Schemex: Interactive Structural Abstraction from Examples with Contrastive Refinement

Sitong Wang
Columbia University
New York, NY, USA
sw3504@columbia.edu

Samia Menon
Columbia University
New York, NY, USA
sm4788@columbia.edu

Dingzeyu Li
Adobe Research
Seattle, WA, USA
dinli@adobe.com

Xiaojuan Ma
Hong Kong University of Science and Technology
Hong Kong, China
mxj@cse.ust.hk

Richard Zemel
Columbia University
New York, NY, USA
zemel@cs.columbia.edu

Lydia B. Chilton
Columbia University
New York, NY, USA
chilton@cs.columbia.edu

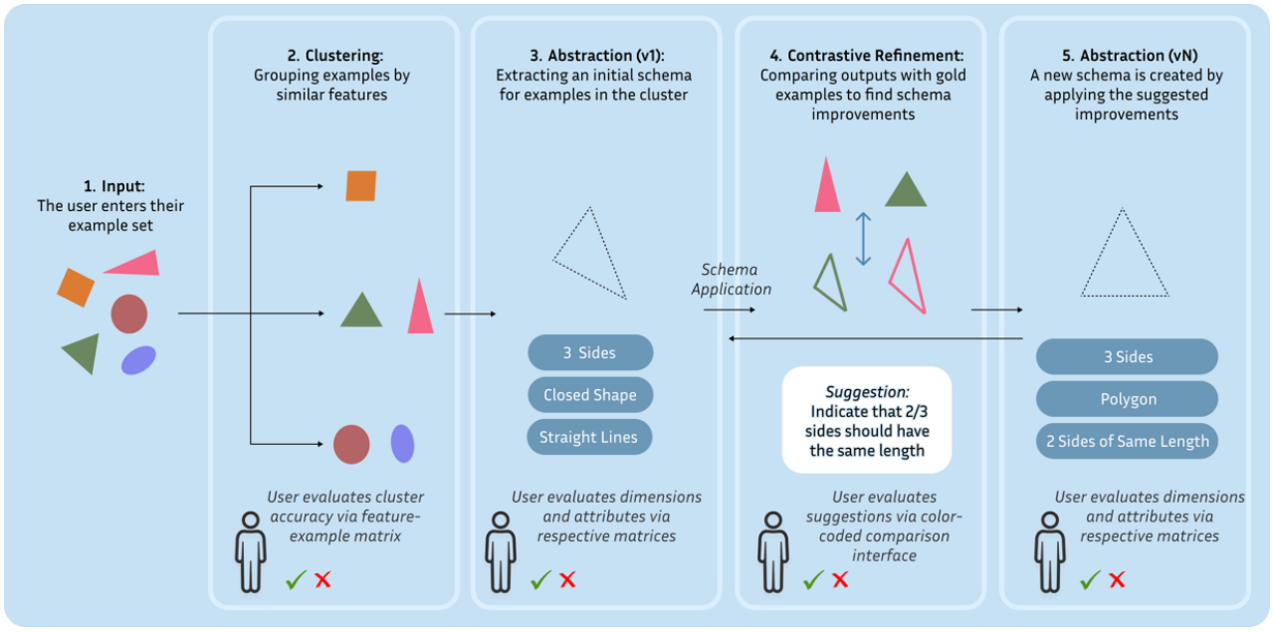

Figure 1: Schemex is an interactive, AI-powered visual workflow that helps users induce schemas from examples. It comprises three main stages: clustering, abstraction, and contrastive refinement.

ABSTRACT

Each type of creative or communicative work is underpinned by an implicit structure. People learn these structures from examples—a process known in cognitive science as schema induction. However, inducing schemas is challenging, as structural patterns are often obscured by surface-level variation. We present Schemex, an interactive visual workflow that scaffolds schema induction through clustering, abstraction, and contrastive refinement. Schemex supports users through visual representations and interactive exploration that connect abstract structures to concrete examples, promoting transparency, adaptability, and effective human-AI collaboration. In our user study, participants reported significantly greater insight and confidence in the schemas developed with Schemex compared to those created using a baseline of an AI reasoning model. We conclude by discussing the broader implications of structural abstraction and contrastive refinement across domains.

CCS CONCEPTS

• **Human-centered computing** → **Interactive systems and tools**.

KEYWORDS

generative AI, structural abstraction, contrastive refinement, schema induction, structure discovery

1 INTRODUCTION

All creative or communicative works are underpinned by structural frameworks. When people undertake tasks such as writing a research abstract, designing a slide deck, or making a product demo, they are rarely starting from a blank slate. Instead, they draw on an internalized understanding of how such artifacts are typically organized—what components are expected, how they are

sequenced, and what makes the result coherent and effective. This structural knowledge, while often tacit rather than explicit, plays a critical role in both interpreting existing examples and producing new ones. By examining multiple instances of a genre or task, people begin to identify recurring patterns—mental templates that organize content and guide its construction. The clearer and more explicit this structure becomes, the better it supports purposeful, flexible creation rather than mere surface-level imitation. In cognitive science, such internalized structural representations are known as **schemas**: abstract models that capture the essential components of a concept and the relationships among them.

Schema induction is the process of discovering the abstract structure from unstructured examples. Experts often create schemas implicitly from years of analyzing and creating examples. However, schemas can also be made explicit—these explicit schemas (also known as design patterns) are extremely helpful to novices in structuring and guiding their creative process. The cognitive process of schema induction broadly resembles the iterative process of sensemaking, incorporating both inductive and deductive approaches. One must iteratively verify that the abstractions and structures induced from the examples actually fit those examples, and when discrepancies arise, deduce how to reconcile them. For example, from looking at examples of HCI papers, one might learn that the first sentence is typically broad motivation, meant to show why the problem or areas the paper addresses are important. By looking at multiple examples, the recurring patterns—and variations within them, can be deduced.

However, inducing schemas from examples is challenging because of the vast amount of ambiguity in identifying hidden structures. First, structural patterns are often obscured by surface variation, as examples use diverse language and achieve similar goals by different means. For example, motivation sentences in abstracts all motivate different topics and in different ways using different words, but they all achieve the same goal. Second, examples within a set may adhere to different underlying schemas. For example, a system paper and a study paper contain different elements—while their schemas are related, they are distinct. Trying to generate a schema that explains both will likely be overly general. Third, schemas are difficult to evaluate—often the abstractions and structures in a schema can only truly be evaluated when they are applied to multiple examples to test the "fit" and generalizability. Lastly, schemas must balance between specificity and generalizability: overly abstract schemas offer insufficient guidance, while excessively specific ones transfer poorly across contexts.

To address these challenges, we frame schema induction as an interactive process grounded in real-world examples and guided by AI-assisted pattern discovery. Drawing on recent advances in AI reasoning—such as DeepSeek R1's "slow thinking" capabilities [22]—we explore how AI can scaffold the core cognitive steps of clustering, abstraction, and contrastive refinement.

- Clustering: Starting from a set of examples, the AI identifies latent groupings based on structural similarities and divides examples into groups.
- Abstraction: Within each cluster, it infers underlying dimensions and derives both overall and dimension-specific attributes.
- Contrastive Refinement: The current schema is used to generate outputs, which are then compared to real ("gold") examples and iterate the schema. To guard against overfitting, unseen validation examples are also included.

Throughout the process, humans play a central role as reflective evaluators—they make critical judgments about coherence and utility, direct the exploration process, and retain decision agency over when to advance or refine the schema. This approach grounds schema induction in concrete examples, promoting transparency, adaptability, and effective human-AI collaboration.

To operationalize this approach, we developed Schemex, an interactive visual workflow for schema induction. Schemex scaffolds interactive structural abstraction and constrastive refinement through AI-assisted reasoning and visualizations that keep users grounded in real examples. Users initiate the process by specifying a goal (e.g., "Write HCI paper abstracts") and providing a set of examples (e.g., CHI best paper abstracts). These examples can be multimodal; Schemex employs models such as GPT-4V and Whisper to convert images or videos into structured textual descriptions. The system clusters examples based on structural similarity and displays them as interactive visual nodes. Users can explore these clusters to inspect shared features and example-level mappings, validating the coherence of the clustering. They can then select a cluster of interest (e.g., empirical studies) and prompt the system to extract key structural dimensions (e.g., Motivation, Method, Findings). A visualized matrix allows users to examine how strongly each example reflects each dimension, supported by in-line explanations and citations. Users guide the next step by requesting inferences of dimension-level attributes (e.g., "Challenge Statement" under Motivation) and overall attributes (e.g., tone, length, or style). They evaluate the coherence of these attributes across examples and explore implementation details through contextual snippets. Users then apply the derived schema: given an input (e.g., a paper title), the system generates content guided by the schema. Schemex enables users to compare generated outputs and original examples via a color-coded dimensional analysis interface. Differences are highlighted, and the system suggests schema refinements that can be incorporated in the iteration. Validation cases not used in earlier steps are also provided, allowing further testing of schema robustness. This iterative cycle—observe, apply, compare, refine—continues until users arrive at a coherent, actionable schema tailored to their goal. In our user study, participants reported significantly greater insight and confidence in the schemas developed with Schemex compared to those created with the baseline condition, OpenAI o1-pro. Moreover, both schemas and outputs generated with Schemex were rated significantly higher in quality and depth by expert evaluators compared to those produced using the baseline condition.

The paper makes the following contributions:

- A structural abstraction technique that leverages clustering and pattern analysis to infer dimensions and attributes from example sets, including multimodal artifacts;
- A contrastive refinement method that supports structured generation tasks by comparing AI-generated and real examples to guide schema evolution;

- An interactive visual workflow that links inferred schema elements to source examples, enabling interpretable and revisable schema induction;
- An empirical evaluation showing that schemas and outputs produced with Schemex are rated significantly higher in quality and example alignment than those from a strong AI baseline (o1-pro).

2 RELATED WORK

2.1 Schema Induction and Cognitive Foundations

Humans can infer schemas from concrete experiences. Cognitive science theories of analogy and structural alignment suggest that comparing multiple examples enables people to abstract their common relational structure [18, 20]. In classic analogical problem-solving studies, Gick and Holyoak [20] demonstrated that examining two analogs can lead to the induction of a schema—a generalized solution framework that preserves shared relations while ignoring surface differences. Such analogical schemas support the transfer of knowledge to novel problems. These cognitive theories provide a foundation for the idea that people can induce schemas or conceptual models by observing recurring patterns across multiple instances. An effective schema abstracts away surface details to highlight underlying structure. For example, prior work on schema induction [49] mentions that, to capture the essence of an invention, it is more useful to focus on its purpose (e.g., detaching components) and mechanism (e.g., using comb-like features), rather than vague notions (e.g., "making things easy") or overly specific traits (e.g., color). However, striking the right balance between specificity and generalizability remains a challenge in schema induction.

