# Textual-to-Visual Iterative Self-Verification Slide Generation Agent

Anonymous ACL submission

## Abstract

The generation of presentation slides is a time-consuming and repetitive task that demands automation to improve efficiency and reduce manual effort. Due to their limited flexibility and lack of automated refinement mechanisms, existing autonomous agents face constraints in real-world applicability. In this work, we decompose the task of generating missing slides into two key components—**content generation** and **layout generation**—which correspond respectively to determining what information should be conveyed and how it should be visually organized. For content generation, we introduce an approach that enhances coherence and relevance by incorporating context from surrounding slides and leveraging section retrieval strategies. For layout generation, we propose a novel **textual-to-visual self-verification process** with an **LLM-based Reviewer + Refiner workflow**, converting complex textual layouts into intuitive visual formats. This modality transformation simplifies the task, enabling accurate and human-like review and refinement. Experiments show that our approach significantly outperforms baseline methods in terms of alignment, logical flow, visual appeal, and readability.

## 1 Introduction

Effectively summarizing and presenting research findings through academic presentation slides is an essential part of scientific communication, allowing researchers to present key contributions and engage audiences at conferences and seminars (Guo et al., 2024; Mondal et al., 2024). However, creating these slides is a time-consuming process that requires extracting core information from lengthy papers, organizing it coherently, and designing visually consistent layouts across multiple slides (Fu et al., 2021). With the rapid growth in the volume of research, the demand for automated solutions has increased significantly. Recent advances in large language models (LLMs) (OpenAI, 2023; Touvron et al., 2023; Templeton et al., 2024) have demonstrated remarkable capabilities in mimicking human behavior for complex tasks (Hong et al., 2023; Park et al., 2023; Yao et al., 2022b; Zala et al., 2024; Ma et al., 2024a) beyond text generation (Yao et al., 2022b,a; Xi et al., 2024; Yang et al., 2024). Building on these strengths, LLM-based agents offer a promising opportunity to automate tasks like slide generation (Zheng et al., 2025), reducing manual effort while ensuring coherence and visual quality.

Despite its potential, generating high-quality academic presentation slides presents two major challenges: **how to assign reasonable and adaptive layouts for generated content** and **how to ensure layout quality and consistency**.

The first challenge lies in generating layout information that adapts to the unique visual structure for different textual contents. Some methods focus solely on textual content, neglecting structural aspects like positioning, spacing, and alignment, leading to impractical outputs (Sun et al., 2021; Bandyopadhyay et al., 2024). Existing rule-based methods provide a quick and straightforward solution by populating predefined slots with generated content. However, they overlook the unique structural style of each presentation, often leading to rigid layouts that break the visual coherence.

The second challenge lies in achieving consistent textual-visual results, complicated by the inherent difficulty of representing slide layouts in structured textual formats. Unlike visual representations, where spatial relationships and element alignment are easy to interpret, textual formats lack this visual clarity (Xu et al., 2024; Hu et al., 2024). This makes it difficult for models to fully understand the spatial and structural aspects of slide design, leading to frequent errors such as text overflow, misalignment, and inconsistent spacing.

Furthermore, correcting these errors directly in

the textual format is non-trivial. Without a visual reference, detecting overlapping elements or misalignments becomes challenging, particularly in slides with complex layouts.

The key component of our framework is a textual-to-visual iterative self-verification process to refine initial output. The initial slide layouts are generated in a textual format, which—while structured and machine-readable—often contains errors due to the complexity of representing slide information in a non-visual form. Additionally, reviewing and refining these layouts in their original format is challenging and unintuitive. To address this, we introduce a **modality transformation** (Li et al., 2025) that converts the textual format into a visualized form. This transformation significantly reduces the complexity of the task, making it easier for the LLM-based Reviewer + Refiner workflow to detect and correct issues such as alignment and text overflow in a human-like, intuitive manner. The reviewer provides feedback by analyzing the visual representation of the slide layout. The feedback is then passed to the refiner, who applies the suggested adjustments to the structured layout in textual format. This iterative refinement process ensures higher-quality final outputs with improved coherence and visual consistency.

Our key contributions are as follows:

- An agentic framework for slide generation including content and layout generation approaches, ensuring thematic consistency and visual coherence.
- A textual-to-visual iterative self-verification process with modality transformation, enabling intuitive and accurate refinement for slide layout.
- Extensive analyses and systematic evaluation, demonstrating the precision and effectiveness of our framework for automated academic slide generation.

## 2 Related Work

In this section, we introduce the background of the LLM-based agent and existed studies on slides generations.

### 2.1 LLM-based Agent

LLMs have demonstrated impressive capabilities for complicated, interactive tasks (Yao et al., 2022b,a; Xi et al., 2024; Yang et al., 2024; Ma et al., 2024b). LLM-based autonomous agents have achieved remarkable progress in a wide range of domains, including logic reasoning (Qi et al., 2024; Khattab et al., 2022), tool use (Qin et al., 2024; Zhang et al., 2023a), and social activities (Park et al., 2023). The current paradigm of agents relies on the language intelligence of LLMs. The mainstream work pattern encompasses environment perceiving, planning, reasoning, and executing, forming a workflow to dive and conquer intricate challenges.

Empowered by the recent progress of multimodal pre-training, those agents can understand image, video, and audio channels (Wu et al., 2023; Liu et al., 2023). (i) Visual knowledge can largely facilitate reasoning and is integrated into Chain-of-Thoughts (Zhang et al., 2023b; Xu et al., 2024). (ii) Multi-modal reasoning enables divergent thinking cross modalities and takes advantage of those different modalities. Sketchpad (Hu et al., 2024) allows LLMs to draw drafts to assist its planning and reasoning, i.e., to draw auxiliary lines for geometry problems. Visualization-of-Thought (Wu et al., 2024) generates visual rationales for spatial reasoning tasks like mazes. For each stage of complex multi-modal tasks, selecting an appropriate modality as the main modality for reasoning can leverage the natural characteristics of the modality and stimulate the potential of LLMs (Park et al., 2025).

### 2.2 Slide Generation

Previous studies have explored extractive methods and simplified this task as sentence selection, e.g., to calculate the importance score and extract top sentences (Wang et al., 2017). With the development of small language models (Lewis et al., 2020; Raffel et al., 2020), slide generation is unified as abstractive, query-based document summarization (Sun et al., 2021).

