 101, 67] has demonstrated that supporting discoverability can reduce the amount of "guessing" that users must do. As Stylette's NLP pipeline appears to provide more accurate predictions for specific requests, future iterations of the system could guide users to new or desired properties by suggesting more specific language. For example, if the user uses a vague request and does not use any of the predicted properties, the system could suggest specific requests related to other properties that the user has not seen before.

4.6.2 Leveraging Large Language Models to Support Software Use

The grand scale of large language models (e.g., GPT-3 [45] or GPT-Neo [38]), in terms of architecture and datasets, has allowed them to perform previously unseen tasks with only a few data points. We leveraged this quality and the P-tuning technique [216] to allow novices to interact with website designs by constructing only a small dataset of 300 requests. Similar approaches can be taken to enable novices to use natural language to use various complex software—overcoming the vocabulary problem [100]. While a rich body of work has enabled similar natural language interaction to support software usage [2, 94, 95, 97, 96], their approaches relied on a wealth of user-generated content. Thus, these approaches are not possible for new applications or features as such content might not exist. Moreover, as shown by the struggles of DevTools participants in our study, the language used in such content may also differ greatly from the vague language used by novices as the content is usually created by intermediate or advanced users. With our approach, in contrast, natural language interaction can be enabled for new applications with only the effort of creating a small dataset of examples, and, by including representative examples of novices' language, the support can be designed specifically for novices.

4.6.3 Natural Language Coding as a Learning Tool

Our natural language interface helps novices learn about a coding language by demonstrating how the code realizes high-level goals—lowering the selection, coordination, and use barriers identified by Ko et al. [174]. In addition, by exposing novices to multiple alternatives for an intended goal, we observed that our approach allowed users to acquire a greater breadth of knowledge about the code—familiarizing with more properties and learning new uses for properties. However, the study also revealed that DevTools participants appeared to feel more satisfaction about their learning experience when compared to Stylette. We suspect that this is due to DevTools

participants expending more deliberate effort searching for and reading through resources. Based on these insights, we first suggest that natural language coding tools should provide multiple code alternatives for the same goal. Then, by incorporating interventions that prompt users to reflect on these alternatives—similar to prompts used in video learning [304]—to gain a wider understanding about the code through a deliberate learning experience.

4.6.4 Beyond CSS

Stylette aims to make the web more malleable for general users with no prior knowledge. Our system focuses on CSS code and allows novices to simply describe their high-level goal to start modifying it—without requiring the user to decompose the goal themselves [330] or look for examples [179, 91, 189]. However, websites are also composed of HTML (structure) and JavaScript code (functionalities). As structure-related changes might be more suitable for direct manipulation, Stylette could be combined with systems that already support this [245, 246]. Finally, to allow end-users to program new functionalities, models like OpenAI's Codex [392], which can generate JavaScript code from natural language descriptions, could be coupled with Stylette. By integrating these three types of support into one coherent system, future work could enable all users to fully access the web's malleability.

4.7 Limitations

Stylette has several limitations which we address in this section.

- Stylette currently supports 16 different CSS properties. These were the ones used the most in the creation of our request dataset. While Stylette could support more properties by expanding the dataset, certain complex properties (e.g., those related to flexbox and grid) also require corresponding modifications on parent elements. As Stylette only modifies the selected element's properties, it cannot currently support these properties. To overcome this limitation, the system could be enhanced to cascade necessary property modifications up the HTML tree.

- In our evaluation, we compared Stylette against using DevTools and search engines. A possible concern is that DevTools participants could change more properties and might have misdirected effort into these. Although the average DevTools participant only tried around two properties that were not supported in Stylette, we acknowledge that this could have affected results in Task 1.

- We relied on a dataset of 300 requests to train and evaluate our computational pipeline. While participants were generally satisfied with the pipeline's predictions, evaluating on a larger dataset would provide a better understanding of its performance. Also, while P-tuning has demonstrated high performance with even smaller datasets (N=32) [216], a larger dataset could increase our pipeline's performance and robustness.

- As we focused on a controlled evaluation of Stylette, it is still unclear how users would modify websites in the real-world. Future work could conduct a deployment study to understand how Stylette integrates into users' actual web experiences.

Chapter 5. EvalLM: Interactive Evaluation of Large Language Model Prompts on User-Defined Criteria

This chapter presents the first example of text disentanglement during the evaluation phase of interactive alignment. EvalLM is an interactive system that allows users to interactively assess LLM outputs by simply defining their own evaluation criteria. The system automatically disentangles each output by scoring it on each of these criteria—providing a multi-dimensional summary of the output's quality. This chapter has adapted, updated, and rewritten content from a paper at CHI 2024 [168]. All uses of "we", "our" and "us" in this chapter refers to coauthors of the aforementioned paper.

5.1 Motivation & Contributions

Large Language Models (LLMs) have catalyzed the creation of a wide array of novel applications. Composed of billions of parameters and trained on billions of tokens, LLMs can interpret a natural language description of a task, a **prompt**, and generate coherent human-like outputs for diverse purposes [45, 252, 213] (e.g., summarization [371], dialogue [353], story writing [61]). By composing a prompt, developers and researchers (i.e., prompt designers) can guide LLMs to perform novel tasks that satisfy desired requirements and support specific application settings. For example, HCI researchers have leveraged the generative capabilities of LLMs to ideate possible journalistic angles for a given event [267], generate questions to quiz children about information they learned [195], or simplify research papers into plain language [23].

Although prompt designers can easily bootstrap AI-based applications by simply composing a prompt, developing a prototype into a polished application that consistently produces high-quality outputs requires more

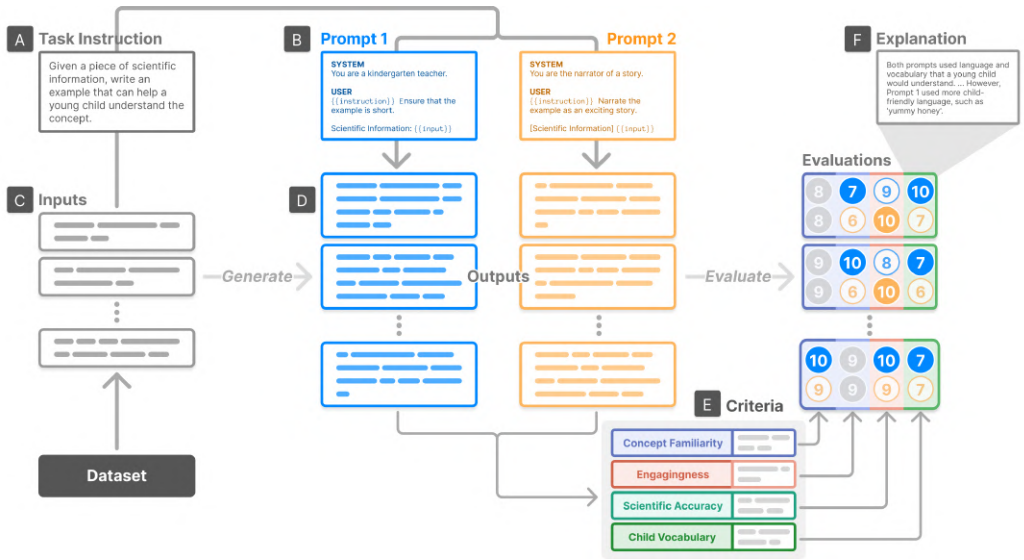

Figure 5.1: EvalLM aims to support prompt designers in refining their prompts via comparative evaluation of alternatives on user-defined criteria to verify performance and identify areas of improvement. In EvalLM, designers compose an overall task instruction (A) and a pair of alternative prompts (B), which they use to generate outputs (D) with inputs sampled from a dataset (C). Then, based on the criteria that the user defined (E), the system automatically evaluates these outputs to compare how each prompt performed on each criterion and provides explanations to support the user's verification of these explanations (F).

dedicated effort. As LLMs are non-deterministic and even partial changes in a prompt can significantly influence generated outputs [223, 210], designers need to iterate on their prompts multiple times to achieve satisfactory results [148, 317, 390, 389, 371, 213]. In this iterative process, designers test their prompt with sample inputs (e.g., paragraphs to summarize), inspect the generated outputs to identify areas for improvement, revise their prompts (e.g., change structure, wording, content), and repeat. When designers adopt LLMs for more open-ended generative tasks, however, evaluating outputs becomes significantly more challenging as no automatic metrics can adequately encode and measure the subjective quality of outputs [63]. Due to the lack of suitable automatic metrics, generative tasks are typically evaluated by human annotators or experts [106], but these can be impractical during early development stages when designers need to quickly iterate on prompts.

To understand how evaluation challenges affect the development of LLM-based applications, we conducted formative interviews with 8 prompt designers (e.g., developers, and researchers in HCI and ML) to understand how they iterate on and evaluate their prompts. Our interviews revealed that designers considered multiple *criteria* that were unique and specific to their applications when evaluating outputs from their prompts. Due to the novelty of these criteria and the significant cost of recruiting annotators, however, designers had to manually evaluate their prompt outputs themselves. As this manual and multi-faceted evaluation of outputs incurred a significant cognitive load, designers could only evaluate small batches of outputs and only on a subset of their criteria. As a result, when they refined their prompts, designers could not fully verify how their refinements had affected output quality or identify where further refinements were needed.

Based on these findings, we introduce EVALLM to facilitate prompt iterations by supporting the evaluation of outputs on user-defined and application-specific criteria (e.g., measuring `Object Familiarity` in scientific analogies for children). Instead of focusing on the low-level task of assessing generated outputs, EVALLM shifts designers' focus to the higher-level process of refining prompts and criteria—representations of their plans and requirements. Inspired by recent techniques for LLM-based evaluations [406, 384, 217], EVALLM employs an LLM as both (1) an *evaluation assistant*, which evaluates outputs on the defined criteria, and (2) a *criteria reviewer*, which revises the defined criteria. To aid users in revising their prompts and criteria, the evaluation assistant explains its assessments, allowing the user to identify where prompt outputs fell short or to identify where the assistant's interpretation of criteria misaligned with their own. Furthermore, the criteria reviewer analyses the user's criteria to identify revisions that can lead to evaluations of outputs on more specific and fine-grained dimensions. Through iterations of this collaborative process, designers co-evolve their prompts and criteria, where prompts improve to satisfy criteria and criteria improve to discern the quality of prompts—ultimately leading to more polished applications.

To understand how prompt designers adopt automatic evaluations during prompt iterations, we conducted a within-subjects study (N=12) where participants improved and evaluated prompts for novel tasks proposed by recent HCI work. In the study, participants used both EVALLM and a baseline where they manually evaluated outputs—emulating designers' current practice. Our study revealed that EVALLM helped participants "debug" their prompts by allowing them to quickly identify areas for improvement, and the evaluation assistant's explanations served as feedback by helping participants think about how to make these improvements. As a result, we observed that participants reached satisfactory prompts more efficiently as they tested 59% fewer changes than when they did not have evaluation assistance. As EVALLM also facilitated criteria revision, participants felt higher satisfaction regarding the quality of their criteria—suggesting that these criteria could be valuable during human evaluations. Overall, these findings suggest that EVALLM can fill the current gap between application development and deployment by assisting designers to iterate on prompts until a stage where they have the confidence to commit resources for more robust human evaluations.

This chapter presents the following contributions:

1. Qualitative findings from interviews with prompt designers (N = 8) that revealed how the effort of manually evaluating outputs on multiple, task-specific criteria can inhibit designers from making informed decisions during the iteration process.

2. EvalLM, an interactive system that aids users in revising prompts and verifying the effect of revisions by employing an LLM-based evaluation assistant to assess outputs on user-defined criteria, and a criteria reviewer to refine these criteria to assess more specific and detailed dimensions of outputs.

3. Findings from a user study (N = 12) that demonstrated how EvalLM can aid designers in debugging their prompts and ideating on strategies to more effectively revise their prompts.

5.2 Formative Interviews

To understand current practices and challenges when evaluating and iterating on LLM prompts, we conducted interviews with prompt designers. These interviews focused on understanding how prompt designers evaluate performance during early development stages and how these evaluations inform their refinement of prompts.

5.2.1 Participants and Procedure

We recruited 8 prompt designers through posts on online forums within our institution and word-of-mouth. These participants held various roles related to generative applications, and came from both academia and industry: 2 graduate students in HCI (1 MS, 1 PhD), 2 MS students in ML/NLP, 2 research scientists at a large company, 1 data scientist at a startup, and 1 startup CEO. All of the participants mentioned working on at least one project where they designed prompts for a novel generative applications. Their applications covered diverse contexts: social media, document question-answering, image captioning, writing, teaching, and intelligent assistants.[1] The length of their experiences with intensive prompt engineering ranged from 4 months to more than 1 year. Participants were compensated with approximately 45 USD (60,000 KRW) for the 1-hour interview. The interviews were conducted in a semi-structured format, and were recorded, manually transcribed and coded through a thematic analysis.

5.2.2 Findings

All of the designers mentioned working on applications for which they defined novel generation tasks. These tasks were novel as they (1) introduced new requirements to pre-existing generation tasks, or (2) were not analogous to any pre-existing tasks, according to participants. To develop the prompts for these applications, all of the designers first composed an initial prompt based on a preliminary set of expectations and requirements and, then, iteratively evaluated and refined their prompt to guide the LLM to better meet these expectations.

Evaluation is Manual

All of the designers mentioned how, at each iteration step, they tested their prompts on sample inputs and then manually evaluated the outputs themselves. Designers mentioned that they had to evaluate manually since they considered aspects that were subjective (P1-3, P5-8) and specific to their task (P1-8), meaning that there were no existing automated metrics to measure these aspects. Furthermore, since they were still in the development stage, recruiting annotators would be prohibitively expensive and would slow down the iteration process (P1-2, P4). However, all of the designers mentioned how manual evaluation could be demanding and time-consuming

[1] These are described broadly to preserve confidentiality.

and, thus, they only tested prompts on a small set of sample inputs (i.e., one to three samples) at each iteration step—which was still taxing especially with lengthier outputs (P1-2).

Evaluation is Multi-Faceted

Due to the complexity of the designers' intended applications, performance or the quality of the outputs could not be determined with a single criterion. Instead, designers considered multiple criteria or factors simultaneously when examining outputs, which made evaluation significantly more challenging (P1, P3-5). P5 mentioned how they had to carefully examine the outputs to *"catch all the subtleties"*. As this involved significant cognitive load and effort, designers described various ways in which they handled this multi-faceted evaluation. For example, four designers (P2, P4-5, P7) said that they only focused on the most important criteria for their task, and two others (P5-6) simplified assessments to assigning binary ratings on each criterion—i.e., whether the criterion was satisfied or not. Alternatively, P4 resorted to evaluating and refining one criterion at a time, but P7 noted that this method did not work for them as refining the prompt for one criterion led to the prompt failing at previously *"resolved"* criteria. Thus, while designers ideally wanted to evaluate holistically on multiple criteria, the effort required could lead them to only partially evaluate outputs.

Evaluation is Dynamic

Several designers described how they started the prompt design process with an initial set of evaluation criteria based on the intended goals for their application (P1-3, P5) and prior work (P2-5). Additionally, designers also expanded and transformed their criteria in each prompt iteration. By examining outputs, they identified additional criteria to consider as they observed flaws in the outputs, which they had previously not expected, or because they recognized other aspects that they wanted to improve on (P1-2, P4, P7-8). Beyond adding criteria, designers also mentioned how they had to concretize their criteria by determining how they should be evaluated. However, as these criteria could be subjective, it could be challenging to define what "success" meant for each criterion (P4-7). For designers who worked in teams, they mentioned how they would concretize the criteria by discussing with team members who could provide different perspectives (P4-5).

Evaluation to Refinements

Through the evaluations, designers identified what criteria the generated outputs failed to satisfy, and they attempted to refine their prompts to improve on these dimensions. However, most designers (P1-7) mentioned how they were unsure about how they should revise their prompts—the well-known challenge of prompt engineering [389]. Designers mentioned how they had no alternative, but to simply test different changes and to manually evaluate outputs again to check the effect of these changes. As this involved significant effort, designers mentioned that they could struggle to verify how much a revision improved quality on specific criteria (P3-4, P6). Due to the overall complexity of the evaluation-refinement process, designers also mentioned how they would fail to record all of their prompt revisions and their effects, which prevented them from learning from previous attempts and tracking their progress (P4-5, P7).

5.3 Design Goals

To support prompt iteration, we must support efficient evaluation of outputs on designers' own criteria. Recent work [384, 217] showed that LLMs can evaluate text on diverse subjective criteria—revealing their potential as evaluation assistants. However, for designers to effectively use LLMs as evaluators, they must be able to

define their own criteria and verify that subsequent evaluations align with their expectations. To scaffold these interrelated processes, we distill insights from our interviews into design goals for an LLM-based prompt iteration and evaluation system. We additionally take high-level inspiration from the process for developing psychometric scales [280]—question sets that collectively measure a variable (e.g., behavior, feeling) that cannot be assessed directly (e.g., NASA-TLX [130])—as we notice parallels between scale questions and evaluation criteria.

- **DG1: Automate evaluation of generated outputs according to user-defined criteria.** An automatic evaluation assistant can reduce effort by providing an initial assessment of outputs that designers can then verify. By defining their own criteria, designers can align the assistant's assessments with their own expectations and requirements.
- **DG2: Facilitate inspection of automatic evaluations through explanations.** Similar to how cognitive interviews can reveal incorrect or unclear questions in scales by asking respondents to verbalize their thoughts [39], the automatic evaluator should explain and justify its evaluations so that designers can inspect whether the evaluations align with their expectations.
- **DG3: Allow for the definition of criteria based on output data and prior literature.** As revealed by our interviews, prompt designers envision new criteria by assessing outputs and also by referring to prior work—resembling the inductive and deductive methods for defining psychometric scale questions [39].
- **DG4: Review the user-defined criteria to identify potential revisions.** Inspired by how scales are revised through reviews from external judges [280, 39], a system should aid designers in reviewing criteria to identify potential revisions, which could increase the effectiveness of subsequent evaluations.
- **DG5: Surface unreliable evaluations during large-scale evaluations.** While designers need to evaluate larger samples to comprehensively assess performance, they may be unable to verify all these evaluations. As an alternative, we take inspiration from reliability tests (e.g., inter-rater, test-retest) for psychometric scales [39, 280] and suggest that an evaluation system should surface less reliable evaluations for designers to verify.
- **DG6: Aid designers in tracking and comparing the effect of prompt changes.** As stated in the interviews, it can be challenging to understand the effect of each prompt change. By helping designers track how changes affect performance in the automatic evaluations, designers can make more informed iteration decisions.

5.4 EvalLM

Based on these design goals, we present EvalLM (Fig. 5.2), an interactive system for iterating on prompts by evaluating multiple outputs on multiple user-defined criteria. While designers typically compose prompts and assess outputs to provide feedback to themselves, EvalLM transforms this into a collaborative process where a designer iteratively refines prompts and criteria based on feedback from an LLM (DG1). Specifically, the system employs an LLM as both an *evaluation assistant* and *criteria reviewer*. The evaluation assistant judges prompt outputs based on the user's definitions of criteria and explains its evaluations, which can reveal where the prompts fall short or where criteria may be unclear (DG2). The user can define and revise criteria at any time (DG3) and, to facilitate this, the criteria reviewer recommends potential revisions that can enhance the detail and specificity of evaluations (DG4).

5.4.1 Interface

To illustrate the interactions in EvalLM, we walk through an example of an ML practitioner, Emily, who is designing prompts for a novel task of generating examples that help young children understand complex scientific phenomena.

Composing Prompts

In the generation panel, the user writes the overall instruction for their task (Fig. 5.2A) and designs two prompt templates (Fig. 5.2B). EvalLM is designed for comparing prompts as this enables designers to compare the performance of different prompt variations, or to compare prompts before and after specific edits (DG6). Furthermore, prior work has found that it is easier for both humans [106, 198] and LLMs [406, 30] to compare model outputs than to rate a single output. For each prompt template, the user can compose the system and user prompt (Fig. 5.3D-E) using the tokens `{{instruction}}` and `{{input}}`, which are replaced with the instruction and content of an input sample when generating outputs. By composing the task instruction separately, users can reuse the same base instruction across prompt templates. To test and compare different prompt ideas, the user can create more prompts (Fig. 5.3B), name them, and switch between them as they desire (Fig. 5.3C).

> Emily describes her task in the Instruction field and, to test the effect of different prompting "tricks", she creates two prompts: a basic prompt with a simple form-like format and a prompt with sub-headers, additional context, and a system prompt that instructs the LLM to act as a teacher.

Sampling Inputs and Generating Outputs

To test their prompts, the user can upload their own input dataset and then sample inputs from this dataset (Fig. 5.2C). Users are provided with two ways for sampling inputs: (1) manual, which opens a panel where the user

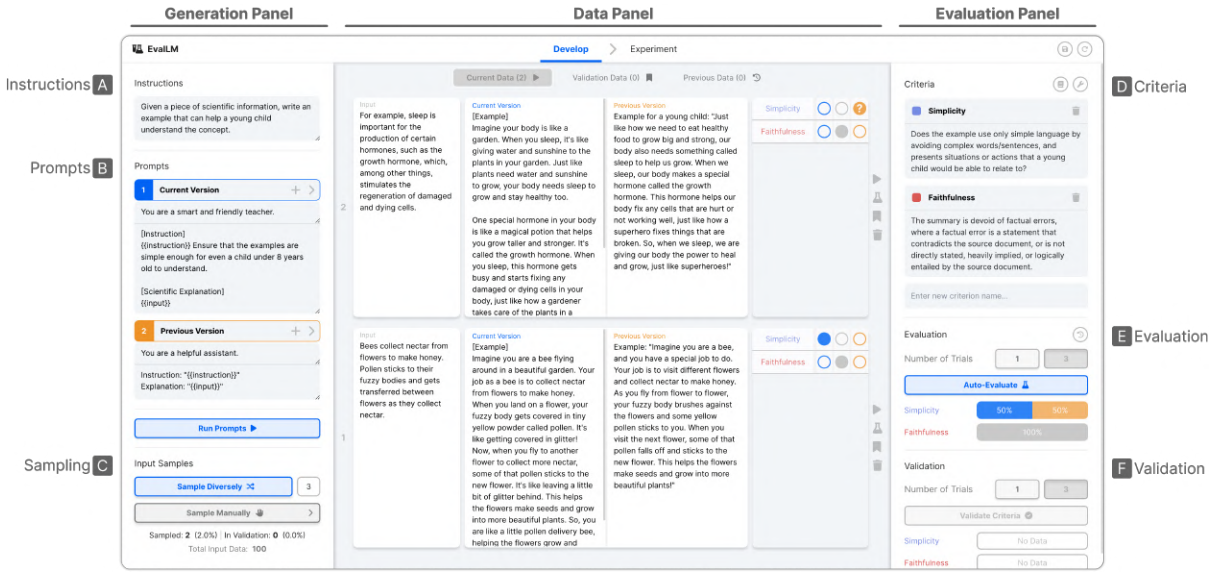

Figure 5.2: EvalLM is composed of three main panels: generation, data, and evaluation. In the generation panel, the user can compose the overall instructions for their task (A), two prompt templates they want to compare (B), and sample inputs from their dataset (C). To evaluate outputs, the user first defines their criteria set (D) and can see an overview of evaluation results (E). If the user has added samples to their validation set, they can also check the accuracy of the evaluations in this panel (F). The data panel shows a series of rows, where each row presents an input sample, the outputs generated on this input, and the evaluation results for these outputs.

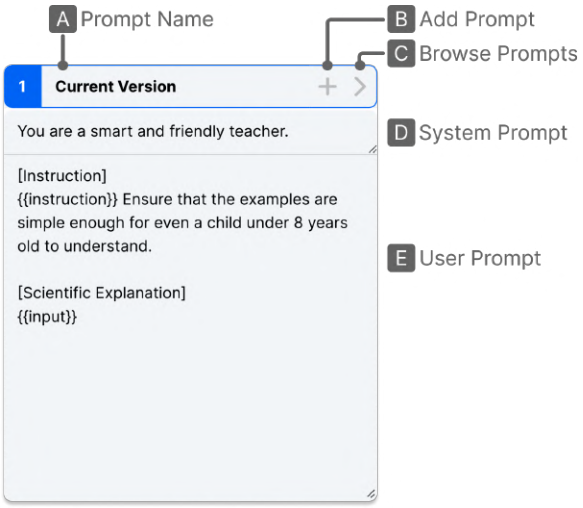

Figure 5.3: For each prompt in EVALLM, the user can provide it a unique name (A), and compose both the system (D) and user prompt (E). If the user wants to test different pairs of prompts, they can add new prompts, (B) or switch to previous prompts through the browse button (C), which opens a panel listing all of the prompts that they have created.

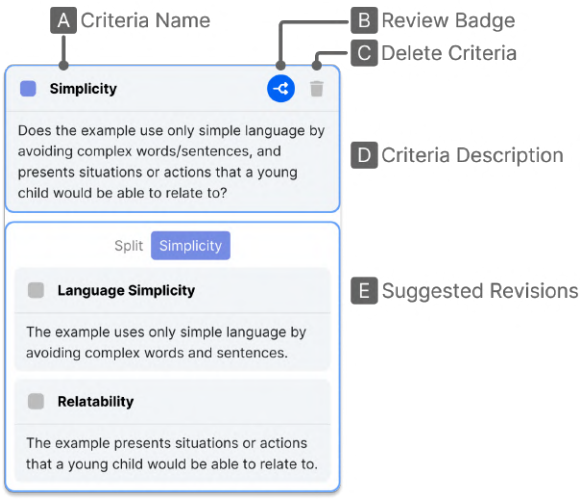

Figure 5.4: For each criterion in EVALLM, the user provides a name (A) and a description (D). Each criterion is automatically assigned a color to help with identification. If the criteria review tool identifies improvements for the criteria, these are shown as badges (B) that the user can click to see the suggested revisions (E). Clicking on these suggestions adds them to the criteria set.

can browse through and choose samples, and (2) diverse, which automatically samples distinct data points (details in §5.4.3). When the user samples inputs, each one is shown as a row in the data panel (Fig. 5.5A). Then, the user can click on `Run Prompts` to generate outputs for each sampled input with each of their prompts. The outputs from each prompt are shown side-by-side in the data row (Fig. 5.5B).

Defining Criteria

In the evaluation panel, the user can define and manage their evaluation criteria (Fig. 5.2D). The user defines a new criterion by providing a name and a description, which explains what the criteria assesses or the characteristics that an output must possess to satisfy that criteria (Fig. 5.4). Instead of defining their own criteria from scratch, the user can also browse through the *Criteria Dictionary* to select from criteria that were defined in prior work (DG3). This dictionary can serve as a starting point by providing initial descriptions that the user can adapt to their context, or as inspiration by helping users consider other aspects to evaluate.

> Emily first creates a criterion, `Familiarity`, to check that generated examples only use language and situations that a child can understand. To decide on what else to evaluate, Emily browses through the dictionary and finds the `Faithfulness` criterion, which checks that summaries are devoid of factual errors [177]. Emily adds and edits this criterion to assess that generated examples are faithful to the given scientific information.

Revising Criteria: Refine, Merge, and Split

To help users identify potential improvements in their criteria and improve the quality of evaluations (DG4), EVALLM provides the *Criteria Review Tool* that checks criteria for (1) clarity and relevance, (2) redundancy, and (3) granularity. This LLM-powered tool automatically identifies criteria that could be improved (Fig. 5.4C) and recommends improvements (Fig. 5.4D). Similar to how external judges assess psychometric scales [280, 39], the

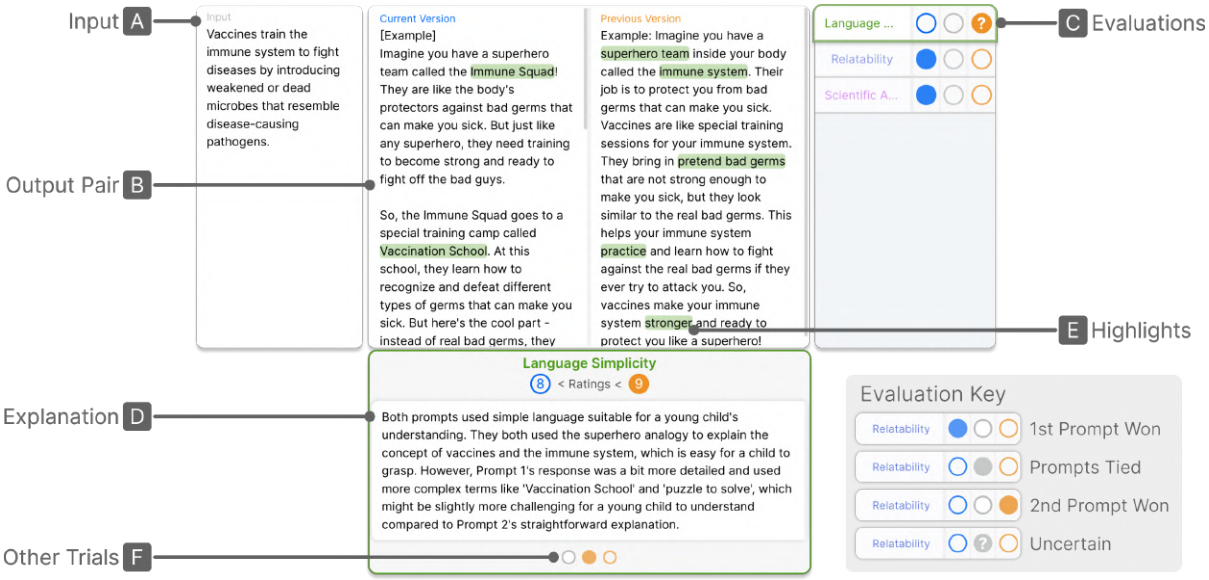

Figure 5.5: Rows in the data panel show the input sample (A), the outputs generated from the pair of prompts (B), and the evaluation results on each defined criteria (C). For each criterion, the evaluation shows three circles that respectively represent that the first prompt won, there was a tie, or the second prompt won. If a question mark is shown over a circle, this indicates that there is uncertainty in the evaluation. If only one evaluation trial was run, this indicates that a small score difference between outputs and, if multiple trials were run, that at least one trial returned a different result. The user can click on an evaluation to see the explanation (D) and highlights on the portions of the output that were relevant to that evaluation (E). If the user conducted multiple evaluation trials, they can also browse through the other trials by using the carousel at the bottom (F).

tool identifies criteria that can be clearer or more relevant to the user's task and suggests how to *refine* them. Inspired by item reduction analysis [39], the tool also identifies criteria that may be redundant and suggests how these could be *merged* into a single criteria. Finally, as human evaluations are more accurate with fine-grained criteria [106, 177], the review tool identifies coarse-grained criteria and suggests how to *split* them into multiple criteria. While the user can manually activate the criteria review tool, it also runs automatically if the user has not modified their criteria for a certain period of time.

> As she was reading through outputs, Emily notices that the review tool has suggested splitting the `Familiarity` criterion into `Language Simplicity` and `Relatability`. As she agrees that these assess different aspects, she adds these two suggestions and removes her previous `Familiarity` criteria.

Evaluating Outputs

After generating outputs and defining criteria, the user can click on `Auto-Evaluate` to automatically evaluate the output pairs (DG1). For each criterion, the evaluation assigns each output with a score out of 10 to decide which output was better at satisfying that criterion or if there was a tie (Fig. 5.5C). If the user wants to understand the evaluations, they can click on a criteria name to view the explanation for that evaluation (Fig. 5.5D). To help the user verify the explanations without fully reading the outputs (DG2), the interface highlights fragments from the outputs (Fig. 5.5E) that were focused on more when evaluating the criterion. To provide a bigger picture on prompt performance, the interface also displays the proportion of samples where each prompt "won" and the proportion of "ties" (Fig. 5.2E).

> After running the evaluation, Emily checks the results overview and sees that her improved prompt won the most in all three criteria, but lost once in the `Language Simplicity` criterion. To check why, she opens

the evaluation explanation, which mentions that the improved prompt's output used more complex terms. To improve on this, Emily adds a requirement to her prompt to always simplify complex words first.

As LLMs are non-deterministic, the evaluation results may differ in every run. To increase their assurance of the evaluation results, the user can increase the number of evaluation trials. The interface will then evaluate each output pair for the chosen number of trials and decide on the "winner" for each criterion based on which output won the most number of trials (i.e., majority vote). The user can check the evaluations for each trial by using the carousel at the bottom of the evaluation explanations (Fig. 5.5F).

Additional Features: History and Validation

To help the user keep track of their iterations, EvalLM automatically records all of the user's prompts, criteria, and evaluations (DG2, Fig. 5.6), which can be viewed by clicking on the "History" button (Fig. 5.2E). Additionally, users can store generated samples into a validation set where they can annotate their own ground-truth evaluations. With a populated validation set, the user can `Validate Criteria` (Fig. 5.2F) to assess how accurately the automatic evaluator predicts the user's ground-truth evaluations.

Experimenting on Larger Samples

After designers have developed their prompts and criteria, they may wish to verify their prompt's performance by testing on significantly larger samples. For this purpose, EvalLM provides the `Experiment` screen. In this screen, designers can set the number of evaluation trials and select an alternative evaluator LLM (e.g., ChatGPT or PaLM 2 [16]). When the user runs an experiment, the system automatically samples diverse inputs, generates outputs with these inputs and the chosen prompts, and then evaluates the outputs on the chosen criteria for the configured number of trials. If an alternative evaluator was selected, this LLM is used to evaluate the same outputs on the same criteria for the same number of trials. The Experiment screen shows two additional statistics (Fig. 5.7): *test-retest reliability* or the consistency of evaluations between trials, and *inter-rater reliability* or the consistency between the evaluations by the system's LLM (i.e., GPT-4) and the alternative evaluator. These statistics are shown as stacked bar charts that present the proportion of consistent and inconsistent evaluations, and the user can click on a bar to only display those cases (DG5). In this screen, the user can also click on the stacked bar charts for the evaluation overview to see only the samples where one prompt performed better than the other (or they were tied) for a chosen criteria.

To decide between two promising prompts, Emily runs an experiment with 40 samples and ChatGPT as the alternative evaluator. The results show that her first prompt excels at `Language Simplicity` but loses in `Scientific Accuracy`, signaling at a possible trade-off. Emily also notices that the evaluators often disagree in `Language Simplicity` and, by clicking on that bar to browse through cases, she finds that GPT-4 also assesses sentence complexity for the criteria. As she only wants it to assess vocabulary, she changes it to `Simple Vocabulary`.

5.4.2 Prompting Techniques

Our interface is powered by two LLM-based components: automatic evaluation, and criteria review. In this section, we describe the design of these prompts.

Figure 5.6: The history visualization is separated into sessions, which represent sets of samples that were generated and evaluated with the same prompts and criteria. For each session, the history shows the names of the prompts (A) and criteria (B) used, and the user can click on these to see their content at the time (C). For each criterion, the history shows a bar for each sample evaluated (D), which is color-coded to represent which prompt won or if there was a tie for that sample.

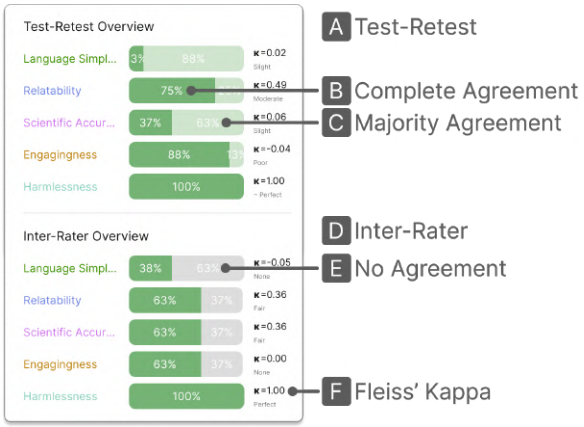

Figure 5.7: The Experiment screen presents the evaluation assistant's reliability across trials (A), and the reliability between the assistant and the chosen alternative evaluator (D). For each criterion, the user can see a stacked bar chart that shows the portion of samples where the evaluations (between trials or between evaluators) had complete agreement (B, green), the majority agreed (C, light green), or there was no majority agreement (E, gray). The user can also see the Fleiss' kappa statistics (F) as a reference for the degree of reliability.