Computational models in AI and cognitive science have sought to replicate this human capacity for structure learning. Kemp and Tenenbaum's hierarchical Bayesian model identifies the most appropriate structural form (such as trees, rings, or clusters) from a predefined space to explain a dataset [26]. It treats structure induction as a model selection problem across graph-based representations and has successfully uncovered latent structures in domains ranging from biology to social networks. Other models emphasize the learning of relational schemas. For example, DORA, a neural-symbolic model, discovers new relational concepts through analogy and represents them as explicit predicates or schema-like structures [10]. However, such models often lack grounding in real-world artifacts, limiting their practical applicability.

Together, these cognitive and computational foundations suggest that inferring schemas through analogical generalization and structure induction is both a core human strategy and a possible computational goal. Schemex builds on this foundation by enabling users to perform schema induction interactively—leveraging human pattern recognition alongside algorithmic assistance to extract structural abstractions from example collections.

2.2 Approaches of Interactive Structural Abstraction and Contrastive Refinement

Interactive systems increasingly support users in forming abstractions from complex data by integrating direct manipulation with automated inference. Early work on mixed-initiative clustering [23] demonstrated how adaptive grouping can incorporate user feedback to resolve tensions between statistical similarity and conceptual relevance. Visual interfaces like those used in Cyclone [11] and semantic interaction systems [13] enable users to reposition, merge, or split clusters, guiding the underlying models toward more meaningful structural representations. These systems treat abstraction as a dialogue between human and machine, where user intentions are surfaced through interactive spatial cues.

A complementary technique for refining abstractions is contrastive refinement—clarifying boundaries by comparing positive, negative, or near-miss examples. MOCHA [17] illustrates this by generating counterexamples that expose the limits of a user's current concept, prompting structural correction and alignment. Flash Fill [21] supports refinement by allowing users to correct inferred transformation rules using contrasting input-output examples. These systems show how surfacing edge cases and exceptions can accelerate convergence toward accurate and generalizable models.

Schemex builds on these traditions of mixed-initiative abstraction and contrastive refinement but targets multimodal design examples and schema-level generalization. Schemex treats structure learning as a collaborative, iterative process. It specifically supports users in moving from scattered examples to coherent, named schemas through clustering, structural abstraction, and contrastive refinement. This workflow makes schema induction more interpretable and actionable for downstream design and authoring tasks.

2.3 HCI Systems for Structure Discovery and Sensemaking

Schemex builds on a growing body of HCI research that supports structure discovery and sensemaking, particularly in design and multimodal content domains. In design, many systems aim to surface reusable patterns by mining common structures at scale. Prior work has emphasized documenting reusable solutions in structured schema formats (e.g., problem, context, solution) [29]. For example, Webzeitgeist mined common UI structures from thousands of websites [30], and other systems have identified interface design patterns across mobile apps through large-scale clustering [4]. Recent systems such as DesignWeaver [43] and Luminate [42] help users explore the design space of AI-generated variations by revealing latent design dimensions and attributes.

Structure discovery is increasingly applied to multimodal content, where visual and textual alignment reveals deeper task structures. For example, RecipeScape [5] supports this by clustering hundreds of recipes for a given dish based on procedural similarity, helping culinary professionals explore variation and identify recurring techniques at scale. Hierarchical tutorial generators [44] break down videos into stepwise instructional schemas, while VideoMix [48] aggregates how-to content into coherent workflows.

Moreover, sensemaking tools support users in organizing data, sharing insights, and externalizing evolving structures [19]. Jigsaw [41] helps users analyze document relationships through interactive visualizations, and Selenite [33] provides overviews of complex corpora using LLM assistance. To assist qualitative analysis, LLooM [31] offers a text-based approach to concept induction by enabling users to collaboratively cluster and label high-level

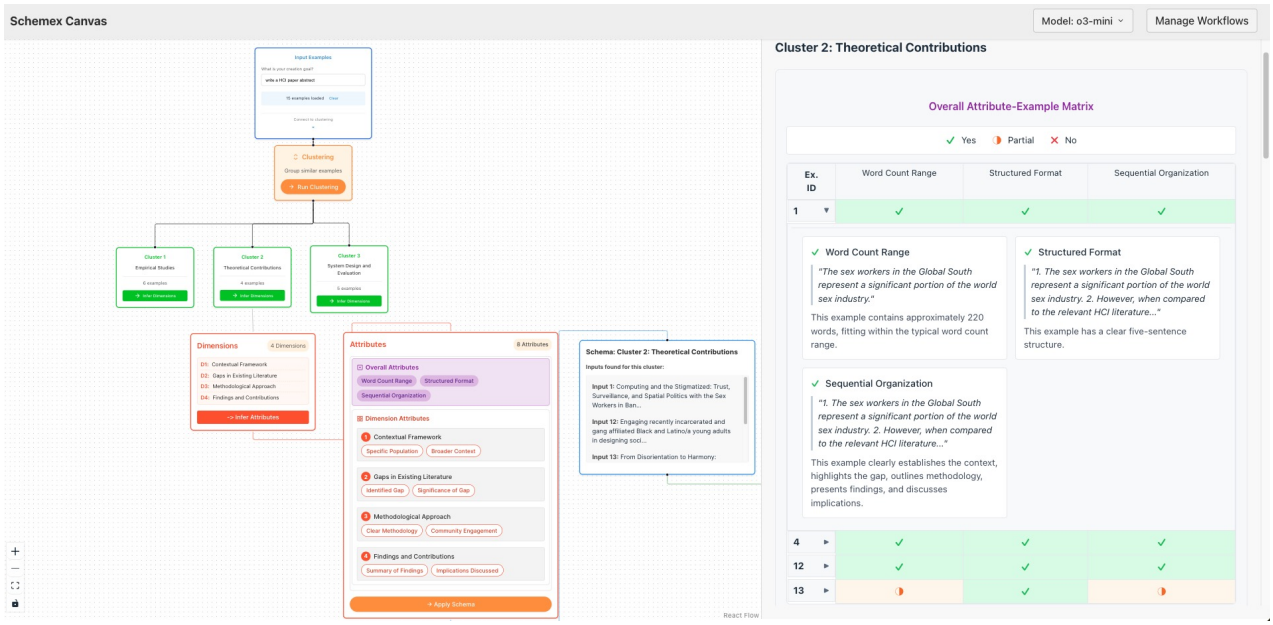

Figure 2: Screenshot of Schemex, an interactive, AI-powered visual workflow that helps users induce schemas from examples. Each node is interactive and features a side panel displaying detailed information, such as definitions and example mapping matrices. The workflow comprises three key stages: clustering, abstraction, and refinement through contrasting examples.

ideas from unstructured documents. CollabCoder [16] supports team-based coding and codebook evolution. Earlier crowdsourced research further shows that externalizing schemas—whether derived by individuals or crowds—can improve ideation, analogy retrieval, and collaborative understanding [28, 49, 50].

Schemex shares the goal of supporting structure discovery through mixed-initiative workflows, targeting schema induction from multimodal content creation examples. Unlike prior systems that focus on AI-generated artifacts (e.g., [42, 43]), Schemex emphasizes human sensemaking from real artifacts. It supports users in articulating and evolving actionable schemas. In this way, Schemex complements and extends prior systems by offering a workflow for interpretable, interactive, and grounded schema authoring.

2.4 Schemas Used in HCI Systems

Because of the generalized scaffolding schemas provide—particularly for novices—schemas are widely used in creativity support tools. Books of design patterns have influenced generations of architects [3], software engineers [15], and web designers [12]. In computer graphics, design patterns can help solve challenges such as laying out furniture in a room [36], generating usable maps [2], illustrating furniture assembly instructions [1], or sequencing cuts in film [32]. In HCI, design patterns have been used to help novice filmmakers structure their videos [27], support human-robot interaction programming [25, 40], and inspire novel product ideas [49, 50]. AI has also been instrumental in helping users apply abstract schemas to creative tasks such as writing stories [37], creating visual blends [6–9, 47], producing trailers and teasers [46], crafting social media videos [35, 45], and generating inclusive, persuasive emergency messages [24]. In nearly all of these systems, authors have had to manually analyze examples and construct schemas by hand—a labor-intensive process. Accelerating schema induction could vastly expand access to structured creativity, enabling more people to design, communicate, and express ideas effectively.

3 SCHEMEX SYSTEM

Schemex (see Figure 2) is an interactive visual workflow that assists users in transforming a set of examples into actionable schemas through three AI-assisted stages: clustering, abstraction, and refinement via contrasting examples. If the input examples are multimodal, the system first performs a preprocessing step that converts them into structured textual representations using models such as GPT-4V and Whisper. Next, the workflow clusters the examples into distinct groups based on their latent similarities. For each cluster, the system examines the examples to derive key dimensions and dimension-specific attributes. Overall attributes of the examples within the cluster are also inferred, forming an initial schema comprising both dimensions and attributes. Finally, Schemex refines the schema through contrastive refinement: it generates outputs based on the initial schema, compares them to the original human-authored examples, and iteratively improves the schema. Throughout the workflow, each intermediate result is presented in an interactive node. Clicking a node reveals a side panel containing more detailed information, such as definitions and example mapping matrices, helping users effectively ground their schema inspection and understanding in the example set.

Schemex is implemented using the Flask/React Flow web framework. It is powered by a reasoning LLM (o3-mini-high) for reasoning and analysis tasks, and a general-purpose LLM (GPT-4o) for generation tasks—particularly in schema application. We chose

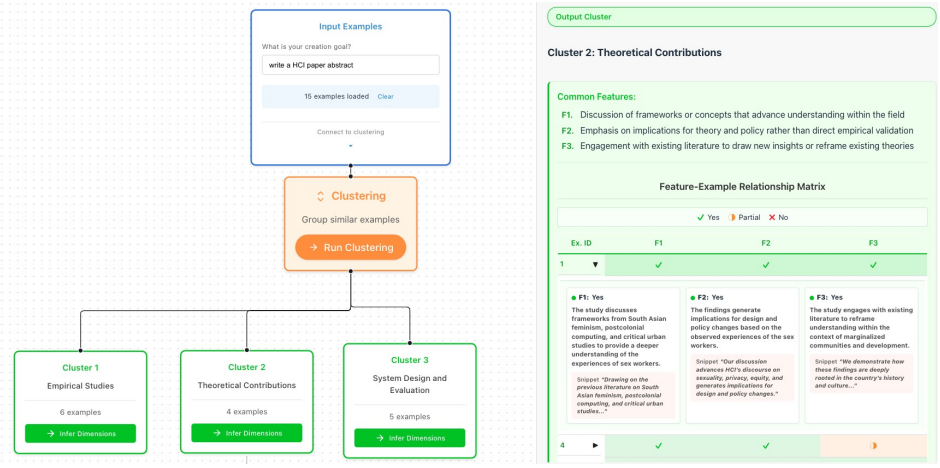

Figure 3: Schemex Stage 1: Clustering.