Despite their early success, the emergence of LLMs exhibits exceptional performance and stimulates the demands of intelligent slide generation. Slide generation poses intricate challenges for autonomous agents, as it requires document reading comprehension and precise tool use to generate layouts. Pioneer work focuses on modifying target elements, asking agents to execute a series of specific instructions (Guo et al., 2024). The agent needs to understand the status of the slide, navigate to the element, and generate precise API calls. Recent studies first plan the outlines and then generate

each page. To further control the style of presentations, Mondal et al. (2024) introduce a reward model trained on human feedback to guide both topic generation and content extraction. Considering the visual quality of slides, Bandyopadhyay et al. (2024) employ a visual LM to insert images. DOC2PPT (Fu et al., 2021) integrates an object placer to predict the position and size of each element by training small models. PPTAgent (Zheng et al., 2025) directly utilizes slide templates to fix the layout and then fill textboxes, ensuring visual harmony and aesthetic appeal.

## 3 Methodology

In this section, we propose an LLM-based agentic workflow to automate the generation of content and layout for academic paper slides.

### 3.1 Task Formulation

We first formally define our slide generation task. In this task, a presentation is represented as a collection of slide pages, where each page consists of multiple elements. Each element $e \in E$ is a tuple $(c, l)$, where $c$ denotes the content (e.g., text, images, tables) and $l$ specifies the corresponding layout information (e.g., position, size, font style).

Our **overall task** is to generate the missing slide $\hat{S}_i$ given the paper $D$, the missing slide topic $T$, and the partially available slide set $S = \{S_1, S_2, \ldots, S_n\}$.

**Input** The input consists of: 1. A paper $D = \{d_1, d_2, \ldots, d_m\}$, where $d_i$ denotes a section or paragraph in the paper. 2. A missing slide topic $T$, describing the main focus of the missing slide. 3. A partially available slide set $S = \{S_1, S_2, \ldots, S_n\}$, where some slides $\hat{S}_i$ are missing. 4. The preceding slide $S_{prev}$ and the following slide $S_{next}$ as contextual information.

**Output** The output is a structured textual file $\hat{S}_i$, which describes the missing slide, including both content $c$ and layout information $l$ for each element $e \in E$. Formally,

$$\hat{S}_i = \{e_j = (c_j, l_j) \mid j = 1, 2, \ldots, k\}$$

where $k$ is the number of elements in the generated slide. The generated textual file can be directly converted into a PowerPoint slide.

### 3.2 Slide Generation Framework

The process of creating a presentation typically involves two key stages: (1) identifying the core content that needs to be presented on each slide, and (2) arranging this information into a visually coherent and consistent layout.

The goal of content generation is to generate $c_j$ for each element $e_j$ based on the paper $D$, the missing slide's topic $T$, and contextual information from the surrounding slides $S_{prev}$ and $S_{next}$:

$$c_j = \mathcal{G}_{\text{content}}(D, T, S_{prev}, S_{next})$$

Here, $\mathcal{G}_{\text{content}}$ represents the content generation process, ensuring that the generated content is accurate, concise, and contextually relevant.

The layout generation task determines the layout $l_j$ for each element $e_j = (c_j, l_j)$ to maintain visual consistency and readability. The initial layout draft $l_j^{(0)}$ is generated using the content $c_j$ and contextual information from the surrounding slides:

$$l_j^{(0)} = \mathcal{G}_{\text{layout_draft}}(c_j, S_{prev}, S_{next})$$

To refine the initial layout, a textual-to-visual iterative self-verification process is applied. The layout at step $k$ ($l_j^{(k)}$) is visualized as Image($l_j^{(k)}$), allowing the LLM-based Reviewer + Refiner workflow to provide feedback and corrections:

$$l_j^{(k+1)} = \mathcal{G}_{\text{refine}}\left(l_j^{(k)}, \text{Image}(l_j^{(k)})\right)$$

This iterative process continues until the layout reaches the desired quality and visual coherence.

#### 3.2.1 Content Generation

Determining the key contents on a slide page involves understanding paper structures, extracting critical texts and figures, and ensuring overall coherence for a logical flow and consistent style.

Our content generation stage adopts a multi-step process with three sub-modules: Text Retriever, Figure Extractor, and Content Generator, consisting of a pipeline to identify relevant text segments, recommend figures and tables, and then decide the contents to present.

**Text Retriever** We build a text retriever to retrieve the most relevant sections of the paper. The paper is divided into section-level granularity, with each segment represented and indexed as a dense embedding. Given the topic of a slide, the retriever selects the most relevant segments by calculating the cosine similarity between the dense embeddings of the slide topic and the indexed sections.

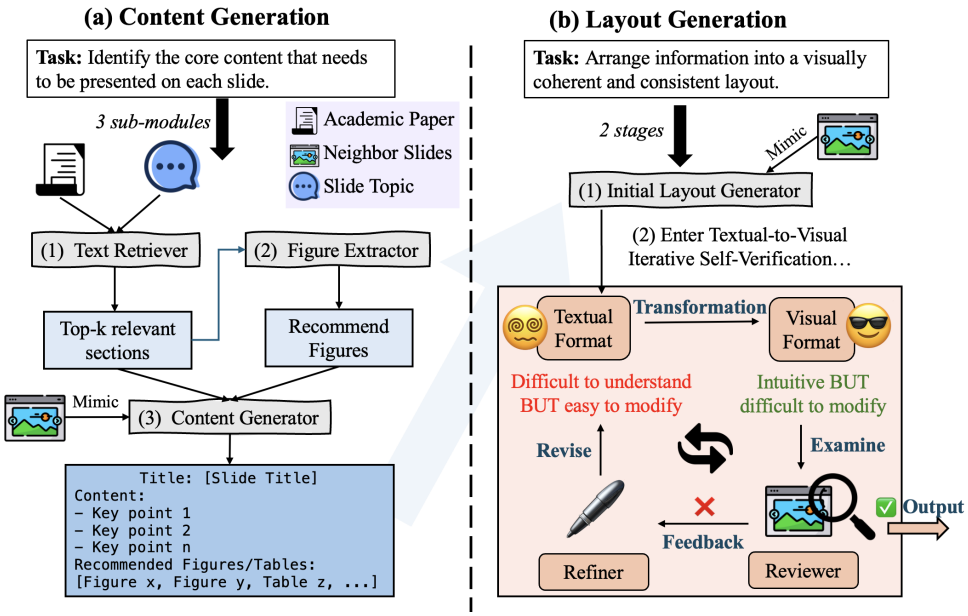


Figure 1: Overall Framework

**Figure Extractor** Beyond the retrieved text, the Figure Extractor identifies candidate figures to support slide content. It scans the top-$k$ retrieved text segments for explicit references (e.g., "Figure 1", "Table 2"), and extracts the corresponding captions from the paper.

At this stage, only candidate figures are collected; the final selection is made by the Content Generator based on the slide content.

**Content Generator** Given the related text segments and candidate figures provided by previous modules, the LLM agent performs three sub-tasks.