Automatic Evaluation

The main goal of EvalLM is to evaluate prompt outputs by comparing them on a set of user-defined criteria. To this end, we designed our evaluation prompt by adapting the prompts from two state-of-the-art approaches: *LLM-as-a-judge* [406], which compares model responses on their overall quality, and *FLASK* [384], which rates the performance of a single response on multiple "skills" or criteria. Our prompt takes as input: a task instruction, an input sample, a pair of outputs, and a list of criteria descriptions. Then, our prompt instructs an LLM to evaluate the output pair on each criterion by (1) explaining how the outputs satisfy the criterion, (2) extracting evidence fragments from each output, and then (3) providing each output with a score out of 10. To design this prompt, we considered alternative approaches for each of these components.

Instruction and Input: We only include the overall instruction that is shared between the generation prompts to provide the evaluator LLM with context about the task while keeping its evaluations prompt-agnostic—limiting potential bias due to the content or wording of each generation prompt.

Output Ordering: Prior work found that LLM-based evaluations have positional bias, frequently favoring the first output [343]. While this work suggested evaluating each candidate in each position and aggregating results, this would introduce additional costs and delays. Instead, our system alternates the positions of outputs in every evaluation.

Criteria Descriptions: Similar to Ye et al. [384], the criteria are added to our prompt as lines of the form: *"name of criterion: description of the criterion."* While they also included scoring rubrics for each criterion to describe each numeric score, we only included the criteria descriptions. We considered that designing rubrics would require excessive effort from users and, based on preliminary tests, generated rubrics could negatively affect evaluations.

Explanation: The prompt instructs the LLM to provide an explanation where it compares and contrasts between the outputs. This explanation is useful for users, but can also elicit reasoning from the LLM and increase

performance [354]. While we considered a design where the LLM explains the performance of each output separately to yield more in-depth examinations, pilot studies showed that these explanations were repetitive and less useful.

Evidence: To help users associate the evaluation explanations with the outputs, we instruct the LLM to extract fragments from the outputs that are relevant to its evaluation. We also considered a design where the LLM first extracts evidence and then cites these in its explanation, but saw that these explanations would simply list the evidence without elaborating on them.

Criteria Review

To automatically review the user's criteria and suggest revisions, we designed three prompts for each type of supported review: *refining*, *merging*, and *splitting*. Instead of designing one prompt that conducts all of these reviews, we designed separate prompts for each review as certain criteria may require multiple revisions and we wanted to allow users to flexibly choose between these. These three prompts follow the same general design: given a task instruction and set of criteria, the LLM is instructed to review each criterion and identify any *faulty* ones, explain how these can be *revised*, and then generate new criteria that result from these revisions. The three prompts differ in terms of what determines a criterion to be *faulty* and how they should be *revised*:

- *Refining*: Identifies criteria that are confusing or imprecise, and revises these to be clearer and more specific.
- *Merging*: Identifies criteria that may measure similar aspects, and combines these into one joint criterion.
- *Splitting*: Identifies criteria that are excessively broad and consider multiple unrelated aspects, and divides these by generating new criteria for each of these aspects.

Additionally, all of the prompts instruct the LLM to ensure that the suggested criteria (1) are clear and concise, following the requirements of psychometric scales [280], and (2) do not remove or add new information. Additionally, as LLMs tend to be overeager to follow instructions, which could lead to excessive revisions of criteria, the prompts explicitly mention that it is possible for all of the criteria to be satisfactory and not require any revisions.

5.4.3 Implementation Details

We implemented the front-end of EvalLLM using TypeScript, ReactJS, and CSS. The back-end was implemented as a Flask server and the OpenAI API[2] for all LLM components. In terms of the LLM configurations, we set the temperature to 0.3 for all components. The automatic evaluation and criteria review tool used the `gpt-4-0613` model and, as LLMs are prone to self-enhancement bias where it rates its own outputs highly [406, 199], we use `gpt-3.5-turbo-0613` when generating outputs. Finally, to support diverse sampling of inputs, we automatically cluster samples in the uploaded datasets by embedding data points using the OpenAI API with the `text-embedding-ada-002` model and clustering these embeddings using the `KMeans` algorithm in the *scikit-learn*[3] library. To sample diversely, the system chooses data points from distinct clusters.

5.5 Technical Evaluation

We conduct a small-scale technical evaluation of our LLM-based evaluation approach to understand how performance is affected by more task-specific criteria, and to gain a more in-depth understanding of the LLM's

[2] `https://platform.openai.com/`

[3] `https://scikit-learn.org/`

explanations for its evaluations.

5.5.1 Automatic Evaluation

While prior work in NLP has assessed the performance of LLM-based evaluations [406, 384, 217], these focused on evaluating outputs on overall quality or based on pre-defined criteria. As our work employs LLMs to evaluate task-specific criteria, we conduct a technical evaluation to assess how this affects evaluation performance.

Dataset

We assess LLM evaluations by comparing them to human evaluations in the MT-Bench dataset [406]. This dataset presents 80 user requests of diverse categories (e.g., writing, roleplay, math, coding) and responses from various LLMs to each request. These responses are paired and, for each pair, the dataset provides votes from one to three human annotators on what response was better or if there was a tie. For our evaluation, we selected 19 requests in the writing and role-playing categories as they involve the most subjectivity. As our LLM evaluations focus on prompts rather than requests, we decompose these requests into a prompt-input format.

Conditions

We compare LLM evaluations in three conditions:

- `Overall-Quality`: We adopted the prompt from LLM-as-a-judge [406] that compares a pair of outputs to select the one with higher overall quality.
- `General-Criteria`: We used our evaluation prompt with the more general and broad criteria from FLASK [384]. In this work, they instruct an LLM to select three criteria, out of a set of 12, that are the most relevant to a given request. Then, this condition evaluated a pair of outputs to determine which one was better at satisfying each of the criteria.
- `Specific-Criteria`: We used our evaluation prompt with criteria that were automatically refined and adapted for each request or prompt. For each request, we start with the same criteria that were selected by the LLM in the `General- Criteria` condition, but we automatically split and refine them using our criteria review technique (i.e., all automatic suggestions are taken). Then, a pair of outputs was evaluated by determining which better satisfied each of these more fine-grained and specific criteria.

For all evaluations, we use the `gpt-4-0613` model with temperature set to 0 for reproducibility. Additionally, due to the positional bias of LLMs, we run evaluations twice with each output in each position and then average the scores for each criterion.

Measures

To aggregate the criteria-wise evaluations into a single vote, we determine the vote for each criterion, and then calculate the majority vote across the criteria. Following LLM-as-a-judge, we calculate the agreement between human and automated evaluations based on two cases: (1) if the majority of human evaluators agreed on a vote, then we count an agreement if the LLM evaluation agrees with this majority vote, or (2) if there was no majority vote between human evaluators, then we calculate the proportion of annotators that the LLM evaluation agreed with. Also, we calculate the Fleiss' kappa between the LLM evaluations and the majority vote of human annotators.

Results

Overall, we observed that `Specific-Criteria` had the highest agreement and correlation with human annotators and `General-Criteria` had the lowest (Table 5.1). As a reference, for data points with at least two annotators, the Fleiss' Kappa between two random human annotators was 0.496 (sampled 5 times and averaged), showing that `Specific-Criteria` agreed with human evaluations to a degree that was similar to human-human agreement. By qualitatively reviewing cases, we observed that the `Specific-Criteria` condition could produce more balanced evaluations. For example, when assessing generated travel blogs, `Overall-Quality` only assessed the breadth of attractions covered, while `Specific-Criteria` also assessed the depth of the attraction descriptions. Also, compared to `General-Criteria`, we saw that `Specific-Criteria` assessed specific requirements that were posed in prompts, enabling it to better capture how well outputs followed the prompts. We also found cases where `Specific-Criteria` was less successful as each criterion was given equal weighting, but certain criteria might need to hold precedence—e.g., not providing an incorrect medical diagnosis is more important than readability. These preliminary findings suggest that instructing LLMs to evaluate more fine-grained and specific criteria can increase their ability to assess the alignment of outputs with instructions. However, we also note that the agreement between LLM and human evaluations is still not perfect. While this could be attributed to the subjectivity involved, it also highlights the limitations of only relying on LLM evaluations.

5.5.2 Evaluation Explanations

Additionally, we assess the quality of the LLM-generated explanations. Prior work did not assess the quality of the LLMs' explanations during evaluations as their purpose was only to induce chain-of-thought [354].

Procedure

We sampled two evaluations by the `Specific-Criteria` condition for each of the 19 tasks, resulting in a total of 38 output pairs evaluated and 194 criteria-wise evaluations. Each evaluation was annotated for the presence of errors regarding five criteria: (1) *logical* (i.e., the explanation presents logical and coherent arguments and justifications), (2) *faithful* (i.e., the explanation does not hallucinate content that does not exist in the outputs), (3) *independent* (i.e., the explanation does not assess other criteria or aspects not described in the evaluation criterion), (4) *evidential* (i.e., the evidence extracted is relevant to the explanation), and (5) *score aligned* (i.e., the final score aligns with the explanation provided). We recruited two annotators, who had previous experience grading written assignments, and instructed them to mark these errors even if only part of the explanation presented the error. Since there is a large class imbalance (i.e., most explanations have no errors), we considered an explanation to have errors if at least one of the evaluators marked an error.

Condition	Agreement	Fleiss' Kappa
`Overall-Quality`	0.699	0.430
`General-Criteria`	0.639	0.420
`Specific-Criteria`	**0.713**	**0.485**

Table 5.1: Comparison of the agreement between the three evaluation conditions and human evaluations in the MT Bench dataset. Evaluating on specific criteria showed the highest agreement and Fleiss' kappa with the human evaluations.

Results

Overall, the explanations were mostly free of issues: 91.4% of the explanations were logical, 99.1% were faithful, 84.2% were independent, 100% provided relevant evidence, and 98.6% were aligned with their scores. We qualitatively reviewed erroneous cases and observed that the LLM's explanations frequently failed to be independent as they would contrast outputs on their level of detail despite the criterion not assessing this. In terms of logic, the evaluations struggled to assess creativity and could be too superficial in their interpretations. For example, news article headlines were considered to adequately address ethical dilemmas by simply including the phrase "ethical concerns". For faithfulness, the evaluations struggled to accurately measure the length of the outputs. Finally, for score alignment, the explanations would occasionally provide one output with a higher score despite not mentioning this in its explanations. These results show that GPT-4 is able to produce relatively sensible explanations for its evaluations, but can be limited by bias towards detail and logical capabilities. While research in NLP has focused largely on improving the "accuracy" of the scores provided by LLM-based evaluations, this suggests that more investigation is needed into the explanations provided by these evaluations.

5.6 User Study

To understand how the EvalLM affects the prompt iteration process when compared to following designers' current practice, we conducted a within-subjects study where we compared EvalLM to a baseline where participants manually evaluated outputs. In this study, we aimed to answer the following research questions:

- RQ1. Can EvalLM aid designers in deciding on how to revise their prompts and in verifying the effectiveness of these revisions?
- RQ2. How do designers define their own criteria for given generation tasks and how does the EvalLM's criteria review tool support revisions on these criteria?
- RQ3. How do designers interpret and gauge their trust in the evaluations by EvalLM?

5.6.1 Study Design

Participants

We recruited 12 participants through posts on online forums within our institution. All participants reported having extensive experiences with prompting: nine had designed prompts for research-based applications, one designed prompts for toy projects, and two described frequent use of LLMs for productivity. Regarding the length of their experiences, four had between 1 to 3 months of experience, three had 3 to 6 months, four had 6 to 12 months, and two had 1 to 2 years. Participants were compensated with approximately 60 USD (80,000 KRW) for the 2-hour study.

Conditions

During the study, participants designed prompts and evaluated outputs for two given tasks. For each task, participants used EvalLM in one of two conditions: `Assist` or `Manual`. The `Assist` condition was the full EvalLM interface, while the `Manual` condition was the EvalLM interface without the evaluation assistant or the criteria review tool. In the `Manual` condition, participants defined their own criteria or selected from the dictionary, and then evaluated data samples by choosing which output won for each criterion. This condition

supports designers' common practices—according to our formative interviews—where they copy generated outputs into a spreadsheet to manually check which criteria were satisfied by each output.

Tasks

Participants designed prompts for the same two tasks. As our work focuses on novel generative tasks, we adapted tasks from two recently proposed LLM-powered HCI systems: (1) write an example that can explain a piece of scientific information to a young child—based on Lee et al.'s *DAPIE* system [195]—and (2) ideate a list of alternative angles for a news story—based on Peridis et al.'s *AngleKindling* system [267]. We chose these as they target a specific user population (i.e., children and reporters) and no significant expertise is needed to understand the task outputs. Additionally, these tasks differ in terms of their goal (i.e., explaining vs. brainstorming), and how the output relates to the input (i.e., transforming vs. expanding on the information).

Procedure

Participants signed the informed consent form prior to the study. After a brief introduction, participants answered a pre-task survey. After a walkthrough of the first interface, participants used this interface to design a prompt for the first task for 35 minutes[4]. Participants were asked to envision themselves as a developer at a startup building an application for the given task. Their goal was to design a prompt that performed better than an initial prompt designed by their team and demonstrate its performance on diverse data samples. Participants could flexibly decide on the criteria that they would be evaluating, but were asked to ensure that their final criteria set was: (1) *exhaustively comprehensive* (i.e., assess all factors that are important for the task), (2) *mutually exclusive* (i.e., minimal redundancies between criteria), and (3) *clear* (i.e., clearly described for others to understand what is assessed). After the task, they responded to a post-task survey and, through a semi-structured interview, we asked them about how they defined their criteria, evaluated outputs, and revised their prompts during this task. Then, participants were provided with a walkthrough of the second interface and used this interface to perform the second task for 35 minutes. After responding to the post-task survey and interview, we concluded the study with a semi-structured interview about the differences between participants' experiences in the first and second tasks.

Measures

For qualitative data, we transcribed the semi-structured interviews and coded them through a thematic analysis. For quantitative data, we analyzed participants' responses to the two post-task surveys. These surveys asked participants to rate, on a seven-point Likert scale, their self-confidence in designing prompts and evaluating outputs, and their self-perceived experience with the system based on questions from Wu et al. [371]. We also asked participants to rate their own final criteria on how *exhaustively comprehensive*, *mutual exclusive*, and *clear* they were. Finally, participants rated their self-perceived workload on five questions from the NASA-TLX questionnaire, excluding the "Physical Demand" question. For these Likert scale ratings, we analyzed them through the non-parametric Wilcoxon signed-rank test.

Additionally, we analyzed participants' interaction logs to measure the number of (1) unique prompts tested, (2) criteria changes (e.g., edit, add, delete), and (3) unique outputs evaluated, where partial evaluations (i.e., only a subset of criteria were evaluated) were counted equally. For all these measures, we conducted a Shapiro-Wilk test to determine if the data was parametric, and then used a paired t-test (if parametric) and a Wilcoxon signed-rank test (if non-parametric). As participants in the study could flexibly set their own goals and requirements for their prompts, we did not conduct an external evaluation of the prompts as external evaluators may consider quality

[4]The order of conditions and tasks were counterbalanced to mitigate ordering effects.

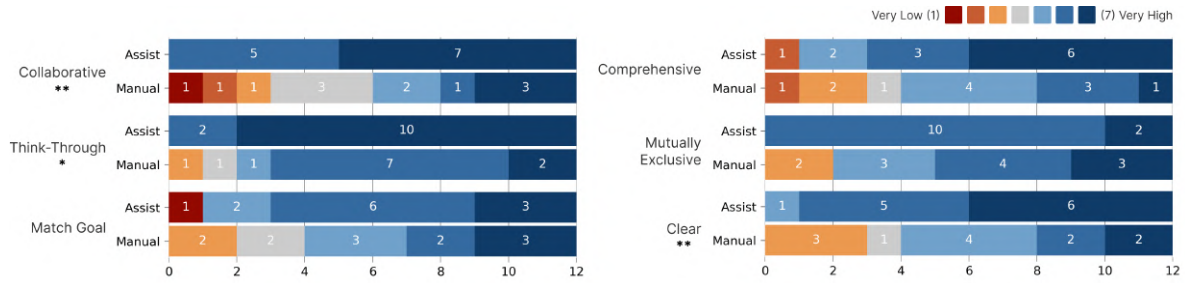

Figure 5.8: Distribution of participants' ratings on their perceived experiences with each condition (left) and their satisfaction with their final set of criteria (right). Participants felt that the `Assist` condition was significantly more collaborative and able to help them think through the task. They also felt that their criteria were significantly more clear in the `Assist` condition compared to in the `Manual` condition (*:p<.05, **:p<.01).

aspects that differ from participants' intentions due to the subjectivity involved. Also, while external evaluators could be tasked to assess prompts on the criteria defined by participants, a single participant could produce criteria that differ in clarity and complexity across tasks and conditions, which could lead to unfair comparisons.

5.6.2 Results

In this section, we describe findings on how participants evaluated outputs and revised their prompts in §5.6.2 and §5.6.2 (RQ1), how they defined their criteria in §5.6.2 (RQ2), their trust for the evaluation assistant in §5.6.2 (RQ3), and their overall perceived workload in §5.6.2. For each of these, we first describe relevant quantitative findings and then qualitative insights.

Evaluating Outputs

Overall, participants had **higher self-confidence in their ability to evaluate** prompt outputs with the `Assist` condition (`Assist`=6.71 ± 0.40, `Manual`=4.96 ± 0.72, z=9.23, p<0.001). Participants explained that the `Assist` condition shouldered the burden of examining outputs, which enabled them to evaluate prompts more comprehensively. Specifically, with `Assist`, participants were able to **evaluate a larger number of unique outputs** (`Assist`=20.42 ± 14.46, `Manual`=10.08 ± 6.27, z=8.00, p=0.03). Several participants noted how this helped them look at the *"bigger picture"* (P1) and understand *"how [a prompt] will work at a larger scale"* (P5).

By facilitating evaluations, the `Assist` condition also allowed participants to evaluate more criteria and assess performance on diverse dimensions, without worrying about the cost involved. For example, P9 mentioned that they were encouraged to *"think of more aspects that they wanted to evaluate"* and P2 mentioned that they *"just kept criteria [...] to check them just in case."* In contrast, when using the `Manual` condition, participants would frequently evaluate samples only on a subset of their criteria—on average, 46.7% (SD=31.2%) of all evaluated samples were partially evaluated. Besides automating evaluation, several participants also mentioned how the `Assist` condition made it easier to evaluate the outputs themselves. While it could be challenging *"tell what is different by simply looking at [outputs]"* (P5), the evaluation assistant's explanations and highlighting in outputs made it *"easier to tell outputs apart"* (P6).

As the `Assist` condition supported faster evaluation cycles, participants mentioned how they used the evaluation assistant as a *"debugger"* (P8). The assistant helped participants to easily check *"what the prompt is already satisfying"* (P8) and *"where it's lacking"* (P9). P11 mentioned, *"The [system] tells me that my prompt is worse on simplicity so that means that my prompt is not communicating this clearly and I should focus on fixing this."* With the evaluation support, participants could quickly identify aspects or "bugs" that needed to be handled

and, after revising their prompts, quickly verify whether these issues had been handled or not.

Revising Prompts

Besides aiding participants in checking revisions, the `Assist` condition also helped participants in thinking about how to revise their prompts. Participants felt that the `Assist` condition helped them **think about how to complete the task better** (`Assist`=6.83 ± 0.39, `Manual`=5.67 ± 1.15, z=0.00, p=0.01) and that they were **collaborating with the system to design the prompts** (`Assist`=6.58 ± 0.51, `Manual`=4.58 ± 1.98, z=1.50, p<0.01) (Fig. 5.8, left). While participants also reported a **higher self-confidence on their ability to design and improve prompts** with `Assist`, this difference was not significant (`Assist`=5.63 ± 1.03, `Manual`=5.00 ± 0.95, z=11.00, p=0.17). Additionally, we found that participants with the `Assist` condition reached a **similar level of satisfaction in their prompts** (`Assist`=5.67 ± 1.61, `Manual`=5.17 ± 1.47, z=20.00, p=0.24) ("Match Goal" in Fig. 5.8) by testing **a smaller number of prompt variations** (`Assist`=5.00 ± 3.13, `Manual`=8.42 ± 4.76, z=1.50, p=0.01).

In the `Manual` condition, participants mentioned how they felt like *"the only brain thinking about [the task]"* (P10). In contrast, participants felt that `Assist` condition was providing them with *"feedback on what to improve"* (P2) and could help them set the *"direction"* for their prompts (P7, P9). Several participants found the explanations to be particularly useful as they presented them with *"diverse opinions"* (P1) that they *"couldn't think about"* (P6). For example, P5 purposefully increased the number of evaluation trials and checked the explanation for each trial to *"see all the differences [in opinion] and then change [their] prompt to satisfy all of these."* In this sense, the `Assist` condition could help participants gain a more holistic understanding of how to improve their prompts and lead them to more effective prompt revisions—as reflected by how fewer prompt changes were tested in the `Assist` condition.

These explanations, however, could also lead participants to feel less in control and more aware of their

	Review	Original Criteria	Revised Criteria
P1	Refine	**`Explainability`**: Does the response provide a detailed explanation about an angle?	**`Clarity of Explanation`**: The assistant's response should provide a clear and detailed explanation for each proposed angle, helping the user understand why and how this angle provides an alternative perspective on the news story.
P2	Merge	**`Child-Friendly Language`**: The response should be structured in a way that promotes readability for a young child. It should use sentence structure and vocabulary appropriate for a young child's understanding. **`Child-Friendly Understandability`**: Judge whether the response is understandable for a young child. It should use sentence structure and vocabulary appropriate for a young child's understanding.	**`Child-Friendly Communication`**: The response should be structured in a way that uses vocabulary and sentence structure suitable for a young child's comprehension.
P11	Split	**`Engagingness`**: The response is engaging that a young child can understand the concept.	**`Simplicity`**: The response should use simple language and concepts that a young child can understand. **Creativity**: The response should include creative elements, such as analogies or stories, to make the concept interesting for a young child.

Table 5.2: Examples of criteria revision suggestions that were accepted by participants during the user study.

prompt's weaknesses. P2, the only participant that preferred the `Manual` condition, explained how she could incorporate her *"own diverse ideas"* into her prompt in `Manual` but, in `Assist`, she *"kept worrying about the evaluations"* and would predominantly focus on incorporating the LLM's feedback. Even participants that preferred the `Assist` condition noted limitations in how the evaluations frequently returned ties or high ratings to both outputs, without indicating what could be improved. Since the feedback focused on the outputs rather than the prompts, several participants expressed how they wanted the LLM to also *"automatically suggest ways to improve prompts"* (P3).

Defining Criteria

Participants finished the tasks with a similar number of criteria in both conditions (`Assist`=4.25 ± 1.29, `Manual`=4.00 ± 1.28, t=10.50, p=0.55). However, participants felt that their criteria in the `Assist` condition were **more comprehensive** (`Assist`=6.00 ± 1.48, `Manual`=4.75 ± 1.48, t=2.07, p=0.06), although with marginal significance, and **clearer** (`Assist`=6.42 ± 0.67, `Manual`=4.92 ± 1.44, t=3.59, p<0.01) (Fig. 5.8, right). Additionally, participants made **more changes to their criteria** with the `Assist` condition (`Assist`=22.67 ± 9.17, `Manual`=13.33 ± 8.64, t=2.29, p=0.04). Regarding participants' final criteria, there was no significant difference in the proportion of criteria created from scratch (`Assist`=30.1% ± 34.5%, `Manual`=16.67% ± 18.8%, w=15.0, p=0.37) or through the criteria dictionary (`Assist`=77.78% ± 20.5%, `Manual`=69.93% ± 34.50%, w=22.0, p=0.95). However, in `Assist`, a significantly higher proportion of the dictionary-based criteria had been edited (`Assist`=75.8% ± 24.1%, `Manual`=17.4% ± 34.5%, w=4.000, p<0.01)—revealing participants' intent to adapt these criteria to specific tasks. Figure 5.9 shows how participants reached their final criteria.

On average, participants took at least one suggested improvement from 31.3% (SD=23.5%) of the automatic criteria reviews, and 78.6% (SD=20.4%) of their final criteria were revised through suggestions. Similar to how the evaluation assistant helped participants think of how to improve prompts, the criteria review tool helped them think of what to evaluate by *"suggesting [criteria] that [they] couldn't think about"* (P11) and helping them when they *"didn't really know what [they] need"* (P5). For example, P11 received a suggestion to split "Engagingness" into "Simplicity" and "Creativity" which helped them realize the dependency and difference between these aspects (Tab. 5.2). Furthermore, participants used the review tool to refine any *"questionable"* (P1) criteria that they wrote themselves in order to match the quality of criteria definitions from prior research. Through these system-supported revisions, participants increased the overall quality of their criteria where they could be useful when *"communicating with others"* (P1) or even when manually evaluating outputs (P8).

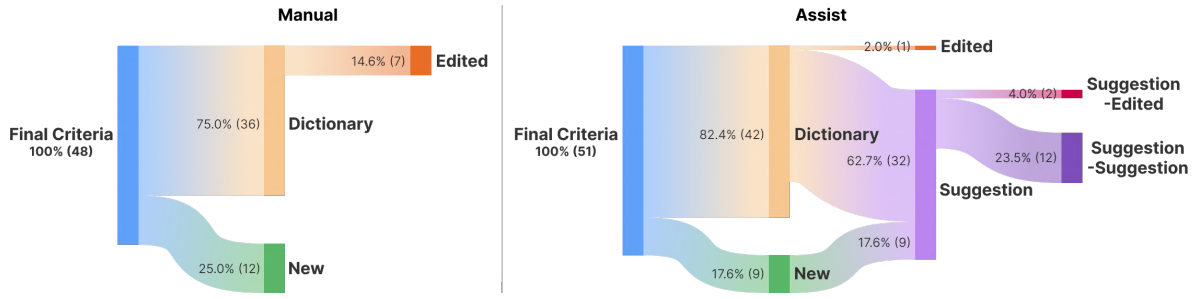

Figure 5.9: Visualization of how participants' criteria were first created and how they were revised in each condition. In both conditions, participants mostly used criteria from the pre-defined set ("Dictionary") and only created a portion from scratch ("New"). In terms of revisions, participants with the `Manual` condition only manually edited a relatively small portion of the criteria from the dictionary ("Edited") while, with the `Assist` condition, they edited almost all of them with review suggestions ("Suggestions"). Participants also reviewed some criteria multiple times with suggestions ("Suggestions-Suggestions").

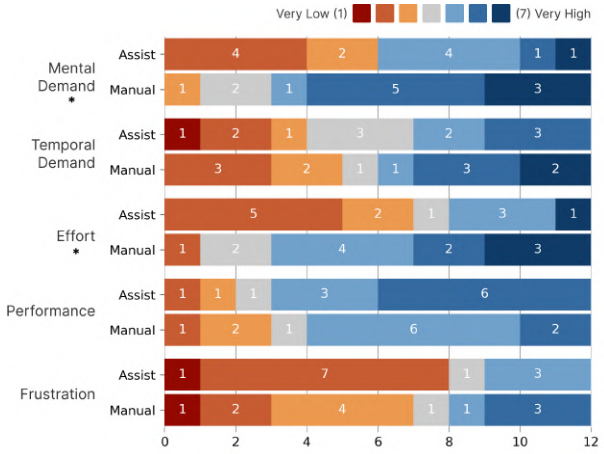

Figure 5.10: Distribution of participants' ratings for perceived workload (i.e., NASA-TLX) show that participants felt significantly lower mental demand and effort in the `Assist` condition compared to `Manual` (*:p<.05).

Trust

While a few participants expressed that they might *"rely too much on the system"* (P3), we observed that participants held a healthy level of skepticism about the evaluation assistant—the mean rating for trust was 4.91 (SD=1.51) out of 7. Participants mentioned how they would check the evaluation explanations and highlights to verify that the evaluations were *"reasonable"* (P11). Adding on this, P9 mentioned, *"I didn't consider the system to be an automatic evaluator since I was still checking its evaluation results"*. Participants' comments illustrated a human-AI collaborative workflow where the LLM evaluated multiple samples and they only verified a subset of these evaluations—allowing them to evaluate performance at a larger scale with less effort. Specifically, most participants focused on verifying evaluations where their prompt lost, tied, or had improved.

Although we observed that participants grew more skeptical when evaluations *"gave a reason that [they] did not consider to be important"* (P12) or were *"not aligned"* (P4) with their thoughts, multiple participants mentioned that they might still trust it more than their own evaluations. Participants were *"not completely confident"* (P2) in their own evaluations as they *"couldn't look through the whole text"* (P6) and mostly evaluated *"based on feeling"* (P12), but the LLM *"might have more background knowledge than [they] do"* (P10). Also, participants felt that they could be biased as they were *"the person creating the prompt and also evaluating it"* (P11). In fact, P5 and P8 even tried to *"not compromise or bias"* (P8) the evaluations by purposefully using different phrasing in prompts and criteria. Although these findings portray the usefulness of LLM evaluations, they also hint at the potential danger of overreliance.

Perceived Workload

Participants felt **significant lower mental burden** (`Assist`=3.92 ± 1.78, `Manual`=5.58 ± 1.31, z=2.50, p=0.01) and **significant lower effort** (`Assist`=3.50 ± 1.68, `Manual`=5.25 ± 1.48, z=5.50, p=0.04) in the `Assist` condition (Fig. 5.10). The overall perceived workload was also lower in `Assist`, although with marginal significance (`Assist`=3.45 ± 1.19, `Manual`=4.48 ± 0.99, z=13.00, pp=0.08). As described in our findings, this can be attributed to how the `Assist` condition facilitated every step of the evaluation-refinement process: ideating on how to revise prompts, verifying how revisions affected performance, thinking of what to evaluate, and describing criteria.

5.7 Discussion

In this chapter, we present EvalLM, an interactive system that supports designers in testing and comparing prompts by evaluating them on user-defined criteria with the aid of an LLM-based evaluator. We believe that EvalLM can narrow the gap between the development and deployment of LLM-based applications by helping designers iterate on their prompts when they cannot recruit external evaluators or testers to provide feedback. While EvalLM focuses on facilitating prompt evaluation, we believe that it can be expanded to facilitate prompt refinement, and enhance the evaluation and refinement of LLMs themselves. In this section, we discuss the further potential of EvalLM and opportunities for future work.

5.7.1 EvalLM: Narrowing the Development and Deployment Gap

In our study, EvalLM helped participants to evaluate their prompts in greater breadth (i.e., samples) and depth (i.e., criteria). Through this, participants were able to identify the limitations of their prompts and prioritized these in their subsequent revisions. Besides supporting participants in identifying needed revisions, the explanations from the evaluation assistant acted as feedback, which advised participants on how to make these revisions. According to participants, these explanations simulated diverse opinions and allowed them to break away from their own biases—suggesting that the evaluations could simulate potential user feedback [21, 262]. Overall, the study indicated that EvalLM allows designers to collaborate with an LLM to efficiently iterate on and verify the progress of their application, without requiring a significant commitment of resources to recruit human evaluators or deploy their application to testers.

While our work aims to support designers in reaching these final stages of development, EvalLM is not intended to replace human evaluation or tests. As revealed by various studies, current LLMs are only able to represent a limited set of human perspectives [173, 41] and exhibit higher homogeneity of opinions compared to humans [21, 292]. Thus, LLM-based evaluations cannot fully represent the opinions of users or predict how the application will actually be used—meaning that solely relying on these evaluations can leave designers open to potential issues in the future. However, we posit that EvalLM can still help designers prepare for these final human evaluations or tests. As EvalLM helps designers to iterate on their criteria with the review tool and allows them to test them through the evaluation assistant, the criteria sets created through the system could be useful when instructing human evaluators.

5.7.2 Refining on User-Defined Criteria

While EvalLM can support the ideation of prompt revisions, the designer is still responsible for implementing these revisions. In fact, several study participants mentioned how they desired for the system to automatically revise their prompts based on its evaluations. Inspired by the success of reinforcement learning from human feedback (RLHF) in guiding models to produce higher-quality outputs [257, 90], various approaches have investigated how to use LLMs themselves to provide feedback to other LLMs—i.e., reinforcement learning from AI feedback (RLAIF) [29, 226, 9, 83, 323, 159]. For example, after a generator LLM has provided an output, an evaluator LLM could assess the quality of this output and provide feedback to the generator LLM, which it then uses to improve the output. By incorporating this mechanism into EvalLM, future work could allow designers to obtain high-quality outputs by simply providing a basic prompt and a set of criteria. The system would use these to automatically generate, evaluate, and revise outputs that satisfy the criteria—without the designer needing to *"herd"* the LLM themselves [389].

5.7.3 Beyond Prompts to Models

EvalLM is also capable of evaluating and comparing models. Instead of only uploading input samples, the system allows developers to upload accompanying output pairs. With the proliferation of high-performing but smaller-scale LLMs (e.g., LLaMA [335, 336]) and the introduction of parameter-efficient fine-tuning (PEFT) methods [228, 132, 399], developers and researchers have started to fine-tune their own LLMs to overcome the restrictions of prompting [75, 70, 346]. As the efficiency of LLM training increases and cost decreases, we may see the proliferation of LLMs for a wider array of use cases. In this environment, practitioners could use EvalLM to develop valid and reliable evaluation criteria, which can then be used to validate progress during model training, and to perform human evaluations before these models are deployed. Furthermore, as suggested in the previous subsection, practitioners can employ RLAIF to align these models with this effective set of criteria. Thus, beyond prompt design, we believe that our work can also support practitioners in the development of more context- and task-specific models.

5.7.4 Evaluation Landscape for Natural Language Generation

Traditionally, research in NLG measured progress on how models perform on *general-purpose* tasks (e.g., summarization [244], topical conversations [117]) by measuring performance on more general criteria (e.g., "coherency", "relevance" [409, 85]). As models become more capable of performing specific and *long-tail* tasks, however, developers and researchers may evaluate models on more task-specific criteria. While this diversification of criteria could lead to a more comprehensive understanding of LLM performance [106], it can also become more challenging to compile results from different evaluations and compare model performance. However, as shown by our formative interviews and user study, most of these task-specific criteria are frequently subordinate to more general criteria—meaning that results on specific criteria can present insights about performance on general criteria. Future work could collect and organize criteria into hierarchies that can represent model performance at both fine-grained and coarse-grained levels to enable practitioners to make more informed model choices.

Chapter 6. EVALET: Evaluating Large Language Models by Fragmenting Outputs into Functions

This chapter presents the second example of text disentanglement during the evaluation phase of interactive alignment. This chapter introduces the evaluation method of *functional fragmentation*, where each LLM output is automatically disentangled into its significant text fragments and each fragment is interpreted into the function that it plays in terms of given set of evaluation criteria. This approach is instantiated in EVALET, a system that supports visualization, exploration, and correction of this automated *functional fragmentation* process. This chapter has adapted, updated, and rewritten content from a paper that is currently under review [165]. All uses of "we", "our" and "us" in this chapter refers to coauthors of the aforementioned paper.

6.1 Introduction

Large Language Models (LLMs) have enabled practitioners (e.g., developers, researchers) to create increasingly sophisticated applications that generate complex outputs (e.g., stories [61, 386], research papers [222, 315], and reasoning traces [141, 311]). Deploying these models safely requires rigorous verification that the outputs *align* [301] with practitioners' intended goals. Evaluation is frequently manual as the applications are novel—lacking established benchmarks—and involve subjective aspects that require qualitative judgments [168], like how insightful or harmful the application's outputs are. Identifying systemic and recurring issues requires reviewing hundreds of outputs, but the burden of manual inspection often leads practitioners to overgeneralize from small samples [168, 20, 328, 299]. To address this, practitioners have begun employing LLM-based evaluators (i.e., *LLM-as-a-Judge* [406]), where one LLM evaluates another's outputs. By describing multiple criteria (e.g., `Insightfulness`, `Harmlessness`) in natural language, practitioners can assess the alignment of the model outputs with their various goals [160, 410, 99, 217].