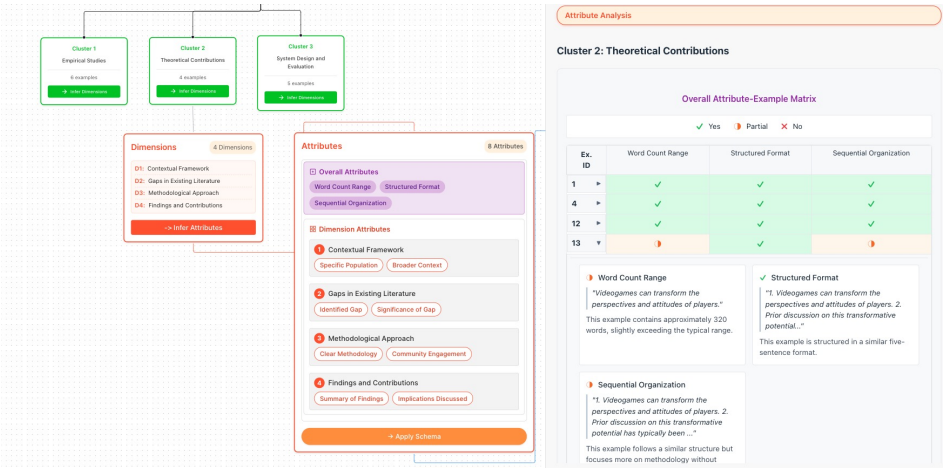

Figure 4: Schemex Stage 2: Abstraction.

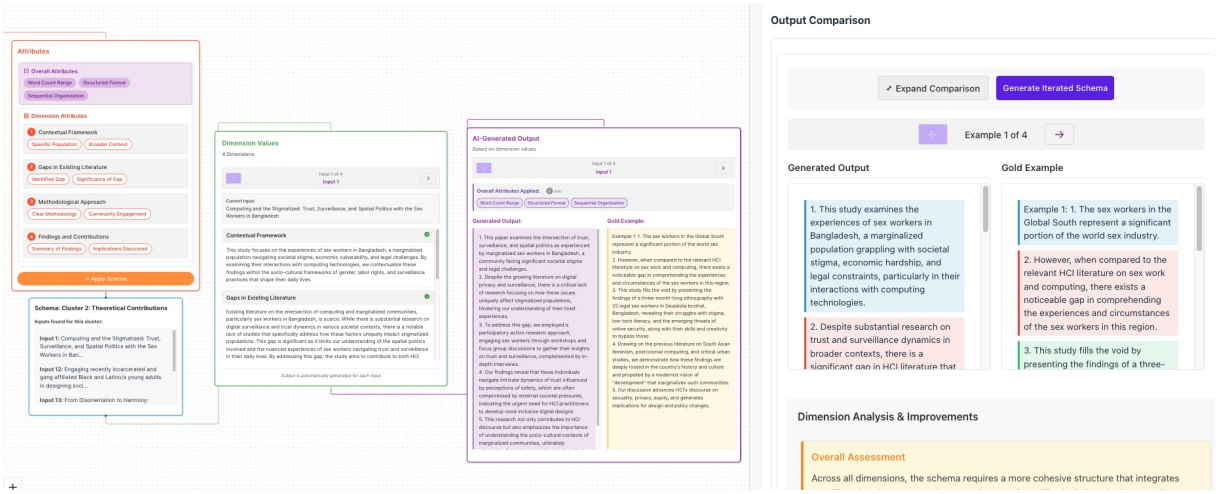

Figure 5: Schemex Stage 3: Contrastive Refinement.

o3-mini-high as our reasoning model due to its balance of accuracy and time efficiency. During early experiments, we found that it tended to generate overly profound outputs, so we reverted to GPT-4o for the generation steps. All prompts used in the system are available in Appendix A.

Next, we illustrate a typical interaction with the system through a walkthrough. In this scenario, the user wants to learn how to write an HCI paper abstract using a sample set of 15 CHI'24 best paper abstracts. The user also gathers the corresponding paper titles as input context.

3.1 Preprocessing Step for Multimodal Input

The user enters the system, which displays an example input node for specifying their goal and example set. To begin, the user enters the creation goal: "Write an HCI paper abstract", and uploads a collection of 15 CHI'24 Best Paper abstracts along with the corresponding paper titles. The examples and corresponding inputs are stored in pairs within the system.

Although not applicable in this specific use case, the system is capable of analyzing multimodal examples. If multi-model content is detected, the system performs additional processing steps. For images, they are sent to GPT-4V to generate visual descriptions. For videos, the system extracts one frame per second as "screenshot" images and sends them to GPT-4V to generate visual descriptions and extract any available captions. The audio tracks of the videos are transcribed using Whisper. All parsed information is then combined into a structured textual file containing visual descriptions, captions (if available), and audio transcriptions.

Once the examples are processed and ready for analysis, the system generates a clustering operation node and prompts the user to click the "Run Clustering" button.

3.2 Stage 1: Clustering

Clustering (see Figure 3) identifies subclasses of examples and is crucial in preventing schema overgeneralization. When given a set of examples, it is often unclear if they share the same schema. Generalizing across structurally different examples can lead to weak and bland schemas. Therefore, we must perform clustering first to identify examples that share latent similarities with each other and then find the schema specific to each cluster.

After the user clicks the "Run Clustering" button, the system instructs the reasoning model to cluster the examples and provide reasoning. The system processes all of the examples simultaneously, producing and labeling clusters. The resulting clusters are then displayed as individual nodes on the Schemex interface.

For instance, Schemex maps the abstracts into 3 clusters: Empirical Studies (6 examples), Theoretical Contributions (4 examples) and System Design and Evaluation (5 examples).

The user can easily verify the clustering results by selecting a cluster node to view a side panel that explains the common features shared among its examples, including a matrix describing how closely each example aligns with these features. For example, Schemex identifies the common features of Theoretical Contributions papers to be: 1) Discussion of frameworks or concepts that advance understanding within the field, 2) Emphasis on implications for theory and policy, 3) Engagement with existing literature to draw new insights or reframe existing theories.

After checking the side panel, the user is satisfied with the clustering and decides to explore the Theoretical Contributions cluster in more depth. They click the "Infer Dimensions" button on the cluster node.

3.3 Stage 2: Abstraction

Abstraction (see Figure 4) identifies structural commonalities among examples within each cluster. Unlike "features" that are used for clustering, the schemas formed during "abstraction" synthesize multiple examples into a general pattern of specific dimensions with descriptive attributes. It synthesizes multiple examples and extracts generalizable insights, a task that can often be overwhelming for individuals due to the high cognitive load involved.

Once the user clicks "Infer Dimensions" for the selected cluster, Schemex prompts the reasoning model to analyze the examples and extract the core elements or dimensions of the examples. For the Theoretical Contributions cluster, the key dimensions include: Contextual Framework , Gaps in Existing Literature, Methodological Approach, Findings and Contributions. The user can verify the generated schema by clicking on the dimension node to view further details about each dimension. This panel also includes an interactive Dimension-Example matrix that provides a visualization of how well each example conforms to each dimension. The matrix provides specific citations and justifications regarding whether each example fully, partially, or did not incorporate each dimension, enabling users to assess the system's coherence by mapping its analysis back to the examples.

The next step is to "Infer Attributes", including the dimension attributes, the key features within each dimension, and the overall attributes, the key descriptors of the full output. For example, for the Contextual Framework dimension, key attributes include a clearly defined population from which theoretical insights emerge or to which they apply, and the broader sociotechnical, cultural, or historical context that shapes the significance and applicability of the theory. For overall attributes, examples include a target word count range of 140–150 words and a narrative structure that presents the dimensions sequentially.

This approach distills executable guidelines for the specific cluster, ensuring that the output not only captures the key components (Dimensions) but also details what makes each component effective (Dimension-specific Attributes) and how to integrate these components into a cohesive output (Overall Attributes). At this point, the user has the first version of a usable schema.

3.4 Stage 3: Refinement via Contrasting Examples

The schema derived from Stage 2 is not perfect. For instance, while it specifies that important findings should be summarized, it does not clearly define how they should be written. To make the schema more precise and actionable, we employ an apply-and-test approach, iterating on the schema using contrastive refinement (see Figure 5).

Specifically, once the user has created an initial schema, they can click "Apply Schema" to prompt GPT-4o to generate Dimension

values based on the current schema and an example's input context (here is the paper title). This intermediate step allows users to 1) assess the accuracy of the identified dimensions and attributes before generating a final output 2) provide insight into the individual components behind the final product, deepening the user's understanding of the system's reasoning, as well as the task. Next, the user can generate a complete output (e.g., an HCI paper abstract) based on the generated dimension values and overall attributes. Note that the user can pick from a validation set (examples that came from the cluster was set separately and not used in the training) for schema application to check the schema effectiveness.

Schemex provides a color-coded interface that allows to user to easily compare the generated outputs with the original examples by dimension, highlighting potential improvements extracted by the reasoning model. These improvements are then used to iterate the schema. For example, the system might suggest that findings should be explicitly linked to existing literature and frameworks, rather than presented in isolation. After one iteration, the system generated a more detailed schema incorporating this improvement for the Findings dimension, such as adding the attribute "theoretical integration." The iterated schema can then be used to generate a new set of dimension values and complete outputs. The cycle repeats until users are satisfied with the final schema. The user can keep exploring another cluster they are interested in.

Throughout their interaction with Schemex, users are involved in the process by evaluating whether the generated content meets their standards. By highlighting and citing content relevant to the AI's reasoning decisions, Schemex encourages critical analysis of system output and constant comparison to the source material. These features support the user in their understanding of the AI and the paper abstract writing ask, and the original inputs, so they can more effectively act as a reflective evaluator.

Ultimately, users can utilize the resulting schemas as a guide to write their own HCI paper abstracts.

4 TECHNICAL EVALUATION

To evaluate how structural abstraction and contrastive refinement contribute to schema quality, we conducted a controlled study in which domain experts assessed schemas generated under three different conditions:

- Condition 1: o1-pro—an advanced reasoning model capable of interpreting task descriptions, decomposing them into steps, and executing those steps to return results.
- Condition 2: Schemex Initial—a structured workflow involving clustering, followed by abstraction into dimensions and attributes.
- Condition 3: Schemex Full—our full system implementation with one iteration, which builds on Schemex Initial by introducing a contrastive refinement step.