First, it generates slide text aligned with the slide's topic and context. Second, it selects the most relevant figures or tables from the candidate set. Finally, it incorporates adjacent slides to maintain logical flow and ensure seamless transitions.

The Content Generator's output is then used for Layout Generation, which focuses on organizing content into a visually coherent slide.

### 3.2.2 Layout Generation

Slide layouts need to be flexible and controllable, rather than fully randomized or constrained by rigid templates. However, generating adaptive layouts is challenging and prone to issues such as text overflow, misalignment, and inconsistent spacing, especially when handling diverse content and styles.

To address this, we design a **textual-to-visual iterative self-verification process**. The initial layout draft mimics surrounding slides for style consistency but remains difficult to review in its structured textual format. We design an LLM-based *Reviewer* + *Refiner* workflow that validates and refines the layout respectively, improving accuracy and coherence through iterative corrections.

**Stage 1: Initial Layout Generation** The initial attempt is conducted by directly asking the LLM to arrange the layout for each element of the generated contents, specifying each element's position, size, font, and color. We also append surrounding slide pages as demonstrations and carefully optimize the prompt to instruct the LLM to mimic their layout patterns for a visually consistent design. The layout is normalized as a JSON format.

While this initial layout serves as a foundation, our pilot experiments show that several factors contribute to potential errors:

(i) Textual slide layout is inherently complex, requiring detailed key-value pairs for positions, sizes, fonts, and colors. Any inconsistency in this structured data can cause significant visual defects.

(ii) LLMs lack direct visual feedback and cannot accurately assess how the generated layout will appear in its final form. Unlike models specifically trained for visual tasks, LLMs rely on textual context and structural patterns to predict layout information. This process is inherently limited, as it depends heavily on imitation and pattern recognition without understanding visual balance or spatial relationships. Consequently, the generated layouts may exhibit issues such as poor alignment, over-

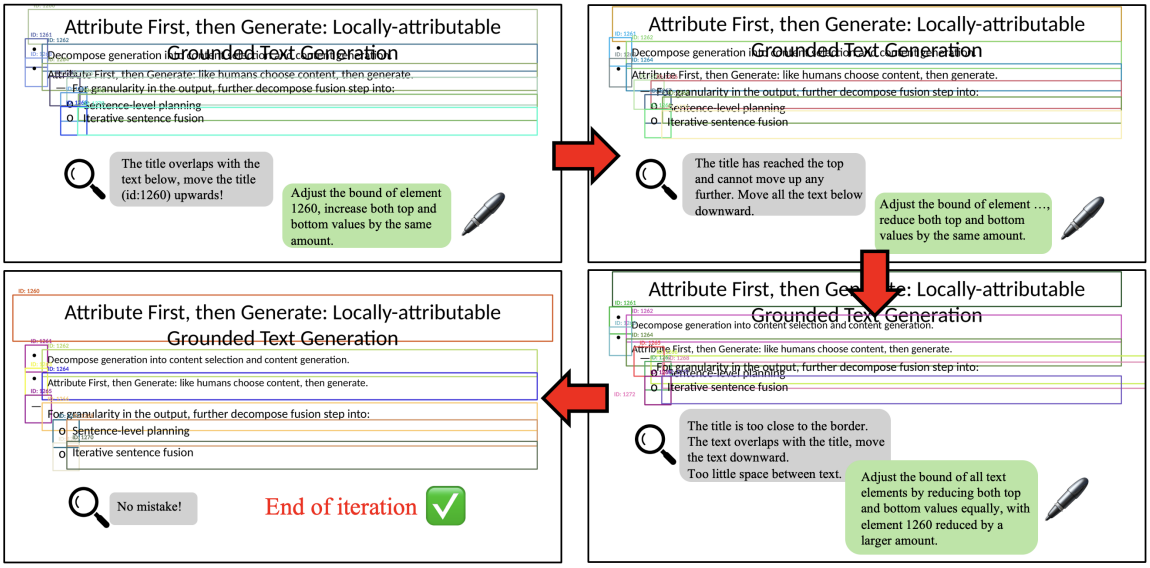


Figure 2: Iterative Layout Refinement in the Reviewer + Refiner Workflow

lapping elements, or inconsistent spacing, which require further refinement to ensure high-quality results.

**Stage 2: Textual-to-Visual Iterative Self-Verification** To refine the initial layout, we introduce a self-verification process that combines modality transformation and a LLM-based agentic workflow.

**1. Modality Transformation** We first convert the initial LLM layout output, which is in JSON format, into a visualized slide. While the JSON format allows easy modifications, it is not visually intuitive. To address this, the layout is rendered as an image with each visual element enclosed in a colored bounding box, labeled with a unique ID that corresponds to the original JSON elements. This visualization simplifies the review process by allowing the Reviewer to quickly reference specific elements and identify potential issues.

**2. Reviewer** The Reviewer simulates how a human would evaluate slide quality, following a predefined set of evaluation criteria and adjustment rules. Specifically, it performs the following tasks: Object overlapping detection, Image quality and distortion analysis, Element bounding and text overflow correction, Element positioning and alignment, Text formatting consistency and Overall composition and visual balance.

Each recommendation is output as a structured list of suggestions, identifying specific elements by their **ID** and providing precise numerical values for adjustments. For example, the Reviewer might suggest increasing a text box's height by 1.2x to accommodate overflowing text or shifting an image downward by 10% of its height to resolve an overlap. Such a definite, specific advice format makes it easier for the Refiner to implement precise corrections in the subsequent refinement stage.

**3. Refiner** The Refiner plays a role for execution, translating the Reviewer's visual feedback into precise modifications within the textual layout. To ensure accurate modifications, the Refiner follows a set of predefined rules based on the type of feedback received. For example, when the Reviewer suggests repositioning an element, the Refiner adjusts its bounding box coordinates accordingly while ensuring it remains within slide boundaries. Each rule is applied systematically based on the Reviewer's feedback. The Refiner's task is to modify only the necessary fields while maintaining the basic structure, resulting in a complete and refined file that reflects the intended adjustments. Details provided in the Appendix A.