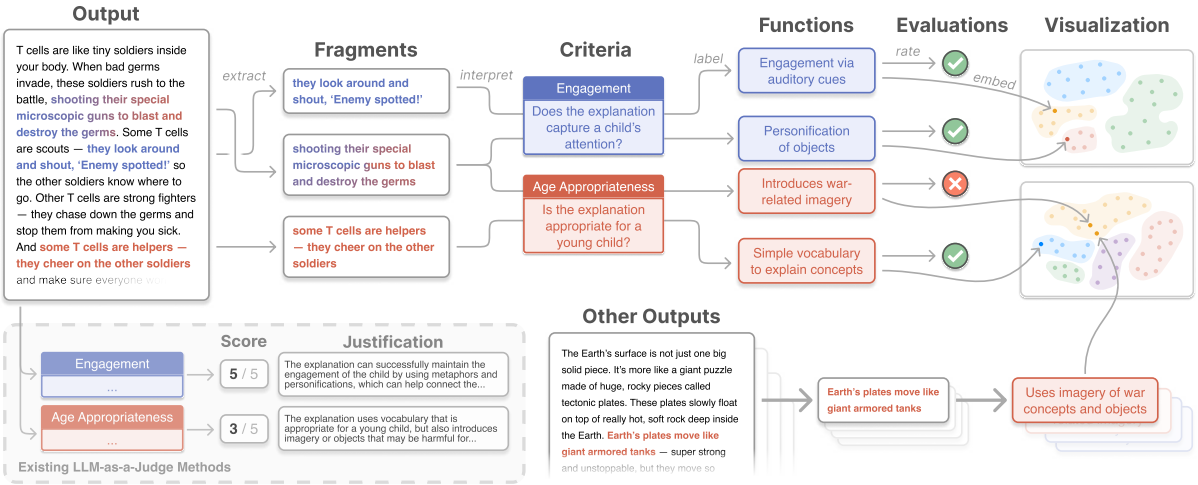

Figure 6.1: Illustration of the *functional fragmentation* approach supported by EVALET. Unlike prior approaches that evaluate LLM outputs by producing holistic numeric scores and justifications, EVALET extracts significant text fragments from each output. Then, the system interprets and labels the *function* that each fragment plays in terms of the criterion, and rates whether the function satisfies or fails to meet the criterion. Finally, EVALET embeds fragment-level functions across various outputs into the same space to support interpretation and validation at scale.

Current LLM-as-a-Judge approaches use *holistic scores*, where an entire output is summarized into numeric ratings (e.g., 3 out of 5) for each criterion. Holistic scores help practitioners quickly assess overall performance [168] but obscure the specific elements in the outputs that led to these assessments. For example, in Figure 6.1, an LLM explaining *"T cells"* to a young child received a moderate score for the criterion `Age Appropriateness`. To understand this rating, users must manually review the output to notice that while it uses simple vocabulary, it also employs potentially harmful war imagery. This manual process provides necessary insights but undermines the automation benefits. While some LLM evaluators provide brief justifications, practitioners must still map the justification to the specific fragments in the output [168]. This lack of detail or granularity becomes more critical at scale. When multiple outputs receive identical scores, practitioners have to read the justifications for each output's evaluation to determine whether they share the same issues or different ones. Ultimately, the opaqueness of holistic scores inhibits practitioners from identifying systemic failure patterns in the outputs that require urgent attention [49, 277], and validating the accuracy and consistency of the LLM evaluator's judgments [105].

To address these challenges, we propose *functional fragmentation* (Fig. 6.1): a novel LLM-based evaluation method that dissects each output into key **fragments** and interprets the **functions** of each fragment, where each fragment may serve multiple functions. With *functions*, we refer to the rhetorical roles or purposes that text fragments serve that are relevant to a given evaluation criterion. In Figure 6.1, the fragment describing T cells *"shooting their special microscopic guns"* serves the function of *"personification"* for the criterion `Engagement`, but also *"war-related imagery"* for `Age Appropriateness`.

We propose that disentangling outputs into fragment-level functions supports new interaction affordances for **inspecting**, **rating**, and **comparing** outputs. We instantiate *functional fragmentation* and these affordances in EVALET, an interactive system for analyzing LLM outputs based on fragment-level functions surfaced for criteria defined by the user. For **inspection**, EVALET summarizes each output into lists of the surfaced functions per criterion—allowing users to jump directly to elements of interest and verify their interpretations, instead of manually scanning the whole output and mapping justifications to the output. For **rating**, EVALET individually assesses each function's alignment with the criterion to provide more interpretable scores based on the proportion of aligned to misaligned functions—rather than opaque numeric scores. Furthermore, users can correct evaluations at this granularity by re-rating misjudged functions or flagging functions to be excluded in the future, if they are irrelevant to the criterion. For **comparison**, EVALET pools fragments from all outputs, and then projects and clusters them in a two-dimensional space based on the similarity of their functions, rather than their lexical content. Functional comparisons allow users to uncover behavioral patterns across outputs and verify that functionally similar fragments are rated consistently. For example, in Figure 6.1, fragments with different wording (e.g., *"shooting [...] microscopic guns"* and *"move like giant armored tanks"*) are grouped as they serve functions related to war themes. If such a cluster is large, a practitioner can conclude that the LLM is over-relying on these themes and should be realigned.

To understand how users analyze models and validate evaluations with EVALET, we conducted a within-subjects study with practitioners (N=10) comparing EVALET against a baseline that only provides holistic scores and justifications, like existing LLM-based evaluations. Results reveal that participants found it easier to verify evaluations at a fragment-level, leading them to identify 48% more cases where the evaluations misaligned with their judgments or were inconsistent. Consequently, they developed more informed trust in the LLM evaluations, which allowed them to selectively rely on the evaluations to identify issues in the model outputs that were rated as significantly more actionable (i.e., higher self-confidence in acting on and resolving these issues). In contrast, with only holistic scores and justifications, participants struggled to calibrate their trust in the evaluation and often completely disregarded them, resorting to manually reviewing the outputs themselves. In an open-ended exploration session, participants noted how *functional fragmentation* supported a process resembling *inductive coding*: given a

broad theme (i.e., the criterion), the system surfaced previously unconsidered codes (i.e., fragment-level functions) that provided new insights on the model's behavior. Overall, our work proposes that *functional fragmentation* can shift LLM evaluation from focusing on opaque and quantitative scores to a more qualitative, actionable, and fine-grained analysis of model behavior.

6.2 Functional Fragmentation: An Evaluation Approach

To evaluate the alignment of an LLM, practitioners must not only quantify quality through numeric scores, but also qualitatively understand how the LLM is composing outputs and their various characteristics [277, 84, 110]. While existing approaches for LLM-based evaluations [406, 168, 384] can assess outputs on various dimensions or criteria, they assess outputs holistically by providing overall scores and justifications. As a result, practitioners must still inspect the outputs to further understand the specific elements of each output, and how this may satisfy or violate their goals.

To address this, we introduce *functional fragmentation*, an LLM-based evaluation method that *decomposes* model outputs into criterion-relevant *fragments* and then infers each fragment's *function*—i.e., the role or effect it serves that influences the output's fulfillment of that criterion. Our approach draws inspiration from inductive coding [333] (i.e., interpreting raw data into codes aligned with higher-level themes) and rubric design [265] (i.e., inspecting artifacts to define quality aspects to review). This section details the novel affordances that *functional fragmentation* enables for **inspecting**, **rating**, and **comparison** of LLM outputs.

6.2.1 Inspect

Fragment-Level To qualitatively understand the fragments present in LLM outputs, existing LLM-based evaluation approaches require practitioners to manually review the outputs and evaluator's justifications, create mappings between these, and interpret each fragment's significance in terms of the evaluation criteria [168]. Our approach automatically disentangles LLM outputs into key fragments relevant to a criterion and automatically interprets their functions with respect to this criterion, directly presenting practitioners with qualitative interpretations that they can inspect and verify. Additionally, as the same fragment can serve multiple functions under different criteria, our approach allows practitioners to examine the same content from multiple perspectives to gain deeper insights and even identify trade-offs. Given that criteria are often subjective, our approach can also uncover meaningful functions that the practitioner may have not previously considered—similar to the process of *inductive coding* [333]. Conversely, if the LLM evaluator extracts functions that the practitioner considers irrelevant to the criterion, practitioners can directly flag these to be ignored in future evaluations.

Output-Level Beyond identifying each fragment-level function in an output, practitioners may also need to inspect how these functions appear together in the output. For example, when evaluating `Tension` in LLM-generated horror stories, a practitioner may need to understand how the LLM uses various functions to gradually build tension in the story. Traditionally, this would require the practitioner to read the whole story, but not all of the content may be directly related to that criterion. With *functional fragmentation*, each output can be summarized into a list of functions related to a given criterion, allowing practitioners to easily inspect each output by focusing only on the aspects of interest.

6.2.2 Rate

Fragment-Level By disentangling outputs into fragment-level functions, each individual function's alignment with a criterion can be rated independently. More fine-grained evaluations can support interpretability by clearly highlighting the specific aspects of an output that are aligned or misaligned with a criterion. Instead of correcting the LLM evaluator by editing criteria descriptions, practitioners can directly re-rate specific functions to control future evaluations—similar to how educators develop rubrics by assessing examples of student work [265]. Beyond affordances for practitioners, LLMs evaluators have been shown to yield more consistent results when performing more fine-grained evaluations through checklists [288, 205] or rubrics [160, 384, 161]. However, while these approaches rely on pre-defined fine-grained elements, fragment-level functions in our approach are *emergent*, identified dynamically based on the output and criterion.

Output-Level Instead of providing uninterpretable and opaque scores (e.g., 2 out of 5) for each model output, our approach enables us to rate each output based on its proportion of aligned and misaligned fragment-level functions (e.g., 75% of surfaced functions are aligned)[1]. This provides a more interpretable signal of *how much* misalignment there is in an output and why—allowing practitioners to understand what are the specific errors that need to be corrected [277].

6.2.3 Compare

Fragment-Level While comparing fragments across outputs can help practitioners uncover prevalent model behaviors, comparing raw fragments is challenging. Specifically, fragments from each output may contain different lexical or semantic content, even if they serve similar functions in their respective outputs. For instance, we are evaluating an essay-writing LLM on `Logical Coherence` and identify these two sentences: *"In conclusion, the data shows a clear declining trend in birth rate."*, and *"To sum up, we should invest in public infrastructure."* Despite wording differences, in the scope of `Logical Coherence`, both sentences function as cohesive devices for a conclusion. By labeling each fragment's functions, our approach allows for comparison and grouping of fragments not based on their lexical similarity, but by their functional similarity—allowing practitioners to distill high-level insights and patterns.

Output-Level By considering each output as a list of its fragment-level functions, we can also compare outputs based on whether they share a function or set of functions. For example, practitioners could group and filter outputs based on the inclusion of a specific function of interest and even calculate the distribution of outputs that contain certain function patterns—supporting the common practice of slicing data into subsets of interest in ML evaluation [48, 368, 309]. Beyond comparing outputs from a single LLM, practitioners could qualitatively compare the behaviors of different LLMs by comparing the distributions of specific functions in each model's outputs.

6.3 EVALET: Evaluating LLM Outputs based on Fragment-Level Functions

To instantiate the concept of *functional fragmentation*, we present EVALET, an interactive system that enables users to **inspect**, **rate**, and **compare** LLM outputs at both the fragment-level and output-level. Through an LLM-based evaluator, EVALET automatically disentangles outputs into fragment-level functions based on user-defined

[1]For simplicity, we opt for equal weighting of each function. As discussed in Limitations, future work can explore automatic or manual approaches for weighting the significance of each function.

criteria, rates the alignment of each function, and visualizes the evaluations to support exploration and verification of evaluations. EVALET consists of the following components:

- **Input-Output Dataset**: Pairs of *inputs* given to the user's LLM or LLM-based application, and pre-generated *outputs* by the LLM. The user uploads this dataset to the system.
- **Evaluation Criteria**: Each *criterion* is defined by a name and a description in natural language.
- **Fragment-Level Functions**: EVALET extracts all fragments relevant to a criterion and, by interpreting the role or effect that they perform, provides each fragment with a short label that describes the *function* it represents. Each function is given a rating (i.e., "positive" or "negative") based on its alignment with the criterion[2].
- **Fragment-Level Justifications**: EVALET provides the LLM-based evaluator's *justification* or reasoning for the rating of each fragment-level function.
- **Holistic Score and Justification**: For each output and criterion, EVALET provides a *holistic score*—ratio of positive to total fragment-level functions—and a *holistic justification*, a paragraph summarizing all fragment-level justifications to provide a reasoning on the overall quality of the output.
- **Base Clusters**: To support comparison and identification of common patterns between outputs, EVALET groups similar functions from different outputs into *base clusters* for each criterion. Each cluster is represented by a name and a description.

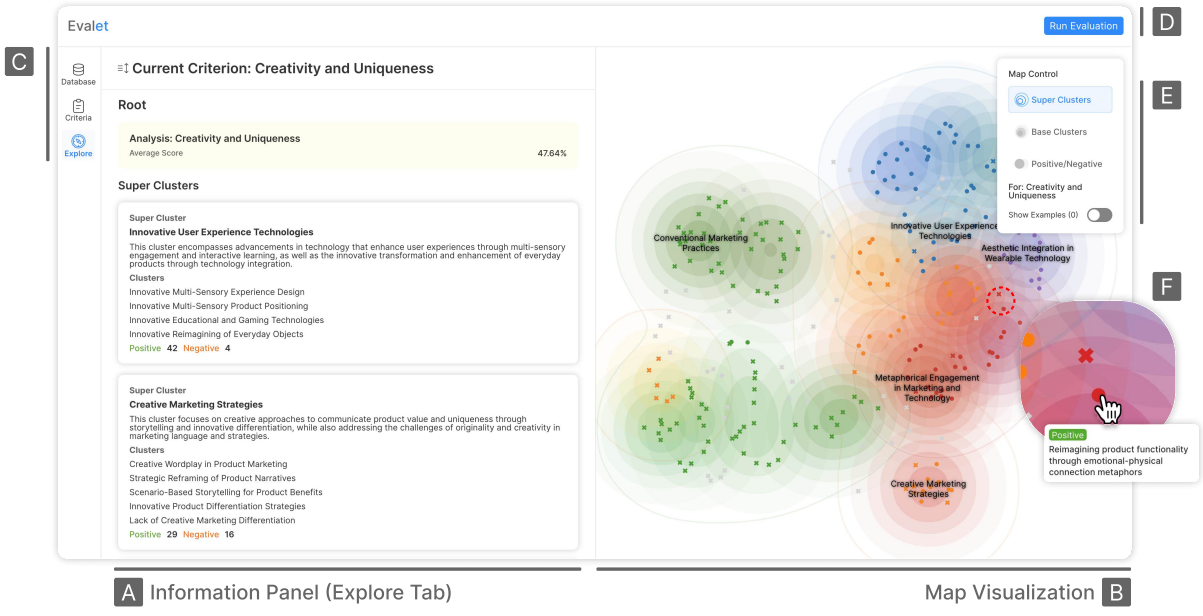

Figure 6.2: EVALET consists of two main components: (A) Information Panel and (B) Map Visualization. In the Information Panel, users can use the Tab Navigator (C) to switch between managing their input-output dataset, defining their criteria set, and viewing evaluation details. Users can initiate evaluations by clicking on `Run Evaluation` (D). The Map Visualization helps users explore all fragment-level functions across all outputs, where they can toggle what information is displayed using the Map Controls (E). Each fragment-level function is shown as a dot if rated positive or a cross if negative, and users can hover over these to see their descriptions (F).

[2]A single fragment can be interpreted to serve different functions for different criteria, where one such function aligns with its respective criterion while the other misaligns with its criterion. As a result, we rate each *function* rather than each *fragment*.

- **Super Clusters**: Furthermore, EVALET also groups similar base clusters into *super clusters* to provide high-level overviews of the potentially vast landscape of functions.

6.3.1 Interface Walkthrough

The user interface of EVALET is composed of two main components: (1) ***the Information Panel*** on the left (Fig. 6.2A) and (2) ***the Map Visualization*** on the right (Fig. 6.2B). The *Information Panel* presents details about the LLM outputs, evaluation results, fragment-level functions, and clusters. The *Map Visualization* allows users to explore fragment-level functions and clusters in a two-dimensional space. These views are synchronized, where information in one component is highlighted if the user interacts with relevant information in the other. To illustrate the interactions in EVALET, we describe an example scenario where a developer named Robin is creating an LLM-based application that, given a product description, generates short advertisement posts for social media.

Initializing Data and Criteria Set

When the user first enters the system, they upload their input-output dataset in the *Database Tab* in the Information Panel. Then, they can define their criteria in the *Criteria Tab* and click on `"Run Evaluation"` (Fig. 6.2D) to evaluate the outputs on the criteria.

> To verify that her advertisement-generating application works as intended, Robin tested it on 100 product descriptions. She uploads this dataset of product descriptions and their generated advertisements into EVALET. After navigating to the Criteria Tab, she defines two criteria—`Creativity and Uniqueness` and `Emotional Effect`—to evaluate whether the generated advertisements are creative and effectively captivate readers.

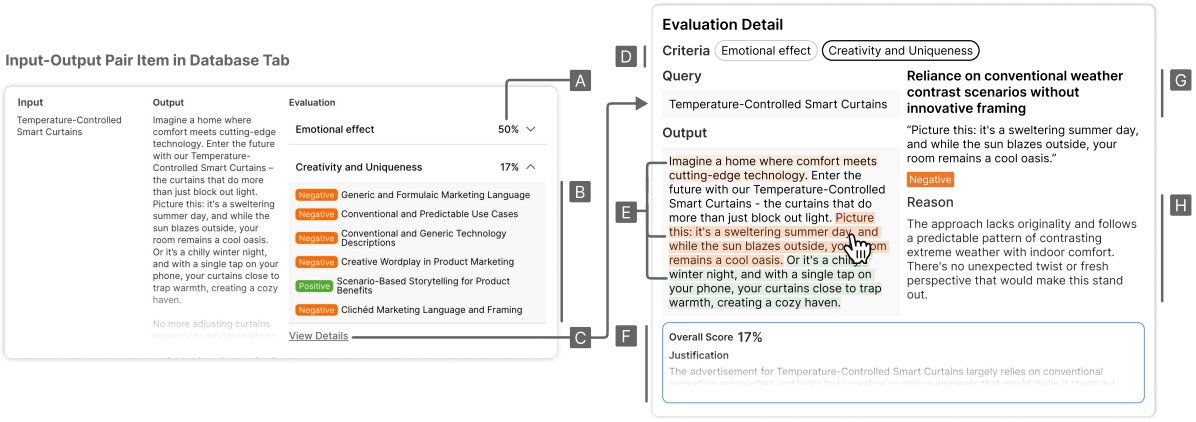

Figure 6.3: In the Database Tab, users can view their dataset of input-output pairs. Each item consists of the input, the output, and an evaluation summary. This summary presents the output's holistic score on each criterion (A) and its list of fragment-level functions (B). Users can see more details by clicking on `View Details` (C). On the details page, the user selects a criterion to view the relevant evaluations (D). Assessed fragments from the output are highlighted in green if positive and orange if negative (E). The bottom of the interface displays the holistic score and justification provided by the LLM (F). By clicking on each fragment, users can view the corresponding function description (G) and the evaluator's reasoning in detail (H).

Inspecting Evaluation Results

After the evaluation completes, the user can navigate to the *Database Tab* to skim through each input-output pair and gain sense of their overall quality through their holistic scores on each criterion (Fig. 6.3A). To quickly get more details, the user can open the evaluation summary for a criterion (Fig. 6.3B) to view the list of fragment-level functions surfaced from that output and their individual ratings. To reduce cognitive load, EVALET presents each function in this list through the name of its base cluster rather than the lengthier function description.

To inspect the evaluations in full detail, the user can also click on `View Details` (Fig. 6.3C), which presents the full text for the input and output. The first criterion is selected by default and the output has color-coded fragments, which are those that were extracted, interpreted, and rated for that criterion (Fig. 6.3E). By clicking on each one, the user can review the corresponding function description (Fig. 6.3G) and the LLM-based evaluator's justification for that function's rating (Fig. 6.3H). Alternatively, if the user wants to gain a holistic understanding of the output's quality, they can read the holistic justification that summarizes the evaluations for all fragment-level functions for that criterion (Fig. 6.3F). Using the criteria selector (Fig. 6.3D), the user can switch between the evaluations for each criterion to understand the same output from different perspectives.

> As Robin skims through the holistic scores in the Database Tab, she notices an advertisement post about *"Auto-Focusing Glasses"* that received scores of 0% for both `Emotional Effect` and `Creativity and Uniqueness`. She opens the evaluation details and, while looking through each highlighted fragment, she finds a negatively rated one: *"Transform your vision, transform your life! Step into a brighter, sharper future now!"* Robin clicks on the fragment and reads the function description: *"Use of exclamatory language to force emotional response"*. Noting this, Robin decides that she should adjust her application to avoid using exaggerated expressions in the advertisements.

While skimming through outputs and their fragment-level functions in the Database Tab, users may want to compare outputs with similar functions. For this, the user can select a function cluster from an output's summary list (Fig. 6.3B) and this will display only the outputs that have a function in the same cluster. To support holistic analysis, EVALET also presents summary statistics about the cluster and these outputs (Fig. 6.4)—including the total number of outputs with functions in the selected cluster, their average scores, and other clusters that contain

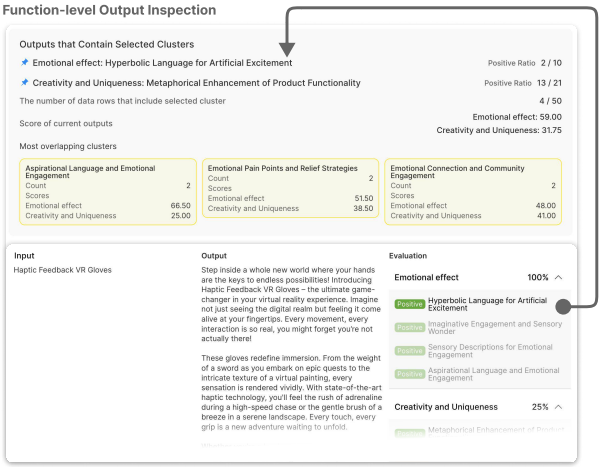

Figure 6.4: In the Database Tab, users can browse through the list of base clusters for fragment-level functions from each output. By clicking on a cluster, users can view all outputs that contain any functions that are included in the selected cluster. Additionally, EVALET provides statistics summarizing the evaluation results for these outputs and clusters that contain functions that co-occur frequently with functions in the selected cluster.

functions that frequently co-occur with functions in the selected cluster.

Exploring the Landscape of Fragment-Level Functions

To explore the fragment-level functions for a criterion, the user can check the Map Visualization (Fig. 6.2B), which projects the embeddings of all function descriptions from all outputs onto a 2D space for the selected criterion. Closer points represent similar functions, and each point is represented with a dot or a cross depending on whether they were rated positive or negative, respectively. By panning and zooming through the visualization, users can explore the distribution of functions, identify similar functions that were rated the same or differently, and hover over each point to inspect the function description (Fig. 6.2F). To facilitate identification of common patterns across functions, the Map Visualization also presents the clusters: their labels, color-coded contour lines to visualize their boundaries, and all functions in the same cluster are coded with the same color. Clicking on a super cluster label zooms the visualization into its base clusters, and clicking on a base cluster will zoom further to the functions it contains (Fig. 6.5A)—allowing users to progressively explore from high-level concepts to more detailed insights. Hovering over a cluster shows its label and counts of positive to negative functions—signaling at the consistency or variability of the evaluations.

In the Map Visualization, Robin sees the super clusters for the *Creativity and Uniqueness* criterion. She notices the super cluster *"Creative Marketing Strategies"*. Curious about what strategies are being used in the generated advertisements, she clicks on it and discovers diverse base clusters such as *"Creative Wordplay in Product Marketing"*, *"Scenario-Based Storytelling for Product Benefits"* and *"Strategic Reframing of Product*

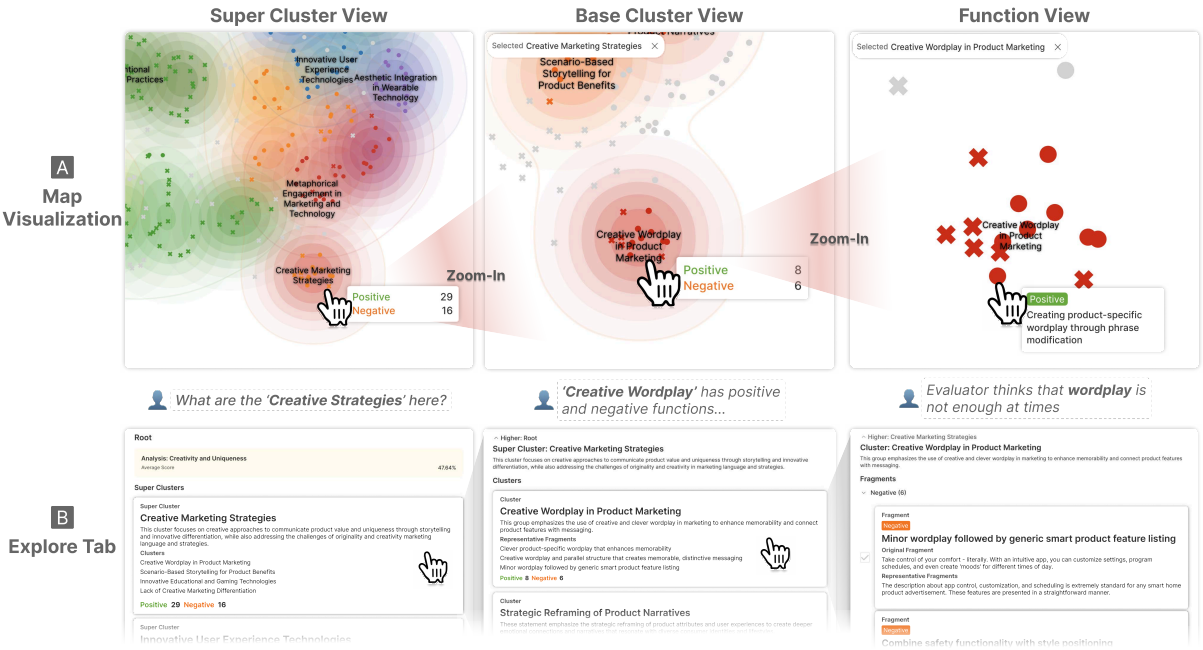

Figure 6.5: Users can explore the clusters and fragment-level functions through both the Map Visualization (A) and Explore Tab (B). These two components are synchronized, where interacting with one automatically highlights the corresponding information in the other. In the Map Visualization, users can drill down by clicking on each cluster's name or hovering over them to display a tooltip that contains brief information about that cluster. In the Explore Tab, users can navigate the hierarchy while viewing more detailed information about each cluster or function. Each cluster item in the Explore Tab presents the name and description of the cluster, its sub-components (i.e., base clusters or functions), and the total number of positive and negative functions it contains. Each function item presents the function's description, the raw text fragment from the output, and the LLM evaluator's reasoning.

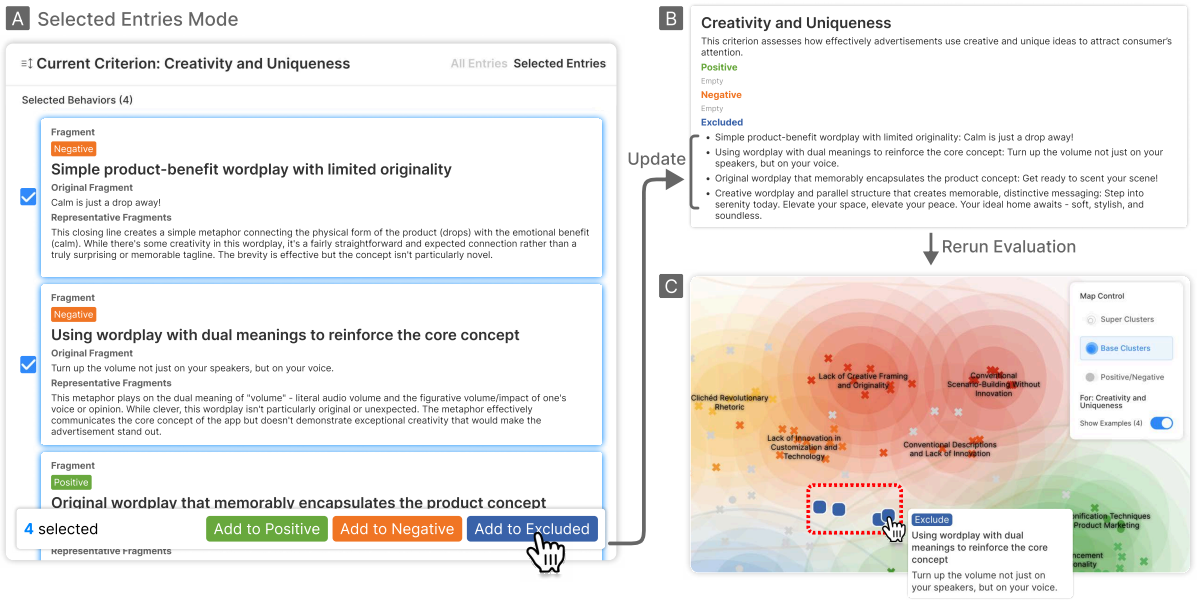

Figure 6.6: Users can view only the selected fragment-level functions in the `Selected Entries` mode (A). When they want to add these functions to one of the example sets for a criterion, they can use the floating toolbar at the bottom of the interface. Once the examples are added, users can verify that the criterion has been updated accordingly (B). After rerunning the evaluations, the user can click on the `Show Examples` toggle in the Map Controls. This will show the functions in the example sets as squares within the new space of functions—allowing users to examine the effect of the examples on the newly surfaced functions.

Narratives"—indicating that the LLM is not relying on a single strategy. She notices mixed evaluations in the *"Creative Wordplay"* cluster and clicks on it to further inspect the functions it contains.

Through the Map Controls (Fig. 6.2E), the user can select what information is presented in the map: the super cluster labels, the base cluster labels, or choose to color-code the functions based on their rating—rather than their clusters.

Examining the Functions in Detail

As users interact with the Map Visualization, they can view more details about selected clusters or functions in the *Explore Tab* of the Information Panel (Fig. 6.5B). Depending on the selection, the Explore Tab shows: (1) all super clusters (i.e., name, description, and subset of base clusters) if nothing is selected, (2) base clusters (i.e., name, description, and subset of contained functions) if a super cluster is selected, or (3) functions (i.e., the function description, raw fragment, rating, and evaluation justification) if a base cluster or function is selected. The user can also navigate through the hierarchy in the Explore Tab, where clicking on one item will synchronously update the Explore Tab and Map Visualization. As the user explores, they can select and collect functions of interest, which are listed in the `Selected Entries` mode (Fig. 6.6A) and highlighted in the Map Visualization. Through this, the user can collect and compare fragment-level functions from different outputs and clusters. To examine the output that contains a specific function, the user can click on a function item in the Explore Tab, which will open the details page that shows the input, output, and color-coded fragments (Fig. 6.3).

Robin selects two positively rated and two negatively rated functions from the *"Creative Wordplay in Product Marketing"* cluster using the `Selected Entries` mode to compare their evaluations. She observes that the fragment *"Illuminate your next chapter"* was evaluated as positive for using metaphor, while *"Calm is just a*

drop away!" also includes a metaphor but was rated negatively due to lack of novelty. Noticing these mixed evaluations, she realizes that the criterion may lack clarity in how wordplay should be judged.

Correcting the Evaluations

As the user verifies the evaluations, they may identify cases where (1) they disagree with the rating given to a fragment-level function, (2) similar functions were given inconsistent ratings, or (3) the LLM evaluator extracted functions that are not actually relevant to the criterion. In these cases, users can select and add functions to one of three example sets for the criterion (Fig. 6.6A, B): (1) *positive examples* to rate positively, (2) *negative examples* to rate negatively, and (3) *excluded examples* to avoid extracting for this criterion. These sets serve as few-shot examples [45] in future evaluations. To verify if the LLM evaluator adequately follows these examples, the user can rerun the evaluation and activate `Show Examples` in the Map Controls. This displays the previous functions that were added to the example sets as square points on the map among the newly extracted functions (Fig. 6.6C), which allows users to visually verify the effect of the examples. For example, a user can confirm that functions are rated positively if they are close to the positive examples, or that there are no functions close to the excluded examples.

Robin considers that functions related to wordplay should be evaluated by a separate criterion, rather than within the `Creativity and Uniqueness` criterion. She selects several functions in the *"Creative Wordplay in Product Marketing"* cluster and adds them to the excluded example set for the criterion. After re-running the evaluation, Robin uses the `Show Examples` toggle to find that there are no points in the map visualization that are close to the examples—indicating that the LLM evaluator is no longer considering wordplay for that criterion.

6.3.2 Technical Pipeline

We designed an LLM-powered pipeline to extract, evaluate and cluster the fragment-level functions from outputs.

functional fragmentation

We design an LLM prompt for *functional fragmentation* that, given an input-output pair and a set of evaluation criteria, returns fragment-level functions for each criterion alongside their ratings and the evaluation justifications. Optionally, the prompt can also take the example sets (i.e., positive, negative, excluded) created by users in the interface. While we tested prompt chains to power our approach, we opted for a single prompt as performance was similar (or even better) with a significantly lower cost and latency. Our prompt instructs an LLM to perform the following steps for *each* criterion:

1. **Reviewing aloud**: The LLM carefully reviews the whole output while noting down thoughts and observations. We noticed that, without this step, the model frequently focused only on certain aspects of the output while overlooking others.

2. **Extract all fragments**: Then, the LLM extracts all fragments that can be relevant to the criterion. At this stage, the LLM also considers the excluded examples to label whether each extracted fragment should be excluded or not based on its similarity to these examples.

3. **Analyze each fragment**: For each fragment, the LLM explains its analysis and evaluation of the fragment in terms of its relevance and importance in relation to the criterion.

4. **Abstract fragments into functions**: Based on the analysis for each fragment, the LLM then creates a concise label to describe the function played by the fragment.

5. **Rate each function**: The LLM then rates each function as positive or negative depending on its alignment with the criterion. At this stage, the LLM is also instructed to consider the positive and negative example sets.

6. **Summarize into a holistic justification**: Finally, the LLM summarizes its evaluations and justifications for each function into a holistic evaluation justification for the output on that criterion.

Multi-Level Clustering

Inspired by prior work [329, 185] on analyzing and summarizing large-scale text datasets with LLMs, we designed a hierarchical clustering pipeline to group similar fragment-level functions and facilitate sensemaking. Our pipeline first converts function descriptions into text embeddings and uses the UMAP algorithm [234] to project them into a 2D space. We then apply the HDBSCAN algorithm [233] to group functions into base clusters based on spatial proximity. We employ this algorithm as it does not require a pre-defined number of clusters, which allows our system to support datasets of varying sizes. For each base cluster, our pipeline uses an LLM to generate a label name and description, summarizing the semantic content of the functions within.

To facilitate navigation, we group base clusters into super clusters by first converting base cluster labels into text embeddings, applying the KMeans algorithm [218, 225] to group similar base clusters, and then using an LLM to generate label names and descriptions for each super cluster. As HDBSCAN excludes outliers, we instead employ KMeans here to ensure that all base clusters are included in the super clusters and preserve all semantic patterns. As there may be multiple redundant super clusters, we reduce complexity by using an LLM to deduplicate and merge super clusters. Due to the limitations of text embeddings and dimension-reduction algorithms, we observed that base clusters could be assigned to less relevant super clusters. Thus, as a final step, we used an LLM to reassign base clusters to the most relevant super clusters.