We hypothesize that schemas generated under the Schemex Full condition will outperform those from the other conditions across seven dimensions: (1) cluster quality, (2) dimension and attribute quality, (3) coverage and generalizability, (4) example fit, (5) usefulness for content creation and critique, (6) alignment of schema-guided generations with original examples, and (7) overall effectiveness of schema-guided generations.

4.1 Data Preparation

We collected 6 different topics as the test suite, each containing 15 randomly picked examples. The topic span different domains of academia, business and media, and span different modalities:

- Task 1: EE/CS faculty candidate job talk abstracts
- Task 2: UIST teaser video
- Task 3: LinkedIn Profile summary
- Task 4: Pitch deck for start-ups
- Task 5: News headline
- Task 6: News TikToks [38]

Example creations are collected from trustworthy real-world resources, for example, the EE/CS faculty candidate job talk abstracts were collected from the university emails, the UIST teaser videos were downloaded from the UIST YouTube channel. All examples were processed using our multi-modal analysis pipeline to get the structured textual dataset.

To generate schema with o1-pro, we use the following prompt, which is equivalent to what we give Schemex as the task description and input. The prompt can be found in Appendix B. We ran models or pipelines in three conditions all once for each of the task. After getting all the schemas, we used GPT-4o to generate the outputs (as described in the System section "schema application" step). For each schema (of each cluster), we generated outputs for two randomly picked inputs from the cluster. Note that if a cluster just has one example, we just generated output for one input. Overall, we generated 20 schemas and 38 outputs for Condition 1, and 17 schemas and 33 outputs for Condition 2 and Condition 3, respectively.

4.2 Participants and Procedure

We invited twelve experts (two experts for each topic) to evaluate the schema and the accordingly outputs. The experts (average age 27.0, 7 female, 5 male) all have experience in the relevant domain and the specific task. For example, for news TikTok tasks, we invited journalism graduate students who had two years of experience in making news TikToks. Before the evaluation, we presented experts with examples of the schemas. We provided an evaluation rubric of the seven dimensions and guided them through it using practical examples. We requested them to read the original examples thoroughly before commencing ratings on schemas and outputs. For each of the tasks, two experts independently evaluated each dimension on a 7-point scale with justifications. When presenting the data to the experts, the condition information is removed, and the order of the condition shown to experts are counterbalanced between all the participants. The experts were compensated $50/hour.

4.3 Evaluation Results and Findings

The interrater Agreement (within ±1 point) is 0.78, showing substantial agreement between the two experts' judgment. This is deemed acceptable due to its highly subjective nature. We average the two experts' scores for each schema and output. We run paired-sample Wilcoxon tests to compare the scores. See results in Table 1.

	Condition1: o1-pro	Condition2: Schemex Initial	Condition3: Schemex Full	p-value (1 vs 2)	p-value (2 vs 3)	p-value (1 vs 3)
D1-Cluster Quality	5.1 (0.79)	5.8 (0.72)	6.4 (0.51)	**0.046**	**0.02**	**0.004**
D2-Dimensions & Attributes Quality	5.0 (0.85)	5.9 (0.67)	6.8 (0.39)	**0.026**	**0.002**	**0.003**
D3-Coverage & Generalizability	5.0 (1.1)	5.5 (0.67)	5.8 (0.75)	0.19	0.26	0.07
D4-Example Fit	5.6 (0.9)	6.1 (0.67)	6.6 (0.51)	0.19	**0.014**	**0.018**
D5-Usefulness for Content Creation and Critique	5.3 (0.98)	6.2 (0.75)	6.7 (0.49)	**0.032**	0.059	**0.007**
D6-Alignment of Generations with Original Examples	5.1 (0.79)	5.7 (0.65)	6.4 (0.67)	0.07	**0.007**	**0.007**
D7-Overall Effectiveness of Generations	5.5 (0.52)	5.8 (0.58)	6.3 (0.65)	0.16	**0.014**	**0.013**

Table 1: Evaluation results comparing o1-pro, Schemex Initial, and Schemex Full, where means, standard deviations (in parentheses), and p-values for the paired-sample Wilcoxon tests are reported. Bolded p-values are statistically significant.

4.3.1 Comparing Schemex Initial to o1-pro: Structural abstraction significantly improves schema quality and usefulness. Schemex Initial outperforms o1-pro in all seven dimensions, and it shows statistically significant improvements in three of the seven evaluation dimensions: D1: cluster quality (p=.046), D2: dimension and attribute quality (p=.026) and D5: usefulness of the schema (p=.032).

In terms of cluster quality, Schemex clusters are functionally exclusive and grounded in clear structural commonalities. In contrast, o1-pro's clusters often blur together and overlap, reducing precision and limiting their usefulness. For example, in the EE/CS faculty candidate job talk case, although o1-pro-derived clusters are thematically distinct: Cluster 1: Technical or Algorithmic-focused; Cluster 2: Systems and Architecture-Focused; Cluster 3: Human and Societal Impact-Focused. Example 10 (which is about supporting user-centered analytical interface at scale) was assigned to Cluster 2 due to its system focus, but can also be placed to Cluster 3 as it also highlights the "human" perspective. While Schemex-derived clusters have a clear structural divide which each example can be cleared mapped to: Cluster 1: Sequentially Ordered Roadmap Abstracts and Cluster 2: Integrated Narrative Abstracts.

In terms of dimension and attribute quality (D1), Schemex offers more granular, example-set-specific attributes that closely reflect structural patterns in the data. o1-pro, by comparison, relies on broad, vague guidelines that lack depth and fail to capture nuances. For example, in the LinkedIn profile summary case (D2), while o1-pro notes that they "often include structured bullet points" or "end with an invitation to subscribe," Schemex identifies precise narrative moves like a "pivotal challenge or turning point," to detail career evolution—offering clearer insights into how personal storytelling and structural choices actually function in the examples.

In terms of the usefulness of schema (D5), Schemex provides practical, actionable guidance that helps users emulate gold examples effectively. o1-pro is less descriptive and offers limited support for creation or critique. For example, for the UIST demo system walkthrough cluster, while o1-pro mentions key elements like on-screen annotations, Schemex goes further by detailing how annotations are layered to guide user understanding—making it more actionable for creators aiming to emulate complex interactive walkthroughs like those in the example set.

4.3.2 Comparing Schemex Full to Schemex Initial: Contrastive refinement unlocks additional gains for example fit and generation quality. Schemex Full goes a step further, yielding statistically significant improvements in 5 out of 7 dimensions compared to Schemex Initial: D1: cluster quality (p=.02), D2: dimension and attribute quality (p=.002), D4: Example fit (p=.014), D6: Alignment of Generations with Original Example (p=.007) and D7: Overall Effectiveness of Generations (p=.014).

As Schemex Full went through another cycle of contrastive analysis and abstraction, it is not surprising that it can generate higher-quality schema (D1 and D2). In terms of example fit (D4), through contrastive refinement, Schemex Full enables finer-grained alignment between schema attributes and examples, capturing subtle structural cues that Schemex Initial often missed. For example, for the news headline case, the initial version emphasizes using active verbs and focusing on a single event. The full version adds guidance for compound action handling—such as using "and," "as," or "after" to connect multiple developments—which better reflects several gold examples that combine actions and consequences in a single headline. This also explains why schemas derived by Schemex Full can generate outputs that better align with the original examples (D6) and serve the creation purpose more effectively (D7), as it captures more nuanced rules from good examples for AI to follow.

4.3.3 Limitations of Contrastive Refinement: Coverage and Generalizability. Interestingly, D3—Coverage—is the only dimension where Schemex Iterated did not show a statistically significant improvement over o1-pro (p = 0.07). This dimension evaluates how well a schema accounts for the range of structural patterns that might appear in unseen or future examples, without overfitting to the initial sample set. This result suggests that while contrastive refinement improves precision, depth, and stylistic alignment, it may not inherently expand the generalizability of the schema. In fact, by refining based on specific gaps relative to a gold example, the iteration might even bias the schema further toward that particular instance—narrowing rather than broadening its scope. By contrast, o1-pro, as a general-purpose reasoning model, may maintain broader coverage.

For example, for the pitch deck test suite, one limitation of the Schemex full-version schema is that it places greater emphasis on specific storytelling patterns—such as step-by-step, roadmap-style narratives—which were prominent in the example set. While this leads to more precise alignment with those patterns, it may not generalize as well to pitches that follow less conventional structures. For instance, a founder who presents their mission, product, and traction in a fluid, vision-led narrative—without using explicit

Participant	Example set	Number of Examples
P1	Personal creative writing	10
P2	NBA team's Instagram posts	13
P3	Iambic Pentameter	15
P4	Subheaders from NPR Articles	14
P5	Tech Career LinkedIn Bios	8
P6	HCI PhD SoP First Paragraph	12
P7	Fandom Wiki for Harry Potter	10
P8	Hopecore TikTok	8

Table 2: Participants and the example sets they choose.

sequencing—might be harder to fully capture within the refined schema. The core logic is still there, but the schema may not recognize it as strongly. In contrast, the o1-pro schema is more focused on identifying the underlying reasoning—problem, solution, and evidence—regardless of how the story is told. This gives it more flexibility in handling diverse and novel formats, even when they diverge from patterns seen in the example set.

Overall, our findings demonstrate that structural abstraction (Schemex Initial) significantly improves schema quality and practical utility over a strong baseline (o1-pro), particularly in cluster coherence, attribute granularity, and actionable guidance. Building on this, the addition of contrastive refinement (Schemex Full) yields further gains in alignment with examples and generation effectiveness by capturing subtler structural nuances. However, this added precision may come at the cost of generalizability, suggesting a trade-off between depth and breadth in schema induction.

5 USER STUDY

To understand how Schemex assists users in schema induction, we conducted a within-subjects study with eight participants, comparing their interaction with Schemex to a baseline of ChatGPT (o1-pro). We chose o1-pro as a strong language model baseline that excels in complex problem-solving, enabling us to isolate the specific benefits of Schemex for schema induction tasks.

5.1 Protocol

We recruited eight participants (average age=26.6, four female, four male) for this study through a university mailing list and word-of-mouth. All participants reported being familiar or very familiar with ChatGPT. Each participant selected an example set of their choice. The example sets spanned diverse areas and modalities with 8-15 examples per set (see Table 2).

The participant was then given two tasks, (*Task A*) inducing a satisfactory schema using Schemex, and (*Task B*) inducing a satisfactory schema using o1-pro for the same example set. Half began with Task A, and half with Task B, with a brief demonstration before the Schemex task. Each task had a time limit of 30 minutes. After each task, participants were interviewed regarding their experience with the corresponding system. In these interviews, users rated questions (see questions in Section 5.2) on a 1-7 scale and provided reasons for their ratings. Each study session lasted approximately 80 minutes, with participants compensated $25/hour. We run paired-sample Wilcoxon tests to compare the ratings.