**Stage 3: Integration and Rendering** The final output of this process is a refined JSON-formatted layout description that accurately represents the corrected slide. This JSON is passed to the rendering module to produce the final PowerPoint slide, ensuring that the layout visually reasonable and aligns with the overall presentation style.

| LLM | Method | Coverage | ROUGE-1 | | | ROUGE-2 | | | ROUGE-L | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | P | R | F1 | P | R | F1 | P | R | F1 |
| | | | **Baseline Methods** | | | | | | | | |
| - | D2S | 24.38 | 18.30 | 30.31 | 20.47 | 4.73 | 7.79 | 5.26 | 16.86 | 27.21 | 19.08 |
| GPT-4o | GPT-4o-Image | 22.70 | 18.72 | 31.82 | 19.84 | 5.51 | 9.92 | 6.26 | 14.44 | 26.52 | 15.77 |
| GPT-4o | GPT-4o-LaTeX | 26.55 | 29.79 | 32.02 | 24.19 | 10.27 | 13.09 | 9.84 | 23.19 | 28.33 | 19.86 |
| **LLaMA3.1-8B** | Vanilla | 29.81 | 24.56 | 47.74 | 28.02 | 8.94 | 19.96 | 10.34 | 17.54 | 37.58 | 20.46 |
| | w/o Retriever | 28.18 | 30.06 | 42.04 | 29.35 | 12.44 | 19.45 | 12.54 | 23.19 | 34.85 | 22.99 |
| | w/o Neighbor | 32.36 | 25.31 | 42.31 | 26.79 | 9.78 | 19.03 | 10.72 | 19.00 | 34.07 | 20.42 |
| | Ours (3S) | 32.76 | 28.64 | 39.30 | 27.47 | 11.23 | 17.13 | 11.15 | 21.99 | 32.18 | 21.36 |
| | Ours (5S) | 31.18 | 28.52 | 42.63 | 28.40 | 11.38 | 19.33 | 11.68 | 21.76 | 34.99 | 21.97 |
| **Qwen2.5-7B** | Vanilla | 25.28 | 24.27 | 44.92 | 26.02 | 9.06 | 19.69 | 10.10 | 17.89 | 36.24 | 19.65 |
| | w/o Retriever | 26.18 | 31.47 | 36.77 | 27.92 | 12.60 | 17.11 | 11.60 | 24.66 | 30.39 | 22.14 |
| | w/o Neighbor | 30.08 | 24.13 | 44.93 | 25.91 | 9.01 | 19.69 | 10.06 | 17.78 | 36.26 | 19.57 |
| | Ours (3S) | 28.79 | 29.78 | 36.26 | 25.99 | 11.63 | 16.58 | 10.56 | 24.17 | 30.76 | 21.21 |
| | Ours (5S) | 27.67 | 28.31 | 37.17 | 26.01 | 10.29 | 15.71 | 9.87 | 21.60 | 30.21 | 20.18 |
| **GPT-4o** | Vanilla | 28.41 | 23.29 | 43.97 | 25.65 | 7.15 | 16.86 | 8.20 | 16.23 | 34.09 | 18.31 |
| | w/o Retriever | 29.01 | 32.48 | 37.68 | 28.36 | 11.15 | 15.88 | 10.05 | 24.45 | 30.35 | 21.64 |
| | w/o Neighbor | 28.81 | 29.11 | 34.60 | 26.13 | 10.18 | 15.43 | 9.61 | 22.79 | 29.21 | 20.88 |
| | Ours (3S) | 29.49 | 31.63 | 32.86 | 26.10 | 11.30 | 14.91 | 9.84 | 24.34 | 27.81 | 20.76 |
| | Ours (5S) | 29.41 | 31.75 | 37.68 | 28.39 | 10.89 | 15.71 | 10.28 | 24.09 | 30.60 | 21.97 |
| **DeepSeek-R1** | Vanilla | 27.30 | 24.21 | 42.34 | 26.14 | 7.96 | 17.81 | 9.15 | 17.56 | 33.65 | 19.40 |
| | w/o Neighbor | 28.56 | 24.19 | 37.99 | 24.53 | 8.39 | 16.65 | 8.99 | 18.10 | 30.88 | 18.74 |
| | w/o Retriever | 25.35 | 28.73 | 39.26 | 27.53 | 9.94 | 17.23 | 10.12 | 21.29 | 31.27 | 20.86 |
| | Ours (3S) | 28.66 | 27.64 | 38.41 | 26.77 | 9.68 | 17.02 | 9.98 | 20.92 | 31.72 | 20.77 |
| | Ours (5S) | 28.65 | 27.09 | 37.51 | 26.02 | 9.43 | 16.28 | 9.68 | 20.69 | 30.80 | 20.33 |

Table 1: Evaluation results of content generation.

# 4 Experiments

## 4.1 Dataset Construction

The dataset consists of aligned slide–paper pairs collected from the ACL 2024 In-Person Poster Session 1 on the Underline. Each pair includes a research paper and its corresponding slides, both in PDF format.

For **paper processing**, we extract structured text using GROBID (Kermitt2, 2020) to obtain section-level hierarchy, paragraphs, titles, and references. Figures and captions are extracted using PDFFigures 2.0 (Clark and Divvala, 2016). For **slide processing**, we parse each slide into element-level components (e.g. text boxes, images), extracting their textual content, font styles and layout attributes.

## 4.2 Settings and Implementation

We evaluate two categories of baselines:

(i) Traditional methods, which represent early approaches to document-to-slide generation. **D2S** (Sun et al., 2021) adopts a two-stage pipeline: first retrieving relevant content using slide titles, then summarizing it into bullet points via long-form QA. **Doc2ppt** (Fu et al., 2021) formulates the task as end-to-end generation using a hierarchical seq2seq model that jointly predicts content and layout.

(ii) End-to-end generative methods, where GPT-4o directly produces presentation slides without intermediate structure. **GPT-4o-Image** generates full slide images in a single step, based on slide titles and retrieved content. The output is purely visual and not editable. **GPT-4o-LaTeX** outputs Beamer-style LaTeX code that can be compiled into slides, offering a code-based approach.

We implement our framework using Python 3.10. Experiments are conducted on an Ubuntu 20.04 server equipped with NVIDIA A800-SXM4 GPUs.

In the content generation stage, we evaluate the performance of four large language models: Llama-31-8B-Instruct (Grattafiori et al., 2024), Qwen-2.5-7B (Qwen et al., 2025), GPT-4o (OpenAI et al., 2024), and Deepseek-R1 (DeepSeek-AI et al., 2025). The best-performing model is selected as the default generator for structured slide content. For the retriever, we use the **Salesforce SFR-Embedding-Mistral** retriever (Wang et al., 2024) to compute similarity scores between the

slide title and segmented sections of the paper. We conduct systematic comparisons across different configurations: **Ours**: the full pipeline using top-$k$ retrieved sections with adjacent slides, **3S / 5S** : only the top-3 or top-5 most relevant sections are used, **w/o Retriever**: the entire paper is used without retrieval, and **w/o Neighbor Slides**: the adjacent slides are excluded, leaving only the retrieved sections.