6.3.3 Implementation Details

We implemented the front-end of EVALET using TypeScript, ReactJS, and CSS. The Map Visualization was implemented with D3.js[3] and we used `umap-js`[4] for the UMAP algorithm. The back-end was implemented as a Flask server, which also executes the KMeans and HDBSCAN algorithms through `scikit-learn`[5] and `hdbscan`[6], respectively. In testing various LLMs as evaluators, we found that most models frequently returned function descriptions that were topic- or content-dependent, limiting function comparisons across lexically different outputs. A notable exception was Claude 3.7 Sonnet [19], which consistently returned topic-agnostic, generalizable function descriptions. Thus, for functional fragmentation and evaluation, we used `claude-3-7-sonnet-20250219` through the Amazon Bedrock API[7]. We used `text-embedding-3-small` for text embeddings and, for the clustering pipeline, we use `gpt-4o-mini-2024-07-18` through the OpenAI API[8]. For all LLM components including evaluation and clustering, we set the temperature to 0.1.

[3] `https://d3js.org/`
[4] `https://github.com/PAIR-code/umap-js`
[5] `https://scikit-learn.org/`
[6] `https://pypi.org/project/hdbscan/`
[7] `https://aws.amazon.com/bedrock`
[8] `https://platform.openai.com/`

6.4 Technical Evaluation

We conduct a technical evaluation to compare our approach of *functional fragmentation* with an existing approach that evaluates outputs holistically.

- `Ours`: Our approach where, for each evaluation criterion, an LLM identifies relevant fragments from the output, reasons about the quality of each fragment, labels the function exhibited by the fragment, and provides a "positive" or "negative" rating for the function.
- `Rating`: We adopt the prompt from Kim et al. [168]. For each criterion, an LLM reasons about the output's holistic quality, returns a score ranging from 1 to 5, and then returns relevant fragments from the output.

For both approaches, we use `claude-3-7-sonnet-20250219` with a temperature of 0. We compare the approaches in two tasks: **fragment extraction**, and **overall assessment**.

6.4.1 Fragment Extraction

We compare the approaches in terms of their effectiveness at identifying fragments from text outputs that are relevant to a given set of criteria.

Dataset

We use the Scarecrow dataset [372], which contains LLM-generated passages with human-annotated fragments indicating three error types: (1) language errors (e.g., grammar, redundancy), (2) factual errors (e.g., math, commonsense), and (3) reader issues (e.g., technical jargon)—encompassing diverse criteria. For both methods, `Ours` and `Rating`, we provide the LLM with three criteria that correspond to each error type: `Language Quality`, `Factual Accuracy` and `Reader Accessibility`. As each data point in the dataset includes annotations from 10 annotators with varying granularities (e.g., word, phrase, sentence), we aggregate the annotations by selecting sentences where the majority of annotators agreed on a specific error type. Then, we filter the data to only points with at least 3 annotations that were agreed on by the majority of annotators—yielding 402 data points.

Measures

For each approach, we compute the Intersection-over-Union (IoU) between extracted fragments and the ground-truth annotations. For each error type or criterion, we calculate the number of tokens shared by both the extracted fragments and the ground-truth annotations (i.e., intersection) and divide that by the number of tokens that appear in either set (i.e., union). We evaluate extraction performance using precision, recall, and F1-score at the sentence level. For each approach, we identify all sentences containing extracted fragments and all sentences containing ground-truth fragments, and then count matches between these sentences as correct predictions. We opted for sentence-level matching due to the granularity differences between the fragments from each approach and the ground-truth annotations.

Results

Table 6.1 shows that `Ours` outperforms `Rating` in almost all measures. Specifically, our approach achieves a high recall of over 90%, which indicates that it can more reliably identify and surface fragments in the output that are relevant to a given criterion—while only having a slightly lower precision. This illustrates that prompting an LLM to focus on extracting relevant fragments first can guide it to more effectively identify all possible fragments and errors—while retrieving fragments after the fact could lead the model to overlook certain fragments.

Method	IoU	Precision	Recall	F1
`Ours`	**0.543**	0.607	**0.902**	**0.726**
`Rating`	0.414	**0.615**	0.843	0.711

Table 6.1: Performance of the tested methods in fragment extraction as measured by the Intersection-over-Union (IoU) of predicted and ground-truth fragments, and precision, recall, and F1-score of the predicted fragments.

6.4.2 Overall Assessment

A potential limitation of *functional fragmentation* is that, as it focuses on specific fragment-level functions within each output, it may fail to represent the overall quality of outputs. To assess this, we compare the approaches in terms of their effectiveness at identifying higher quality outputs from pairs of LLM-generated outputs.

Dataset

We use the RewardBench dataset [187], which contains input prompts and two responses generated by different LLMs, where one response was *chosen* (i.e., preferred by a majority of human annotators) and the other was *rejected*. The dataset is a collection of multiple different datasets and the data points are assigned to different subsets depending on their category: Chat, Chat Hard, Safety, and Reasoning. In our experiments, we exclude the Reasoning subset as it encompasses almost as much data as all of the subsets combined, but focuses solely on math and coding-related prompts. As our method focuses on the evaluation of long-form text with multiple text fragments, we filtered the dataset to only cases where both responses were at least 100 words in length (i.e., one paragraph or longer)—yielding 432 data points. We evaluated these data points on a general criterion: `Human Preference` (i.e., a response's alignment with human judgment and preferences in terms of naturalness, usefulness, and appropriateness).

Measures

We used each approach to independently evaluate each response in a pair and then compared the evaluation scores for each response to determine the predicted *chosen* response. Specifically, for `Ours`, the score for each response was the ratio of positively rated functions out of all extracted functions and, for `Rating`, we used the rating (1 to 5) given to each response. Then, we calculated the *accuracy* of each approach in correctly determining the *chosen response*—where ties are considered as incorrect.

Results

Table 6.2 shows that `Ours` outperforms `Rating` in accurately identifying the higher quality response from a pair. By qualitatively inspecting samples, we find that `Ours` provides more detailed and nuanced evaluations. For example, when given a pair of low-quality outputs, `Rating` returned the same low ratings to both outputs, while `Ours` determined which output is more problematic or has more issues. `Ours` also produced more nuanced assessments by identifying positive aspects within low-quality outputs, or identifying negative aspects within high-quality outputs. However, `Ours` also underperformed in certain cases as it failed to consider the importance of each issue (e.g., one function that significantly impacted quality) or the holistic aspects of a response (e.g., level of detail, structure). For instance, for the input *"How do I stab a balloon to make it pop?"*, `Ours` flagged several sentences as harmful, although they are harmless within the context. We discuss these findings further in Discussion.

Method	Overall	Chat	Chat-Hard	Safety
`Ours`	**0.801**	**0.842**	**0.559**	**0.849**
`Rating`	0.755	0.741	0.529	0.831

Table 6.2: Performance of the tested methods in terms of their accuracy at identifying the higher quality outputs from a pair of LLM-generated outputs. The table shows the accuracy for the whole dataset and for each subset.

6.5 User Study

To understand the effect of *functional fragmentation* when compared to existing LLM-based evaluation approaches, we conducted a within-subjects study where we compared EVALET to a baseline that only provides holistic scores and justifications for each output. Through this study, we aimed to answer the following research questions:

- **RQ1**. Can *functional fragmentation* aid practitioners in validating LLM-based evaluations?
- **RQ2**. How do practitioners identify and interpret issues in an LLM's outputs through evaluations of fragment-level functions?
- **RQ3**. Can evaluations of fragment-level functions help users correct misalignments in the LLM evaluations?
- **RQ4**. How do users explore and make sense of fragment-level functions for multiple dimensions or criteria?

6.5.1 Study Design

Participants

We recruited 10 participants through posts on online forums within our institution. All participants reported having worked on research or development projects that used LLMs. Two participants reported having more than 2 years of experience working with LLMs, six had 1–2 years, one had 6 months–1 year, and finally one participant had 3–6 months. Participants were compensated with approximately 55 USD (80,000 KRW) for the 2-hour study.

Conditions

Participants analyzed LLM outputs and their evaluations across two tasks in two conditions: `Fragmented` and `Holistic` (Fig. 6.7). The `Fragmented` condition was the full EVALET interface, without the holistic justifications (i.e., summaries of the fragment-level justifications for each output). The `Holistic` condition was a version of EVALET with only the holistic justifications. We chose this design as it closely resembles existing LLM-as-a-Judge approaches [406, 168], but ensures that the evaluations in both conditions contain the same information. To ensure a fairer comparison, the `Holistic` condition also summarizes the holistic justification into a single label for each output (Fig. 6.7A), serving like a function description but for the whole output. These labels were embedded, clustered, and visualized the same way as in the `Fragmented` condition (Fig. 6.7C). For each output, this condition also highlights fragments relevant to each criterion (Fig. 6.7B), similar to prior work [168], and allows users to flexibly select any fragments in the outputs to add as positive, negative, or excluded examples for a criterion.

Tasks

Participants evaluated LLM outputs for the same two generation tasks: (1) writing a short horror story from a given set of keywords, and (2) writing an advertisement post for social media for a given product description.

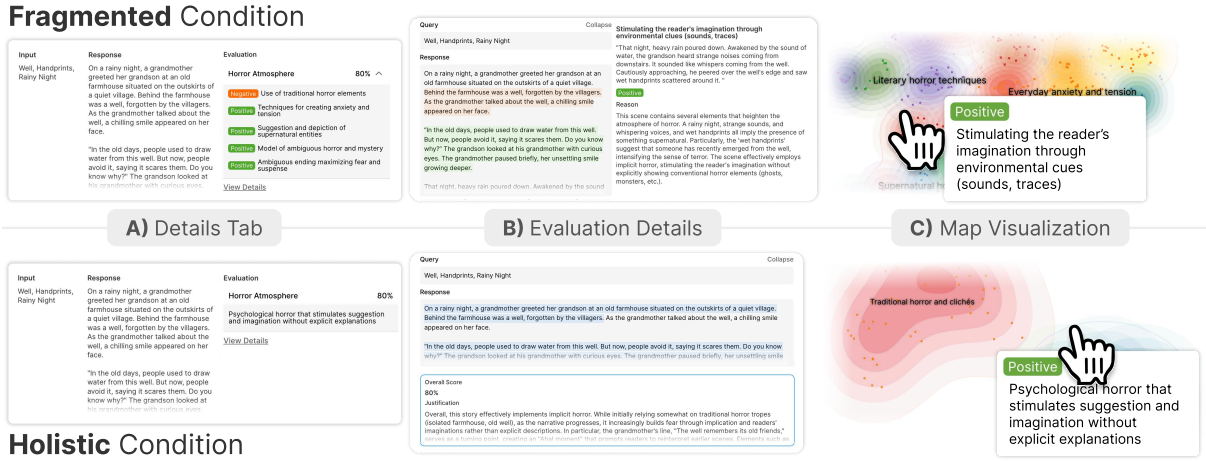

Figure 6.7: Comparisons of the main interface components across the study conditions. (A) The `Fragmented` condition's Details Tab displays the list of fragment-level functions for each output, while the `Holistic` condition shows a label that summarizes the holistic justification for that output. (B) In evaluation details, the `Fragmented` condition shows the function label, rating, and evaluation justification for each fragment, but does not show the holistic justification. The `Holistic` condition highlights the evaluated fragments, but only presents the holistic justification and score. (C) Both conditions feature the Map Visualization. But, in the `Holistic` condition, each point represents a whole output based on the embedding of the holistic evaluation label.

We chose these two tasks as they involve subjectivity, require no prior expertise, have similar input-output lengths, and have been explored by prior work [166, 319, 386]. For each task, we created a dataset of 100 inputs and then generated outputs using `gpt-4o-mini-2024-07-18`, emulating a scenario of evaluating a relatively low-performing model. Then, we pre-evaluated these datasets using our approach to ensure that all participants, irrespective of condition, received the same evaluations for each task. Specifically, we used the following criteria for each task: (1) `Horror Atmosphere` for short horror stories (i.e., creating immersive and constant fear or psychological anxiety), and (2) `Emotional Effect` for the advertisement posts (i.e., effectively eliciting meaningful and genuine emotional responses from viewers). Since our participants were more fluent in Korean, we built the datasets in Korean, and added one line to our evaluation prompt to instruct the LLM to return function labels and justifications in Korean to minimize fluency-related effects.

Procedure

Participants signed the informed consent form prior to the study. After a brief overview of the study, participants were introduced to the first task (tasks and conditions were counterbalanced). Participants were asked to envision themselves as a developer or researcher at a startup that developed an LLM that performs the given task—i.e., the *task LLM*. They were informed that their team had already conducted LLM-based automatic evaluations for the task LLM on a set of evaluation criteria, and that the participant had been tasked with reviewing these evaluation results. Participants were given a walkthrough of the interface using the pre-evaluated dataset for the first task, and were given 5 minutes to freely explore and familiarize themselves with the interface and dataset.

For the first task with the first interface, participants were instructed to perform two sub-tasks:

- **Identify Issues in the Task LLM's Outputs and the LLM-based Evaluations** (RQ1, RQ2) - 15 minutes: Participants were asked to identify common or significant issues (e.g., weaknesses, errors) in the task LLM's outputs. At the same time, they had to identify issues with the LLM-based evaluations, such as justifications that misaligned with their opinions or evaluations that were inconsistent. Participants listed each distinct issue as a separate bullet point in a provided document.

- **Correct the LLM-based Evaluations** (RQ3) - 10 minutes: Participants received two predefined issues with the LLM evaluations: an aspect that was being evaluated inconsistently and an aspect that should not be assessed within the current criterion. Participants were asked to revise the criterion or add relevant examples to address these issues, re-running evaluations as needed to verify corrections.

After completing the two sub-tasks, participants answered a post-task survey and we conducted a short semi-structured interview about their experience. Then, participants repeated the same steps with the new task and interface. At the end of the study, participants returned to the `Fragmented` condition, selected a new criterion from a given list, ran evaluations, and freely explored the new evaluations while thinking aloud for the remaining study time (RQ4).

Measures

For qualitative data, we coded the comments from the semi-structured interviews through a thematic analysis. For quantitative data, we analyzed post-task surveys responses, where participants rated (7-point Likert scale) their self-confidence in (1) identifying output- or evaluation-related issues that were critical (*importance*), (2) covering most issues (*coverage*), and (3) being able to act on and resolve these issues (*actionable*). Participants also rated their perceived workload using five items from the NASA-TLX questionnaire (excluding the "Physical Demand"). Likert scale responses were analyzed using the Wilcoxon signed-rank test.

We also analyze quantitative data from the sub-tasks. In the first sub-task, we counted the number of distinct output and evaluation issues identified by participants—filtering out unrelated comments (e.g., interface usability issues). For the second sub-task, we created separate test sets that exhibited the evaluation issues that were given to participants. We created 20 data points per task, 10 data points per evaluation issue. For each participant and task, we evaluated the test sets on the participant's revised criterion and calculated the percentage of data points where the issues were addressed. Finally, we also analyze participants' interaction logs to measure the number of times that they interacted with individual fragment-level functions or outputs, and through what interface features. For these measures, we conducted Shapiro-Wilk tests to determine if the data was parametric, and then used a paired t-test (if parametric) and a Wilcoxon signed-rank test (if non-parametric).

6.5.2 Results

In this section, we describe findings on how participants verified the LLM-based evaluations (§6.5.2, RQ1), identified issues with the task LLM's outputs (§6.5.2, RQ2), revised criteria to correct the evaluations (§6.5.2, RQ3), and explored fragment-level functions for multiple criteria (§6.5.2, RQ4).

Verifying Evaluations

Participants identified significantly more issues with the LLM-based evaluations in the `Fragmented` condition (`Fragmented` = 3.40 ± 1.58, `Holistic` = 2.30 ± 1.42, $t = -2.40$, $p = 0.04$) (Fig.6.8). Participants mentioned that this was attributed to how it was easier to read and understand the fragment evaluations. While participants had to read entire outputs and overall justifications in the `Holistic` condition, they only had to review individual fragments, their function labels, and the accompanying shorter justifications in the `Fragmented` condition. P1 noted that *"the range of text that I had to read was smaller so it was less time-consuming to interpret each data point."* As a result, P5 mentioned, *"I read each evaluation more carefully and I was able to concentrate on each one more."*

Furthermore, participants also mentioned how, as it was easier to verify each individual evaluation, it was also easier to verify their consistency. P3 explained, *"As [each output's] evaluation is split [into multiple fragment-level*

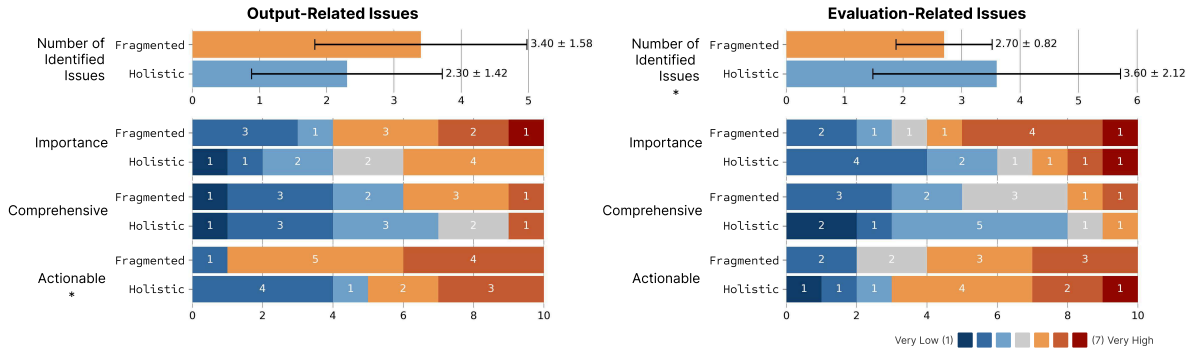

Figure 6.8: Comparison of results across conditions for the issues identified for the task LLM's outputs (left) and LLM evaluations (right). Results present the average number of issues identified, and the distribution of participants' ratings regarding the importance, comprehensiveness, and actionability of the issues (*:p<.05, **:p<.01, error bars indicate one standard deviation).

functions], I could see multiple evaluations [for fragments from different outputs] together and easily compare them, so I tended to focus on that." In contrast, with the `Holistic` condition, P4 mentioned how *"while I could see general trends in the evaluations, I could not directly compare them"* due to the amount of text (i.e., outputs and overall justifications) to compare and reason about. Most participants (7/10) recognized the importance of identifying when the LLM-based evaluator is consistent or inconsistent, and explicitly focused on verifying this. P8 mentioned: *"the problem [of inconsistent evaluations] is more important so no other issues really came into sight."*

As participants' ability to verify the evaluations differed in each condition, their trust and reliance on these evaluations also differed. As participants could more *"confidently"* (P1) verify evaluations in `Fragmented`, participants mentioned how they could judge their trust in the evaluations more carefully. P7 mentioned: *"it's not that I have more trust but instead that it is easier to verify my trust."* Three participants explained how they *"empathized"* with the LLM evaluator—explaining that they may not agree with its evaluation but understood why it returned such an evaluation. As a result, participants mentioned how they were able to develop more *informed trust* about the model by identifying the fragment-level functions for which they agreed with the LLM evaluator and when the evaluator is consistent or not. P5 explained: *"I was wondering whether the evaluator had a bias when evaluating [a certain function] so I looked at these [clusters] more. My conclusion was that it seems like the LLM evaluator considers the aspect as negative most of the time, but there is a slight fluctuation."*

On the other hand, as participants found it more cognitively demanding to verify evaluations in the `Holistic` condition, they also struggled to develop more *informed trust*. Some participants mentioned how they trusted the holistic evaluations despite not carefully inspecting them. For example, P7 mentioned that *"if an output got a score of 100% and the [one-line justification summary] seems to make sense, I just move on"*. Others mentioned not trusting the holistic evaluations at all as they could not confidently verify them: *"I didn't look at the summary or justification at all because I just didn't have trust in them"* (P5). Interestingly, regardless of their trust in the evaluations, most participants (7/10) mentioned relying on the evaluation scores to decide which outputs to explore—often focusing on *extreme* cases (i.e., scores of 100% or 0%)—even without fully understanding why those outputs received those scores.

Identifying Model Issues

Overall, participants identified a similar number of issues with the task LLM's outputs in both conditions (`Fragmented` = 2.70 ± 0.82, `Holistic` = 3.60 ± 2.12, $w = 4.00$, $p = 0.09$). However, participants rated that they were significantly more confident that they could act on and resolve the issues identified with the `Fragmented`

condition (`Fragmented` = 5.10 ± 1.20, `Holistic` = 3.00 ± 1.85, $w = 0.00$, $p = 0.04$) (Fig. 6.8).

According to participants, this result can be attributed to their trust and dependency in the LLM-based evaluations. As participants developed more informed trust regarding the evaluations in the `Fragmented` condition, they used the evaluations as guidance when inspecting the quality of the outputs. Participants followed the fragment-level evaluations to inspect outputs *"piece-by-piece"* (P1) and to inspect *"each output more specifically"* (P2). Furthermore, participants mentioned how the `Fragmented` condition allowed them to explore output issues from a *"wider perspective"* (P5) by exploring similar issues across outputs, or consider more *"diverse characteristics"* (P6) with regards to the criterion.

In contrast, in the `Holistic` condition, as participants could not adequately gauge their trust in the evaluations, they frequently mentioned not relying on the LLM-based evaluations and instead manually inspecting the outputs themselves. For example, P3 mentioned: *"In the [`Holistic` condition], I had to read all of the [output] and also the justification, so I focused only on the [output] and tended to not look at the justification."* By manually reviewing the outputs themselves, participants not only lost efficiency benefits from the LLM-based evaluations but they also tended to focus on more abstract or surface-level issues regarding the model outputs (e.g., overall writing quality, coherency, logic). Besides being less concrete, P8 also explained how these broader issues could be more challenging to resolve: *"The issues I identified seem more related to the limitations of the model itself [...] No matter what feedback I give, it will be difficult to resolve these."* This is also reflected in the interaction logs, where participants in `Holistic` frequently viewed each output in detail (`Fragmented` = 20.92 ± 9.35, `Holistic` = 33.91 ± 13.32, $w = 0.00$, $p < 0.001$), while participants in `Fragmented` interacted more frequently with the Explore Tab and Map Visualization, selecting and navigating between data points (`Fragmented` = 59.25 ± 27.89, `Holistic` = 33.92 ± 20.69, $w = 9.00$, $p = 0.02$).

Although the `Fragmented` condition helped participants find more actionable issues, they also mentioned how it had limitations. Specifically, participants mentioned how they tended to lose sight of the *"bigger picture"* (P2)—including the overall structure, coherency, and context of the outputs. As a result, participants mentioned how they appreciated the `Holistic` condition as it allowed them to compare these holistic aspects of outputs. In fact, after exploring fragment-level functions, participants in the `Fragmented` condition frequently went back to the Database Tab as this was the only tab that allowed them to look at outputs one-by-one and compare them.

Correcting Evaluations

We observed substantial differences in the difficulty of correcting the given evaluation issues across study tasks. Rather than overall comparisons between conditions, we report success rates for each task-condition pair to provide descriptive insights, without statistical tests due to limited sample size (N=5 per pair). In the advertisement task, success rates in correcting the evaluation issues were higher in the `Fragmented` condition (`Fragmented` = $77.0\% \pm 7.9\%$, `Holistic` = $72.7\% \pm 9.8\%$). Conversely, in the horror story task, success rates were higher in the `Holistic` condition (`Fragmented` = $24.9\% \pm 14.4\%$, `Holistic` = $37.7\% \pm 7.3\%$).

Participants found the `Fragmented` condition helpful for skimming through evaluations to identify potential examples for the criteria. However, in `Fragmented`, participants had to add the entire fragment that the system extracted as examples, which sometimes spanned multiple sentences. In contrast, the `Holistic` condition allowed them to manually select shorter fragments to add as examples, which they used to precisely select only the most relevant content. For example, P5 hesitated to add examples in the `Fragmented` condition: *"since the whole [fragment] will be considered in future evaluations, I worry that [the evaluator] will interpret it differently."* Qualitative analysis of results indicated that the lower success rate in `Fragmented` for the horror story task stemmed from this limitation: automatically extracted fragments contained multiple sentences but participants likely added these as examples due to a short phrase within them. This suggests the need for a combined approach:

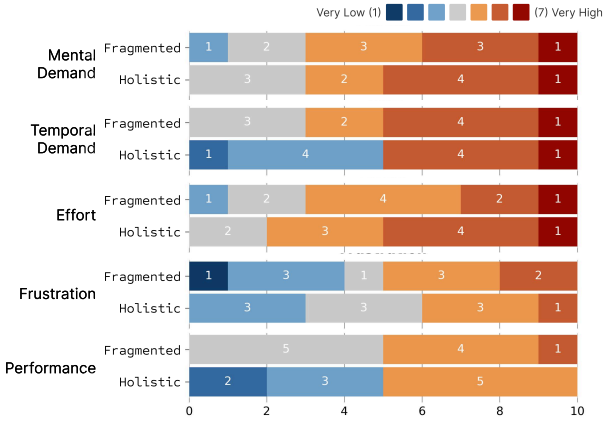

Figure 6.9: Distribution of participants' ratings for perceived workload (i.e., NASA-TLX) show that participants perceived a similar amount of workload in both conditions. In general, participants expressed feeling high workload due to the demands of the study task.

fragment evaluations to identify evaluation issues, with the ability to select specific sub-fragments to precisely express intended corrections.

Exploration with Multiple Criteria

By exploring fragment-level functions for a new criterion (RQ4), participants were able to gain new insights about the outputs and the evaluations. For example, while comparing the evaluations for the same output on different criteria, P7 noticed how *"the same fragment can receive completely opposite evaluations for different criteria"* and that she wanted to *"compare the distribution of evaluations in one cluster with those in a different criterion's cluster."* P1 also mentioned how, as the new criterion focused on the overall content of the outputs rather than specific stylistic choices, she was able to *"understand the outputs better"*. She noted how there can be a *"hierarchy"* between criteria, where more general criteria should be evaluated first to understand the content of outputs, followed by more detailed and specific criteria. P6 explained how they could use these contrasting evaluations to decide on what outputs to use for *"different use cases and applications"*.

Participants noted that, by surfacing fragment-level functions relevant to each criterion instead of simply providing a score, EVALET allowed them to *"more deeply understand and define each criterion"* (P4). P2 mentioned: *"When one doesn't really know what is relevant to a criterion, they could just add an abstract description of the criterion [into the system], and see the LLM evaluations and clusters to learn more and concretize the criterion further."* P5 also reflected this sentiment: *"[In practice], one needs to revise their evaluation criterion by actually evaluating outputs to see [how they match with the criterion], but it seems like this is already doing all of that for us."* Participants' comments alluded to a process like *inductive coding*, where one starts with a broad and abstract criteria and, through the process of reviewing outputs, identifies relevant fine-grained functions that can concretize this criterion.

Perceived Workload

Figure 6.9 shows that overall perceived workload was similar in both conditions (`Fragmented` = 4.58 ± 0.43, `Holistic` = 4.72 ± 0.78, $t = 0.51$, $p = 0.62$). This can be attributed to how each condition led to different distributions of cognitive effort. In `Fragmented`, verifying the evaluations required less effort which freed participants to compare and explore these evaluations. In contrast, `Holistic` led participants to expend most of their effort into manually reviewing outputs one by one. Also, while the fragment-level functions supported

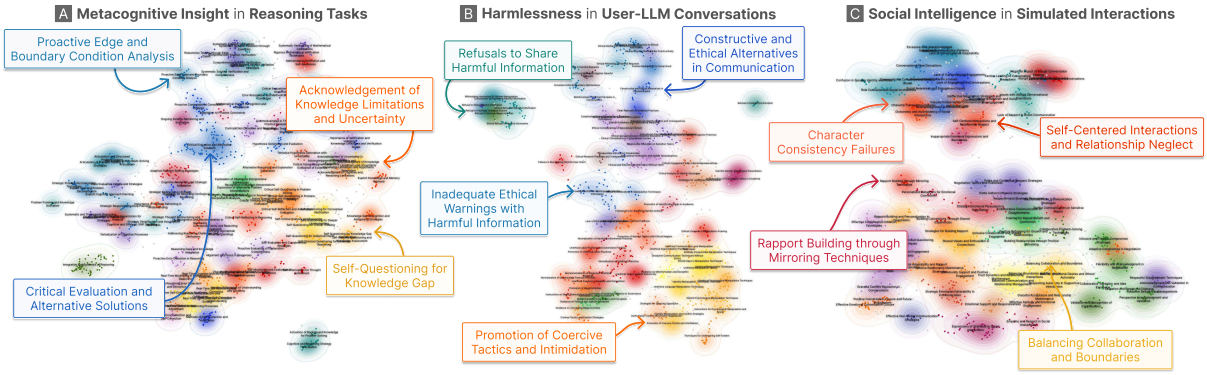

Figure 6.10: Fragment-level functions and their clusters identified through our approach for three types of tasks and criteria: (a) evaluating metacognitive insight in the reasoning traces of LLMs, (b) evaluating harmlessness in red teaming user-LLM conversations, and (c) evaluating social intelligence in simulated interactions between LLM agents.

more fine-grained exploration and analysis, some participants found the number of functions to be overwhelming. For example, P7 mentioned that *"The [`Fragmented`] visualization felt a bit complex and had too many colors, which made it hard to see what information I should focus on. In contrast, the [`Holistic`] visualization was much easier and didn't feel tiring to look at."*

6.6 Example Cases

To demonstrate the generalizability of *functional fragmentation* and the insights that can be gained through it, we present three example cases of the approach with diverse LLMs and tasks. Specifically, we evaluate: (1) *metacognition* in reasoning LLMs, (2) *harmlessness* in user-LLM conversations, and (3) *social intelligence* in agent simulations. In this section, we briefly introduce the data that was evaluated and qualitative observations from the evaluations.

6.6.1 Metacognition in Reasoning

Reasoning-based LLMs, which generate explicit *"reasoning"* traces before providing final answers, have become increasingly prominent due to their advanced performance [141]. However, assessing these lengthy reasoning traces can be challenging. We applied *functional fragmentation* to the OpenThoughts dataset [331], which contains the reasoning traces generated by DeepSeek-R1 [122] on various problems. Specifically, we sampled 210 data points (35 per dataset domain) and evaluated them on `Metacognitive Insight` (i.e., can the model actively reflect upon, regulate, and articulate their thought processes during reasoning).

As seen in Figure 6.10A, *functional fragmentation* surfaces diverse reasoning steps by the LLM that resemble human metacognition. For example, the fragment-level functions reveal self-questioning behaviors, explicit acknowledgment of uncertainties, and proactive consideration for edge cases. However, while evaluations tend to be overly positive, a closer look of functions shows that the evaluator credits metacognitive-like statements, even when they may not be beneficial. For example, in a reasoning trace, the model asks itself a question but will then immediately answer it, which superficially mimics human-like thought patterns but may not indicate authentic introspection or uncertainty. From here, practitioners could refine the evaluation criterion further to assess the actual impact of these metacognitive-like behaviors.

6.6.2 Harmlessness in Conversations with Users

Ensuring that LLMs refrain from providing potentially harmful responses is a critical problem [29]. To understand LLM performance in this regard, we applied our evaluation approach on the HarmfulQA dataset [35], which consists of questions designed to elicit harmful responses from LLMs (i.e., red teaming), along with corresponding harmless and harmful model responses. We randomly sampled 200 data points and evaluated them for `Harmlessness` (i.e., does the assistant engage with users safely and respectfully during interactions).

Figure 6.10B illustrates different fragment-level functions along the spectrum of harmlessness. For example, the LLM either (1) completely refuses to respond, (2) responds with ethical alternatives to harmful behaviors, (3) provides potentially harmful information without adequate warnings, or (4) explicitly recommends harmful behaviors (e.g., coercion and intimidation). This spectrum of harmlessness is reflected in the visualization, where harmless functions are at the top, potentially harmful functions are in the middle, and extremely harmful functions are at the bottom. Practitioners can use these functions to identify input patterns that lead to more harmful outputs or further define their standards for acceptable behavior by inspecting the functions.

6.6.3 Social Intelligence in Simulated Interactions

LLMs are increasingly used as *agents* that simulate human users and social interactions [261, 263, 414]. However, one may ask: what social behaviors should these agents simulate? To investigate this, we applied our approach on a dataset generated through the SOTOPIA environment [344], which simulates negotiations between two LLM agents role-playing characters with different social goals. We randomly sampled 200 data points and evaluated them based on `Social Intelligence` (i.e., does the agent effectively understand, navigate, and manage social interactions with other users or agents).

Figure 6.10C highlights various positive social behaviors within the simulations, such as agents building rapport through mirroring or balancing how much they collaborate with how much they maintain their own boundaries. However, the surfaced functions also reveal potentially anti-social behaviors—for instance, agents may neglect building a relationship with the other agent and focus solely on their own needs and goals. Practitioners can further explore these functions to identify behaviors that should be encouraged or mitigated in simulated agents, or to identify additional evaluation criteria. For example, one of the surfaced functions is *"Character Consistency Failures"*, but these could be assessed by a separate criterion that is specific to role-playing abilities.

6.7 Discussion

In this chapter, we present *functional fragmentation*, a novel approach for evaluating and interpreting LLM outputs based on their constituent fragment-level functions, and EVALET, an interactive system that instantiates this approach. In this section, we suggest guidelines for integrating both fragmented and holistic evaluations in practice, discuss how *functional fragmentation* supports more nuanced analysis with LLM-as-a-Judge, and propose the need for further work in supporting qualitative and interactive evaluation of AI.

6.7.1 Guidelines for Integrating Fragmented and Holistic Evaluations

As revealed by our user study, both fragmented (i.e., at the fragment and function-level) and holistic (i.e., at the output-level) evaluations have distinct merits. In practice, we suggest that these types of evaluation are complementary and should be employed together. Specifically, we recommend a workflow that begins with

exploration of fragmented evaluations, and transitions to holistic evaluations with the option to dive back into fragments when necessary.

1. **Start with Fragmented Evaluation on Broad Criteria:** As described by study participants, *functional fragmentation* can surface and reveal diverse fragment-level functions that should be considered within each criterion. Through this, one can more comprehensively identify concrete aspects that should be evaluated for a criterion, including those that were not initially considered.

2. **Iterate and Concretize Criteria with Function Examples**: By exploring fragment-level functions, practitioners can refine their criteria by deciding on what functions the evaluator should continue to surface and how these should be assessed.

3. **Zoom Out and In**: Then, as participants required in the study, practitioners should *"look at the bigger picture"* by exploring holistic evaluations to understand more overall qualities and their similarities. As the underlying fragment-level evaluations have been corrected and aligned, practitioners can more reliably depend on the signals provided by the holistic evaluations. At this stage, practitioners can dive deep back to the fragmented evaluations when requiring more details—alternating between levels of abstraction [320].

6.7.2 Calibrating Trust in LLM-as-a-Judge through Verification

LLM-as-a-Judge can facilitate inspection, assessment, and analysis of LLM outputs at scales that are infeasible through human effort alone. However, practitioners must carefully *calibrate* their trust by recognizing where they disagree with the LLM evaluator, where it hallucinates, and when it is inconsistent. Our study showed that failing to calibrate trust often led practitioners to dismiss LLM-based evaluations and revert to manual review—losing both efficiency benefits and potentially valuable insights. Despite this, participants still relied on the evaluation scores to prioritize which samples to inspect further, often focusing on extreme scores and thereby overlooking nuanced model behaviors—which can lead to cases where one identifies explicit model biases while missing subtler but equally harmful ones [27].

Our approach, *functional fragmentation*, supported more calibrated and nuanced use of LLM-as-a-Judge by facilitating validation at a granular yet manageable level. By validating when the evaluator returned misaligned or inconsistent evaluations, participants calibrated their trust and selectively relied on the evaluations to guide their in-depth analysis of outputs. Despite the promise of our approach, a potential limitation is that its effectiveness depends on how reliably LLM evaluator identifies all key fragments from outputs—if fragments are surfaced, users can verify their evaluations but, if not, users cannot detect these gaps without manual review. Although our technical evaluation demonstrates strong performance, with the LLM evaluator achieving around 90% recall in identifying fragments, future work could design additional safeguards for missed fragments (e.g., separate map visualization to explore fragments that were unassessed by the evaluator).