5.2 Findings

Schemex significantly outperformed ChatGPT (o1-pro) in 4 out of 5 areas that were analyzed, including structural insight, novel reflections, confidence, and visual presentation. In terms of clarity, both Schemex and ChatGPT scored similarly. To assess performance across these areas, users rated the five research questions on a 1–7 scale. See results in Figure 6.

- RQ1 - Schema Clarity: How clear and understandable was the schema provided by the system?
- RQ2 - Structural Insight: How much new insight did you gain into the structure or norms of this genre?
- RQ3 - Novel Reflection: Did interacting with the system help reflect or notice things you otherwise wouldn't have?
- RQ4 - Confidence: After using this system, how confident do you feel about the schema?
- RQ5 - Visual Presentation: Do you think the UI/visual presentation helps you learn or enhance your understanding?

Qualitative analysis of user interviews suggests that the high ratings for Schemex stemmed not just from the outputs, but from the *interactive workflow* that guided users through the schema creation process. These interactions deepened users' understanding of their original material and increased trust in the system's output. We discuss specific quantitative and qualitative findings for each research question below:

5.2.1 RQ1 - Schema Clarity. Schemex provided a similarly clear schema (5.56/7) as ChatGPT (5.69), p=.680.

While both systems were provided comparable scores, the clarity of the two system's final schemas came from very different areas. P8 (Hopecore TikToks) scored both Schemex and ChatGPT 6/7. She explains that while ChatGPT "was pretty readable [and] stayed kinda high-level," Schemex had "a lot going on, but it was laid out super visually - it was easy to follow." Overall, users found that GPT's simple, text-based output was comparably clear to Schemex's structured, information-rich output.

5.2.2 RQ2 - Structural Insight. Schemex provided significantly greater structural insight (5.56/7) than ChatGPT (2.25/7), p = .004.

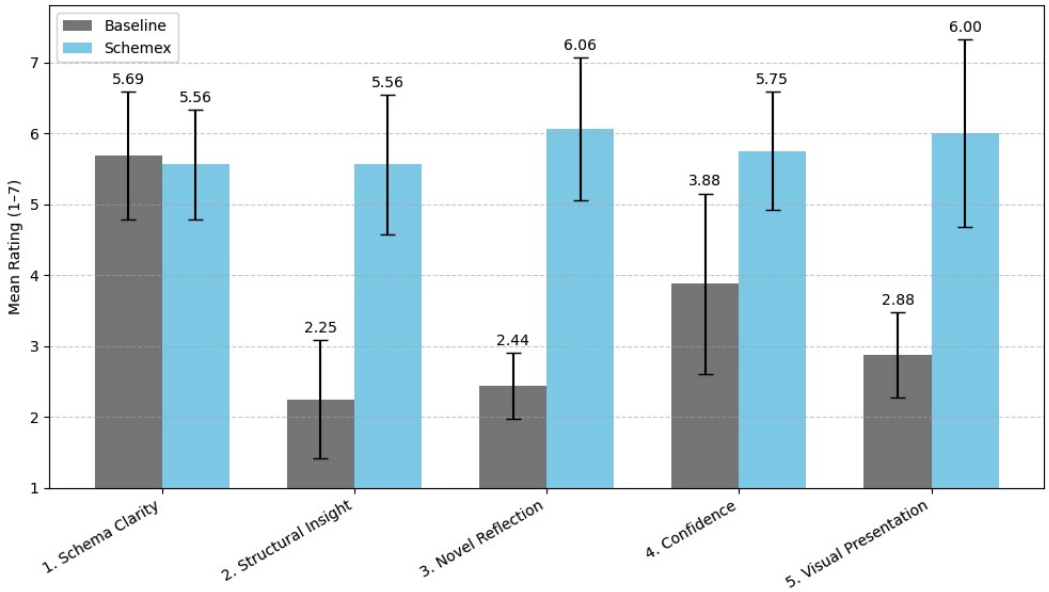

Figure 6: Participant ratings over five research questions for baseline vs. Schemex.

Schemex was able to capture deeper semantic structures than the baseline. P1, who analyzed diverse samples of her writing, rated Schemex a 7/7, citing its identification of common thematic threads in her work, despite their structural differences: "it found the implicit connections and anxiety around time." The ability to gradually break down the inferred commonalities into specific examples through interaction with Schemex encouraged intellectual engagement. P2 continued, "[Schemex] tells you - 'this is a thing you do and here's the precise way it manifests.' It makes me reflect."

Additionally, **Schemex outputs were more detailed, providing more precise information than the o1-pro model.** P3, who was analyzing various examples of iambic pentameter, rated Schemex 7/7. His final schema provided specific instructions regarding meter, language style, and how to form the rhetorical questions characteristic of classical poetry in the genre.

Schemex's depth and detail made it the participants' preference for tasks requiring a personal understanding of the material. For example, P3 (Iambic Pentameter) noted that Schemex would be very helpful if he was trying to teach a course on the subject. P5 (LinkedIn Bio), who provided Schemex with a score of 4 and the baseline with a score of 2, echoed, "If I was trying to *understand* what a LinkedIn bio is, I would go to [Schemex]."

5.2.3 RQ3 - Novel Reflection. Schemex encouraged a higher level of reflection and discovery (6.06/7) compared to ChatGPT (2.44/7), p = .004.

Interacting with Schemex sparked new insights participants had not noticed on their own. Many users initially believed their example sets were straightforward. However, **Schemex's intermediate steps, particularly clustering, helped surface unexpected distinctions.** P4, analyzing NPR subheaders, discovered meaningful categories like "Direct Factual Summaries" and "Composite and Contrasting Narratives." See Figure 7 for the detailed schema. He scored Schemex 5/7, explaining: "It helps you understand something that seems like 'the same thing' [and find] there's different types of them."

Furthermore, **exploration options provided by Schemex provided avenues for refinement they had not considered.** For example, when using ChatGPT, P2 (Instagram Posts) had difficulty determining how to improve its outputs: "You have to know what to ask it in order to get what you want. It's a little bit more frustrating." He scored ChatGPT 2/7. P1 explains how the options provided by Schemex (scored 7/7) remedy this problem, citing the suggestions provided in the comparison step: "Schemex helps you with how you should iterate and provide you with an understanding of all of the opportunities." He compares this to the experience using ChatGPT (scored 2/7): "I know what needs to change, but I don't know what to change to. [I have to] act in a very corrective way."

5.2.4 RQ4 - Confidence. Participants expressed greater confidence in schemas produced by Schemex (5.75/7) than ChatGPT (3.88/7), p = .011.

Schemex's layered, interactive workflow fostered faith and confidence in outputs. P3 shared: "Because [at each level] I could see the added context, it was a lot easier to trust - which wasn't the case with ChatGPT. If I had used ChatGPT second, I would have given [ChatGPT] lower scores."

Schemex further instilled confidence in users by providing them with accessible and efficient evaluation features. P2 emphasized the example matrix as a confidence-boosting element: "It was just an easy way to sort through [the information] … and not have to go back and figure out each one." P5 added: "I liked the self-reflection of the AI. It made me feel more confident in what it was telling me."

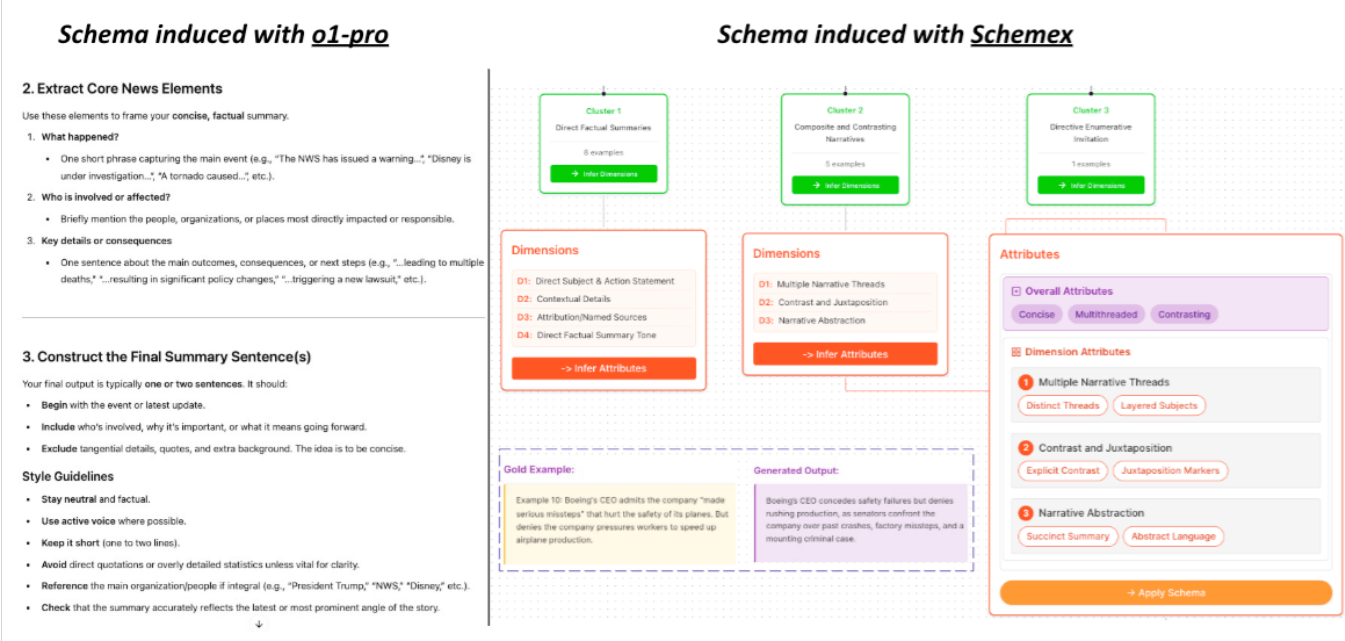

Figure 7: Schemas P4 induced with o1-pro (left) and Schemex (right) for the example set of NPR article subheaders. While ChatGPT gave a generic outcome, Schemex provides clusters, dimensions, and attributes that P4 found insightful and specific to the given example set. Inside the dotted box is one output from Schemex and the according gold example.

Schemex developed users' trust by grounding abstract representations in real data. P1, when discussing the limitations of AI systems, noted, "I love that it helped balance the abstractness...I don't think an AI-empowered interface could do without examples." In contrast, the same participant highlighted a key shortcoming of ChatGPT, stating, "the way ChatGPT engaged with the examples was generally summarizing." P5 similarly emphasized the role of concrete examples in fostering trust, explaining their 7/7 rating by saying, "I think it also goes to the trust in the system. Examples are the most helpful for me."