In the layout generation stage, both the *Reviewer* and *Refiner* modules are built on top of GPT-4o. To evaluate the effectiveness of different refinement strategies, we compare the following result types: **Reference Slide** (human-written): slides provided by the original paper authors, serving as the upper bound; **Vanilla**: the initial layout directly generated by the model without any refinement; **Textual-Based Refinement**: a rule-based pipeline where the Reviewer analyzes the JSON structure to detect layout issues, and the Refiner modifies the JSON file accordingly; **Ours**: by introducing **modality transformation**, we convert the layout JSON into a slide image, enabling GPT-4o to perform visual review and refinement more effectively.

### 4.3 Evaluation

Our evaluation method combines quantitative metrics and qualitative assessment to measure both content and layout generation.

**Content Evaluation** We evaluate generated slide content using ROUGE (Lin, 2004) and Coverage (Kothawade et al., 2020). ROUGE measures lexical overlap with reference slides, while Coverage computes the average cosine similarity between sentence embeddings from the source document and generated bullet points, reflecting semantic alignment. Higher Coverage indicates better semantic alignment with the original content.

**Layout Evaluation** We adopt **LLM-as-Judge**(Chen et al., 2024) to evaluate slide layout across three levels. See Appendix B for details.

- Element Level: Assesses alignment and spacing to ensure a well-structured layout.
- Slide Level: Focuses on logical flow and text-visual consistency, ensuring information is presented clearly and supported by relevant visuals.
- Overall Impression: Evaluates visual appeal and readability, ensuring cohesive design, appropriate font size, and clear figures.

### 4.4 Main Results

**Content Generation** Across all models, a clear trend emerges: decomposing slide generation into retrieval + context-aware generation significantly outperforms direct end-to-end generation. Within each model group, retrieval enhances content relevance (higher ROUGE-1/2), and adjacent slides contribute to more coherent structure and expression. Our full method not only outperforms the baseline methods (D2S and end-to-end generation) and ablation variants. Incorporating adjacent slides provides valuable contextual cues that help the model determine what has already been covered and what remains to be discussed. Moreover, this contextual grounding implicitly conveys the author's writing style and presentation logic, enabling the model to produce text that aligns better with the original tone and narrative flow. Detailed results are shown in Table 1.

**Layout Generation** For layout evaluation, Table 2 summarizes the results of layout generation.

**Vanilla**: Due to the complexity of multi-element layouts and the lack of visual feedback, the initial output without refinement often contains errors such as misalignment, overflow.

**Textual-Based Refinement**: Although this approach improves some metrics, such as Coherence (3.4), it still struggles with Visual Appeal (1.8) and Alignment (2.1), indicating the limitations of rule-based refinement without visual feedback.

**Our Method**: By introducing **modality transformation**, we convert the textual layout into a visualized slide image, allowing the Reviewer + Refiner workflow to detect and correct issues more intuitively. This approach yields significant improvements, especially in Alignment and Spacing (3.0) and Logical Flow (3.8), closely approaching the quality of the reference slides.
The results indicate that incorporating the Reviewer + Refiner workflow and modality transformation significantly improves layout quality. For additional analysis, please refer to Appendix C.

## 5 Analysis

### 5.1 Ablation

**Effect of Neighbor Slides** Neighbor slides significantly impact the quality of content generation. For instance, removing neighbor slides in Llama-31-8B (w/o Neighbor Slides) leads to a noticeable

| Result Type | Align & Space | Logic | Coherence | Visual Appeal | Readability |
|---|---|---|---|---|---|
| DOC2PPT | 3.1 | 1.9 | 3.2 | 2.0 | 3.0 |
| GPT-4o-Image | 3.3 | 3.2 | 3.0 | 2.3 | 3.3 |
| GPT-4o-LaTeX | 3.8 | 3.5 | 3.7 | 2.2 | 3.7 |
| Vanilla | 2.0 | 3.0 | 3.3 | 2.0 | 2.5 |
| Textual-Based Refinement | 2.1 | 2.6 | 3.4 | 1.8 | 2.4 |
| Ours | 3.0 | 3.8 | 3.4 | 2.8 | 3.1 |
| Reference Slide | 4.5 | 3.7 | 3.8 | 3.5 | 3.8 |

Table 2: Evaluation results of layout generation.

decrease in ROUGE-1 F1 (28.40 to 26.79) and ROUGE-2 F1 (11.68 to 10.72). Similar trends are observed in GPT-4o and Qwen2.5-7B, highlighting the importance of contextual information in maintaining logical coherence and reducing redundancy.

**Balancing Full Context vs. Section Retrieval** While using a section retriever helps reduce input length and improve efficiency, it can also cause minor variations in ROUGE scores. For example, Llama-31-8B with Section Retriever achieves slightly lower recall compared to its full-input counterpart. When provided with the full paper, they can better understand the broader context and underlying relationships, resulting in more accurate and coherent slide content.

However, in situations where the input length exceeds the model's context window or when the paper contains many irrelevant information, **Section Retrieval** is essential. Selecting an optimal number of sections (e.g. 3 vs. 5) helps balance relevance and completeness. According to the results, **Ours (5S)** generally offers better recall and F1 compared to fewer sections, as it provides more comprehensive contextual information without overwhelming the model with unnecessary details.

### 5.2 Impact of Text-to-Visual Transformation on Layout Quality

Alignment and Spacing metrics evaluate whether elements are properly positioned, evenly spaced, and free from overlap. As shown in Table 2, our method achieved a notable improvement in the Alignment and Spacing score (3.0) compared to the Baseline (2.0) and JSON-Based Refinement (2.1). Specifically, we observed that self-verification on textual layout cannot improve the layout quality, even compromise the Logic, Visual Appeal, and Readability. Our method eliminates this problem and achieves consistent improvement by introducing the textual-to-visual modality transformation.

### 5.3 Complete Presentation Generation

While our framework focuses on generating missing slides, the method can be extended to automate the generation of a complete presentation.

**Topic Generation and Slide Planning** The first step in generating a full presentation is to extract key topics from the input material. Each generated topic corresponds to a unique slide, ensuring that the presentation covers the material.

**Multi-Page Content Generation** Once the topics are generated, the framework applies the content generation strategy iteratively for each slide. By incorporating context from the previously generated slides, the model maintains logical flow and coherence across the entire presentation.

**Consistent Layout and Visual Style** The Reviewer + Refiner review process can be fully reused to ensure layout consistency across all slides.

We have conducted experiments on product documentation, validating the applicability of this method in other domains as well. Detailed input-output examples are provided in Appendix E.

## 6 Conclusion

In this paper, we propose a novel framework for generating academic presentation slides. By decomposing the task into content generation and layout generation, our method ensures adaptive layouts and visually consistent slides. We introduce a textual-to-visual iterative self-verification process using an LLM-based Reviewer + Refiner workflow, transforming complex textual layouts into visual representations for intuitive review and refinement. Experiments demonstrate that our approach significantly improves alignment, logical flow, visual appeal, and readability.