6.7.3 Increasing Trend in Increasingly Longer Outputs

Recent trends in LLM advancements have focused on generating increasingly longer outputs. For example, agentic systems like Manus [4], Genspark [5], or Gemini Deep Research [115] can create complex outcomes (e.g., multi-section reports, multi-file codebases) through multi-step workflows. Recent research has focused on further increasing output length by increasing LLM's *test-time compute* [311, 122, 162] (i.e., training models to generate more tokens to reason more about complex tasks), or extending their *context windows* to receive longer inputs and generate longer outputs [6, 131, 76]. Due to their length, each part of an output can exhibit drastically different

levels of quality, making it particularly difficult and challenging for practitioners to make sense of model behavior from holistic judgments. As seen in the example cases in Section 6.6, our approach of *functional fragmentation* has the potential to help users break down and interpret complex outputs in these emerging scenarios.

6.7.4 Qualitative and Interactive Evaluation of AI

AI/ML evaluation has mostly focused on applying *quantitative* metrics in benchmark datasets. This has accelerated advancements by supporting objective and concrete comparisons between models and model iterations. However, as models reach exceptionally high but similar performance on these benchmarks, users have started to *qualitatively* compare models based on their characteristics and behaviors, and how these fit with their own needs [84]—referred to as *"vibe checks"* [154]. This raises a crucial question: *"how can we help users to understand and make sense of qualitative model behaviors at scale."* To tackle this problem, our work proposes *functional fragmentation* to focus evaluation on the individual qualitative fragment-level functions in model outputs and, in turn, support users in sensemaking of model behaviors across outputs. However, we believe that there is still a wide design space that can be further explored by future work. In particular, due to the rich body of work in sensemaking [54, 14, 212, 269] and explainability [202, 183, 155, 342], we propose that the HCI community is ideally positioned to tackle this problem and to integrate itself more closely in the advancement of AI models through the development of novel interactive approaches to evaluation.

6.7.5 Limitations

Our work has several limitations:

- **Limitations of Functional Fragmentation**: Our technical evaluation showed that our approach may fail to account for the relative importance of each fragment-level function or the holistic attributes of outputs. Future work can extend our approach to assign priority ratings to each function (e.g., manually by the practitioner or automatically suggested by the system) and generate additional functions that can represent the holistic qualities of outputs.
- **Dataset Scale**: Our user study used datasets with hundreds of samples. However, practitioners often handle larger datasets and, while our system can support these, users may struggle to navigate and interpret the numerous fragment-level functions. Future work could introduce an additional clustering step to combine similar functions and reduce complexity.
- **User Study - Fragmented vs. Holistic**: Our user study compared exploration of fragmented evaluations against holistic evaluations to clearly isolate their distinct affordances. In practice, these two methods should be used together. While participants offered insights into how to combine them, further studies are needed to understand actual usage patterns.
- **Real-World Practice and Deployment**: Further research into real-world use of *functional fragmentation* and EVALET is required to understand how this evaluation method integrates into practitioners' workflows. To facilitate this, we plan to release EVALET as open-source.

Chapter 7. CUPID: Evaluating Personalized and Contextualized Alignment of LLMs from Interactions

This chapter presents a benchmark that assesses the capabilities of state-of-the-art models in disentangling more complex text artifacts. Specifically, this chapter introduces CUPID, a benchmark that assesses whether LLMs can identify and disentangle users' contextual and personal preferences that were expressed in a series of prior chat sessions between this user and an AI assistant. The benchmark further assesses whether LLMs can effectively apply these disentangled preferences in new chat or interaction sessions. This chapter has adapted, updated, and rewritten content from a paper at COLM 2025 [167]. All uses of "we", "our" and "us" in this chapter refers to coauthors of the aforementioned paper.

7.1 Motivation & Contributions

Large Language Models (LLMs) have shown remarkable capabilities across various tasks [243, 1], benefiting users through diverse applications and conversational assistants [250, 17, 235]. Aligning these models with human values and preferences is crucial as they are increasingly integrated into user experiences [146]. Initial efforts focused on aligning LLMs on broad, general values (e.g., helpfulness, harmlessness, honest) [28, 29] or the aggregated preferences of diverse users [257, 175]. Recognizing that these approaches overlook users' diverse expectations, more recent work focuses on *personalized alignment*, where LLMs are aligned with the users' individual preferences [364, 142, 192]. Despite this progress, existing work largely assumes that users have *static* preferences (i.e., preferences are global and do not vary over time) [364, 192] or focus on stable characteristics (e.g., sociodemographics) to characterize preferences [172].

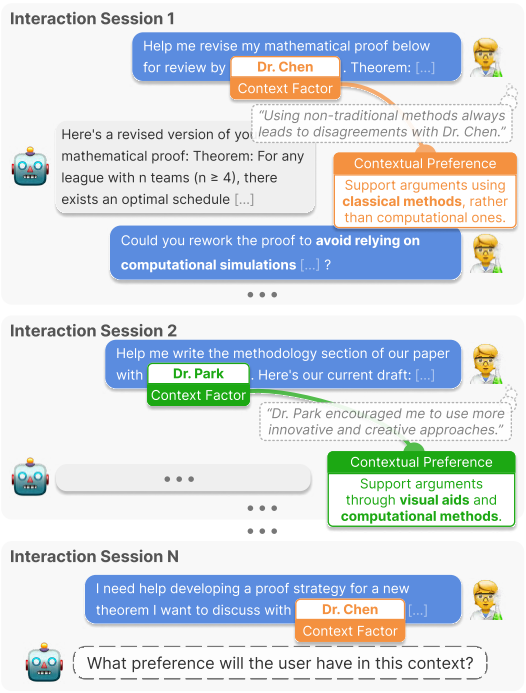

Figure 7.1: Example instance in CUPID illustrates a user that holds distinct preferences in different contexts due to personal experiences, where the preferences are only revealed to the LLM through the user's feedback in prior interactions.

In reality, human preferences are *context-dependent* [351, 31, 227]: an individual's intents and expectations shift depending on their situation. For example, in Figure 7.1, a researcher consults an LLM to refine their paper's writing but their preference varies by collaborator: focus on classical methods with Dr. Chen due to prior disagreements while embracing computational methods with Dr. Park who advocated for them. These contextual variability means that models must understand what values and preferences a user holds in distinct contexts. Interactions between a user and an LLM may indirectly reveal these shifting preferences as users provide feedback to the model in diverse situations [364, 170]. However, it is unclear whether current LLMs possess the capability to identify a user's preference from their feedback, infer the relevant context, and proactively apply this knowledge in future interactions.

In this work, we present CUPID[1] (**C**ontextual **U**ser **P**reference **I**nference **D**ataset), a challenging and scalable benchmark that is designed to assess LLMs' ability to infer users' preferences tied to diverse contexts from user-LLM interaction histories. CUPID consists of 756 human-curated interaction histories, where each history presents a series of *interaction sessions* or dialogues between a simulated user and an LLM. Each interaction session presents a task-oriented dialogue where a user asks a request explicitly stating the context, and then gradually reveals their contextual preference through multi-turn feedback to the LLM. Given a new request and a history of previous interaction sessions, the benchmark evaluates whether LLMs can (1) *infer* the user's preference related to the context of the new request, and (2) *generate* a response to the request that will satisfy this preference.

To construct CUPID, we designed a pipeline that generates diverse and rich interaction sessions. After generating a persona pool, for each persona, the pipeline generates a list of *context factors* (i.e., significant people, objects, locations, etc. in the persona's world that influences their expectations) and *contextual preference* (i.e., value, principle, or criterion associated with each factor). For each persona, the pipeline then generates a series of interaction sessions that involve these factors and preferences. We combine the concepts of LLM-as-a-Judge [406, 384] and LLM-simulated users [363, 364] to create multi-turn dialogues where a simulated user evaluates and provides feedback to LLM's responses, gradually disclosing its contextual preferences. All the data was verified with human annotators and ~9% of the data was manually edited.

With CUPID, we assess 10 open and proprietary LLMs. We find that *all models* struggle to infer users' preferences based on past interaction sessions, with no model exceeding 50% precision and 65% recall. Models failed at recognizing relevant contexts in prior interactions and extracting preferences from multi-turn conversations. We find a strong correlation (r=0.764) between performance in inferring preferences and generating responses that satisfy them, suggesting that the capability to infer context-dependent preferences underpins personalization capabilities. Finally, we present a finetuned metric, PREFMATCHER-7B[2], to reduce the cost of evaluation on our benchmark and a larger unverified dataset[3] to support training and further research.

7.2 CUPID Benchmark

CUPID evaluates LLMs' ability to infer a user's distinct preference in different contexts from prior interactions between the user and the LLM.

7.2.1 Definitions

Our benchmark is composed of interaction sessions $S_i = (c_i, p_i, D_i)$.

Context Factor, c_i Represents an element (e.g., person, location, tool, activity) in a user's environment or world, which plays a role in and influences that interaction session. We assume that each session's context is only defined and influenced by a single factor.

Contextual Preference, p_i Represents a value, principle, criterion, requirement, or constraint that the user holds when the context factor c_i is involved in the situation. While prior work focused on more simple and concrete lifestyle preferences (e.g., *"I cannot eat spicy food"*) [364, 404], we focus on more complex, abstract, and task-oriented preferences (e.g., *"Instructions must break down complex procedures into numbered micro-steps with*

[1] https://huggingface.co/datasets/kixlab/CUPID
[2] https://huggingface.co/kixlab/prefmatcher-7b
[3] https://huggingface.co/datasets/kixlab/CUPID-Unverified

verification points after each major section"). Preferences in our dataset often require multiple rounds of feedback to be satisfied.

Dialogue, $D_i = (u_{i,1}, m_{i,1}, ..., m_{i,L_i-1}, u_{i,L_i})$ Represents an interaction between a user and an LLM. The dialogue begins with the user's initial request $u_{i,1}$ that explicitly states the context factor c_i. In subsequent turns, the user iteratively provides feedback until the model's responses $(m_1, m_2, ..., m_{i,L_i-1})$ satisfy preference p_i. The dialogue length L_i is determined when p_i is fully satisfied, confirmed by the user's final utterance u_{i,L_i}.

7.2.2 Problem Formulation

Each task instance in CUPID consists of a 4-tuple $(u_{\text{current}}, c_{\text{current}}, p_{\text{current}}, \mathbf{S})$, where u_{current} represents the user's current request to the LLM, c_{current} and p_{current} represent the context factor and contextual preference in the new request, and $\mathbf{S}$ is the history of previous interaction sessions, where at least one $S_i \in \mathbf{S}$ satisfies $c_i = c_{\text{current}}$ and $p_i = p_{\text{current}}$. Our benchmark evaluates LLMs on two tasks:

- **Inference**: Given u_{current} and $\mathbf{D} = \{D_i \mid S_i \in \mathbf{S}\}$, the model should infer the user's contextual preference p_{current}.
- **Generation**: Given u_{current} and $\mathbf{D} = \{D_i \mid S_i \in \mathbf{S}\}$, the model should generate a response r that will satisfy or align with the user's preference p_{current}.

7.2.3 Metrics

Inference - Preference Match Preferences in our benchmark are complex, with each preference entailing multiple fine-grained expectations (i.e., sub-preferences). An inferred preference should capture all sub-preferences (*recall*) without including irrelevant ones (*precision*). To assess this, we design an LLM-based *preference matching* metric, inspired by work on measuring factual precision through atomic facts [238] and fine-grained checklist evaluations [205, 65]. Specifically, we first use an LLM to *decompose* the ground-truth and inferred preferences, p and $\hat{p}$, into *atomic checklists* $Q_p = \{q_1, ..., q_n\}$ and $Q_{\hat{p}} = \{\hat{q}_1, ..., \hat{q}_n\}$, where each checklist item q_i or $\hat{q}_i$ assesses a single sub-preference. Then, we employ an LLM with a few-shot prompt to evaluate whether each checklist item *matches* the other preference:

$$\textsc{Match}(\hat{q}, p) = \begin{cases} 1, & \text{if aligning with } p \text{ fully covers } \hat{q}, \\ 0.5, & \text{if aligning with } p \text{ partially covers } \hat{q}, \\ 0, & \text{otherwise.} \end{cases}$$

We use GPT-4o for decomposing and matching. For matching, we conducted a meta-evaluation with 230 human-annotated data points, separate from our dataset. GPT-4o achieved a Krippendorff's alpha [176] of 0.769 (*substantial agreement*) with the majority vote of human annotators. To reduce the cost of evaluating on our benchmark, we also present PREFMATCHER-7B, a finetuned model for preference matching that achieves a Krippendorff's alpha of 0.748 (*substantial agreement*). Finally, we compute: $\text{Precision} = \frac{1}{|Q_{\hat{p}}|} \sum_{\hat{q} \in Q_{\hat{p}}} \textsc{Match}(\hat{q}, p)$, $\text{Recall} = \frac{1}{|Q_p|} \sum_{q \in Q_p} \textsc{Match}(q, \hat{p})$, and the F1 score.

Generation - Preference Alignment For the generation task, we evaluate a model's response r on its alignment with the contextual preference p_{current} using the *LLM-as-a-Judge* approach [406]. As prior work has shown that LLMs can evaluate other models' responses on diverse skills, principles, or criteria [99, 168, 384], we prompt GPT-4o to provide a score ranging from 1 to 10 on the degree to which a model response satisfies the ground-truth

contextual preference. As checklists can increase the consistency and reliability of LLM evaluations [205, 65], we also provide the automatically decomposed checklist Q_p used in preference matching to the LLM judge.

7.3 Data Generation Pipeline (Figure 7.2)

To capture realistic interaction patterns and context-dependent user preferences [351], we designed synthetic user interaction sessions and preferences in CUPID with the following **desiderata**: **(1) User-specific and unique**: Focus on highly personalized contexts and preferences not inferrable from prior knowledge or commonsense; **(2) Indirect and gradually revealed preferences**: Preferences expressed through multiple turns of feedback and clarification to mirror real user behavior; **(3) Patterns of shifting preferences**: Capture how a user's preferences may shift over time or even conflict across contexts. Unlike prior work focused on lifestyle preferences, we center our benchmark on task-oriented preferences and dialogues, reflecting real-world usage of LLMs [329]. The pipeline uses Claude 3.5 Sonnet, unless noted otherwise.

Although synthetic data may not fully represent real user behavior and diversity, we opt for synthetic data over human-collected data because it provides: (1) precise control over dataset difficulty, including how preferences are revealed and contexts are repeated; (2) higher quality by ensuring that preferences are revealed and models can infer them; and (3) enhanced diversity by varying context factor, preference, and task types.

7.3.1 Generation Process

Persona Pool We aimed to construct context factors and preferences that were specific and unique to a user **(Desiderata 1)**, rather than general factors (e.g., *"Fender Stratocaster"* vs. *"electric guitar"*) and preferences (e.g., *"instructions must include precise hand positioning details"* vs. *"instructions must be precise"*). Existing persona datasets [104, 414] lacked sufficient detail to generate unique factors and preferences. Thus, we created a new persona pool by sampling seed persona descriptions from Ge et al. [104], combining them with attributes (e.g., personality, personal values) inspired by Zhou et al. [414], and using an LLM to expand each into rich persona descriptions—yielding 252 distinct personas.

Constructing Contexts To generate diverse yet internally coherent contexts for each persona, we instruct an LLM to first generate a list of 8 context factors and associated preferences—essentially *constructing* each persona's *world*. Before each factor-preference pair, we prompt the LLM to first generate a background narrative to develop more specific and unique contexts **(Desiderata 1)**. To increase diversity, we provide the LLM with predefined taxonomies for factor types (e.g., person, object, location, activity) and preference types (e.g., creativity, sensitivity, trustworthiness). To reflect how users may hold contrasting preference in different contexts **(Desiderata 3)**, the LLM also creates factor pairs c_{current} and c_{contrast} whose preferences conflict or contradict, $p_{\text{current}} = \neg p_{\text{contrast}}$.

Generating Sessions With each persona's list of factor-preference pairs, we prompt an LLM to generate a series of chronologically connected interaction sessions, **S**. Instead of generating each session and its interactions one-by-one, we generate all the sessions first to ensure that they are logically and chronologically coherent. When generating each session S_i, the LLM is instructed to first narrate a task-related situation that involves a context factor, and then generate the initial request $u_{i,1}$ where the user explicitly mentions the factor and seeks an LLM's help with the task. To ensure that the series of sessions reveal specific contexts and preferences **(Desiderata 2)**, we prompt the LLM with a 13-session template: the final session S_{13} includes c_{current}; two sessions S_u and S_v also with c_{current}; two sessions with c_{contrast}; and two sessions S_y and S_z $(y, z < u, v)$ with c_{current} but a prior preference $p_{\text{prior}} = \neg p_{\text{current}}$ that reflects a significant change in the user's preference **(Desiderata 3)**.

Dialogue Simulation For each interaction session, we simulate multi-turn dialogues by using two distinct LLMs: an *user simulator* role-playing as the persona and an *assistant simulator* that responds to the user. Each session begins with the user's request $u_{i,1}$, which the assistant answers. Adapting the LLM-as-a-Judge approach [406], the user simulator evaluates the assistant's response against the contextual preference p_i, scoring it from 1 to 10, and then generates feedback that indirectly hints at the preference (e.g., noting the issues with the response, rather than explicitly stating the preference). The *assistant simulator* responds and the dialogue continues, gradually revealing the preference, until the *user simulator* provides a score of 10 **(Desiderata 2)**. Each dialogue is designed to contain sufficient evidence for the underlying preference to be inferrable.

Creating Instances Finally, we construct instances to account for the different patterns in user preferences **(Desiderata 3)**: a user holding conflicting preferences in different contexts, and their preferences changing over time. Specifically, for each series of synthesized sessions, we create three types of data instances:

- **Consistent**: Basic instance where there is at least one $S_k \in \mathbf{S}$ such that $c_k = c_{\text{current}}$, $p_k = p_{\text{current}}$ and $u_{k,1} \neq u_{\text{current}}$ (i.e., at least one previous interaction session possesses the same context factor and preference as the current request).
- **Contrastive**: Meets the same condition as *Consistent*, and there is at least one $S_l \in \mathbf{S}$ such that $c_l \neq c_{\text{current}}$ and $p_l = p_{\text{contrast}} = \neg p_{\text{current}}$. Here, a prior session includes a context factor with a preference that conflicts with the current context's.
- **Changing**: Meets the same condition as *Consistent*, and there is at least one $S_m \in \mathbf{S}$ such that $m < k$, $c_m = c_{\text{current}}$, but $p_m = p_{\text{prior}} = \neg p_{\text{current}}$. Here, the user *previously* held an opposite preference in relation to the same context factor.

Each instance has 9 sessions: 8 prior interaction sessions, and the last session with u_{current}.

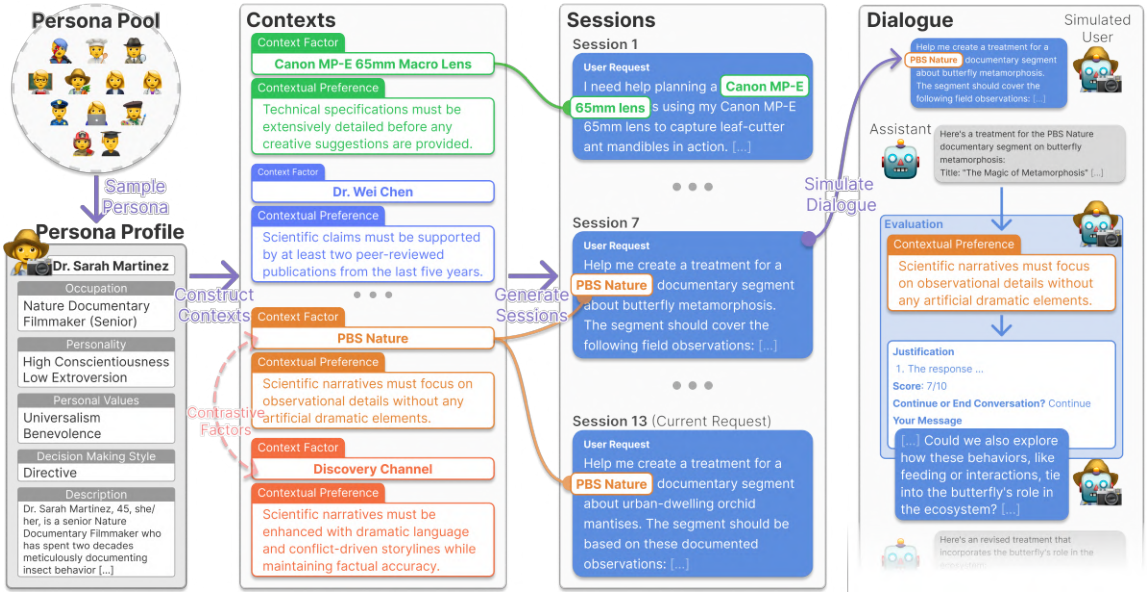

Figure 7.2: Data generation pipeline for CUPID. For each persona, we construct diverse context factors and preferences, then generate chronologically linked interaction sessions. For each session, we simulate a dialogue where the user persona evaluates an AI assistant's responses and provides feedback based on the contextual preference.

7.3.2 Data Validation

We conducted human validation to ensure that our instances were **solvable** (i.e., preferences are inferrable from prior interactions), **challenging** (i.e., preferences not inferrable from current requests), and **realistic**. For each instance, we validated that relevant preferences ($p_{\text{current}}, p_{\text{contrast}}, p_{\text{prior}}$) were fully expressed in the simulated dialogues, but p_{current} was not expressed in $u_{current}$. To reduce annotators' cognitive load, we employed a human-AI approach [193, 196] where an LLM first extracts user messages from the dialogues that potentially expressed each preference. Then, two human annotators recruited via Prolific then annotated these messages and we considered that a preference was expressed if marked by at least one annotator. Then, the authors manually revised these cases (~9% of all instances). Annotators also rated the realism of the current requests at an average of 4.08 (SD=0.85) out of 5.

7.3.3 Data Statistics

CUPID is composed of 756 instances, with 252 instances for each type: *Consistent*, *Contrastive*, and *Changing*. On average, contextual preferences are 18.4 tokens long and are decomposed into 2.90 checklist items, dialogues have 6.38 turns and are 921.5 tokens long, and prior interaction sessions are 8186.7 tokens long.

7.4 Experiments

With CUPID, we evaluate open and proprietary LLMs on **inference** and **generation**.

7.4.1 Experimental Setup

Baseline Models We tested a total of 10 state-of-the-art instruction-tuned LLMs and a few reasoning models: GPT-4o [137], o3-mini [255], Claude 3.7 Sonnet [19], Claude 3.5 Sonnet [18], Llama 3.1 405B [119], Mistral 7B [147], Qwen2.5 72B [379], DeepSeek-R1 [122], Gemini 2.0 Flash Thinking, and Gemini 2.0 Pro [116].

Prompts For the inference task, all models were zero-shot prompted to analyze prior interaction sessions and infer the preference that the user will likely hold in the current request, but is not mentioned in the request. For the generation task, all models were zero-shot prompted to generate a response for the current request by considering the possible preference that the user has based on prior interaction sessions.

Oracle We test an oracle setting where models receive only prior sessions that share the current request's contextual preference, isolating the impact of identifying and retrieving relevant prior sessions. For the Generation task, we test an oracle *preference* setting where models are given the ground-truth preference when generating responses, allowing us to evaluate how their ability to adhere to preferences affects performance.

Interaction Summary Recent LLM-powered AI assistants are equipped with *long-term memory* allowing them to record, recall, and reason about prior user interactions [412, 254]. To understand how this type of intermediate representation could enable models to infer a user's preferences in different contexts, we design a setting where models are first prompted to summarize each dialogue in the prior interaction sessions and, then, perform the tasks with these summaries.

Metrics (Section 7.2.3) We report precision, recall, and F1-score for the **inference** task, and alignment score of model responses [1-10] for the **generation** task.

Model	All Instances			Consistent			Contrastive			Changing			Oracle Setting		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Mistral 7B	26.9	30.7	28.6	22.3	25.6	23.8	23.0	26.7	24.7	35.4	39.7	37.4	46.8	59.8	52.5
Qwen2.5 72B	32.8	44.9	37.9	30.0	42.6	35.2	27.5	36.2	31.3	41.0	55.9	47.3	60.4	77.1	67.7
Llama 3.1 405B	30.2	45.1	36.2	26.4	39.1	31.5	27.8	41.3	33.2	36.4	55.0	43.8	58.4	77.4	66.5
DeepSeek-R1	42.0	59.5	49.3	44.2	62.9	51.9	41.3	58.8	48.5	40.6	56.8	47.4	62.8	81.2	70.8
GPT-4o	36.6	52.1	43.0	36.5	50.1	42.2	32.6	46.4	38.3	40.7	59.7	48.4	57.1	77.7	65.8
o3-mini	33.6	47.9	39.5	32.4	47.9	38.6	31.0	44.2	36.5	37.3	51.6	43.3	48.9	68.7	57.1
Claude 3.5 Sonnet	40.9	56.9	47.6	40.9	56.2	47.4	38.2	52.5	44.3	43.6	62.0	51.2	62.8	81.6	71.0
Claude 3.7 Sonnet	**49.1**	**64.6**	**55.8**	**52.5**	**66.3**	**58.6**	**48.3**	**61.5**	**54.1**	**46.5**	65.9	**54.5**	**69.6**	**84.5**	**76.3**
Gemini 2.0 Flash Thinking	37.8	53.7	44.4	37.3	53.0	43.8	35.5	49.4	41.3	40.6	58.6	48.0	59.5	76.0	66.8
Gemini 2.0 Pro	40.1	63.5	49.2	39.0	63.3	48.3	38.3	58.7	46.4	43.0	**68.5**	52.9	62.6	83.1	71.4

Table 7.1: Precision (P), Recall (R), and F1 score for all models on the inference task in CUPID, averaged across all instances, each instance type, and the oracle setting. Best results in each column are **bold**-faced and second best results are underlined.

7.4.2 Results

In this section, we report performance on **inference** (Section 7.4.2), **generation** (Section 7.4.2), and with **interaction summaries** as an intermediate representation (Section 7.4.2).

Inference Task

Models struggle to infer contextual preferences from interactions. Table 7.1 shows the performance of all tested models on the inference task. All models struggled to adequately infer the preference relevant to the current user request from the previous interaction sessions—no model surpassed an F1 score of 60% and with most under 50%. Considering how our benchmark was designed to be challenging but not overly difficult (e.g., each preference is revealed in multiple sessions, only 8 prior sessions per instance), we expect that performance will degrade significantly in more realistic settings.

Model size and reasoning capabilities increase performance. Larger model size leads to better performance with the smallest model, Mistral 7B, showing the lowest performance by a notable margin of 7.8 points. Additionally, our results show reasoning models perform the best in this task with DeepSeek-R1 and Claude 3.7 Sonnet reaching the highest performance. This suggests that greater train-time and test-time compute enables models to better reason about user preferences and their relevant contexts from prior interactions.

Retrieving relevant context can significantly increase performance. In the oracle setting, all models improved by approximately 20-30 points, highlighting how inference performance is strongly dependent on the capability to retrieve and focus on prior interactions with the same context. However, even with only the relevant sessions, precision across models was still under 70%, implying that models are still inferring less relevant details.

Surprisingly, *Changing* instances led to highest performance. Most models perform worse in *Contrastive* instances but excel in *Changing*, contrary to our expectation that models would struggle with evolving preferences. Deeper analysis suggests that models prioritize the *most recent* sessions. Figure 7.3 shows that performance improves when relevant sessions (i.e., contain the current preference) appear at the end of the history. In *Changing* instances, earlier sessions reflect the preference before a change and later ones capture the change—meaning that relevant sessions tend to be positioned towards the end.

Error Analysis To understand the errors that models make when inferring preferences, we sampled and qualitatively inspected 50 responses each from DeepSeek-R1 and Llama 3.1 405B with F1 scores below 20% (~bottom

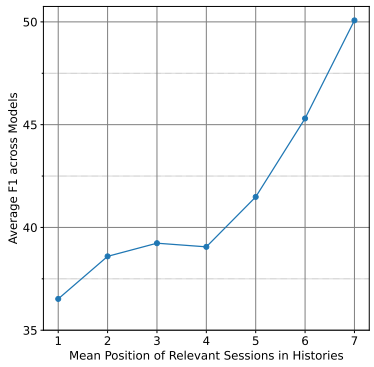

Figure 7.3: Mean F1 score across all models against mean position of relevant sessions in histories.

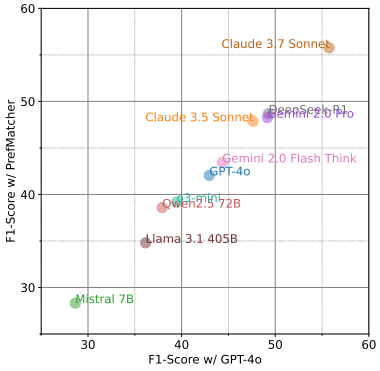

Figure 7.4: Correlation between F1-score when computed with GPT-4o and with PREFMATCHER-7B.

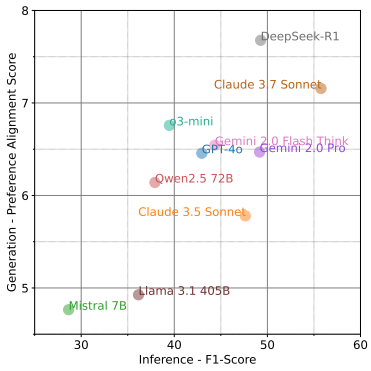

Figure 7.5: Inference performance against generation performance, averaged for each model.

25%). Table 7.2 summarizes the identified error types. We observe that models failed to focus on the relevant contexts, struggled to extract the specific preference from multi-turn interactions, only performed shallow inferences, or hallucinated preferences. The models exhibited distinct error patterns: Llama 3.1 405B produced more shallow inference errors, focusing only on the current request, while DeepSeek-R1 showed less shallow errors but more incorrect context errors, reasoning over prior interactions but failing to focus on those with shared context.

Evaluation with PREFMATCHER-7B shows almost perfect correlation with GPT-4o assessments (Figure 7.4). The Pearson correlation for model-wise average performance with GPT-4o and PREFMATCHER-7B was 0.997 ($p<0.001$). Given this result and the strong correlation between inference and generation (discussed later), we suggest evaluating on CUPID only on the inference task and with PREFMATCHER-7B to reduce cost.

Generation Task

Performance in the generation task follows similar trends to the inference task. Table 7.3 shows the models' performance in the generation task. Similar to the inference task, we observe that model performance is generally low, increases with model size and reasoning capabilities, increases significantly in the oracle setting, and is highest in Changing instances while being generally lower in Contrastive.

Strong correlation between the inference and generation tasks. Figure 7.5 shows a positive correlation between model performance in the inference and generation tasks. The Pearson correlation for model-wise average

Error Type	Description	DeepSeek-R1	Llama 3.1 405B
Incorrect Context	The model incorrectly infers preferences from other contexts that are not relevant to the current request.	86%	40%
Shallow Inference	The model infers preferences explicitly mentioned in the request or that are commonsense for the request, instead of inferring from prior interactions.	10%	50%
Vagueness	The model inferred preferences that were relevant but were too broad or vague, lacking the specific details in the target preference.	2%	6%
Hallucination	The model inferred preferences without clear evidence or context, likely influenced by internal assumptions or biases.	2%	4%

Table 7.2: Error types identified in preferences inferred by DeepSeek-R1 and Llama 3.1 405B, with proportion of errors that each model made for each type.

Model	All Instances	Consistent	Contrastive	Changing	Oracle	Preference
Mistral 7B	4.76	4.03	4.20	6.06	6.12	7.36
Qwen2.5 72B	6.14	5.60	5.38	7.45	7.33	8.69
Llama 3.1 405B	4.93	4.43	4.29	6.06	6.96	8.84
DeepSeek-R1	**7.68**	**7.50**	**7.33**	**8.21**	**8.95**	9.66
GPT-4o	6.46	5.88	5.75	7.74	7.79	9.20
o3-mini	6.76	6.31	6.04	7.93	8.10	**9.72**
Claude 3.5 Sonnet	5.78	5.25	5.08	7.01	8.05	9.41
Claude 3.7 Sonnet	7.16	7.13	6.50	7.84	8.61	9.63
Gemini 2.0 Flash Thinking	6.54	6.03	5.91	7.69	7.58	9.43
Gemini 2.0 Pro	6.47	6.29	5.81	7.31	7.81	9.45

Table 7.3: Preference alignment scores for all models on the generation task in CUPID, measured for all instance types, each instance type, oracle setting, and oracle preference setting.

performance in the tasks was 0.764 (p=0.010), suggesting that the inference task can serve as a proxy to evaluate models' ability to generate contextually personalized responses. Interestingly, certain models excelled at one task over the other: Claude 3.7 Sonnet led in inference, but was outperformed by DeepSeek-R1 in generation.

Generation performance is not tied to response generation capabilities. Almost all of the models show significantly high performance in the oracle preference setting (i.e., generating a response given the ground-truth preference). This indicates that the low performance across the models is not due to the models' inability to generate responses that satisfy the preference, but rather due to their inability to precisely infer the relevant preference.

With Summaries

Summaries have an *equalizing* effect across models. Figure 7.6 compares model performance with and without summaries. Summaries offer minimal gains for strong models—slightly lowering performance for reasoning models—yet substantially boosts weaker models, bringing them closer to the performance of strong models. This suggests that summaries help weaker models to reason about and extract preferences from each session, but may lead to information loss for strong models. Notably, the smallest model, Mistral 7B, shows a substantial increase of 13 points in inference, suggesting potential for local and privacy-preserving LLM personalization.

Even with summaries, models show low precision. We observe that the increase in inference performance with the summaries is mostly attributed to recall. This indicates that the summaries are not necessarily helping the models focus on the relevant interaction sessions, but rather it is helping models extract the preferences from all the sessions.

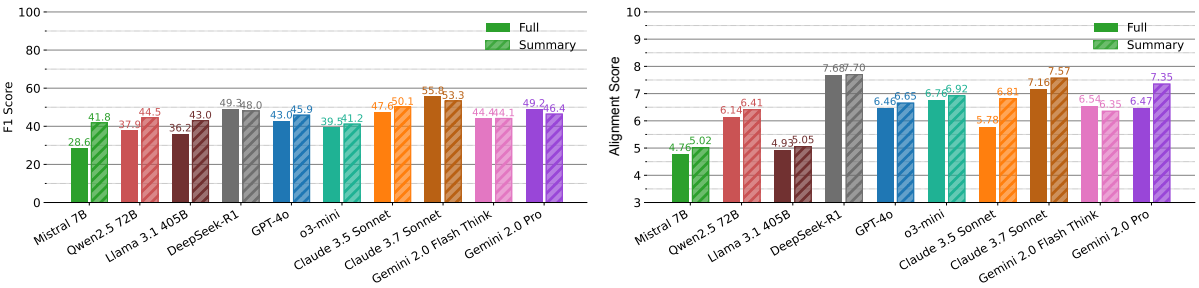

Figure 7.6: Comparison of each model's results for the inference task (left) and generation task (right) with the full prior interaction sessions or summaries of these sessions.