Ultimately, users were confident in the precision, depth, and actionability of the Schemex schemas. P3 noted: "It was just so much more descriptive and aligned with what I hope I would take away if I had a lot of time spent studying this." P5 echoed this confidence regarding her Schemex output: "I would have been able to write a LinkedIn bio [from it]."

5.2.5 RQ5 - Visual Presentation. Schemex's visual interface (6.0/7) was rated significantly higher than ChatGPT's (2.88/7), $p = .008$.

Schemex's "tree-like" layout more closely matched participants' conceptual model of a schema than linear text. P1 explains why the Schemex interface better represents schemas when compared to ChatGPT: "Schemas are abstractions. They're less linear. It's harder to conceptualize and bucket things in pure text." P3 agreed that Schemex allowed for improved information processing: "Schemex was by far the most granular at visualizing and demonstrating all the specific details."

Schemex's interface enhanced user understanding by allowing users to "drill down" from abstract groups to general insights to specific details. The "overview first, zoom and filter, then details on demand" mantra, or Visual Information-Seeking Mantra, suggests presenting data effectively by first offering a broad overview, then allowing users to explore specific details through interactive tools. P1 enjoyed how this structure allowed her to gradually break down patterns in her writing. She explains, "It started off with broad themes, [then] an integrated framework, and [then] how you capture 'vivid imagery' in your writing specifically with an operational definition." P2 also found that this step-by-step breakdown made the logic of the system easier to follow, stating, "I like how it drilled down, it gave me more insight into the AI and how it determined the less obvious stuff."

However, not all participants preferred the Schemex interface. P5 scored Schemex 3/7 and ChatGPT 4/7, expressing her preference for more condensed, linear representations of information: "I am the kind of person who likes to see what I am focusing on... [Schemex] was a little too much for me." Despite this, the majority of participants found the visualization to significantly support their understanding.

6 DISCUSSION

6.1 Supporting Schema Discovery through Interactive Structural Abstraction

Schemex scaffolds interactive structural abstraction through AI-assisted reasoning and visualizations. In our study, participants consistently preferred the interactive visual workflow of Schemex over chat-based interfaces and reported gaining deeper insights. This preference is not surprising in retrospect: learning benefits from active engagement. Cognitive theories such as constructivism [14]

emphasize that people learn more effectively when they can manipulate, reflect on, and iterate over ideas. In contrast, chat interfaces like GPT-style models often promote passive interaction—users tend to skim for answers rather than engage in deeper exploration. By making structure visible and refinable, Schemex encourages users to think more critically and deliberately, leading to more meaningful learning and insight.

Discovering underlying structure from examples is rarely a linear task. Like other sensemaking processes, schema induction involves generating hypotheses, testing them against data, and refining them based on feedback. Users often need to zoom in on fine-grained patterns and zoom out to adjust the overall schema. Visual workflows like Schemex make this iterative process manageable by scaffolding multiple levels of abstraction—from clusters to dimensions to example-level attributes. This mirrors how experts build understanding over time and enables novices to follow a similar path through guided exploration.

The effectiveness of interactive schema discovery also depends on the capabilities of the underlying AI. Large language models—especially those optimized for reasoning—have recently become capable of meaningful structural analysis: clustering examples, identifying latent dimensions, and proposing abstractions that align with human interpretations. Schemex leverages these capabilities through a guided, multi-stage process: the AI suggests patterns, but always in a way that users can inspect and refine. This makes the system a collaborative partner rather than a black box, aligning well with best practices in human-AI interaction.

While Schemex was designed and evaluated for schema induction in creative and communicative tasks, the underlying approach is applicable to a wide range of domains. Many real-world problems involve extracting structure from complex, nonlinear data—for example, mapping workflows in organizational process mining, modeling user flows on websites, or tracing state transitions in software systems. These domains all require iterative exploration and abstraction to surface useful insights. The interactive structural abstraction model supported by Schemex provides a general framework for approaching such problems, suggesting promising directions for future applications.

6.2 Contrastive Refinement as a Mechanism for Evolving Schema Understanding

People often learn by comparing examples—noticing variations, identifying what holds constant, and refining their understanding through contrast. Schemex builds on this principle through contrastive refinement, where it compares AI-generated outputs to real examples to evaluate and adjust the underlying schema. This process helps recognize mismatches, clarify vague structures, and iteratively improve schema quality.

Contrastive methods are already central in AI and machine learning, particularly in contrastive learning approaches, where models improve by distinguishing between similar and dissimilar instances. In Schemex, contrast serves a different but related purpose: it helps refine abstract representations through guided comparison. This aligns with cognitive theories of structural alignment [34], which emphasize that abstraction is often clearest when comparing systematically varied cases.

Importantly, schemas are rarely static. As genres shift or new formats emerge, existing schemas may no longer fit. For instance, an HCI researcher familiar with traditional system papers may encounter first-person HCI work and struggle to interpret its structure using previous templates. Contrastive refinement supports this kind of reflection by highlighting where existing schemas fall short and how they might adapt. It enables users to remain flexible and responsive as domains evolve.

Future work could extend this idea toward longer-term schema tracking. Schemas may shift over time within individuals or communities: evolving, merging, or splitting as practices change. Supporting experts in monitoring these shifts could help them stay current, inform pedagogical tools, or uncover emerging patterns in fields. Contrastive refinement offers a foundation for such reflective exploration by anchoring schema development in real, observable differences.

6.3 Limitations and Future Work

Schemex currently supports multi-modal example analysis by converting images, videos, or audio into structured textual descriptions to enable LLM-based reasoning. While this approach allows schema induction across diverse content types, it limits the evaluation and application of schemas to a primarily textual modality. Future work could explore fully multi-modal interaction—for example, aligning schema elements with visual or interactive outputs—to help users better assess how schemas function across different media.

Schemex also includes a basic schema application workflow: given a user input and a derived schema, the AI generates content by filling in values for each dimension. This feature helps users test the schema in context and refine it through contrastive evaluation. However, supporting real-world schema application more effectively presents open challenges. What kinds of inputs should users provide? Given any schema, how can we scaffold users through the process of applying it meaningfully? What forms of intervention or guidance should the system offer? Future research could explore how to tailor this workflow to different content domains, input types, and levels of user expertise.

Finally, the potential misuse of Schemex raises ethical concerns, such as scraping others' examples or imitating stylistic patterns without proper attribution [39]. Addressing these issues will be essential as the system moves toward broader deployment. Future versions of Schemex could incorporate mechanisms for attribution tracking, usage transparency, and community norms to encourage responsible use.

7 CONCLUSION

We introduced Schemex, an interactive system that supports schema induction through AI-assisted structural abstraction and contrastive refinement. By grounding abstract patterns in real examples and enabling iterative comparison, Schemex helps users uncover, evaluate, and refine the underlying structure of creative artifacts. Our study shows that Schemex improves user insight, confidence, and schema quality compared to a reasoning AI model baseline. This work highlights the potential of interactive, example-driven workflows to scaffold complex reasoning tasks, and opens new directions for supporting structure discovery across domains.

REFERENCES

[1] Maneesh Agrawala, Wilmot Li, and Floraine Berthouzoz. 2011. Design Principles for Visual Communication. *Commun. ACM* 54, 4 (April 2011), 60–69. https://doi.org/10.1145/1924421.1924439

[2] Maneesh Agrawala and Chris Stolte. 2001. Rendering Effective Route Maps: Improving Usability Through Generalization. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. ACM, New York, NY, USA, 241–249. https://doi.org/10.1145/383259.383286

[3] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, and Shlomo Angel. 1977. *A Pattern Language - Towns, Buildings, Construction*. Oxford University Press.

[4] Khalid Alharbi and Tom Yeh. 2015. Collect, decompile, extract, stats, and diff: Mining design pattern changes in Android apps. In *Proceedings of the 17th international conference on human-computer interaction with mobile devices and services*. 515–524.

[5] Minsuk Chang, Léonore V Guillain, Hyeungshik Jung, Vivian M Hare, Juho Kim, and Maneesh Agrawala. 2018. Recipescape: An interactive tool for analyzing cooking instructions at scale. In *Proceedings of the 2018 CHI conference on human factors in computing systems*. 1–12.

[6] Lydia B. Chilton, Ecenaz Jen Ozmen, Sam H. Ross, and Vivian Liu. 2021. VisiFit: Structuring Iterative Improvement for Novice Designers. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3411764.3445089

[7] Lydia B. Chilton, Savvas Petridis, and Maneesh Agrawala. 2019. VisiBlends: A Flexible Workflow for Visual Blends. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, 172:1–172:14. https://doi.org/10.1145/3290605.3300402

[8] João M. Cunha, João Gonçalves, Pedro Martins, Penousal Machado, and Amílcar Cardoso. 2017. A Pig, an Angel and a Cactus Walk Into a Blender: A Descriptive Approach to Visual Blending. In *Proceedings of the Eighth International Conference on Computational Creativity*. 80–87. https://arxiv.org/abs/1706.09076

[9] João Miguel Cunha, Pedro Martins, and Penousal Machado. 2018. How Shell and Horn Make a Unicorn: Experimenting with Visual Blending in Emoji. In *Proceedings of the Ninth International Conference on Computational Creativity*. 145–152. https://openreview.net/forum?id=u1mCZAjwX7

[10] Leonidas AA Doumas, John E Hummel, and Catherine M Sandhofer. 2008. A theory of the discovery and predication of relational concepts. *Psychological review* 115, 1 (2008), 1.

[11] Hakan Duman, Alex Healing, and Robert Ghanea-Hercock. 2009. Adaptive Visual Clustering for Mixed-Initiative Information Structuring. In *Human Interface and the Management of Information. Designing Information Environments: Symposium on Human Interface 2009, Held as Part of HCI International 2009, San Diego, CA, USA, July 19-24, 2009, Procceedings, Part I*. Springer, 384–393.

[12] Douglas K. Van Duyne, James Landay, and Jason I. Hong. 2002. *The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-Centered Web Experience*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[13] Alex Endert, Patrick Fiaux, and Chris North. 2012. Semantic interaction for visual text analytics. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 473–482.

[14] Catherine Twomey Fosnot. 2013. *Constructivism: Theory, perspectives, and practice*. Teachers College Press.

[15] Erich Gamma, Richard Helm, Ralph E. Johnson, and John Vlissides. 1995. *Design patterns: elements of reusable object-oriented software*. Vol. 206. 395 pages. https://doi.org/10.1093/carcin/bgs084

[16] Jie Gao, Yuchen Guo, Gionnieve Lim, Tianqin Zhang, Zheng Zhang, Toby Jia-Jun Li, and Simon Tangi Perrault. 2024. CollabCoder: a lower-barrier, rigorous workflow for inductive collaborative qualitative analysis with large language models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–29.