## Limitations

While our framework shows promising results in generating academic slides, it has two main limitations. First, the generated slides are primarily designed for summarizing and synthesizing content, but they lack advanced visual design features such as animations and aesthetic enhancements, which are crucial for creating more engaging presentations. Second, the framework currently relies on picture and chart elements exclusively from the provided input, which limits flexibility in incorporating external or dynamically generated visual elements.

## Ethics Statement

Our study utilizes academic papers and their associated presentation slides that are publicly accessible on official conference platforms. These materials are collected under the fair use principle and are used strictly for non-commercial, analytical research purposes. We do not use these data for model training, but only for performance evaluation.

## References


Sambaran Bandyopadhyay, Himanshu Maheshwari, Anandhavelu Natarajan, and Apoorv Saxena. 2024. Enhancing presentation slide generation by LLMs with a multi-staged end-to-end approach. In *Proceedings of the 17th International Natural Language Generation Conference*, pages 222–229, Tokyo, Japan. Association for Computational Linguistics.

Dongping Chen, Ruoxi Chen, Shilin Zhang, Yinuo Liu, Yaochen Wang, Huichi Zhou, Qihui Zhang, Yao Wan, Pan Zhou, and Lichao Sun. 2024. Mllm-as-a-judge: Assessing multimodal llm-as-a-judge with vision-language benchmark. *Preprint*, arXiv:2402.04788.

Christopher Clark and Santosh Divvala. 2016. Pdffigures 2.0: Mining figures from research papers. In *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries*, JCDL '16, page 143–152, New York, NY, USA. Association for Computing Machinery.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Tsu-Jui Fu, William Yang Wang, Daniel J. McDuff, and Yale Song. 2021. Doc2ppt: Automatic presentation slides generation from scientific documents. In *AAAI Conference on Artificial Intelligence*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Yiduo Guo, Zekai Zhang, Yaobo Liang, Dongyan Zhao, and Nan Duan. 2024. PPTC benchmark: Evaluating large language models for PowerPoint task completion. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 8682–8701, Bangkok, Thailand. Association for Computational Linguistics.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and 1 others. 2023. Cogagent: A visual language model for gui agents. *ArXiv preprint*, abs/2312.08914.

Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Ranjay Krishna. 2024. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Kermitt2. 2020. Grobid: Machine learning for extracting information from scholarly documents. https://github.com/kermitt2/grobid. Accessed: 2025-02-16.

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2022. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp. *arXiv preprint arXiv:2212.14024*.

Suraj Kothawade, Jiten Girdhar, Chandrashekhar Lavania, and Rishabh Iyer. 2020. Deep submodular networks for extractive data summarization. *Preprint*, arXiv:2010.08593.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Chengzu Li, Wenshan Wu, Huanyu Zhang, Yan Xia, Shaoguang Mao, Li Dong, Ivan Vulić, and Furu Wei. 2025. Imagine while reasoning in space: Multimodal visualization-of-thought. *Preprint*, arXiv:2501.07542.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning.

Xinbei Ma, Yiting Wang, Yao Yao, Tongxin Yuan, Aston Zhang, Zhuosheng Zhang, and Hai Zhao. 2024a. Caution for the environment: Multimodal agents are susceptible to environmental distractions. *arXiv preprint arXiv:2408.02544*.

Xinbei Ma, Zhuosheng Zhang, and Hai Zhao. 2024b. Coco-agent: A comprehensive cognitive mllm agent for smartphone gui automation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 9097–9110.

Ishani Mondal, Shwetha S, Anandhavelu Natarajan, Aparna Garimella, Sambaran Bandyopadhyay, and Jordan Boyd-Graber. 2024. Presentations by the humans and for the humans: Harnessing LLMs for generating persona-aware slides from documents. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2664–2684, St. Julian's, Malta. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report. *ArXiv preprint*, abs/2303.08774.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *In the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*, UIST '23, New York, NY, USA. Association for Computing Machinery.

Simon Park, Abhishek Panigrahi, Yun Cheng, Dingli Yu, Anirudh Goyal, and Sanjeev Arora. 2025. Generalizing from simple to hard visual reasoning: Can we mitigate modality imbalance in vlms? *Preprint*, arXiv:2501.02669.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2024. Mutual reasoning makes smaller llms stronger problem-solvers. *Preprint*, arXiv:2408.06195.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Edward Sun, Yufang Hou, Dakuo Wang, Yunfeng Zhang, and Nancy X. R. Wang. 2021. D2S: Document-to-slide generation via query-based text summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1405–1418, Online. Association for Computational Linguistics.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, and 3 others. 2024. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv preprint*, abs/2307.09288.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Improving text embeddings with large language models. *Preprint*, arXiv:2401.00368.

Sida Wang, Xiaojun Wan, and Shikang Du. 2017. Phrase-based presentation slides generation for academic papers. In *AAAI Conference on Artificial Intelligence*.

Shengqiong Wu, Hao Fei, Leigang Qu, Wei Ji, and Tat-Seng Chua. 2023. Next-gpt: Any-to-any multimodal llm.

Wenshan Wu, Shaoguang Mao, Yadong Zhang, Yan Xia, Li Dong, Lei Cui, and Furu Wei. 2024. Mind's eye of LLMs: Visualization-of-thought elicits spatial reasoning in large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. 2024. Agentgym: Evolving large language model-based agents across diverse environments. *Preprint*, arXiv:2406.04151.

Guowei Xu, Peng Jin, Hao Li, Yibing Song, Lichao Sun, and Li Yuan. 2024. Llava-cot: Let vision language models reason step-by-step. *Preprint*, arXiv:2411.10440.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. Swe-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022a. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022b. ReAct: Synergizing reasoning and acting in language models. volume abs/2210.03629.

Abhay Zala, Han Lin, Jaemin Cho, and Mohit Bansal. 2024. Diagrammergpt: Generating open-domain, open-platform diagrams via llm planning. In *COLM*.

Zhuosheng Zhang, Yao Yao, Aston Zhang, Xiangru Tang, Xinbei Ma, Zhiwei He, Yiming Wang, Mark Gerstein, Rui Wang, Gongshen Liu, and Hai Zhao. 2023a. Igniting language intelligence: The hitchhiker's guide from chain-of-thought reasoning to language agents. *Preprint*, arXiv:2311.11797.

Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. 2023b. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*.

Hao Zheng, Xinyan Guan, Hao Kong, Jia Zheng, Hongyu Lin, Yaojie Lu, Ben He, Xianpei Han, and Le Sun. 2025. Pptagent: Generating and evaluating presentations beyond text-to-slides. *arXiv preprint arXiv:2501.03936*.