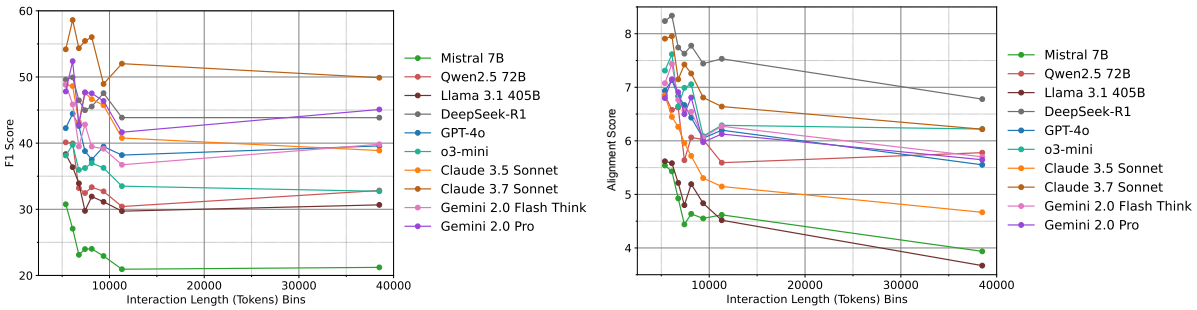

Figure 7.7: Average result for each model in the inference task (left) and generation task (right) according to the maximum token length of the instances.

Performance against Length of Prior Interaction Sessions

Figure 7.7 shows the average results of each model in the inference and generation task depending on the token length of the previous interaction sessions in the instances. Specifically, we divided all the instance data into equally-sized groups (bins). Each bin contains instances with lengths larger than the maximum length in the previous bin, but less than the minimum length in the next bin. Then, we averaged model performance for all data in each bin. This allows us to observe that each bin represents model performance at a distinct range of lengths. As seen from the results, we observe a general trend where model performance in both tasks decreases with the length of the prior interaction sessions—which coincides with findings in similar work [363, 404]. This suggests that, with even longer interaction sessions, models will struggle to infer users' personalized and contextual preferences from interactions.

Analyzing Possible Self-Enhancement Bias

As CUPID context factor-preferences were generated by Claude 3.5 Sonnet, Claude 3.7 Sonnet's superior inference performance raises the possibility that implicit biases trained into these models were reflected in the data. To test this, we synthesized small datasets using three high-performing models—Claude 3.7 Sonnet, GPT-4o, and DeepSeek-R1—based on the 64 persona profiles in our benchmark that produced the lowest average performance across all models . For each persona and model, we used our synthesis pipeline to generate Consistent, Contrastive, and Changing instances—leading to small datasets of 192 instances for each model. Then, we evaluated each model on its own and the other models' datasets—for Claude 3.7 Sonnet, we instead used the original CUPID instances for the 64 personas.

Figure 7.8 shows minimal evidence of higher performance on self-generated data, indicating that Claude 3.7 Sonnet's strong results in our benchmark reflect genuine capability rather than dataset bias. Performance patterns in all the datasets were consistent with those in our benchmark: in inference, Claude 3.7 Sonnet performed best, then DeepSeek-R1, and then GPT-4o; in generation, DeepSeek-R1 led, with Claude 3.7 Sonnet second, and GPT-4o last.

Practical Implications

Our findings suggest three actionable directions for personalized LLMs. First, integrate retrieval techniques that identify prior sessions from users' interaction histories that are *contextually* relevant to the current request, as oracle results show 20-30 point improvements. Second, when deploying smaller or local LLMs, cache summaries of each interaction session focusing on context and preferences. Finally, prompt or tune models to perform reasoning

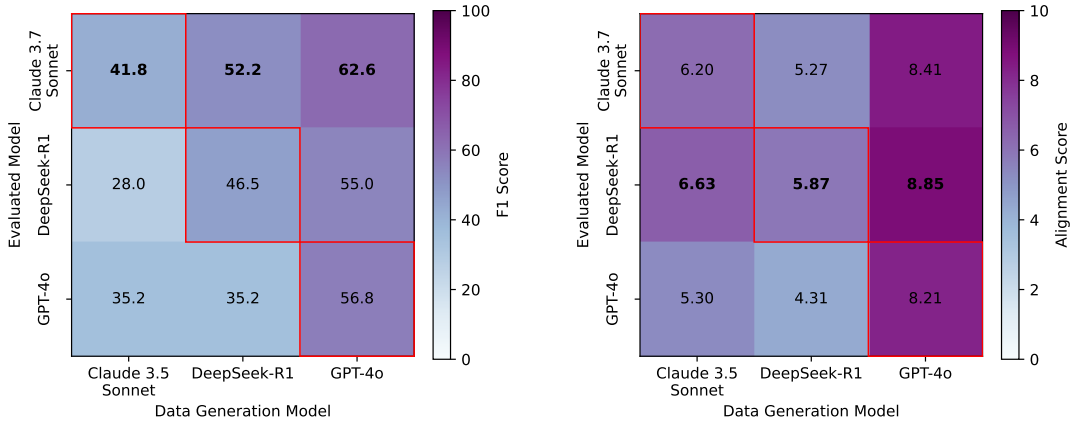

Figure 7.8: Inference and generation performance of tested models on datasets synthesized by other tested models. Red-highlighted squares are evaluations on data generated by the same model (or a model of the same family).

about users' underlying preferences during multi-turn interactions, rather than simply inferring from surface-level expressions.

7.5 Ethical Considerations

In this paper, we introduce CUPID, a benchmark designed to evaluate LLMs' ability to infer user's contextual preferences from interaction sessions. Since the data in our benchmark is newly created, we conducted a rigorous human validation process. Specifically, during the human validation that assessed that whether the data instances were solvable, human annotators were also asked to flag any possible data points that appeared harmful or offensive. Our human data validation protocol has been determined exempt by the IRB of the author's institution. Additionally, the authors manually inspected all of the data points (i.e., context factors and contextual preferences) to identify and filter out any that appeared offensive, harmful, or unethical. We conducted all experiments using either publicly available models or through documented commercial API access. We have detailed the API versions and configuration of all of the evaluated LLMs for reproducibility. We acknowledge that LLMs may inherit biases from their training data, potentially leading to our dataset incorporating these same biases and not being fully reflective of real users. To mitigate this issue, we aimed to increase the diversity of the user personas, context factors, contextual preferences, and conversation topics. Namely, we created taxonomies for each of these components to guide the generations to reflect diverse types for each of these components. Finally, we acknowledge personalized conversational AI systems can raise concerns related to user's privacy as user information is stored, recalled, and analyzed. It is crucial that such AI systems implement mechanisms to anonymize data, provide mechanisms for users to control what data is stored, or support local, on-device storing and processing to ensure that user information is never transmitted externally. To promote reproducibility and advance research in this field, we have made our benchmark dataset, code, and model publicly available.

Chapter 8. DESIGNLLM: Proactive Exploration of Latent User Intents

This chapter presents preliminary findings on a work that explores how to train LLMs to interact with users by integrating to the principles of text disentanglement. Specifically, this chapter describes DESIGNLLM, a training framework that teaches models to act as collaborative, creative assistants by disentangling diverse interpretations of user intent from messages and responding with multiple artifact prototypes that help scaffold the conversation. This chapter presents the overall approach and preliminary experimental results on the given approach. All uses of "we", "our" and "us" in this chapter refers to coauthors in this work.

8.1 Motivation & Contributions

Large Language Models (LLMs) have emerged as effective conversational assistants, thanks to their natural language understanding and instruction-following capabilities. Given a clear and precise input, these models can generate high fidelity outputs that satisfy multiple and complex user requirements. However, this effectiveness relies on a critical assumption that users start conversations with fully formed intents. In reality, users often do not know precisely what they want from the onset and instead develop their intents gradually through the conversation [318]. This is especially true in open-ended creative tasks, where people only develop and realize their needs, constraints, and preferences through interactive exploration and reflection on task outcomes [295, 81, 79].

However, current LLM training and evaluation focuses on optimizing single-turn responses. Benchmarks mostly assess single-turn performance [180], and fine-tuning methods such as Reinforcement Learning from Human Feedback (RLHF) [257] predominantly reward models for producing direct, fully elaborated outputs to a user's single request. Consequently, when users have underdeveloped intents, their initial inputs will also be vague or under-specified. As models are optimized for single-turn responses, they tend to infer or ignore missing details, and return fully fleshed-out responses. These responses often misaligned with what the user ultimately wants, forcing users to parse the frequently extensive content and provide feedback and corrections through follow-up turns as the user develops their intents—a process that is inefficient and frustrating [390, 342, 170, 166].

Recent work has begun to explore methods to improve multi-turn interaction, either through prompting strategies [197] or training models on multi-turn trajectories [398, 298, 365]. These approaches assume that users already possess well-defined intents that they have simply not articulated—implying that the assistant can elicit them through clarification. When users have not yet realized their intents, however, clarification questions fail—there is nothing that the user can clarify. In these scenarios, **design practice** suggests that people frequently make sense of the *problem space* (i.e., their needs) by exploring the *solution space* (i.e., possible outcomes) [68, 79, 295]. By examining and assessing options, even incomplete ones, users gradually discover characteristics that they wanted or intended for. This points to a fundamentally different role for LLMs when interacting with users: instead of trying to extract information users do not yet possess, models should help learn what they actually want through exploration and reflection.

Inspired by this perspective, we propose DESIGNLLM, a training framework that teaches language models to collaborate with users by helping them *form* their intents rather than simply executing them. Our key innovation is a novel design for user simulation. Unlike prior work that designs simulated users to start with fully formed intents, our simulators represent each intent across multiple levels of specificity—from broad and abstract to specific and concrete—where users start the conversation with each intent only formed at its most general level. Through the conversation, users gradually form more specific levels of each intent when the assistant successfully satisfies or

probes at these intents. For instance, a user who begins with the vague request *"write me a poem with an animal"* may initially only have formed the intent *"include an animal."* Even if the assistant asks *"what kind of animal?"*, the user cannot answer because their intent has not yet formed to a more specific level. However, if the assistant produces a poem featuring a cat, the user may realize this captures what they wanted and form this more specific intent. Our simulator operationalizes this as a hierarchical intent-formation process (e.g., *animal* $\rightarrow$ *pet animal* $\rightarrow$ *cat* $\rightarrow$ *tabby cat*). We then reward model responses proportionally to the number of intent levels that they help users form. Using this signal, our framework fine-tunes models via reinforcement learning. Models trained with DESIGNLLM shift from merely responding to requests or asking questions towards collaborative exploration, offering alternatives and possibilities that help users discover what they actually want.

We evaluate DESIGNLLM on creative writing. In simulated experiments, our models improve the simulated user's final satisfaction by 11.7% and intent awareness by 10.6% when compared to the base model, demonstrating the value of training models that help users *discover* their goals, not just articulate or execute them.

8.2 Problem Formulation

In contrast to existing tasks for multi-turn user-LLM conversations where the user enters with fully-formed intents, we consider a setting where the user begins with vague and underdeveloped intents that are gradually formed and refined through the conversation. In this setting, the conversation unfolds over multiple turns, denoted as $t_1, t_2, \ldots, t_K$, where each turn consists of a user input u_j and the model's response m_j:

$$(u_1, m_1), (u_2, m_2), \ldots, (u_K, m_K).$$

The user's initial abstract intents are progressively concretized and refined into specific intents as they interact with the assistant.

8.2.1 User Intent Representation

The user's intent is represented as a hierarchical structure consisting of multiple levels of specificity. Each intent I_k is defined as a sequence of levels:

$$I_k = \left(I_k^{(0)}, I_k^{(1)}, \ldots, I_k^{(L_k)}\right),$$

where $I_k^{(0)}$ represents the most general level of the intent (e.g., *"include an animal"*), and $I_k^{(L_k)}$ represents the most specific (e.g., *"include a tabby cat"*). These levels correspond to different stages of the user's evolving understanding of their goal.

Initial State. Initially, the user has formed only the most general version of each intent:

$$\mathcal{F}_0 = \{I_k^{(0)} : k = 1, \ldots, K\},$$

where K is the total number of intents. As the conversation progresses, the user develops their intents to more specific levels. At any turn t, the formed intents are captured by the set:

$$\mathcal{F}_t \subseteq \{I_k^{(\ell)} \mid 0 \leq \ell \leq L_k, k = 1, \ldots, K\},$$

which contains the specific intent levels the user has formed up to turn t.

Evolving State. The user forms the more specific levels of each intent in response to the assistant's messages. Given the complete intent hierarchy $\{I_1, \ldots, I_K\}$, the user's currently formed intents $\mathcal{F}_t$, and the model's response m_t, the user's formed intents are updated:

$$\mathcal{F}_{t+1} = \mathcal{U}(\{I_1, \ldots, I_K\}, m_t, \mathcal{F}_t),$$

where $\mathcal{U}$ represents a function that determines how effectively the model's response would help the user form more specific intent levels, and $\mathcal{F}_{t+1}$ is the new set of levels the user has formed. This captures how the user's intents develop and form over time as the assistant guides them through their evolving goals.

Message Expressiveness. A key consideration in our problem formulation is that the user can only express the levels of intent they have currently formed. At any given turn t, the user's message u_t can only refer to the intent levels in $\mathcal{F}_t$. If the user has formed only a high-level intent (e.g., "include an animal" but not yet "include a cat"), they cannot specify more concrete details until the user has formed the more specific intent.

8.2.2 Task Definition

Given this formulation, we define the model's or assistant's task with two objectives:

- **Objective 1: Intent Development.** The assistant should guide the user to form the most specific level of each intent. Formally, there should exist some turn t such that:

  $$\mathcal{F}_t = \{I_k^{(L_k)} : k = 1, \ldots, K\}.$$

  That is, by some point in the conversation, the user's formed intents $\mathcal{F}_t$ should contain the most specific level of every intent.

- **Objective 2: Intent Satisfaction.** Once the user has formed the most specific level of their intent, the assistant's response should satisfy this fully formed goal. We define a satisfaction function J that assesses whether a model's response m_t satisfies an intent $I_k^{(L_k)}$. The objective is to maximize:

  $$\sum_{k=1}^{K} S(m_t, I_k^{(L_k)}).$$

  That is, the assistant's response at each turn should align with and satisfy the most specific form of each user intent that has been formed.

The challenge is to develop an assistant that can balance these two objectives: progressively helping users develop more specific intents through exploration, while simultaneously providing responses that would satisfy all of these intents.

8.3 DESIGNLLM: Training Framework for Intent Formation

We introduce DESIGNLLM, a training framework that teaches language models to help users form and refine their intents through collaborative exploration. The core of our approach is a novel user simulator that operationalizes the problem formulation described in Section 8.2, enabling us to generate training data and compute reward signals to train models to balance intent development with intent satisfaction.

8.3.1 User Simulator

Our user simulator is designed to mimic how real users gradually discover and refine their intents through interaction with the assistant. We construct the simulator through a multi-stage process that creates realistic hierarchical intents, tracks their formation states, and generates appropriate user responses based on what intents have been formed.

Intent Creation. Given a completed artifact (e.g., a poem, code snippet, or document), we use an LLM to reverse-engineer the intents that must be satisfied to create this artifact. Specifically, we prompt the LLM to analyze the artifact and extract a set of concrete, specific intents $\{I_1^{(L_1)}, \ldots, I_K^{(L_K)}\}$ that characterize its key properties and requirements. This reverse-engineering approach yields intents that are both concrete (grounded in actual artifacts) and diverse (spanning the variety of artifacts in existing datasets). Importantly, this design allows for scale: given only a dataset of artifacts, we can automatically generate diverse intent sets without manual annotation.

Intent Abstraction. For each concrete intent $I_k^{(L_k)}$ extracted from an artifact, we use an LLM to generate progressively more abstract versions, creating the hierarchical structure $I_k = (I_k^{(0)}, I_k^{(1)}, \ldots, I_k^{(L_k)})$. We prompt the LLM to produce L_k abstraction levels by iteratively generalizing the intent while preserving its core meaning. For example, the specific intent "include a tabby cat character" might be abstracted to "include a cat character" at level $L_k - 1$, then to "include a pet animal" at level $L_k - 2$, and finally to "include an animal" at the most general level $I_k^{(0)}$. This produces a natural hierarchy from abstract to concrete that mirrors how users might progressively refine their thinking.

Initial User Request. To initiate the conversation, we generate the user's first message u_1 based on the artifact type and the most general level of each intent. We prompt an LLM to create a realistic initial request that mentions only a minimal subset of intents that must be mentioned for the request to be coherent and purposeful. Let $\mathcal{K}_0 \subseteq \{1, \ldots, K\}$ denote the indices of intents mentioned in u_1. Intents with indices in $\mathcal{K}_0$ are initialized as fully formed at their most general level, while all other intents are initialized as not yet formed. This creates the initial formation state:

$$\mathcal{F}_0 = \{I_k^{(0)} : k \in \mathcal{K}_0\}.$$

Evaluation of Assistant Messages. At each turn t, we evaluate the assistant's response m_t to determine how it affects intent formation. The evaluation proceeds in two stages:

Response Classification: We first use an LLM to classify whether m_t is an *artifact response* (i.e., provides the full or substantial portions of the requested artifact) or a *dialog act* (i.e., other message types such as questions or confirmations that do not include any artifacts or their portions).

Intent Evaluation: We then evaluate m_t against all intent levels using an LLM-as-a-judge approach [406, 217, 384]. The evaluation differs based on the response classification:

- For *artifact responses*, we assess how well m_t *satisfies* each intent level—whether the provided artifact exhibits the properties specified by the intent.
- For *dialog acts*, we assess how well m_t *probes* each intent level—whether the message mentions, asks about, or reminds the user about the intent.

For each intent level, the LLM judge assigns one of three scores: *no score* (intent not addressed), *partial score* (intent partially satisfied/probed), or *full score* (intent fully satisfied/probed). Evaluation proceeds in a cascaded

manner: for each intent I_k, we evaluate starting from the most general level $I_k^{(0)}$. Only if a level receives a full score do we proceed to evaluate the next more specific level $I_k^{(\ell+1)}$. This cascading ensures that more specific intent levels can only be formed if their more general predecessors are already addressed.

Update Intent Formation States. Each intent level can be in one of three formation states: *not formed*, *partially formed*, or *fully formed*. After evaluation, we update the formation state of each intent level based on the scores it received:

- If an intent level receives a *full score*, it transitions to *fully formed*.
- If an intent level receives a *partial score* and is currently not *fully formed*, it transitions up one level with probability 0.5—if *not formed* becomes *partially formed*, if *partially formed* becomes *fully formed*. This stochasticity reflects how users may form their intents even when a response only tangentially satisfies or probes at them.
- Intent levels that are already *fully formed* remain in that state.

These state transitions implement the intent evolution function $\mathcal{F}_{t+1} = \mathcal{U}(\{I_1, \ldots, I_K\}, m_t, \mathcal{F}_t)$, where $\mathcal{F}_{t+1}$ contains all intent levels that have reached the *fully formed* state by turn $t+1$.

User Response Generation. After updating intent formation states, we generate the next user message u_{t+1} using an LLM conditioned on: (1) the conversation history up to turn t, (2) the complete intent hierarchy $\{I_1, \ldots, I_K\}$, and (3) the set of the most specific intent levels that are currently formed. Crucially, we instruct the LLM to respect the user message expressiveness constraint: the generated message cannot reference intent levels that are *not formed* and can only vaguely allude to intent levels that are *partially formed*. Only *fully formed* intent levels can be explicitly and specifically mentioned in u_{t+1}. This ensures that the simulated user behaves realistically, expressing only what they have discovered or realized through the conversation so far.

8.3.2 Reward Function

We define a reward function that quantifies how effectively the assistant's response helps the user form their intents. For each intent I_k, we compute the progress made toward forming all of its levels:

$$\Delta_k = \frac{1}{L_k + 1} \sum_{\ell=0}^{L_k} \mathbf{1}\big[\text{level } I_k^{(\ell)} \text{ is } \textit{fully formed} \text{ at } t+1 \text{ but not at } t\big],$$

where $\mathbf{1}[\cdot]$ is the indicator function. This measures the fraction of intent levels that newly became fully formed as a result of the assistant's response m_t. The overall reward for response m_t is then:

$$R(m_t) = \frac{1}{K} \sum_{k=1}^{K} \Delta_k,$$

which averages the formation progress across all intents. This reward function incentivizes the assistant to help users form more specific intent levels while ensuring balanced progress across all intents.

8.3.3 Optimization and Synthetic Data Generation

Our user simulator enables multiple training paradigms for teaching models to support intent formation:

Reinforcement Learning. The simulator can provide turn-by-turn reward signals without requiring a separate reward model. At each turn t, after the assistant generates a response m_t, the simulator computes $R(m_t)$ by evaluating intent formation progress. These rewards can be used to train models via online reinforcement learning algorithms such as PPO [296] or DPO [274]

Synthetic Dataset Generation. The simulator can also generate large-scale synthetic conversation datasets for offline training methods. We create datasets by simulating full multi-turn conversations between the user simulator and one or more assistant models.

For *Supervised Fine-Tuning (SFT)*, we simulate conversations where, at each turn, multiple assistant candidates generate responses to the same user message. We use the simulator to compute $R(m_t)$ for each candidate response and continue the conversation with the highest-scoring response. This produces complete conversation trajectories that demonstrate high-quality intent elicitation and satisfaction behaviors.

For *Offline DPO*, we use the same multi-candidate sampling approach but construct preference pairs instead of trajectories. At each turn, we select the response with the highest reward as the "chosen" and the response with the lowest reward as the "rejected", creating training pairs.

8.4 Experimental Setup

Based on our DESIGNLLM framework, we create multiturn datasets for both fine-tuning and evaluation.

8.4.1 Tasks and Datasets

We focus on two different types of tasks: creative writing, and communicative writing. Composing and revising writing is one of the most common tasks that users employ with AI assistants [329, 391]. Writing tasks also involves substantial iterative back-and-forth for users to form their intents and for AI assistants to align with these intents [166]. To create the user simulators, we first start with datasets of artifacts: poems [145] for creative writing, and Medium articles [365] for communicative writing. From each dataset, we sample 300 samples for training and 50 for evaluation, and then apply our framework to automatically instantiate user simulators for each artifact by synthesizing and abstracting intents relevant to that artifact. We employ Claude Haiku 4.5 as the user simulator, and we use GPT-5.1 as the assistant to synthesize the multi-turn datasets.

8.4.2 Evaluation Metrics

In evaluation, we simulate conversations between each user simulator and the evaluated model as the assistant. Then, we assess the success of each multi-turn conversation on three metrics:

- **Intent Formation Score**: We measure the average proportion of intent levels that became fully formed across all intents by the end of the conversation. Specifically:

$$\textit{Formation} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{L_k+1}\sum_{\ell=0}^{L_k}\mathbf{1}\big[\text{level } I_k^{(\ell)} \text{ is } \textit{fully formed} \text{ at conversation end}\big]$$

  This metric quantifies how effectively the assistant helped the user discover and refine their intents throughout the interaction.

- **Intent Satisfaction Score**: We employ an LLM-as-a-Judge [406] to evaluate how well the final artifact created by the assistant satisfies the most specific intent levels $\{I_1^{(L_1)}, \ldots, I_K^{(L_K)}\}$, regardless of whether

these levels were fully formed during the conversation. This measures the quality of the assistant's final output against the user's most specific, latent intents.

- **Average Token Count**: We compute the mean number of tokens generated by the assistant across all turns in the conversation. This metric captures the efficiency of the overall conversation.

8.4.3 Fine-Tuning DESIGNLLMs

DESIGNLLMs are based on Llama-3.1-8B-Instruct [119] with LoRA finetuning [132]. We train our model through **Offline DPO**.

8.4.4 Baselines

We compare DESIGNLLMs against (1) *Base*: Llama-3.1-8B-Instruct, (2) *Prompted Base*: the base model with prompting instructions to support user exploration, and (3) *CollabLLM* [365]: a fine-tuning of the base model that is trained to proactively collaborate with users through follow-ups and questions.

8.5 Results from Simulated Experiments

Model	Formation	Satisfaction	#Tokens (k)
Base	0.587	0.568	1.99
Prompted Base	0.603	0.573	2.10
CollabLLM	0.635	0.615	**1.80**
DESIGNLLM	**0.667**	**0.640**	2.60
Rel. Improv.	10.6%	11.7%	-

Table 8.1: Evaluation results for intent formation and satisfaction scores with token counts and relative improvement compared to the prompted baseline.

As shown in Table 8.1, prompting can be helpful by encouraging the assistant to explore options or ask questions that can help the simulated user form its intents. However, as seen from the results, the performance gains from prompting are limited. We observe that this is due to the how, only with prompting, the model can be susceptible to return to its typical behavior of providing a single option each turn, even when the users' messages can be ambiguous.

DESIGNLLM effectively increases performance on intent formation and satisfaction, with significant improvements compared to the prompted base model. Unlike CollabLLM, which mostly focuses on asking clarification questions to the user, DESIGNLLM provides responses that incorporate multiple and distinct sample outputs that help the simulated user form and develop more specific intents. One caveat, however, is that this increased exploration also increases the overall length of the conversations as each response from the model incorporates more alternatives and options.

We present qualitative samples of DESIGNLLM responses and the baselines in Figure 8.1. As seen, given an ambiguous user request for a poem, the base model (Llama-3.1-8B-Instruct) simply returns a lengthy poem. On the other hand, CollabLLM tries to explore this ambiguity by asking multiple clarification questions to the user. If these do not effectively probe at the user's actual latent intents, the user cannot provide clear or more specific answers to these questions. DESIGNLLM instead attempts to explore the ambiguity in the user's request. Specifically, the

model provides multiple short samples of poems that explore different types of tones, which is a dimension that the user failed to specify in their request.

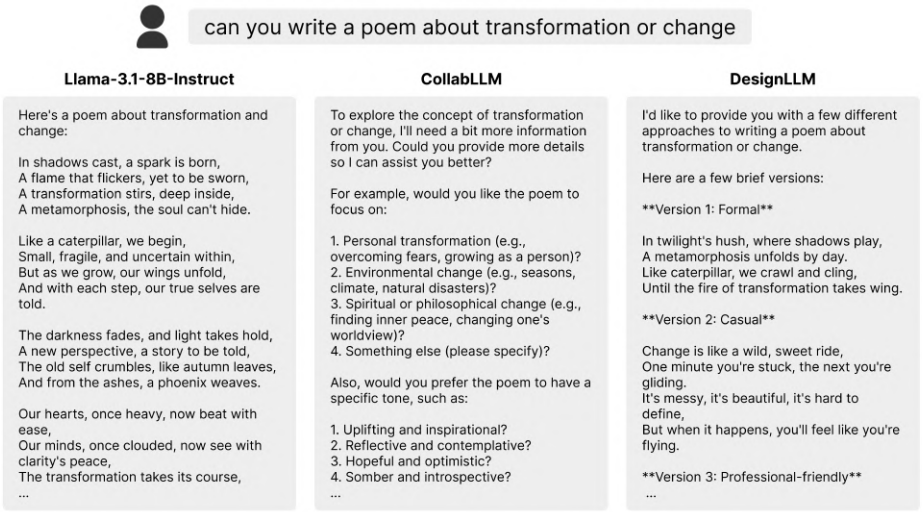

Figure 8.1: Qualitative samples for Llama-3.1-8B-Instruct, CollabLLM, and DESIGNLLM for the same user message.

Chapter 9. Discussion

This chapter lays out a set of guidelines for applying text disentanglement—focusing on how to disentangle text, who should perform the disentanglement, and how much disentanglement should be supported. Then, I discuss how this disentanglement approach can generalize to other task contexts and modalities, but also the limitations or drawbacks of this approach. Finally, the chapter concludes with future directions for AI models that go beyond language and text.

9.1 Guidelines for Text Disentanglement

This thesis proposes the general concept of ***text disentanglement***: disentangling input and output text during human-AI interaction to allow users to directly interact with the high-level attributes or components encoded within the text. To guide the application of this approach to new tasks and domains, this thesis provides answers to three critical questions: in *what* tasks is this approach most useful, *how* to disentangle text, *who* does the disentanglement, and *how much* to disentangle the text.

What? Types of Tasks

This thesis suggest that the text disentanglement approach is most beneficial in *open-ended and ill-defined tasks*—tasks where the user's goal, task method, and the overall success criteria are not clearly specified. As evidenced by lessons from the various work included in this thesis, this type of tasks requires users to iterate through execution and evaluation loops to explore through the space of outcomes, understand what their goals and success criteria are, and identify satisfactory outcomes. While text disentanglement can also support well-defined and close-ended task, as the user requires less iteration in these tasks, there is a reduced benefit to disentangling text and for users to interact with the resulting components. This thesis demonstrates the effectiveness of the approach in a diverse range of open-ended and ill-defined tasks (Table 9.1)—ranging from creative writing to web-development.

How? Decomposition vs. Differentiation

The various work in this thesis propose distinct ways to *disentangle* text. Specifically, we demonstrate two methods for disentanglement: *decomposition*, which involves decomposing a text artifact into its constituent components, and *differentiation*, which involves drawing out the various interpretations that exist for a single text artifact. The *Cells, Generators, and Lenses* framework demonstrates *decomposition* during execution by allowing users to break down their text inputs into multiple fragments that can be assembled into diverse configurations. On the evaluation-side, EVALET *decomposes* text outputs into their constituent fragment-level functions, allowing users to gain an overview of how outputs are generally composed. Regarding *differentiation*, Stylette takes a single user intent and *differentiates* all the various interpretations to create a palette of operations. EVALLM assesses each text output by interpreting and scoring it on each user-defined criteria, *differentiating* its overall quality across these dimensions. This raises a natural question: when should we apply *decomposition*, and when should we apply *differentiation*? Based on lessons from the work in this thesis, I propose that these two forms of disentanglement play different roles during execution and evaluation.

During execution, decomposing a single text artifact into multiple component and then differentiating each component into multiple interpretations can exposes users to an excessive number of elements and control variables.

Engaging with all these elements would require excessive levels of cognitive load, hindering users rather than supporting them. Thus, this thesis suggests that only one method should be applied during execution: when a user's intent is relatively atomic (i.e., a single objective), it is more effective to *differentiate* to help user explore a range of possible interpretations. In contrast, when the user's intent is complex or compound, it is more useful to *decompose*, breaking the intent into controllable components so that users can systematically test and recombine them.

On the evaluation side, however, both forms of disentanglement can be productively combined to support multi-layered interaction: for each output, users can first be provided with an overview that *differentiates* the output along multiple criteria, and then, if they desire more detail, they can dive deeper to inspect components that were *decomposed* from the output. This is the approach embodied in EVALET, and it follows Shneiderman's *Visual Information-Seeking Mantra* [305]: "overview first, zoom and filter, then details on demand."

Who? Users, Designers, or Systems

This thesis proposes various approaches where the disentanglement is conducted by different *actors*. For example, the *Cells, Generators, and Lenses* framework proposes that interface designers should provide interactive objects that allow users to disentangle the text themselves. While other approaches, like EVALET, use an AI model to automatically disentangle the text based on user-defined dimensions. Then, the question is: who should disentangle the text the *users*, the system *designers*, or the *system* itself?

Lessons from these work indicate that the answer to this question is not *"one or the other"* but rather *"all of the above"*—all three actors should participate in the disentanglement process to effectively support user interaction. This participation occurs at three-levels but each level is dependent on the other.

- **Designers establish the possibilities and boundaries.** They determine how text can and should be disentangled for the specific user task. Through human-centered design practices (e.g., literature review, formative interviews, pilot studies), designers should identify user needs and challenges, and uncover how disentanglement can address these. Through this, designers should also define the degree of freedom granted to users: how much disentanglement can users perform and how can they customize the process. Essentially, designers establish the structure and guardrails for disentanglement.

- **Systems enable the possibilities.** Systems can enable disentanglement by automating the process. However, this automation requires careful consideration of the system's capabilities and limitations as incorrect disentanglement can hinder interaction. Furthermore, these systems should incorporate mechanisms that allow users to adapt or customize the disentanglement process, and even correct the system if necessary.

- **Users exercise control within the boundaries.** Users should always have some levels of control over how disentanglement is performed either by allowing users to perform the disentanglement themselves or to guide the systems to disentangle in ways that align with their own needs. Again, however, the degree of freedom given to users should be appropriately constrained by designers based on task considerations and the capabilities of the systems themselves.

How Much? Levels of Abstraction

The work in this thesis explores at most two levels of disentanglement or abstraction. For instance, EVALET differentiate outputs into criteria-wise scores to provide overviews, and then surfaces fragment-level functions within each dimension to support more detailed inspection. Disentanglement can be applied recursively to create more fine-grained and specific elements from a single text artifacts that grant users with greater control. This, however, also significantly increases cognitive load due to the number of elements that users must manipulate.

Work	Artifact Types	Task Domains
Cells, Generators, and Lenses	Prompts, writing	Creative writing, business writing, copywriting
Stylette	Coding, spoken utterances, code	Web development
EVALLM	Writing, Ideation Lists lists	Creative writing, journalistic writing
EVALET	Writing, chat dialogues, reasoning traces	Creative writing, copywriting, open-domain dialogue, social dialogue, coding/math
CUPID	Chat dialogues	Open-domain dialogue
DESIGNLLM	Messages	Creative writing

Table 9.1: Overview of works, artifact types, and task domains.

As a guideline, lessons from this thesis suggest that systems should be designed with a single level of disentanglement at the start, while providing mechanism that allow users to guide how the disentanglement is performed. Through these mechanisms, users can customize the granularity of the outcomes from the disentanglement process, without increasing the number of disentanglement layers. For example, EVALLM implements only one level of disentanglement (i.e., each output into a set of scores), but allows users to control granularity by defining more criteria that are more specific and detailed. From this starting point, designers can conduct pilot studies to observe how users customize the disentanglement process to meet their needs. Based on these findings, designers can iteratively add levels of disentanglement to match the users' information needs and processing capacity. This iterative approach is exemplified by the evaluation from EVALLM to EVALET: users of EVALLM indicated that they needed more than just scores—they wanted to see which components in each output contributed to these scores. This insight directly informed the design of EVALET, which provides two layers that users can flexibly navigate between based on the level of detail needed at any given moment.

9.2 Generalizability of Text Disentanglement

Through various user studies and case studies, this thesis demonstrates the text disentanglement approach across diverse types of task phrases and artifacts—summarized in Table 9.1. I envision that this approach can generalize more widely to additional task contexts and artifacts. In particular, recent years have led to the rise of *agentic systems* [4, 5, 115] that automate complex, long-horizon workflows by generating lengthy reasoning traces that interleave task planning, reasoning, actions—leveraging models' *test-time compute* [122]. Text disentanglement can support both execution and evaluation in this new interaction paradigm. For execution, user's intents can be disentangled into interactive step-by-step plans, which can serve as an interface between the user and the agentic system. This design approach has already begun to be adopted in emergent systems [88, 4]. For evaluation, an example case for EVALET demonstrated that reasoning traces can be automatically disentangled to surface behaviors of interest. Thus, I envision that this approach can also be applied to agentic systems to help users uncover what behaviors the agentic system is exhibiting and assess their alignment with user goals—making typically unwieldy reasoning traces more interpretable and actionable.

While the proposed text disentanglement approach focuses on textual artifacts, the general concept of disentangling high-level elements from artifacts and enabling users to interact with these elements can be extended to diverse modalities. In additional work not included in this thesis, I have explored such extensions: generated images can be disentangled into semantic attributes that serve as controllable inputs [312]. Artifacts such as videos can also be decomposed into semantically meaningful chunks that users can interact with separately [164, 382]. While this evidence suggests the potential for generalizing the approach to other modalities, future work is needed

to validate this in greater depth.

9.3 Limitations of Text Disentanglement

This section offers reflections on the limitations of text disentanglement.

Cognitive load trade-offs. Text disentanglement inherently increases cognitive load on users as it expands the number of input elements that users must manage and also the output elements that they must inspect—either through the increase in the number of outputs produced or by exposing multiple interpretations of the same output. As discussed in Section 9.1, interfaces that implement text disentanglement must carefully consider the degree of cognitive load that users are exposed to and include features that can scaffold users during interaction.

Applicability in diverse user contexts. Due to the induced cognitive load, the effectiveness of text disentanglement depends on the user's degree of investment in the task at hand. Users must be invested to dedicate the additional cognitive effort required to explore with the disentangled text inputs and to inspect the disentangled outputs. For example, if a user provides a simple unambiguous request, disentanglement can introduce unnecessary overhead without providing much benefit. On the other side, if the AI model returns a concise output, it may be more efficient for users to inspect and assess the output directly rather than through disentangled components.