[17] Simret Araya Gebreegziabher, Yukun Yang, Elena L Glassman, and Toby Jia-Jun Li. 2024. Supporting Co-Adaptive Machine Teaching through Human Concept Learning and Cognitive Theories. *arXiv preprint arXiv:2409.16561* (2024).

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

[19] 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.

[20] Mary L Gick and Keith J Holyoak. 1983. Schema induction and analogical transfer. *Cognitive psychology* 15, 1 (1983), 1–38.

[21] Sumit Gulwani. 2011. Automating string processing in spreadsheets using input-output examples. *ACM Sigplan Notices* 46, 1 (2011), 317–330.

[22] 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).

[23] Yifen Huang. 2010. *Mixed-Initiative Clustering*. Carnegie Mellon University.

[24] Sophia S. Jit, Jennifer Spinney, Priyank Chandra, Lydia B. Chilton, and Robert Soden. 2024. Writing out the Storm: Designing and Evaluating Tools for Weather Risk Messaging. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. Association for Computing Machinery, New York, NY, USA, 502:1–502:16. https://doi.org/10.1145/3613904.3641926

[25] Peter H. Kahn, Nathan G. Freier, Takayuki Kanda, Hiroshi Ishiguro, Jolina H. Ruckert, Rachel L. Severson, and Shaun K. Kane. 2008. Design Patterns for Sociality in Human-robot Interaction. In *Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction* (Amsterdam, The Netherlands) *(HRI '08)*. ACM, New York, NY, USA, 97–104. https://doi.org/10.1145/1349822.1349836

[26] Charles Kemp and Joshua B Tenenbaum. 2008. The discovery of structural form. *Proceedings of the National Academy of Sciences* 105, 31 (2008), 10687–10692.

[27] Joy Kim, Mira Dontcheva, Wilmot Li, Michael S. Bernstein, and Daniela Steinsapir. 2015. Motif: Supporting Novice Creativity Through Expert Patterns. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. ACM, New York, NY, USA, 1211–1220. https://doi.org/10.1145/2702123.2702507

[28] Aniket Kittur, Andrew M Peters, Abdigani Diriye, and Michael Bove. 2014. Standing on the schemas of giants: socially augmented information foraging. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*. 999–1010.

[29] Christian Kruschitz and Martin Hitz. 2010. Human-computer interaction design patterns: structure, methods, and tools. *Int. J. Adv. Softw* 3, 1 (2010).

[30] 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*. 3083–3092.

[31] 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.

[32] Mackenzie Leake, Abe Davis, Anh Truong, and Maneesh Agrawala. 2017. Computational Video Editing for Dialogue-driven Scenes. *ACM Trans. Graph.* 36, 4, Article 130 (July 2017), 14 pages. https://doi.org/10.1145/3072959.3073653

[33] 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.

[34] Arthur B Markman and Dedre Gentner. 1993. Structural alignment during similarity comparisons. *Cognitive psychology* 25, 4 (1993), 431–467.

[35] Samia Menon, Sitong Wang, and Lydia Chilton. 2024. MoodSmith: Enabling Mood-Consistent Multimedia for AI-Generated Advocacy Campaigns. *arXiv preprint arXiv:2403.12356* (2024).

[36] Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. 2011. Interactive Furniture Layout Using Interior Design Guidelines. In *ACM SIGGRAPH 2011 Papers* (Vancouver, British Columbia, Canada) *(SIGGRAPH '11)*. ACM, New York, NY, USA, Article 87, 10 pages. https://doi.org/10.1145/1964921.1964982

[37] Piotr Mirowski, Kory W Mathewson, Jaylen Pittman, and Richard Evans. 2023. Co-writing screenplays and theatre scripts with language models: Evaluation by industry professionals. In *Proceedings of the 2023 CHI conference on human factors in computing systems*. 1–34.

[38] Nic Newman. 2022. How publishers are learning to create and distribute news on TikTok. (2022).

[39] Julien Porquet, Sitong Wang, and Lydia B Chilton. 2024. Copying style, Extracting value: Illustrators' Perception of AI Style Transfer and its Impact on Creative Labor. *arXiv preprint arXiv:2409.17410* (2024).

[40] Allison Sauppé and Bilge Mutlu. 2014. Design Patterns for Exploring and Prototyping Human-robot Interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. ACM, New York, NY, USA, 1439–1448. https://doi.org/10.1145/2556288.2557057

[41] John Stasko, Carsten Gorg, Zhicheng Liu, and Kanupriya Singhal. 2007. Jigsaw: supporting investigative analysis through interactive visualization. In *2007 IEEE Symposium on Visual Analytics Science and Technology*. IEEE, 131–138.

[42] 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.

[43] Sirui Tao, Ivan Liang, Cindy Peng, Zhiqing Wang, Srishti Palani, and Steven P Dow. 2025. DesignWeaver: Dimensional Scaffolding for Text-to-Image Product Design. *arXiv preprint arXiv:2502.09867* (2025).

[44] Anh Truong, Peggy Chi, David Salesin, Irfan Essa, and Maneesh Agrawala. 2021. Automatic generation of two-level hierarchical tutorials from instructional makeup videos. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–16.

[45] Sitong Wang, Samia Menon, Tao Long, Keren Henderson, Dingzeyu Li, Kevin Crowston, Mark Hansen, Jeffrey V. Nickerson, and Lydia B. Chilton. 2024. ReelFramer: Human-AI Co-Creation for News-to-Video Translation. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3613904.3642868

[46] Sitong Wang, Zheng Ning, Anh Truong, Mira Dontcheva, Dingzeyu Li, and Lydia B. Chilton. 2024. PodReels: Human-AI Co-Creation of Video Podcast Teasers. In *Proceedings of the 2024 ACM Designing Interactive Systems Conference (DIS '24)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3643834.3661591

[47] Sitong Wang, Savvas Petridis, Taeahn Kwon, Xiaojuan Ma, and Lydia B. Chilton. 2023. PopBlends: Strategies for Conceptual Blending with Large Language Models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. Association for Computing Machinery, New York, NY, USA, 435:1–435:19. https://doi.org/10.1145/3544548.3580948

[48] Saelyne Yang, Anh Truong, Juho Kim, and Dingzeyu Li. 2025. VideoMix: Aggregating How-To Videos for Task-Oriented Learning. In *Proceedings of the 30th International Conference on Intelligent User Interfaces*. 1564–1580.

[49] Lixiu Yu, Aniket Kittur, and Robert E Kraut. 2014. Distributed analogical idea generation: inventing with crowds. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 1245–1254.

[50] Lixiu Yu, Aniket Kittur, and Robert E Kraut. 2014. Searching for analogical ideas with crowds. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1225–1234.

A PROMPTS USED IN SCHEMEX

A.1 Get clusters

```
I'm learning how to {content_type}.
Analyze the following examples to identify clusters
based on STRUCTURAL and RHETORICAL patterns.
Focus on how the content is constructed and
presented—not just what it's about.
For textual examples, examine rhetorical organization.
For multimodal examples, consider how different
modalities interact structurally and rhetorically.

For each cluster:
Name the structural approach (not just the topic)
List 2–3 shared structural features
Include the example IDs in that cluster

Clustering Rules:
Assign every example to exactly one cluster
Use all examples—no omissions or duplicates
Group based on structure, not content themes or ID
order

Response Format:
Cluster 1: [Cluster Name]
Common Features:
- [Feature 1]
- [Feature 2]
Examples:
- Example [ID]
- Example [ID]
...
Total number of examples: [Number]

Cluster 2: [Cluster Name]
...
Examples to analyze: {examples}{input_context}
```

A.2 Get the feature-example matrix

```
I'm learning how to {content_type}.
Help me analyze how each example in the cluster
{cluster_name} demonstrates the identified common
features.
Common Features: {common_features}
Examples: {examples}

Your Task:
For each example:
Indicate whether it demonstrates each feature:
"Yes" = Clearly demonstrates the feature
"Partial" = Feature is present but limited, implied,
or modified
"No" = Feature is not present or is inconsistent with
the definition

Include:
A brief explanation
A direct snippet from the example (if marked "Yes" or
"Partial")

Format Requirements:
Return your output as a valid JSON object with the
following structure:
{
  "mapping": [
    {
      "example_index": "EXAMPLE_ID",
      "example_snippet": "First 50 characters of the
      example...",
      "feature_mapping": [
        {
          "feature": "Feature 1",
          "feature_id": "F1",
          "applies": "Yes/No/Partial",
          "explanation": "Brief explanation of
          classification",
          "snippet": "Verbatim quote from example that
          demonstrates the feature"
        },
        ...
      ]
    },
    ...
  ]
}
```

A.3 Infer dimensions and get the dimension-example matrix

```
I'm learning how to {user_goal}.
Analyze the following examples from the cluster:
{cluster_name}
```

```
Examples: {examples_str}{input_context}

Your Task:
Help me infer the structural commonalities that
define this cluster. Focus on identifying the key
content dimensions that shape the narrative and
composition.

Important Constraints — AVOID OVERFITTING:
Only identify dimensions shared across multiple
examples
Focus on cluster-wide structural patterns, not
individual quirks
Broaden dimensions only if needed to make them general
enough for the cluster

For Each Dimension:
Give it a clear, descriptive name
Briefly explain what the dimension captures
Include verbatim snippets from examples that
demonstrate it
For Each Example:
For every dimension, show:
Whether it applies: Yes, Partial, or No
A short explanation of why
A verbatim snippet from the example (if marked "Yes")
If no snippet can be found, use "Partial" or "No"
Snippet Rules:
Only use exact text from the examples—no paraphrasing
or made-up text
If a snippet can't be found, say so—never fabricate
one

Applies Judgment:
Yes = Strong, specific snippet available
Partial = Present, but limited or implicit
No = Absent or not applicable

Output Format:
Return your response as a valid JSON object.
{
  "dimensions": [
    {
      "name": "Dimension name",
      "description": "Brief explanation",
      "examples": [
        {
          "example_id": "1",
          "snippet": "Verbatim snippet from Example 1"
        },
        ...
      ]
    },
    ...
  ],
  "example_mappings": [
    {
      "example_id": "1",
      "dimension_applications": [
        {
          "dimension": "Dimension name",
          "applies": "Yes/No/Partial",
          "explanation": "Brief explanation",
          "snippet": "Exact text from this example"
        },
        ...
      ]
    },
    ...
  ]
}
```