# A Detailed Descriptions of Reviewer and Refiner Modules

## A.1 Reviewer Module

The Reviewer module analyzes the visual representation of the slide, identifies layout issues, and provides precise feedback for improvements. This feedback focuses on alignment, spacing, text overflow, and distortion. The primary goal of the Reviewer is to detect errors and ensure that all elements are properly positioned and formatted for a visually coherent slide.

### A.1.1 Evaluation Criteria and Feedback Rules

The Reviewer module evaluates slides based on the following criteria:

- **Object Overlapping**: Identifies overlapping elements and suggests repositioning or resizing to maintain separation.
- **Image Quality and Distortion**: Detects blurry or distorted images and recommends proportional scaling.
- **Element Bounding and Text Overflow**: Ensures text fits within its bounding box and suggests expanding the box or reducing font size.
- **Element Positioning and Alignment**: Checks alignment and spacing, adjusting misaligned elements to the nearest grid line.
- **Text Formatting Consistency**: Verifies font family and text hierarchy, ensuring the title is larger than body text.
- **Overall Composition and Visual Balance**: Evaluates symmetry and visual balance, recommending layout adjustments for better harmony.

### A.1.2 Example Output

The output of the Reviewer module is a structured JSON list, detailing necessary modifications for each slide element.

```
[
  {
    "element": 302,
    "recommendation": "Increase text box
        height by 1.2x to fit
       overflowing text."
  },
  {
    "element": 303,
    "recommendation": "Move downward by
       10% of its height to resolve
       overlap with ID 302."
  },
  {
    "element": 304,
    "recommendation": "Reduce font size
       by 2pt to fit within the
       bounding box."
  }
]
```

### A.2 Refiner Module

The Refiner module applies the Reviewer's feedback by modifying the structured layout described in JSON format. This module focuses on correcting bounding box positions, resizing elements, and preventing overlaps.

#### A.2.1 Input to the Refiner

The input to the Refiner module consists of the following components:

- **JSON File**: Describes the position, size, font, and content of each element on the slide.
- **Reviewer's Feedback**: Provides detailed recommendations for modifying elements (e.g., move, resize, align).
- **Slide Dimensions**: Ensures all adjustments remain within the boundaries of the slide.

#### A.2.2 Modification Instructions

The Refiner applies modifications based on the Reviewer's feedback, following these rules:

- **Move an Element**: Adjust the element's bounding box values to reposition it. Modify the top, bottom, left, and right values as required.
- **Resize or Scale an Element**: Modify the width and height of an element proportionally while preserving its aspect ratio.
- **Avoid Overlap**: Ensure no two elements overlap by repositioning or resizing conflicting elements.
- **Maintain Slide Boundaries**: Prevent elements from exceeding the slide's width or height.

#### A.2.3 Example Input and Output

The following example illustrates how the Refiner module processes input and produces a refined layout.

```
{
  "element": 302,
  "Bounds": [100, 200, 300, 400],
  "Font": {"size": 16},
  "Text": "Sample Text"
}
```

```
{
  "element": 302,
  "Bounds": [100, 220, 300, 420],
  "Font": {"size": 14},
  "Text": "Sample Text"
}
```

By applying these refinements iteratively, the Refiner ensures that the final slide layout meets high visual and structural standards, resulting in an accurate and human-like output.

## B Layout Evaluation Criteria and Scoring Standards

This section provides a detailed explanation of the evaluation criteria used to assess the quality of the generated slides. The evaluation process covers multiple aspects of slide design, including alignment, logical flow, text-visual consistency, visual appeal, and readability. Each criterion is scored on a five-point scale from 1 (Poor) to 5 (Excellent).

### B.1 Alignment and Spacing

This criterion evaluates whether elements on the slide are properly positioned, evenly spaced, and free from overlap. It ensures that the layout maintains visual balance and clarity.

- **1 Point (Poor)**: Severe misalignment; text overlaps with visuals, creating a chaotic layout.
- **3 Points (Average)**: Most elements are aligned, but minor misplacements exist.
- **5 Points (Excellent)**: Perfect alignment and spacing with a professional layout.

**Example Output:**

```
{
  "reason": "Most elements are well-
      aligned, but the spacing between
      the title and body text is
      inconsistent.",
  "score": 4
}
```

### B.2 Logical Flow

This criterion assesses the logical sequence of content, ensuring that the information presented in the slide is clear and structured for easy audience understanding.

- **1 Point (Poor)**: Disorganized content; key points do not follow a logical sequence.
- **3 Points (Average)**: Basic logical structure; minor reordering could improve the flow.
- **5 Points (Excellent)**: Seamless logical sequence with clear and structured information.

**Example Output:**

```
{
  "reason": "The information is
      structured logically, but the
      second point would be clearer if
      placed before the third.",
  "score": 4
}
```

### B.3 Text-Visual Consistency

This criterion evaluates the consistency between text and visual elements such as images and charts. It ensures that visuals effectively support the textual information.

- **1 Point (Poor)**: Visuals are irrelevant or contradict the text.
- **3 Points (Average)**: Somewhat aligned, but better integration is needed.
- **5 Points (Excellent)**: Perfectly integrated visuals that reinforce the message.

**Example Output:**

```
{
  "reason": "The visuals effectively
      support the content, but the chart
       could be labeled more clearly.",
  "score": 4
}
```

### B.4 Visual Appeal

This criterion assesses the overall aesthetic quality of the slide, focusing on color harmony, typography, and visual balance.

- **1 Point (Poor)**: Inconsistent styling; visually unappealing design.
- **3 Points (Average)**: Basic but functional color scheme; lacks enhancements.
- **5 Points (Excellent)**: Cohesive and visually appealing design with engaging elements.

**Example Output:**

```
{
  "reason": "The color scheme is
      visually appealing and harmonious,
       but the background contrasts too
      strongly with the text.",
  "score": 4
}
```

### B.5 Readability

This criterion evaluates the readability and clarity of the text and graphical elements, ensuring that all content is easily understandable.

- **1 Point (Poor)**: Text is too small or has low contrast, making it unreadable.
- **3 Points (Average)**: Generally clear, but some areas need better contrast or spacing.
- **5 Points (Excellent)**: Highly readable with optimal font size, spacing, and contrast.

**Example Output:**

```
{
  "reason": "The text is clear, well-
      spaced, and maintains good
      contrast. The charts are easy to
      read and properly scaled.",
  "score": 5
}
```

These evaluation criteria ensure a comprehensive and structured assessment of the generated slides. By adhering to these standards, the evaluation process becomes interpretable, consistent, and reliable.