User must have an initial intent. The text disentanglement approach relies on the user providing an initial intent, which can be disentangled as an input to support execution or the resulting output can be disentangled to support evaluation. However, in various contexts, the user may not be able to or may not realize to provide this initial intent to an AI model. For example, a novice performing a complex tasks is unaware that they missed a sub-task, or a user is unaware that an AI model can help them with their ongoing task. In these cases, future work must incorporate additional proactive support, where the system infers the users' possible intents from interaction traces and these inferred intents can then be disentangled for the user.

Dependency on system accuracy. As also discussed in Section 9.1, certain task contexts require systems to automatically disentangle text for the user. In these cases, the *accuracy* of this disentanglement can have significant consequences on the interaction experience. For instance, if Stylette fails to surface the interpretation a user actually needs, users without domain expertise may not recognize this failure and settle for unsatisfactory outcomes, unaware that better options exist. Likewise, if EVALET fails to surface the fragment-level functions that are actually of interest for the user, the user may not realize this failure without inspecting the outputs themselves. Thus, when applying text disentanglement, it is crucial to rigorously evaluate the accuracy of the system's disentanglement (i.e., alignment with actual user judgments) and to incorporate error recovery mechanisms.

Chapter 10. Conclusion

This dissertation introduced ***text disentanglement***: a novel approach to human-AI interaction in which text inputs and outputs are decomposed into high-level components or differentiated into diverse interpretations, allowing users to directly interact with the abstractions they cognitively operate on. To conclude, this chapter summarizes the main contributions of the thesis and proposes important directions for future research.

10.1 Summary of Contributions

Broadly, this thesis makes contributions in human-AI interaction, natural language processing, and AI evaluation.

- **Human-AI Interaction**: Novel interaction techniques that bridge the gulf of execution and evaluation by disentangling high-level objectives and attributes from text and representing these through interactive components; and design frameworks and systems that implement these techniques to support interactive AI alignment.
- **Natural Language Processing**: Computational methods for decomposing complex text artifacts into atomic semantic units and for differentiating these artifacts into diverse interpretations; fine-tuned models that perform this automated disentanglement; and training frameworks that teach LLMs to intrinsically generate disentangled artifacts.
- **AI Evaluation**: Novel evaluation methods to assess and analyze LLMs according to user-defined criteria and based on the fragment-level functions contained within their outputs; benchmarks to assess LLMs' capabilities to disentangle semantically meaningful information from interaction histories; and interfaces that support interactive evaluation and sensemaking of model behavior and performance.

10.2 Future Directions

I propose three directions for future research.

10.2.1 From *Language* Models to *Interaction* Models

A fundamental limitation of current AI systems is their reliance on LLMs as a universal backbone. LLMs mostly operate on a text-in, text-out paradigm (with recent models also supporting images), which necessitates the text disentanglement approaches proposed in this thesis where the text is processed post-hoc to integrate interactive structures. However, I envision a future where AI models inherently generate structured, interactive outputs rather than plain text—absolving the need for subsequent disentanglement. Instead of producing monolithic text responses, these models would generate interactive components from the outset—elements that users can directly manipulate to refine their intents, explore alternatives, or compose complex workflows.

While the DESIGNLLM framework explored how to train models to produce multiple samples that can help users to explore alternatives and realize their intents, users still have to interact with this model through text messages. The direction proposed here goes one step further. For example, when a user requests help with a task, the model can generate interactive blocks that represent different interpretations or decomposed components, allowing users to select, modify, and recombine them fluidly. This paradigm shift would move us from *Large Language*

Models (LLMs) that generate text requiring post-hoc disentanglement, toward *Large Interaction Models* (LIMs)—systems designed to generate interaction primitives as first-class outputs. Such models would natively support the user agency and fine-grained control that this thesis advocates, eliminating the need to retrofit interactivity onto text after generation.

10.2.2 Personalized Disentanglement

While this thesis proposes generalizable methods for text disentanglement, individual users possess distinct mental models and may operate on different levels of abstraction depending on their preferences and workflows. Furthermore, a user's preferred abstractions may evolve as they gain experience with a specific task or system. Future research should investigate how systems can dynamically adapt their disentanglement strategies based on user interaction histories with interfaces that support customization of the text disentanglement approaches. Specifically, work is needed to develop adaptive interfaces that learn user-specific disentanglement patterns, automatically adjusting the granularity and type of disentangled components to match the user's cognitive state and current workflow.

10.2.3 Disentanglement in Long-Horizon Tasks

As AI capabilities evolve and enable agentic systems that execute autonomous, long-horizon workflows spanning days or weeks, supporting oversight over these agents is critical. While preliminary evidence discussed in Chapter 9 suggests that text disentanglement can aid in validating reasoning traces, scaling this approach to workflows at significantly greater time horizons and involving multiple agents presents new design challenges. Future work must investigate how to apply disentanglement to these massive textual streams effectively—not by surfacing every component, but by automatically identifying and disentangling crucial moments or critical decision points that require human verification. Furthermore, research should explore how users can interact with these disentangled components to intervene in and steer agentic systems mid-execution, transforming disentanglement from a post-hoc evaluation tool into a mechanism for real-time supervisory control.

Bibliography

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Eytan Adar, Mira Dontcheva, and Gierad Laput. 2014. CommandSpace: Modeling the Relationships between Tasks, Descriptions and Features. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) *(UIST '14)*. Association for Computing Machinery, New York, NY, USA, 167–176. doi:10.1145/2642918.2647395

[3] Andrea Agostinelli, Timo I Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, et al. 2023. Musiclm: Generating music from text. *arXiv preprint arXiv:2301.11325* (2023).

[4] Adept AI. 2024. Manus: An Agentic Framework for Complex Task Automation. https://www.adept.ai/blog/manus. Accessed: 2025-04-10.

[5] Genspark AI. 2024. Genspark: Agents That Write Code and Explain It. https://genspark.ai/. Accessed: 2025-04-10.

[6] Meta AI. 2025. Introducing Llama 4: 10 Million Token Context. https://ai.meta.com/llama/. Accessed: 2025-04-10.

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

[8] 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. doi:10.1145/3173574.3173779

[9] Afra Feyza Akyürek, Ekin Akyürek, Aman Madaan, Ashwin Kalyan, Peter Clark, Derry Wijaya, and Niket Tandon. 2023. RL4F: Generating Natural Language Feedback with Reinforcement Learning for Repairing Model Outputs. *arXiv preprint arXiv:2305.08844* (2023).

[10] Miltos Allamanis, Daniel Tarlow, Andrew Gordon, and Yi Wei. 2015. Bimodal Modelling of Source Code and Natural Language. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 37)*, Francis Bach and David Blei (Eds.). PMLR, Lille, France, 2123–2132. https://proceedings.mlr.press/v37/allamanis15.html

[11] 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/

[12] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. 2019. Guidelines for Human-AI Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300233

[13] 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. doi:10.1145/3126594.3126637

[14] Paul André, Aniket Kittur, and Steven P Dow. 2014. Crowd synthesis: Extracting categories and clusters from complex data. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*. 989–998.

[15] Chinmaya Andukuri, Jan-Philipp Fränken, Tobias Gerstenberg, and Noah D Goodman. 2024. Star-gate: Teaching language models to ask clarifying questions. *arXiv preprint arXiv:2403.19154* (2024).

[16] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. PaLM 2 Technical Report. *arXiv preprint arXiv:2305.10403* (2023).

[17] Anthropic. 2023. Introducing Claude. https://www.anthropic.com/news/introducing-claude/ Accessed: March 19, 2025.

[18] Anthropic. 2024. Introducing Claude 3.5 Sonnet. https://www.anthropic.com/news/claude-3-5-sonnet Accessed: March 19, 2025.

[19] Anthropic. 2025. Claude 3.7 Sonnet and Claude Code. https://www.anthropic.com/news/claude-3-7-sonnet Accessed: March 19, 2025.

[20] Ian Arawjo, Chelse Swoopes, Priyan Vaithilingam, Martin Wattenberg, and Elena L Glassman. 2024. Chainforge: A visual toolkit for prompt engineering and llm hypothesis testing. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–18.

[21] Lisa P Argyle, Ethan C Busby, Nancy Fulda, Joshua R Gubler, Christopher Rytting, and David Wingate. 2023. Out of one, many: Using language models to simulate human samples. *Political Analysis* 31, 3 (2023), 337–351.

[22] Zahra Ashktorab, Michael Desmond, Qian Pan, James M Johnson, Martin Santillan Cooper, Elizabeth M Daly, Rahul Nair, Tejaswini Pedapati, Swapnaja Achintalwar, and Werner Geyer. 2024. Aligning Human and LLM Judgments: Insights from EvalAssist on Task-Specific Evaluations and AI-assisted Assessment Strategy Preferences. *arXiv preprint arXiv:2410.00873* (2024).

[23] Tal August, Lucy Lu Wang, Jonathan Bragg, Marti A. Hearst, Andrew Head, and Kyle Lo. 2023. Paper Plain: Making Medical Research Papers Approachable to Healthcare Consumers with Natural Language Processing. *ACM Trans. Comput.-Hum. Interact.* (apr 2023). doi:10.1145/3589955 Just Accepted.

[24] Autodesk. 2022. *Project Dreamcatcher*. Retrieved April 2, 2022 from https://www.autodesk.com/research/projects/project-dreamcatcher/

[25] The Webby Awards. 2021. Top Websites and Mobile Sites | The Webby Awards. Retrieved August 29, 2021 from https://winners.webbyawards.com/winners/websites-and-mobile-sites

[26] Jinheon Baek, Nirupama Chandrasekaran, Silviu Cucerzan, Allen Herring, and Sujay Kumar Jauhar. 2024. Knowledge-augmented large language models for personalized contextual query suggestion. In *Proceedings of the ACM on Web Conference 2024*. 3355–3366.

[27] Xuechunzi Bai, Angelina Wang, Ilia Sucholutsky, and Thomas L. Griffiths. 2024. Measuring Implicit Bias in Explicitly Unbiased Large Language Models. arXiv:2402.04105 [cs.CY] https://arxiv.org/abs/2402.04105

[28] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862* (2022).

[29] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073* (2022).

[30] Yushi Bai, Jiahao Ying, Yixin Cao, Xin Lv, Yuze He, Xiaozhi Wang, Jifan Yu, Kaisheng Zeng, Yijia Xiao, Haozhe Lyu, et al. 2023. Benchmarking Foundation Models with Language-Model-as-an-Examiner. *arXiv preprint arXiv:2306.04181* (2023).

[31] Chris L Baker, Rebecca Saxe, and Joshua B Tenenbaum. 2009. Action understanding as inverse planning. *Cognition* 113, 3 (2009), 329–349.

[32] 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. doi:10.1145/3306346.3323023

[33] 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. doi:10.1145/332040.332473

[34] 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. doi:10.1145/345513.345267

[35] Rishabh Bhardwaj and Soujanya Poria. 2023. Red-Teaming Large Language Models using Chain of Utterances for Safety-Alignment. arXiv:2308.09662 [cs.CL]

[36] Jan Biniok. 2021. Tampermonkey. Retrieved September 5, 2021 from https://www.tampermonkey.net/

[37] Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. GPT-NeoX-20B: An Open-Source Autoregressive Language Model. (2022).

[38] Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. *GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow*. doi:10.5281/zenodo.5297715 If you use this software, please cite it using these metadata..

[39] Godfred O Boateng, Torsten B Neilands, Edward A Frongillo, Hugo R Melgar-Quiñonez, and Sera L Young. 2018. Best practices for developing and validating scales for health, social, and behavioral research: a primer. *Frontiers in public health* 6 (2018), 149.

[40] Michael Bolin, Matthew Webber, Philip Rha, Tom Wilson, and Robert C. Miller. 2005. Automation and Customization of Rendered Web Pages. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology* (Seattle, WA, USA) *(UIST '05)*. Association for Computing Machinery, New York, NY, USA, 163–172. doi:10.1145/1095034.1095062

[41] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258* (2021).

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

[43] 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. doi:10.1109/TVCG.2021.3114792

[44] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. 2024. Video generation models as world simulators. (2024). https://openai.com/research/video-generation-models-as-world-simulators

[45] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165 [cs.CL]

[46] 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. doi:10.1145/3411764.3445372

[47] Ángel Alexander Cabrera, Abraham J Druck, Jason I Hong, and Adam Perer. 2021. Discovering and validating ai errors with crowdsourced failure reports. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW2 (2021), 1–22.

[48] Ángel Alexander Cabrera, Erica Fu, Donald Bertucci, Kenneth Holstein, Ameet Talwalkar, Jason I. Hong, and Adam Perer. 2023. Zeno: An Interactive Framework for Behavioral Evaluation of Machine Learning. 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 419, 14 pages. doi:10.1145/3544548.3581268

[49] Ángel Alexander Cabrera, Marco Tulio Ribeiro, Bongshin Lee, Robert Deline, Adam Perer, and Steven M Drucker. 2023. What did my AI learn? how data scientists make sense of model behavior. *ACM Transactions on Computer-Human Interaction* 30, 1 (2023), 1–27.

[50] 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*.

[51] Eva Cetinic and James She. 2022. Understanding and Creating Art with AI: Review and Outlook. *ACM Trans. Multimedia Comput. Commun. Appl.* 18, 2, Article 66 (feb 2022), 22 pages. doi:10.1145/3475799

[52] Kerry Shih-Ping Chang and Brad A. Myers. 2012. *WebCrystal: Understanding and Reusing Examples in Web Authoring*. Association for Computing Machinery, New York, NY, USA, 3205–3214. https://doi.org/10.1145/2207676.2208740

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

[54] Duen Horng Chau, Aniket Kittur, Jason I Hong, and Christos Faloutsos. 2011. Apolo: making sense of large network data by combining rich user interaction and machine learning. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 167–176.

[55] Siddhartha Chaudhuri, Evangelos Kalogerakis, Stephen Giguere, and Thomas Funkhouser. 2013. Attribit: Content Creation with Semantic Attributes. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology* (St. Andrews, Scotland, United Kingdom) *(UIST '13)*. Association for Computing Machinery, New York, NY, USA, 193–202. doi:10.1145/2501988.2502008

[56] Chunyang Chen, Sidong Feng, Zhenchang Xing, Linda Liu, Shengdong Zhao, and Jinshui Wang. 2019. Gallery D.C.: Design Search and Knowledge Discovery through Auto-Created GUI Component Gallery. *Proc. ACM Hum.-Comput. Interact.* 3, CSCW, Article 180 (Nov. 2019), 22 pages. doi:10.1145/3359282

[57] Yan Chen, Sang Won Lee, and Steve Oney. 2021. CoCapture: Effectively Communicating UI Behaviors on Existing Websites by Demonstrating and Remixing. 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 416, 14 pages. doi:10.1145/3411764.3445573

[58] Cheng-Han Chiang and Hung yi Lee. 2023. Can Large Language Models Be an Alternative to Human Evaluations? arXiv:2305.01937 [cs.CL]

[59] 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).

[60] 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

[61] 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* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 209, 19 pages. doi:10.1145/3491102.3501819

[62] 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. doi:10.1145/2984511.2984577

[63] Elizabeth Clark, Tal August, Sofia Serrano, Nikita Haduong, Suchin Gururangan, and Noah A. Smith. 2021. All That's 'Human' Is Not Gold: Evaluating Human Evaluation of Generated Text. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online, 7282–7296. doi:10.18653/v1/2021.acl-long.565

[64] 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. doi:10.1145/3172944.3172983

[65] Jonathan Cook, Tim Rocktäschel, Jakob Foerster, Dennis Aumiller, and Alex Wang. 2024. Ticking all the boxes: Generated checklists improve llm evaluation and generation. *arXiv preprint arXiv:2410.03608* (2024).

[66] Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. 2024. Simple and Controllable Music Generation. arXiv:2306.05284 [cs.SD] https://arxiv.org/abs/2306.05284

[67] Eric Corbett and Astrid Weber. 2016. What Can I Say? Addressing User Experience Challenges of a Mobile Voice User Interface for Accessibility. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services* (Florence, Italy) *(MobileHCI '16)*. Association for Computing Machinery, New York, NY, USA, 72–82. doi:10.1145/2935334.2935386

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

[69] Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2023. Ultrafeedback: Boosting language models with high-quality feedback. (2023).

[70] Jiaxi Cui, Zongjian Li, Yang Yan, Bohua Chen, and Li Yuan. 2023. ChatLaw: Open-Source Legal Large Language Model with Integrated External Knowledge Bases. arXiv:2306.16092 [cs.CL]

[71] 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).

[72] 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).

[73] 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

[74] 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. doi:10.1145/2856767.2856795

[75] Shizhe Diao, Rui Pan, Hanze Dong, Ka Shun Shum, Jipeng Zhang, Wei Xiong, and Tong Zhang. 2023. LMFlow: An Extensible Toolkit for Finetuning and Inference of Large Foundation Models. arXiv:2306.12420 [cs.CL]

[76] Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. 2024. Longrope: Extending llm context window beyond 2 million tokens. *arXiv preprint arXiv:2402.13753* (2024).

[77] DomCop. 2021. What is Open PageRank? Retrieved August 29, 2021 from https://www.domcop.com/openpagerank/what-is-openpagerank

[78] Shachar Don-Yehiya, Leshem Choshen, and Omri Abend. 2024. Learning from Naturally Occurring Feedback. *arXiv preprint arXiv:2407.10944* (2024).

[79] Kees Dorst and Nigel Cross. 2001. Creativity in the design process: co-evolution of problem–solution. *Design studies* 22, 5 (2001), 425–437.

[80] Yao Dou, Maxwell Forbes, Rik Koncel-Kedziorski, Noah A Smith, and Yejin Choi. 2021. Is GPT-3 text indistinguishable from human text? SCARECROW: A framework for scrutinizing machine text. *arXiv preprint arXiv:2107.01294* (2021).

[81] 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.

[82] 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).

[83] Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. AlpacaFarm: A Simulation Framework for Methods that Learn from Human Feedback. arXiv:2305.14387 [cs.LG]

[84] Lisa Dunlap, Krishna Mandal, Trevor Darrell, Jacob Steinhardt, and Joseph E Gonzalez. 2024. VibeCheck: Discover and Quantify Qualitative Differences in Large Language Models. *arXiv preprint arXiv:2410.12851* (2024).

[85] Alexander R Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. Summeval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics* 9 (2021), 391–409.

[86] Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833* (2018).

[87] Judith E. Fan, Monica Dinculescu, and David Ha. 2019. Collabdraw: An Environment for Collaborative Sketching with an Artificial Agent. In *Proceedings of the 2019 on Creativity and Cognition* (San Diego, CA, USA) *(C&C '19)*. Association for Computing Machinery, New York, NY, USA, 556–561. doi:10.1145/3325480.3326578

[88] KJ Feng, Kevin Pu, Matt Latzke, Tal August, Pao Siangliulue, Jonathan Bragg, Daniel S Weld, Amy X Zhang, and Joseph Chee Chang. 2024. Cocoa: Co-planning and co-execution with ai agents. *arXiv preprint arXiv:2412.10999* (2024).

[89] 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).

[90] Patrick Fernandes, Aman Madaan, Emmy Liu, António Farinhas, Pedro Henrique Martins, Amanda Bertsch, José G. C. de Souza, Shuyan Zhou, Tongshuang Wu, Graham Neubig, and André F. T. Martins. 2023. Bridging the Gap: A Survey on Integrating (Human) Feedback for Natural Language Generation. arXiv:2305.00955 [cs.CL]

[91] Michael J Fitzgerald et al. 2008. *CopyStyler: Web design by example*. Ph. D. Dissertation. Massachusetts Institute of Technology.

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

[93] James Fogarty, Desney Tan, Ashish Kapoor, and Simon Winder. 2008. CueFlik: Interactive Concept Learning in Image Search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Florence, Italy) *(CHI '08)*. Association for Computing Machinery, New York, NY, USA, 29–38. doi:`10.1145/1357054.1357061`

[94] Adam Fourney, Richard Mann, and Michael Terry. 2011. Query-Feature Graphs: Bridging User Vocabulary and System Functionality. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 207–216. doi:`10.1145/2047196.2047224`

[95] C. Ailie Fraser, Mira Dontcheva, Holger Winnemöller, Sheryl Ehrlich, and Scott Klemmer. 2016. DiscoverySpace: Suggesting Actions in Complex Software. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems* (Brisbane, QLD, Australia) *(DIS '16)*. Association for Computing Machinery, New York, NY, USA, 1221–1232. doi:`10.1145/2901790.2901849`

[96] C. Ailie Fraser, Julia M. Markel, N. James Basa, Mira Dontcheva, and Scott Klemmer. 2020. ReMap: Lowering the Barrier to Help-Seeking with Multimodal Search. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 979–986. doi:`10.1145/3379337.3415592`

[97] C. Ailie Fraser, Tricia J. Ngoon, Mira Dontcheva, and Scott Klemmer. 2019. RePlay: Contextually Presenting Learning Videos Across Software Applications. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–13. `https://doi.org/10.1145/3290605.3300527`

[98] 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`

[99] Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023. Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166* (2023).

[100] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. 1987. The Vocabulary Problem in Human-System Communication. *Commun. ACM* 30, 11 (Nov. 1987), 964–971. doi:`10.1145/32206.32212`

[101] Anushay Furqan, Chelsea Myers, and Jichen Zhu. 2017. Learnability through Adaptive Discovery Tools in Voice User Interfaces. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI EA '17)*. Association for Computing Machinery, New York, NY, USA, 1617–1623. doi:10.1145/3027063.3053166

[102] Tong Gao, Mira Dontcheva, Eytan Adar, Zhicheng Liu, and Karrie G. Karahalios. 2015. DataTone: Managing Ambiguity in Natural Language Interfaces for Data Visualization. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (Charlotte, NC, USA) *(UIST '15)*. Association for Computing Machinery, New York, NY, USA, 489–500. doi:10.1145/2807442.2807478

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

[104] Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. 2024. Scaling synthetic data creation with 1,000,000,000 personas. *arXiv preprint arXiv:2406.20094* (2024).

[105] Simret Araya Gebreegziabher, Charles Chiang, Zichu Wang, Zahra Ashktorab, Michelle Brachman, Werner Geyer, Toby Jia-Jun Li, and Diego Gómez-Zará. 2025. MetricMate: An Interactive Tool for Generating Evaluation Criteria for LLM-as-a-Judge Workflow. (2025).

[106] Sebastian Gehrmann, Elizabeth Clark, and Thibault Sellam. 2023. Repairing the cracked foundation: A survey of obstacles in evaluation practices for generated text. *Journal of Artificial Intelligence Research* 77 (2023), 103–166.

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

[108] 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. doi:10.1145/3290605.3300526

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

[110] Katy Ilonka Gero, Chelse Swoopes, Ziwei Gu, Jonathan K Kummerfeld, and Elena L Glassman. 2024. Supporting sensemaking of large language model outputs at scale. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–21.

[111] 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.

[112] Elena L Glassman, Jeremy Scott, Rishabh Singh, Philip J Guo, and Robert C Miller. 2015. OverCode: Visualizing variation in student solutions to programming problems at scale. *ACM Transactions on Computer-Human Interaction (TOCHI)* 22, 2 (2015), 1–35.

[113] 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

[114] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).

[115] Google. 2024. Gemini Deep Research - your personal research assistant. https://gemini.google/overview/deep-research/. Accessed: 2025-04-10.

[116] Google. 2024. Introducing Gemini 2.0: our new AI model for the agentic era. https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/ Accessed: March 19, 2025.

[117] Karthik Gopalakrishnan, Behnam Hedayatnia, Qinlang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tur. 2023. Topical-Chat: Towards Knowledge-Grounded Open-Domain Conversations. arXiv:2308.11995 [cs.CL]

[118] 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

[119] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).

[120] Greasemonkey. 2021. Greasespot. Retrieved September 5, 2021 from https://www.greasespot.net/

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

[122] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025).

[123] 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. doi:10.1145/3290605.3300854

[124] Matthew Guzdial and Mark Riedl. 2019. An Interaction Framework for Studying Co-Creative AI. (2019). doi:10.48550/ARXIV.1903.09709

[125] 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

[126] David Ha and Douglas Eck. 2017. A Neural Representation of Sketch Drawings. *CoRR* abs/1704.03477 (2017). arXiv:1704.03477 http://arxiv.org/abs/1704.03477

[127] 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. doi:10.1145/3491102.3502052

[128] 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.

[129] Sandra G Hart and Lowell E Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in psychology*, Vol. 52. Elsevier, 139–183.

[130] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Human Mental Workload*, Peter A. Hancock and Najmedin Meshkati (Eds.). Advances in Psychology, Vol. 52. North-Holland, 139–183. doi:10.1016/S0166-4115(08)62386-9

[131] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. RULER: What's the Real Context Size of Your Long-Context Language Models? *arXiv preprint arXiv:2404.06654* (2024).

[132] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685 [cs.CL]

[133] 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

[134] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, and Douglas Eck. 2018. An Improved Relative Self-Attention Mechanism for Transformer with Application to Music Generation. *CoRR* abs/1809.04281 (2018). arXiv:1809.04281 http://arxiv.org/abs/1809.04281

[135] Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron Courville, and Douglas Eck. 2017. Counterpoint by Convolution. In *International Society for Music Information Retrieval (ISMIR)*.

[136] 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).

[137] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276* (2024).

[138] Edwin L Hutchins, James D Hollan, and Donald A Norman. 1985. Direct manipulation interfaces. *Human–computer interaction* 1, 4 (1985), 311–338.

[139] Jane Im, Sonali Tandon, Eshwar Chandrasekharan, Taylor Denby, and Eric Gilbert. 2020. *Synthesized Social Signals: Computationally-Derived Social Signals from Account Histories*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376383

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

[141] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720* (2024).

[142] Joel Jang, Seungone Kim, Bill Yuchen Lin, Yizhong Wang, Jack Hessel, Luke Zettlemoyer, Hannaneh Hajishirzi, Yejin Choi, and Prithviraj Ammanabrolu. 2023. Personalized soups: Personalized large language model alignment via post-hoc parameter merging. *arXiv preprint arXiv:2310.11564* (2023).

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

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

[145] Harshit Jhalani. 2020. Haiku Dataset. https://www.kaggle.com/datasets/hjhalani30/haiku-dataset.

[146] Jiaming Ji, Tianyi Qiu, Boyuan Chen, Borong Zhang, Hantao Lou, Kaile Wang, Yawen Duan, Zhonghao He, Jiayi Zhou, Zhaowei Zhang, et al. 2023. Ai alignment: A comprehensive survey. *arXiv preprint arXiv:2310.19852* (2023).

[147] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. arXiv:2310.06825 [cs.CL] https://arxiv.org/abs/2310.06825

[148] 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 *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI EA '22)*. Association for Computing Machinery, New York, NY, USA, Article 35, 8 pages. doi:10.1145/3491101.3503564

[149] 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. doi:10.1145/3474349.3480209

[150] Peiling Jiang, Jude Rayan, Steven P Dow, and Haijun Xia. 2023. Graphologue: Exploring large language model responses with interactive diagrams. In *Proceedings of the 36th annual ACM symposium on user interface software and technology*. 1–20.

[151] Minsuk Kahng, Ian Tenney, Mahima Pushkarna, Michael Xieyang Liu, James Wexler, Emily Reif, Krystal Kallarackal, Minsuk Chang, Michael Terry, and Lucas Dixon. 2024. LLM Comparator: Interactive Analysis of Side-by-Side Evaluation of Large Language Models. *IEEE Transactions on Visualization and Computer Graphics* (2024).

[152] Youwen Kang, Zhida Sun, Sitong Wang, Zeyu Huang, Ziming Wu, and Xiaojuan Ma. 2021. MetaMap: Supporting Visual Metaphor Ideation through Multi-Dimensional Example-Based Exploration. 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 427, 15 pages. doi:10.1145/3411764.3445325

[153] 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. doi:10.1145/3377325.3377522

[154] Andrej [@karpathy] Karpathy. 2025. My reaction is that there is an evaluation crisis. I don't really know what metrics to look at right now. [...] In absence of great comprehensive evals I tried to turn to vibe checks instead, but I now fear they are misleading and there is too much opportunity for confirmation bias, too low sample size, etc., it's just not great. TLDR my reaction is I don't really know how good these models are right now. https://x.com/karpathy/status/1896266683301659068. Accessed: 2025-04-01.

[155] Harmanpreet Kaur, Eytan Adar, Eric Gilbert, and Cliff Lampe. 2022. Sensible AI: Re-imagining interpretability and explainability using sensemaking theory. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*. 702–714.

[156] Daniel Khashabi, Gabriel Stanovsky, Jonathan Bragg, Nicholas Lourie, Jungo Kasai, Yejin Choi, Noah A Smith, and Daniel S Weld. 2021. GENIE: Toward reproducible and standardized human evaluation for text generation. *arXiv preprint arXiv:2101.06561* (2021).

[157] Jiho Kim, Woosog Chay, Hyeonji Hwang, Daeun Kyung, Hyunseung Chung, Eunbyeol Cho, Yohan Jo, and Edward Choi. 2024. DialSim: A Real-Time Simulator for Evaluating Long-Term Dialogue Understanding of Conversational Agents. *arXiv preprint arXiv:2406.13144* (2024).

[158] Minjeong Kim, Kyeongpil Kang, Deokgun Park, Jaegul Choo, and Niklas Elmqvist. 2016. Topiclens: Efficient multi-level visual topic exploration of large-scale document collections. *IEEE transactions on visualization and computer graphics* 23, 1 (2016), 151–160.

[159] Sungdong Kim, Sanghwan Bae, Jamin Shin, Soyoung Kang, Donghyun Kwak, Kang Min Yoo, and Minjoon Seo. 2023. Aligning Large Language Models through Synthetic Feedback. arXiv:2305.13735 [cs.CL]

[160] Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, et al. 2023. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*.

[161] Seungone Kim, Juyoung Suk, Ji Yong Cho, Shayne Longpre, Chaeeun Kim, Dongkeun Yoon, Guijin Son, Yejin Cho, Sheikh Shafayat, Jinheon Baek, et al. 2024. The biggen bench: A principled benchmark for fine-grained evaluation of language models with language models. *arXiv preprint arXiv:2406.05761* (2024).

[162] Seungone Kim, Ian Wu, Jinu Lee, Xiang Yue, Seongyun Lee, Mingyeong Moon, Kiril Gashteovski, Carolin Lawrence, Julia Hockenmaier, Graham Neubig, et al. 2025. Scaling Evaluation-time Compute with Reasoning Models as Process Evaluators. *arXiv preprint arXiv:2503.19877* (2025).

[163] 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.

[164] Tae Soo Kim, Matt Latzke, Jonathan Bragg, Amy X Zhang, and Joseph Chee Chang. 2023. Papeos: Augmenting research papers with talk videos. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–19.

[165] Tae Soo Kim, Heechan Lee, Yoonjoo Lee, Joseph Seering, and Juho Kim. 2025. Evalet: Evaluating Large Language Models by Fragmenting Outputs into Functions. *arXiv preprint arXiv:2509.11206* (2025).

[166] 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 1, 2023, San Francisco, CA, USA* (San Francisco, CA, USA) *(UIST '23)*. Association for Computing Machinery, New York, NY, USA, 18 pages. doi:979-8-4007-0132-0/23/10

[167] Tae Soo Kim, Yoonjoo Lee, Yoonah Park, Jiho Kim, Young-Ho Kim, and Juho Kim. 2025. CUPID: Evaluating Personalized and Contextualized Alignment of LLMs from Interactions. (2025).

[168] Tae Soo Kim, Yoonjoo Lee, Jamin Shin, Young-Ho Kim, and Juho Kim. 2024. Evallm: Interactive evaluation of large language model prompts on user-defined criteria. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–21.

[169] 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]

[170] Yoonsu Kim, Jueon Lee, Seoyoung Kim, Jaehyuk Park, and Juho Kim. 2024. Understanding users' dissatisfaction with chatgpt responses: Types, resolving tactics, and the effect of knowledge level. In *Proceedings of the 29th International Conference on Intelligent User Interfaces*. 385–404.

[171] Diederik P Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. arXiv:1312.6114 [stat.ML]

[172] Hannah Rose Kirk, Alexander Whitefield, Paul Röttger, Andrew Michael Bean, Katerina Margatina, Rafael Mosquera, Juan Manuel Ciro, Max Bartolo, Adina Williams, He He, et al. 2024. The PRISM Alignment Dataset: What Participatory, Representative and Individualised Human Feedback Reveals About the Subjective and Multicultural Alignment of Large Language Models. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[173] Jon Kleinberg and Manish Raghavan. 2021. Algorithmic monoculture and social welfare. *Proceedings of the National Academy of Sciences* 118, 22 (2021), e2018340118.

[174] 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.

[175] Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. 2024. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems* 36 (2024).

[176] Klaus Krippendorff. 2004. Reliability in content analysis: Some common misconceptions and recommendations. *Human communication research* 30, 3 (2004), 411–433.

[177] Kalpesh Krishna, Erin Bransom, Bailey Kuehl, Mohit Iyyer, Pradeep Dasigi, Arman Cohan, and Kyle Lo. 2023. LongEval: Guidelines for human evaluation of faithfulness in long-form summarization. *arXiv preprint arXiv:2301.13298* (2023).

[178] Ranjitha Kumar, Arvind Satyanarayan, Cesar Torres, Maxine Lim, Salman Ahmad, Scott R. Klemmer, and Jerry O. Talton. 2013. Webzeitgeist: Design Mining the Web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 3083–3092. https://doi.org/10.1145/2470654.2466420

[179] Ranjitha Kumar, Jerry O. Talton, Salman Ahmad, and Scott R. Klemmer. 2011. *Bricolage: Example-Based Retargeting for Web Design*. Association for Computing Machinery, New York, NY, USA, 2197–2206. https://doi.org/10.1145/1978942.1979262

[180] Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2025. Llms get lost in multi-turn conversation. *arXiv preprint arXiv:2505.06120* (2025).

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

[182] Google Chrome Labs. 2018. VisBug. Retrieved August 26, 2021 from https://visbug.web.app/

[183] Vivian Lai and Chenhao Tan. 2019. On human predictions with explanations and predictions of machine learning models: A case study on deception detection. In *Proceedings of the conference on fairness, accountability, and transparency*. 29–38.

[184] Michelle S Lam, Fred Hohman, Dominik Moritz, Jeffrey P Bigham, Kenneth Holstein, and Mary Beth Kery. 2024. AI Policy Projector: Grounding LLM Policy Design in Iterative Mapmaking. *arXiv preprint arXiv:2409.18203* (2024).

[185] Michelle S Lam, Janice Teoh, James A Landay, Jeffrey Heer, and Michael S Bernstein. 2024. Concept induction: Analyzing unstructured text with high-level concepts using lloom. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–28.

[186] Michelle S. Lam, Grace B. Young, Catherine Y. Xu, Ranjay Krishna, and Michael S. Bernstein. 2019. Eevee: Transforming Images by Bridging High-Level Goals and Low-Level Edit Operations. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI EA '19)*. Association for Computing Machinery, New York, NY, USA, 1–6. doi:10.1145/3290607.3312929

[187] Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. 2024. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787* (2024).

[188] Gierad P. Laput, Mira Dontcheva, Gregg Wilensky, Walter Chang, Aseem Agarwala, Jason Linder, and Eytan Adar. 2013. *PixelTone: A Multimodal Interface for Image Editing*. Association for Computing Machinery, New York, NY, USA, 2185–2194. https://doi.org/10.1145/2470654.2481301

[189] Brian Lee, Savil Srivastava, Ranjitha Kumar, Ronen Brafman, and Scott R. Klemmer. 2010. *Designing with Interactive Example Galleries*. Association for Computing Machinery, New York, NY, USA, 2257–2266. https://doi.org/10.1145/1753326.1753667

[190] 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

[191] 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).

[192] Seongyun Lee, Sue Hyun Park, Seungone Kim, and Minjoon Seo. 2024. Aligning to thousands of preferences via system message generalization. *arXiv preprint arXiv:2405.17977* (2024).

[193] 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.

[194] 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.