A.4 Infer dimension-specific attributes and get the attribute-example matrix

```
I'm learning how to {user_goal}.
Please analyze the following examples from the
cluster: {cluster_name}
Examples: {examples_full_text}{input_context}

PART 1: IDENTIFY DIMENSION ATTRIBUTES
You've already identified the following dimensions:
Dimensions: {dimensions_text}
Your task now is to define key attributes under each
dimension, based on patterns that are consistent
across the examples in this cluster.
Use only the following example IDs: Example IDs:
{example_ids_text}

Attribute Requirements:
Each attribute must appear in at least 50% of the
examples
Keep attributes broad enough to apply across examples,
but still specific and observable
Exclude attributes that are too narrow, ambiguous, or
inconsistently implemented
Evaluate every example against every attribute—no
omissions

For Each Dimension, Provide:
Detailed Attributes: Clear, concrete, 1-sentence
descriptions of each attribute
Concise Summaries: 1–2 word phrases summarizing each
attribute (in the same order)

PART 2: ATTRIBUTE IMPLEMENTATION IN EXAMPLES
For every attribute in Part 1, assess how each example
implements it.
Use the following classifications:
"YES" — Clearly and fully present (must include a
direct quote)
"PARTIAL" — Present but limited, modified, or implicit
"NO" — Not present or structurally inconsistent with
the attribute
```

```
Implementation Guidelines:
Always use the actual example IDs from the dataset
Provide a verbatim quote from the example if the
classification is “YES” (never paraphrase)
Include an explanation for every example, even if “NO”
Never fabricate text—if you can’t find a quote, use
"PARTIAL" or "NO"
Evaluate every example for every attribute, even if
it’s “NO”

Critical Format Requirements (Strict JSON Only):
{
  "dimensions": {
    "Dimension Name": {
      "detailed": ["Detailed attribute 1", "Detailed
      attribute 2", ...],
      "concise": ["Summary 1", "Summary 2", ...]
    }
  },
  "attributes_examples": {
    "Dimension Name": {
      "Detailed attribute 1": [
        {
          "example_id": "ACTUAL_ID_1",
          "quote": "Exact quote from example",
          "explanation": "How this quote demonstrates
          the attribute",
          "classification": "YES"
        },
        {
          "example_id": "ACTUAL_ID_2",
          "quote": "",
          "explanation": "Why this example does not
          include this attribute",
          "classification": "NO"
        }
      ],
      ...
    }
  }
}
```

A.5 Infer overall attributes and get the attribute-example matrix

```
I’m learning how to {user_goal}.
Please analyze the following examples to identify
overall structural patterns that are consistent
across most of them.
Examples: {examples_full_text}{input_context}
Use only these example IDs in your analysis: Example
IDs: {example_ids_text}

Goal: Identify Overall Attributes
Your task is to identify broad structural or
compositional attributes that appear across the
majority of examples, regardless of individual
dimensions.
Look for consistent patterns in areas like:
Length – Word/sentence count range that appears in
most examples
Format – Layout elements (e.g. paragraph count,
headers, visual markers)
Tone – Overall voice, formality, or affective stance
Organization – Common sequence or structure across
examples
Other global traits – Any other recurring patterns
seen in most examples
You must include at least one attribute about length
or format, and one about organization.

Attribute Guidelines
Each attribute must appear in at least 50% of the
examples
Keep attributes broad enough to apply across many
examples
Ground attributes in observable patterns (not
interpretations)

For Each Attribute:
Provide a detailed description: a one-sentence
explanation of the observable pattern
Provide a concise summary: a 1–2 word label for each
attribute
The number of concise summaries must match the number
of detailed attributes.

Attribute Application per Example
For each attribute, evaluate every example:
YES = Attribute is clearly and fully present (must
include direct quote)
PARTIAL = Attribute is present but limited, modified,
or implicit
NO = Attribute is absent or structurally inconsistent

For every example, include:
Verbatim quote from the example (if applicable)
A brief explanation of your classification
Classification: "YES", "PARTIAL", or "NO"

STRICT FORMAT (DO NOT DEVIATE):
{
  "overall_attributes": {
    "detailed": ["Detailed attribute 1", "Detailed
    attribute 2", "Detailed attribute 3"],
    "concise": ["Concise1", "Concise2", "Concise3"]
  },
  "overall_attributes_examples": {
    "Detailed attribute 1": [
      {
        "example_id": "ACTUAL_ID_1",
        "quote": "Exact quote from example",
        "explanation": "Why this example demonstrates
```

```
        the attribute",
        "classification": "YES"
      },
      {
        "example_id": "ACTUAL_ID_2",
        "quote": "",
        "explanation": "Why this example does not
        include the attribute",
        "classification": "NO"
      },
      ...
    ],
    "Detailed attribute 2": [...],
    "Detailed attribute 3": [...]
  }
}
```

A.6 Get dimension values

```
I am trying to {current_user_goal}.
Given the following input: {input_text}
I need you to generate only one component of the
output—not the full output.
Component to generate: {dim_name} – {dim_description}
Component Requirements: {attributes_text}

Important:
Focus only on generating the specified component.
Do not include other parts of the output.
Follow the given requirements closely.
```

A.7 Get contrasting examples

```
I am trying to {current_user_goal}.
Given this input: {input_text}
And these dimension values: {dimensions_text}
Help me {current_user_goal} by generating a complete
output.

Make sure to:
Integrate the dimension values effectively
Satisfy the following overall requirements:
{overall_arrtibutes}
Output the final content directly.
```

A.8 Get comparative analysis

```
Your task is to analyze the gap between a generated
output and a gold/reference example, in order to
improve the schema itself.

Goal:
Your focus is on identifying what the schema is
missing that would help the generated output
better align with the gold example in more
meaningful and sophisticated ways.

Inputs:
Schema: {schema_text}
Dimension Values: {dimension_values_text}
Generated Output: {generated_output}
Gold Example (Reference): {gold_example}

For Each Dimension:
Your analysis should address:
Patterns or qualities in the gold example that are not
captured in the current schema
Gaps between the generated output and the gold example
that a stronger schema could help bridge
Deeper or more nuanced attributes that could make the
schema more generative, precise, and aligned with
high-quality outputs

Think About:
What deeper structural or rhetorical qualities exist
in the gold example that is missing from the
schema?
Where is the schema too shallow, vague, or rigid?
How can the schema be refined or extended without
overfitting?

For Each Dimension, Suggest 2–3 Improvements, Using
these Tags:
[ADD] – Introduce a new attribute or pattern
[DEEPEN] – Make an existing attribute more
sophisticated or layered
[REFINE] – Clarify or better define an existing
attribute
[RESTRUCTURE] – Reorganize or reconceptualize the
dimension
Also, include a section for Overall
(cross-dimensional insights).

Output Format (STRICT):
Return your response as valid JSON using the following
structure.
{
  "dimension_analysis": {
    "Dimension Name 1": {
      "analysis": "Analysis of deeper patterns in this
      dimension that aren't captured by the
      schema",
      "improvements": [
        "[ADD] Description of a new attribute",
        "[DEEPEN] Suggestion for making an existing
        attribute more sophisticated",
        "[REFINE] Clarify an attribute definition",
       "[RESTRUCTURE] Suggest restructuring if needed"
      ]
    },
    "Dimension Name 2": {
      "analysis": "Analysis of missing or underdefined
      qualities relevant to this dimension",
      "improvements": [
```

```
        "[ADD] ...",
        "[DEEPEN] ..."
      ]
    },
    "Overall": {
      "analysis": "Insights about patterns or
      improvements needed across dimensions",
      "improvements": [
        "[ADD] Cross-dimensional schema improvement",
        "[RESTRUCTURE] Suggest schema-wide
        restructuring"
      ]
    }
  }
}
```

A.9 Get color coding

```
Your task is to create a detailed, structured
comparison between two texts using schema
dimensions to guide your analysis.
TEXTS TO COMPARE:
TEXT 1 (Generated Output): {generated_output}
TEXT 2 (Gold Example): {gold_example}
SCHEMA DIMENSIONS: {schema_text}

TASK:
Segment both texts and align corresponding parts
side-by-side. For each segment:
Indicate which schema dimension (if any) the segment
primarily reflects
Provide a brief annotation that explains the quality,
relevance, or difference of this segment compared
to its counterpart
Include the start and end character indices from the
original source text
Set "dimension": null if the segment doesn't clearly
relate to any dimension
Rate the importance of this segment as "high",
"medium", or "low" based on its role in conveying
schema-aligned meaning or structure

CRITICAL REQUIREMENTS:
Cover the full content of both texts—no omissions
allowed
Do not duplicate content across segments—each
character should appear in exactly one segment
Accuracy of start_index and end_index is
essential—these must correspond exactly to the
characters in each original text
Use actual JSON null (not the string "null") when a
segment doesn't map to any dimension

OUTPUT FORMAT (Strict):
{
  "segments": [
    {
      "id": "segment_1",
      "source": "generated" | "gold",
      "text": "text of the segment",
      "start_index": 0,
      "end_index": 42,
      "dimension": "Dimension Name" | null,
      "annotation": "Brief analysis of the segment",
      "importance": "high" | "medium" | "low"
    },
    ...
  ],
  "dimension_analysis": {
    "Dimension Name 1": "Summary analysis of how this
    dimension compares across the two texts",
    "Dimension Name 2": "Additional dimension
    summary...",
    ...
  }
}
```

A.10 Get iterated schema

```
You are helping improve a schema for AI-guided
generation.
User Goal: {user_goal}
Context: {context_text}
The original schema includes:
Dimensions (each with detailed + concise attributes)
Overall attributes (for general output quality)

Your Task:
Use the suggestions below to iterate and improve the
schema by:
Revising dimension attributes for clarity, depth, or
generality
Adding new attributes where necessary
Improving overall attributes to enhance global output
quality
Removing or simplifying overly specific attributes
that may cause overfitting

Use These Inputs:
Suggested Improvements: {all_suggested_improvements}
Original Schema: {original_schema}

Critical Format Requirements:
For every dimension:
"detailed" and "concise" arrays must exist
Every "detailed" attribute must have a corresponding
"concise" one, and vice versa
Follow the exact same JSON structure as the original
schema

Final Output:
Return a valid JSON object with the same structure as
the original schema, reflecting all suggested
improvements.
```

B PROMPTS USED FOR O1-PRO IN TECHNICAL EVALUATION

```
I want to learn how to {content_type}.

Here are {number_of_examples} examples. Can you help
me infer the schema (structural commonalities)
among the examples? So that I can use them later
to create my own. You can give me multiple schemas
if there are subgroups amongst the examples - each
subgroup has one different schema.

For each schema, please outline the key dimensions
(characteristic features), their corresponding
attributes (descriptors of the dimension), and
overall attributes (general descriptors of the
structure and content of the final output).

Give me the answer in the format: for each schema,
give it a descriptive name, list corresponding
examples, overall attributes, dimensions, and
attributes for dimension.

Examples to analyze: {examples}{input_context}
```