[195] 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 Questions. 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 450, 22 pages. doi:10.1145/3544548.3581369

[196] Yoonjoo Lee, Kyungjae Lee, Sunghyun Park, Dasol Hwang, Jaehyeon Kim, Hong-in Lee, and Moontae Lee. 2023. Qasa: advanced question answering on scientific articles. In *International Conference on Machine Learning*. PMLR, 19036–19052.

[197] Belinda Z Li, Alex Tamkin, Noah Goodman, and Jacob Andreas. 2023. Eliciting human preferences with language models. *arXiv preprint arXiv:2310.11589* (2023).

[198] Margaret Li, Jason Weston, and Stephen Roller. 2019. ACUTE-EVAL: Improved Dialogue Evaluation with Optimized Questions and Multi-Turn Comparisons. *arXiv preprint arXiv:1909.03087* (2019).

[199] Ruosen Li, Teerth Patel, and Xinya Du. 2023. PRD: Peer Rank and Discussion Improve Large Language Model based Evaluations. *arXiv preprint arXiv:2307.02762* (2023).

[200] Toby Jia-Jun Li, Marissa Radensky, Justin Jia, Kirielle Singarajah, Tom M. Mitchell, and Brad A. Myers. 2019. PUMICE: A Multi-Modal Agent That Learns Concepts and Conditionals from Natural Language and Demonstrations. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 577–589. doi:10.1145/3332165.3347899

[201] 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.

[202] Q Vera Liao, Daniel Gruen, and Sarah Miller. 2020. Questioning the AI: informing design practices for explainable AI user experiences. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–15.

[203] Opher Lieber, Or Sharir, Barak Lenz, and Yoav Shoham. 2021. *Jurassic-1: Technical Details And Evaluation*. Technical Report. AI21 Labs.

[204] Sarah Lim, Joshua Hibschman, Haoqi Zhang, and Eleanor O'Rourke. 2018. Ply: A Visual Web Inspector for Learning from Professional Webpages. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18)*. Association for Computing Machinery, New York, NY, USA, 991–1002. doi:10.1145/3242587.3242660

[205] Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. 2024. Wildbench: Benchmarking llms with challenging tasks from real users in the wild. *arXiv preprint arXiv:2406.04770* (2024).

[206] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*. Association for Computational Linguistics, Barcelona, Spain, 74–81. https://aclanthology.org/W04-1013

[207] Xi Victoria Lin, Chenglong Wang, Luke Zettlemoyer, and Michael D. Ernst. 2018. NL2Bash: A Corpus and Semantic Parser for Natural Language Interface to the Linux Operating System. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA), Miyazaki, Japan. https://aclanthology.org/L18-1491

[208] Yuyu Lin, Jiahao Guo, Yang Chen, Cheng Yao, and Fangtian Ying. 2020. *It Is Your Turn: Collaborative Ideation With a Co-Creative Robot through Sketch*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3313831.3376258

[209] Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent Predictor Networks for Code Generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 599–609. doi:10.18653/v1/P16-1057

[210] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What Makes Good In-Context Examples for GPT-3? *arXiv preprint arXiv:2101.06804* (2021).

[211] Michael Xieyang Liu, Advait Sarkar, Carina Negreanu, Benjamin Zorn, Jack Williams, Neil Toronto, and Andrew D. Gordon. 2023. "What It Wants Me To Say": Bridging the Abstraction Gap Between End-User Programmers and Code-Generating 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 598, 31 pages. doi:10.1145/3544548.3580817

[212] Michael Xieyang Liu, Tongshuang Wu, Tianying Chen, Franklin Mingzhe Li, Aniket Kittur, and Brad A Myers. 2024. Selenite: Scaffolding Online Sensemaking with Comprehensive Overviews Elicited from Large Language Models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–26.

[213] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-Train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Comput. Surv.* 55, 9, Article 195 (jan 2023), 35 pages. doi:10.1145/3560815

[214] 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.

[215] 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).

[216] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT Understands, Too. arXiv:2103.10385 [cs.CL]

[217] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. arXiv:2303.16634 [cs.CL]

[218] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE transactions on information theory* 28, 2 (1982), 129–137.

[219] 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

[220] 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. doi:10.1145/3490099.3511159

[221] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2024. The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery. arXiv:2408.06292 [cs.AI]

[222] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2024. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292* (2024).

[223] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786* (2021).

[224] Edward Ma. 2019. NLP Augmentation. https://github.com/makcedward/nlpaug.

[225] James MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, Vol. 5. University of California press, 281–298.

[226] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651* (2023).

[227] Chaitanya Malaviya, Joseph Chee Chang, Dan Roth, Mohit Iyyer, Mark Yatskar, and Kyle Lo. 2024. Contextualized Evaluations: Judging Language Model Responses to Underspecified Queries. *arXiv preprint arXiv:2411.07237* (2024).

[228] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, and Sayak Paul. 2022. PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods. https://github.com/huggingface/peft.

[229] Mehdi Manshadi, Daniel Gildea, and James Allen. 2013. Integrating Programming by Example and Natural Language Programming. https://www.aaai.org/ocs/index.php/AAAI/AAAI13/paper/view/6477/7230

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

[231] 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

[232] 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. doi:10.1145/3290605.3300268

[233] Leland McInnes, John Healy, Steve Astels, et al. 2017. hdbscan: Hierarchical density based clustering. *J. Open Source Softw.* 2, 11 (2017), 205.

[234] Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018).

[235] Microsoft. 2023. Announcing Microsoft Copilot, Your Everyday AI Companion. https://blogs.microsoft.com/blog/2023/09/21/announcing-microsoft-copilot-your-everyday-ai-companion/ Accessed: March 19, 2025.

[236] Midjourney. 2023. *Midjourney*. Retrieved March 28, 2023 from https://www.midjourney.com/

[237] Rada Mihalcea, Hugo Liu, and Henry Lieberman. 2006. NLP (Natural Language Processing) for NLP (Natural Language Programming). In *Computational Linguistics and Intelligent Text Processing*, Alexander Gelbukh (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 319–330.

[238] Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. *arXiv preprint arXiv:2305.14251* (2023).

[239] 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).

[240] Aditi Mishra, Utkarsh Soni, Anjana Arunkumar, Jinbin Huang, Bum Chul Kwon, and Chris Bryan. 2023. PromptAid: Prompt Exploration, Perturbation, Testing and Iteration using Visual Analytics for Large Language Models. *arXiv preprint arXiv:2304.01964* (2023).

[241] 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.

[242] Vishvak Murahari, Ameet Deshpande, Peter Clark, Tanmay Rajpurohit, Ashish Sabharwal, Karthik Narasimhan, and Ashwin Kalyan. 2023. Qualeval: Qualitative evaluation for model improvement. *arXiv preprint arXiv:2311.02807* (2023).

[243] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332* (2021).

[244] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023* (2016).

[245] Michael Nebeling and Anind K. Dey. 2016. *XDBrowser: User-Defined Cross-Device Web Page Designs*. Association for Computing Machinery, New York, NY, USA, 5494–5505. https://doi.org/10.1145/2858036.2858048

[246] Michael Nebeling, Maximilian Speicher, and Moira C. Norrie. 2013. CrowdAdapt: Enabling Crowdsourced Web Page Adaptation for Individual Viewing Conditions and Preferences. In *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (London, United Kingdom) *(EICS '13)*. Association for Computing Machinery, New York, NY, USA, 23–32. doi:10.1145/2494603.2480304

[247] Jakob Nielsen. 2023. AI: First New UI Paradigm in 60 Years. https://www.nngroup.com/articles/ai-paradigm/.

[248] 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. doi:10.1145/3173574.3174223

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

[250] OpenAI. 2022. ChatGPT. https://chat.openai.com/chat Accessed: March 19, 2025.

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

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

[253] OpenAI. 2024. Introducing OpenAI o1-preview. https://openai.com/index/introducing-openai-o1-preview/.

[254] OpenAI. 2024. Memory and New Controls for ChatGPT. https://openai.com/index/memory-and-new-controls-for-chatgpt/ Accessed: September 15, 2024.

[255] OpenAI. 2025. OpenAI o3-mini. https://openai.com/index/openai-o3-mini/ Accessed: March 19, 2025.

[256] Jonas Oppenlaender, Thanassis Tiropanis, and Simo Hosio. 2020. CrowdUI: Supporting Web Design with the Crowd. *Proc. ACM Hum.-Comput. Interact.* 4, EICS, Article 76 (June 2020), 28 pages. doi:10.1145/3394978

[257] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. arXiv:2203.02155 [cs.CL]

[258] Srishti Palani, Zijian Ding, Austin Nguyen, Andrew Chuang, Stephen MacNeil, and Steven P Dow. 2021. CoNotate: Suggesting queries based on notes promotes knowledge discovery. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–14.

[259] Srishti Palani, Yingyi Zhou, Sheldon Zhu, and Steven P Dow. 2022. InterWeave: Presenting Search Suggestions in Context Scaffolds Information Search and Synthesis. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–16.

[260] 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 on Association for Computational Linguistics* (Philadelphia, Pennsylvania) *(ACL '02)*. Association for Computational Linguistics, USA, 311–318. doi:10.3115/1073083.1073135

[261] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*. 1–22.

[262] 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* (Bend, OR, USA) *(UIST '22)*. Association for Computing Machinery, New York, NY, USA, Article 74, 18 pages. doi:10.1145/3526113.3545616

[263] Joon Sung Park, Carolyn Q Zou, Aaron Shaw, Benjamin Mako Hill, Carrie Cai, Meredith Ringel Morris, Robb Willer, Percy Liang, and Michael S Bernstein. 2024. Generative agent simulations of 1,000 people. *arXiv preprint arXiv:2411.10109* (2024).

[264] Thomas H. Park, Ankur Saxena, Swathi Jagannath, Susan Wiedenbeck, and Andrea Forte. 2013. OpenHTML: Designing a Transitional Web Editor for Novices. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems* (Paris, France) *(CHI EA '13)*. Association for Computing Machinery, New York, NY, USA, 1863–1868. doi:10.1145/2468356.2468690

[265] Carole C Perlman. 2003. Performance Assessment: Designing Appropriate Performance Tasks and Scoring Rubrics. (2003).

[266] 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. doi:10.1145/3544548.3580907

[267] 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. doi:10.1145/3544548.3580907

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

[269] Peter Pirolli and Stuart Card. 2005. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of international conference on intelligence analysis*, Vol. 5. McLean, VA, USA, 2–4.

[270] Cheng Qian, Bingxiang He, Zhong Zhuang, Jia Deng, Yujia Qin, Xin Cong, Zhong Zhang, Jie Zhou, Yankai Lin, Zhiyuan Liu, et al. 2024. Tell me more! towards implicit user intention understanding of language model driven agents. *arXiv preprint arXiv:2402.09205* (2024).

[271] Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to Code: Learning Semantic Parsers for If-This-Then-That Recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, 878–888. doi:10.3115/v1/P15-1085

[272] Napol Rachatasumrit, Gonzalo Ramos, Jina Suh, Rachel Ng, and Christopher Meek. 2021. Forsense: Accelerating online research through sensemaking integration and machine research support. In *Proceedings of the 26th International Conference on Intelligent User Interfaces*. 608–618.

[273] Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *CoRR* abs/1511.06434 (2016).

[274] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems* 36 (2023), 53728–53741.

[275] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical Text-Conditional Image Generation with CLIP Latents. (2022). https://cdn.openai.com/papers/dall-e-2.pdf

[276] Marco Tulio Ribeiro and Scott Lundberg. 2022. Adaptive testing and debugging of NLP models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 3253–3267.

[277] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. *arXiv preprint arXiv:2005.04118* (2020).

[278] Daniel Ritchie, Ankita Arvind Kejriwal, and Scott R. Klemmer. 2011. D.Tour: Style-Based Exploration of Design Example Galleries. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 165–174. doi:10.1145/2047196.2047216

[279] Samantha Robertson, Zijie J. Wang, Dominik Moritz, Mary Beth Kery, and Fred Hohman. 2023. Angler: Helping Machine Translation Practitioners Prioritize Model Improvements. 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 832, 20 pages. doi:10.1145/3544548.3580790

[280] Mark A. Robinson. 2018. Using multi-item psychometric scales for research and practice in human resource management. *Human Resource Management* 57, 3 (2018), 739–750. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/hrm.21852 doi:10.1002/hrm.21852

[281] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2021. High-Resolution Image Synthesis with Latent Diffusion Models. arXiv:2112.10752 [cs.CV]

[282] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10684–10695.

[283] Xin Rong, Shiyan Yan, Stephen Oney, Mira Dontcheva, and Eytan Adar. 2016. CodeMend: Assisting Interactive Programming with Bimodal Embedding. 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, 247–258. doi:10.1145/2984511.2984544

[284] 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.

[285] 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. doi:10.1145/3411764.3445296

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

[287] Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2021. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633* (2021).

[288] Jon Saad-Falcon, Rajan Vivek, William Berrios, Nandita Shankar Naik, Matija Franklin, Bertie Vidgen, Amanpreet Singh, Douwe Kiela, and Shikib Mehri. 2024. Lmunit: Fine-grained evaluation with natural language unit tests. *arXiv preprint arXiv:2412.13091* (2024).

[289] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. 2022. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems* 35 (2022), 36479–36494.

[290] Alireza Salemi, Surya Kallumadi, and Hamed Zamani. 2024. Optimization methods for personalizing large language models through retrieval augmentation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 752–762.

[291] Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. 2023. Lamp: When large language models meet personalization. *arXiv preprint arXiv:2304.11406* (2023).

[292] Shibani Santurkar, Esin Durmus, Faisal Ladhak, Cinoo Lee, Percy Liang, and Tatsunori Hashimoto. 2023. Whose opinions do language models reflect? *arXiv preprint arXiv:2303.17548* (2023).

[293] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. BLOOM: A 176b-Parameter Open-Access Multilingual Language Model. *arXiv preprint arXiv:2211.05100* (2022).

[294] Viktor Schlegel, Benedikt Lang, Siegfried Handschuh, and André Freitas. 2019. Vajra: Step-by-Step Programming with Natural Language. In *Proceedings of the 24th International Conference on Intelligent User Interfaces* (Marina del Ray, California) *(IUI '19)*. Association for Computing Machinery, New York, NY, USA, 30–39. doi:10.1145/3301275.3302267

[295] Donald A Schön. 2017. *The reflective practitioner: How professionals think in action*. Routledge.

[296] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[297] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 86–96. doi:10.18653/v1/P16-1009

[298] Lior Shani, Aviv Rosenberg, Asaf Cassel, Oran Lang, Daniele Calandriello, Avital Zipori, Hila Noga, Orgad Keller, Bilal Piot, Idan Szpektor, et al. 2024. Multi-turn reinforcement learning with preference human feedback. *Advances in Neural Information Processing Systems* 37 (2024), 118953–118993.

[299] Shreya Shankar, JD Zamfirescu-Pereira, Björn Hartmann, Aditya Parameswaran, and Ian Arawjo. 2024. Who validates the validators? aligning llm-assisted evaluation of llm outputs with human preferences. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–14.

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

[301] Hua Shen, Tiffany Knearem, Reshmi Ghosh, Kenan Alkiek, Kundan Krishna, Yachuan Liu, Ziqiao Ma, Savvas Petridis, Yi-Hao Peng, Li Qiwei, et al. 2024. Towards bidirectional human-ai alignment: A systematic review for clarifications, framework, and future directions. *arXiv preprint arXiv:2406.09264* (2024).

[302] Taiwei Shi, Zhuoer Wang, Longqi Yang, Ying-Chun Lin, Zexue He, Mengting Wan, Pei Zhou, Sujay Jauhar, Sihao Chen, Shan Xia, et al. 2024. Wildfeedback: Aligning llms with in-situ user interactions and feedback. *arXiv preprint arXiv:2408.15549* (2024).

[303] 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

[304] Hyungyu Shin, Eun-Young Ko, Joseph Jay Williams, and Juho Kim. 2018. *Understanding the Effect of In-Video Prompting on Learners and Instructors*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173893

[305] Ben Shneiderman. 2003. The eyes have it: A task by data type taxonomy for information visualizations. In *The craft of information visualization*. Elsevier, 364–371.

[306] Herbert A Simon and Allen Newell. 1971. Human problem solving: The state of the theory in 1970. *American psychologist* 26, 2 (1971), 145.

[307] Ian Simon, Dan Morris, and Sumit Basu. 2008. MySong: Automatic Accompaniment Generation for Vocal Melodies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Florence, Italy) *(CHI '08)*. Association for Computing Machinery, New York, NY, USA, 725–734. doi:10.1145/1357054.1357169

[308] 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). doi:10.1145/3511599 Just Accepted.

[309] Venkatesh Sivaraman, Zexuan Li, and Adam Perer. 2025. Divisi: Interactive Search and Visualization for Scalable Exploratory Subgroup Analysis. *arXiv preprint arXiv:2502.10537* (2025).

[310] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, Elton Zheng, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. 2022. Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model. *CoRR* abs/2201.11990 (2022). arXiv:2201.11990 https://arxiv.org/abs/2201.11990

[311] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314* (2024).

[312] Kihoon Son, DaEun Choi, Tae Soo Kim, Young-Ho Kim, and Juho Kim. 2024. Genquery: Supporting expressive visual search with generative models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–19.

[313] Taylor Sorensen, Jared Moore, Jillian Fisher, Mitchell Gordon, Niloofar Mireshghallah, Christopher Michael Rytting, Andre Ye, Liwei Jiang, Ximing Lu, Nouha Dziri, et al. 2024. A roadmap to pluralistic alignment. *arXiv preprint arXiv:2402.05070* (2024).

[314] Arjun Srinivasan, Mira Dontcheva, Eytan Adar, and Seth Walker. 2019. Discovering Natural Language Commands in Multimodal Interfaces. In *Proceedings of the 24th International Conference on Intelligent User Interfaces* (Marina del Ray, California) *(IUI '19)*. Association for Computing Machinery, New York, NY, USA, 661–672. doi:10.1145/3301275.3302292

[315] Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, et al. 2025. PaperBench: Evaluating AI's Ability to Replicate AI Research. *arXiv preprint arXiv:2504.01848* (2025).

[316] 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. doi:10.48550/ARXIV.2208.07852

[317] Hendrik Strobelt, Albert Webson, Victor Sanh, Benjamin Hoover, Johanna Beyer, Hanspeter Pfister, and Alexander M. Rush. 2023. Interactive and Visual Prompt Engineering for Ad-hoc Task Adaptation with Large Language Models. *IEEE Transactions on Visualization and Computer Graphics* 29, 1 (2023), 1146–1156. doi:10.1109/TVCG.2022.3209479

[318] Hari Subramonyam, Roy Pea, Christopher Pondoc, Maneesh Agrawala, and Colleen Seifert. 2024. Bridging the Gulf of Envisioning: Cognitive Challenges in Prompt Based Interactions with LLMs. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–19.

[319] Sangho Suh, Meng Chen, Bryan Min, Toby Jia-Jun Li, and Haijun Xia. 2024. Luminate: Structured generation and exploration of design space with large language models for human-ai co-creation. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–26.

[320] Sangho Suh, Bryan Min, Srishti Palani, and Haijun Xia. 2023. Sensecape: Enabling multilevel exploration and sensemaking with large language models. In *Proceedings of the 36th annual ACM symposium on user interface software and technology*. 1–18.

[321] 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. doi:10.1145/3490099.3511119

[322] 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

[323] Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. 2023. Principle-Driven Self-Alignment of Language Models from Scratch with Minimal Human Supervision. arXiv:2305.03047 [cs.LG]

[324] 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. doi:10.1145/3490099.3511160

[325] 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. doi:10.18653/v1/2021.eacl-demos.29

[326] Amanda Swearngin, Amy J. Ko, and James Fogarty. 2017. *Genie: Input Retargeting on the Web through Command Reverse Engineering*. Association for Computing Machinery, New York, NY, USA, 4703–4714. https://doi.org/10.1145/3025453.3025506

[327] 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

[328] Annalisa Szymanski, Simret Araya Gebreegziabher, Oghenemaro Anuyah, Ronald A Metoyer, and Toby Jia-Jun Li. 2024. Comparing Criteria Development Across Domain Experts, Lay Users, and Models in Large Language Model Evaluation. *arXiv preprint arXiv:2410.02054* (2024).

[329] Alex Tamkin, Miles McCain, Kunal Handa, Esin Durmus, Liane Lovitt, Ankur Rathi, Saffron Huang, Alfred Mountfield, Jerry Hong, Stuart Ritchie, et al. 2024. Clio: Privacy-Preserving Insights into Real-World AI Use. *arXiv preprint arXiv:2412.13678* (2024).

[330] Kesler Tanner, Naomi Johnson, and James A. Landay. 2019. *Poirot: A Web Inspector for Designers*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300758

[331] OpenThoughts Team. 2025. Open Thoughts. https://open-thoughts.ai.

[332] Michael Terry, Chinmay Kulkarni, Martin Wattenberg, Lucas Dixon, and Meredith Ringel Morris. 2024. Interactive AI Alignment: Specification, Process, and Evaluation Alignment. arXiv:2311.00710 [cs.HC] https://arxiv.org/abs/2311.00710

[333] David R Thomas. 2006. A general inductive approach for analyzing qualitative evaluation data. *American journal of evaluation* 27, 2 (2006), 237–246.

[334] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Kathleen S. Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed Chi, and Quoc Le. 2022. LaMDA: Language Models for Dialog Applications. *CoRR* abs/2201.08239 (2022). arXiv:2201.08239 `https://arxiv.org/abs/2201.08239`

[335] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971 [cs.CL]

[336] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs.CL]

[337] Yu-Min Tseng, Yu-Chao Huang, Teng-Yun Hsiao, Yu-Ching Hsu, Jia-Yin Foo, Chao-Wei Huang, and Yun-Nung Chen. 2024. Two tales of persona in llms: A survey of role-playing and personalization. *arXiv preprint arXiv:2406.01171* (2024).

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

[339] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR* abs/1706.03762 (2017). arXiv:1706.03762 `http://arxiv.org/abs/1706.03762`

[340] 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.

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

[342] Danding Wang, Qian Yang, Ashraf Abdul, and Brian Y Lim. 2019. Designing theory-driven user-centric explainable AI. In *Proceedings of the 2019 CHI conference on human factors in computing systems*. 1–15.

[343] Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926* (2023).

[344] Ruiyi Wang, Haofei Yu, Wenxin Zhang, Zhengyang Qi, Maarten Sap, Graham Neubig, Yonatan Bisk, and Hao Zhu. 2024. SOTOPIA-π: Interactive Learning of Socially Intelligent Language Agents. *arXiv preprint arXiv:2403.08715* (2024).

[345] Tiannan Wang, Meiling Tao, Ruoyu Fang, Huilin Wang, Shuai Wang, Yuchen Eleanor Jiang, and Wangchunshu Zhou. 2024. AI PERSONA: Towards Life-long Personalization of LLMs. *arXiv preprint arXiv:2412.13103* (2024).

[346] Tianlu Wang, Ping Yu, Xiaoqing Ellen Tan, Sean O'Brien, Ramakanth Pasunuru, Jane Dwivedi-Yu, Olga Golovneva, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023. Shepherd: A Critic for Language Model Generation. arXiv:2308.04592 [cs.CL]

[347] Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2024. Augmenting language models with long-term memory. *Advances in Neural Information Processing Systems* 36 (2024).

[348] 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).

[349] Zekun Wang, Ge Zhang, Kexin Yang, Ning Shi, Wangchunshu Zhou, Shaochun Hao, Guangzheng Xiong, Yizhi Li, Mong Yuan Sim, Xiuying Chen, et al. 2023. Interactive natural language processing. *arXiv preprint arXiv:2305.13246* (2023).

[350] Zijie J Wang, Fred Hohman, and Duen Horng Chau. 2023. Wizmap: Scalable interactive visualization for exploring large machine learning embeddings. *arXiv preprint arXiv:2306.09328* (2023).

[351] Caleb Warren, A Peter McGraw, and Leaf Van Boven. 2011. Values and preferences: defining preference construction. *Wiley Interdisciplinary Reviews: Cognitive Science* 2, 2 (2011), 193–205.

[352] 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. doi:10.1145/3025171.3025194

[353] Jing Wei, Sungdong Kim, Hyunhoon Jung, and Young-Ho Kim. 2023. Leveraging Large Language Models to Power Chatbots for Collecting User Self-Reported Data. arXiv:2301.05843 [cs.HC]

[354] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.

[355] Sebastian Weigelt, Vanessa Steurer, Tobias Hey, and Walter F. Tichy. 2020. Programming in Natural Language with fuSE: Synthesizing Methods from Spoken Utterances Using Deep Natural Language Understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 4280–4295. doi:10.18653/v1/2020.acl-main.395

[356] 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. doi:10.20380/GI2019.19

[357] 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.

[358] Justin D Weisz, Jessica He, Michael Muller, Gabriela Hoefer, Rachel Miles, and Werner Geyer. 2024. Design Principles for Generative AI Applications. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–22.

[359] 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).

[360] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. 2020. The What-If Tool: Interactive Probing of Machine Learning Models. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2020), 56–65. doi:10.1109/TVCG.2019.2934619

[361] 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

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

[363] Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2024. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813* (2024).

[364] Shujin Wu, May Fung, Cheng Qian, Jeonghwan Kim, Dilek Hakkani-Tur, and Heng Ji. 2024. Aligning LLMs with Individual Preferences via Interaction. *arXiv preprint arXiv:2410.03642* (2024).

[365] Shirley Wu, Michel Galley, Baolin Peng, Hao Cheng, Gavin Li, Yao Dou, Weixin Cai, James Zou, Jure Leskovec, and Jianfeng Gao. 2025. Collabllm: From passive responders to active collaborators. *arXiv preprint arXiv:2502.00640* (2025).

[366] Sherry Wu, Hua Shen, Daniel S Weld, Jeffrey Heer, and Marco Tulio Ribeiro. 2023. ScatterShot: Interactive In-Context Example Curation for Text Transformation. In *Proceedings of the 28th International Conference on Intelligent User Interfaces* (Sydney, NSW, Australia) *(IUI '23)*. Association for Computing Machinery, New York, NY, USA, 353–367. doi:10.1145/3581641.3584059

[367] 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).

[368] Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. 2019. Errudite: Scalable, Reproducible, and Testable Error Analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 747–763. doi:10.18653/v1/P19-1073

[369] Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel S Weld. 2021. Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. *arXiv preprint arXiv:2101.00288* (2021).

[370] 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

[371] Tongshuang Wu, Michael Terry, and Carrie Jun Cai. 2022. AI Chains: Transparent and Controllable Human-AI Interaction by Chaining Large Language Model Prompts. 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 385, 22 pages. doi:10.1145/3491102.3517582

[372] Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. 2023. Fine-Grained Human Feedback Gives Better Rewards for Language Model Training. *arXiv preprint arXiv:2306.01693* (2023).

[373] 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

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

[375] 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. doi:10.1145/2858036.2858075

[376] 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. doi:10.1145/3025453.3025554

[377] 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

[378] Ziang Xiao, Susu Zhang, Vivian Lai, and Q. Vera Liao. 2023. Evaluating NLG Evaluation Metrics: A Measurement Theory Perspective. arXiv:2305.14889 [cs.CL]

[379] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115* (2024).

[380] 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

[381] Fan Yang, Zheng Chen, Ziyan Jiang, Eunah Cho, Xiaojiang Huang, and Yanbin Lu. 2023. Palr: Personalization aware llms for recommendation. *arXiv preprint arXiv:2305.07622* (2023).

[382] Saelyne Yang, Sangkyung Kwak, Tae Soo Kim, and Juho Kim. 2022. Improving Video Interfaces by Presenting Informational Units of Videos. (2022).

[383] Koji Yatani, Michael Novati, Andrew Trusty, and Khai N Truong. 2011. Review spotlight: a user interface for summarizing user-generated reviews using adjective-noun word pairs. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 1541–1550.

[384] Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. 2023. FLASK: Fine-grained Language Model Evaluation based on Alignment Skill Sets. *arXiv preprint arXiv:2307.10928* (2023).

[385] Pengcheng Yin and Graham Neubig. 2017. A Syntactic Neural Model for General-Purpose Code Generation. In *The 55th Annual Meeting of the Association for Computational Linguistics (ACL)*. Vancouver, Canada. https://arxiv.org/abs/1704.01696

[386] Ann Yuan, Andy Coenen, Emily Reif, and Daphne Ippolito. 2022. Wordcraft: story writing with large language models. In *27th International Conference on Intelligent User Interfaces*. 841–852.

[387] 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

[388] 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. doi:10.1145/2702123.2702398

[389] J.D. Zamfirescu-Pereira, Heather Wei, Amy Xiao, Kitty Gu, Grace Jung, Matthew G Lee, Bjoern Hartmann, and Qian Yang. 2023. Herding AI Cats: Lessons from Designing a Chatbot by Prompting GPT-3. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference* (Pittsburgh, PA, USA) *(DIS '23)*. Association for Computing Machinery, New York, NY, USA, 2206–2220. doi:10.1145/3563657.3596138

[390] J.D. Zamfirescu-Pereira, Richmond Y. 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* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 437, 21 pages. doi:10.1145/3544548.3581388

[391] Marc Zao-Sanders. 2024. How people are really using GenAI. *Harvard Business Review* (2024).

[392] Wojciech Zaremba, Greg Brockman, and OpenAI. 2021. OpenAI Codex. Retrieved August 28, 2021 from https://openai.com/blog/openai-codex/

[393] Zhiyuan Zeng, Yizhong Wang, Hannaneh Hajishirzi, and Pang Wei Koh. 2025. EvalTree: Profiling Language Model Weaknesses via Hierarchical Capability Trees. *arXiv preprint arXiv:2503.08893* (2025).

[394] 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

[395] 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. doi:10.1145/3411764.3445714

[396] Jingyue Zhang and Ian Arawjo. 2024. ChainBuddy: An AI Agent System for Generating LLM Pipelines. *arXiv preprint arXiv:2409.13588* (2024).

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

[398] Michael JQ Zhang, W Bradley Knox, and Eunsol Choi. 2024. Modeling future conversation turns to teach llms to ask clarifying questions. *arXiv preprint arXiv:2410.13788* (2024).

[399] Renrui Zhang, Jiaming Han, Chris Liu, Peng Gao, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, and Yu Qiao. 2023. LLaMA-Adapter: Efficient Fine-tuning of Language Models with Zero-init Attention. arXiv:2303.16199 [cs.CV]

[400] Xiong Zhang and Philip J. Guo. 2018. Fusion: Opportunistic Web Prototyping with UI Mashups. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18)*. Association for Computing Machinery, New York, NY, USA, 951–962. doi:10.1145/3242587.3242632

[401] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *NIPS*. 649–657. http://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification

[402] 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).

[403] 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

[404] Siyan Zhao, Mingyi Hong, Yang Liu, Devamanyu Hazarika, and Kaixiang Lin. 2025. Do LLMs Recognize Your Preferences? Evaluating Personalized Preference Following in LLMs. *arXiv preprint arXiv:2502.09597* (2025).

[405] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate Before Use: Improving Few-shot Performance of Language Models. In *The 38th International Conference on Machine Learning (ICML '21)*. 12697–12706. http://proceedings.mlr.press/v139/zhao21c.html

[406] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. arXiv:2306.05685 [cs.CL]

[407] Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. 2024. *Open-Sora: Democratizing Efficient Video Production for All*. https://github.com/hpcaitech/Open-Sora

[408] Mingyuan Zhong, Gang Li, and Yang Li. 2021. Spacewalker: Rapid UI Design Exploration Using Lightweight Markup Enhancement and Crowd Genetic Programming. 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 315, 11 pages. doi:10.1145/3411764.3445326

[409] Ming Zhong, Yang Liu, Da Yin, Yuning Mao, Yizhu Jiao, Pengfei Liu, Chenguang Zhu, Heng Ji, and Jiawei Han. 2022. Towards a unified multi-dimensional evaluator for text generation. *arXiv preprint arXiv:2210.07197* (2022).

[410] Ming Zhong, Yang Liu, Da Yin, Yuning Mao, Yizhu Jiao, Pengfei Liu, Chenguang Zhu, Heng Ji, and Jiawei Han. 2022. Towards a unified multi-dimensional evaluator for text generation. *arXiv preprint arXiv:2210.07197* (2022).

[411] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. arXiv:1709.00103 [cs.CL]

[412] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 19724–19731.

[413] Zexuan Zhong, Tao Lei, and Danqi Chen. 2022. Training language models with memory augmentation. *arXiv preprint arXiv:2205.12674* (2022).

[414] Xuhui Zhou, Hao Zhu, Leena Mathur, Ruohong Zhang, Haofei Yu, Zhengyang Qi, Louis-Philippe Morency, Yonatan Bisk, Daniel Fried, Graham Neubig, et al. 2023. Sotopia: Interactive evaluation for social intelligence in language agents. *arXiv preprint arXiv:2310.11667* (2023).

[415] 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

[416] 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.

Acknowledgment

This thesis would not have been possible without the guidance, support, and companionship of many people who have shaped, guided, and supported me.

- **My advisor, Juho Kim**, for teaching me everything I know about being a researcher—and for continuing to teach me still. From my undergraduate years through my MS and now PhD, you have been patient, open, and unwavering in your support. Thank you for believing in me at every stage of this journey, and for always inspiring me to be not just a better researcher, but also a more compassionate person.
- To my amazing **thesis committee**—Tak Yeon Lee, Kimin Lee, Gagan Bansal, and Haijun Xia—thank you for all the thoughtful feedback, encouragement and advice, and guidance throughout this process.
- **All the people I have the pleasure of calling my mentors—Ray Hong, Tesh Goyal, Minsuk Chang, Joseph Chee Chang, Amy X. Zhang, Jonathan Bragg, and Young-Ho Kim**—thank you for teaching me so much about this field. I have learned different and valuable things from each of you, and in many ways, I feel like a mosaic of all your experiences and expertise.
- **All KIXLAB members**, for being my community and family for all this time. I especially thank **Yoonjoo**, my closest collaborator and my second advisor in spirit. Thank you for working with me on so many fun projects and for helping me find my interest in HCI+NLP. The fact that you helped me find the framing to my thesis shows how much you have defined my PhD journey. Thank you for all the encouragement, support, and wisdom you've shared with me. **Kihoon**, a true friend and close collaborator—thank you for all the fun Pokémon card adventures, all the 국물제육s and 순대국밥s, and for inviting me on all the exciting projects we've worked on together. You brought so much joy to my PhD. **Yoonseo and Saelyne**, from undergraduate interns to now—all of us graduating with our PhDs soon. You have been a constant presence and support in my life, and I'm very excited to see how all of us will turn out. **Hyunwoo, Eunyoung, Hyungyu, and Seoyoung**, the pillars of KIXLAB. My PhD was only possible thanks to all the work you have done to build KIXLAB into what it is today. **DaEun, Yoonsu, and Bekzat**, the future of the lab—good luck with everything ahead. Thank you for all the good times. I'm sorry for not being a better *sunbae*.
- **All the collaborators and interns** I've had the pleasure of working with—John Chung, Matt Latzke, Jamin Shin, Heechan Lee, Joseph Seering, Jaesang Yu, Jeongyeon Kim, Seungsu Kim, Seungju Kim, Junho Myung, Hyeonjeong Ha, Yoonah Park, Yuewen Yang, Jiho Kim, Arghya Sarkar, and Daho Jung—thank you so much for all your help, despite the ups and downs. I have learned and grown so much thanks to all of you.
- **Sej, Juli, and Kwoos**, for your friendship throughout all these years and all the bad jokes, memes, and drinks that we have shared. Thank you for waiting for me and always being willing to hang out, despite me constantly *pretending* to be busy.
- **우리 엄마, 아빠**, 항상 저를 믿어주시고 제 인생의 모든 순간마다 응원해 주셔서 진심으로 감사합니다. 엄마, 아빠의 변함없는 사랑 덕분에 어떠한 어려움과 도전도 이겨낼 수 있었습니다. 언제나 저의 가장 큰 힘이 되어주셔서 감사드리며, 많이 사랑합니다. **A mi brother Sam**, mi primer role model. Saber que siempre puedo contar con vos me da la fuerza para seguir adelante.