## C Reliability Verification of Layout Generation Evaluation

To verify the reliability of our layout evaluation framework (LLM-as-Judge), we conducted a human evaluation on a randomly selected subset of generated slides. To ensure consistency with the LLM-based evaluations, we provided human raters with the same scoring rubric and descriptions used by the LLM, including detailed explanations of each criterion. Each dimension was rated on a 1–5 Likert scale, where 1 indicates poor performance and 5 indicates excellent performance. We then computed Pearson correlation coefficients between human scores and the LLM-based assessments.

As illustrated in Figure 9, the average correlation reached 0.6984, suggesting a strong agreement between human judgments and LLM evaluations.

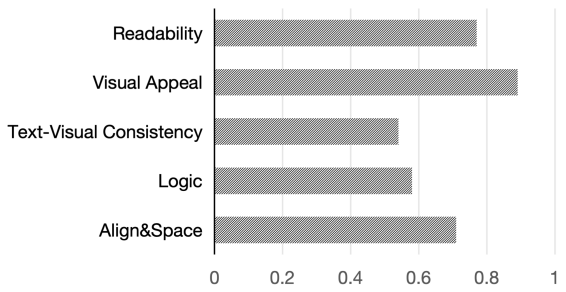


Figure 3: Pearson Correlation Between LLM-as-Judge and Human Evaluation Scores

In particular, dimensions such as *visual appeal* and *readability* achieved the highest consistency, with correlations of 0.89 and 0.77, respectively. These results support the use of LLM-as-Judge as a reliable proxy for human evaluation in layout quality evaluation.

## D Comparison of Text Retrieval Strategy

To assess the impact of retrieval strategy, we compared the embedding-based retriever (Salesforce SFR Embedding-Mistral) with a classical sparse retrieval method, BM25. Both retrievers were applied to retrieve top-5 relevant segments for slide content generation. Quantitatively, the two approaches yielded comparable performance, and further analysis revealed a high degree of overlap in the retrieved segments. This is likely due to the formal and structured nature of academic writing, where key sentences often share significant lexical overlap—making sparse methods like BM25 surprisingly competitive.

Since final content generation is performed by a powerful LLM, minor differences in retrieval results tend to have limited influence on the final ROUGE scores. We retain the embedding-based retriever in our framework for its stronger generalization ability across domains and robustness in semantically diverse settings.

## E Experiment on Product Documentation

Using the method proposed in this paper, we validated our approach on product documentation and generated a complete presentation using PowerPoint's built-in template.

The original input format is HTML (Figure 4). We processed this HTML content into a formatted text and image format, extracting key topics for the slides. Based on these topics, we generated a complete presentation, with each slide corresponding to a specific topic.

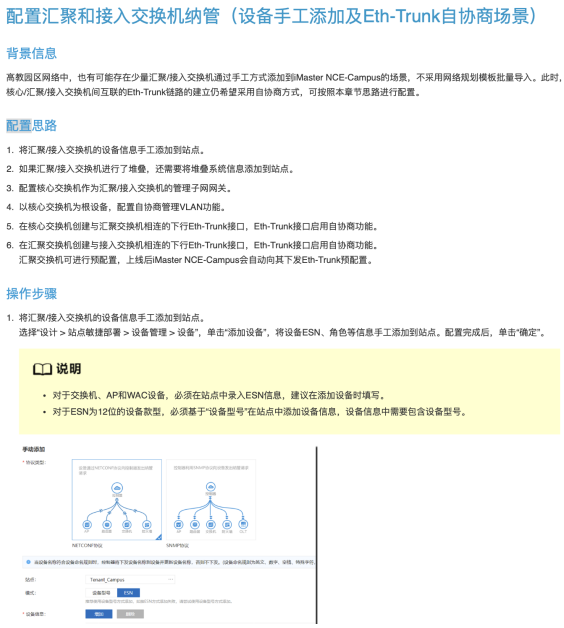


Figure 4: Input Example (HTML content in webpage)

The generated output is shown in Figure 5 and 6.

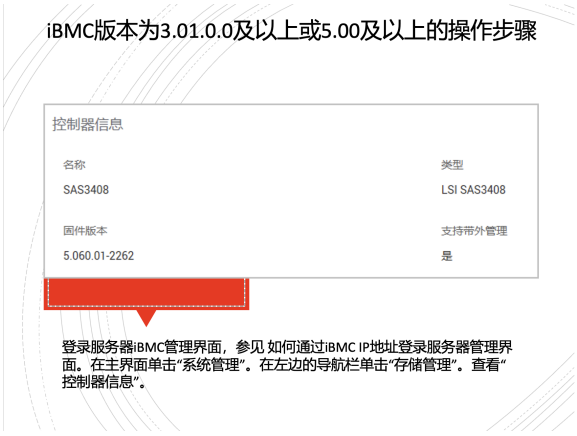


Figure 5: Output Example 1

## F Further Analysis by Slide Type

To better understand how our system performs across different types of slides, we conduct a qualitative analysis based on slide content composition. Specifically, we categorize generated slides into the following three types:

- **Text-only slides:** slides that contain only textual bullet points without any figures or tables.

Figure 6: Output Example 2

- **Text + figure slides:** slides that combine textual content with at least one accompanying figure or table.
- **Figure-only slides:** slides where the primary content consists of visual elements, with minimal or no textual explanation.

For text-only slide, as shown in Figure 7, the refined version improves spacing between elements, making the content more legible and visually organized.

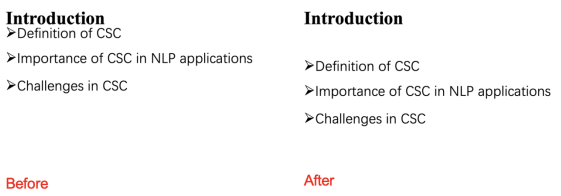


Figure 7: Example of Layout Refinement for a Text-only Slide

For text + figure slide, as shown in Figure 8, in the original version (left), the figure is relatively small and placed in the bottom-right corner, making it visually disconnected from the textual content. In the refined version (right), the figure is enlarged and repositioned to occupy the right half of the slide. The spacing between elements is also improved.

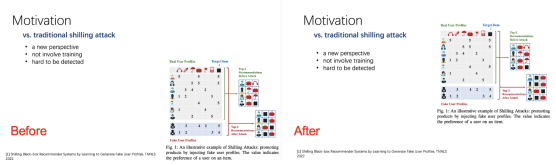


Figure 8: Example of Layout Refinement for a Text+Figure Slide

For figure-only slide, as shown in Figure 9, in the refined version (right), spacing and alignment are improved to reduce clutter and enhance readability, allowing each visual component to stand out more clearly.

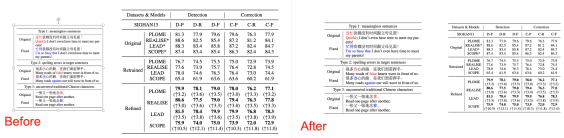


Figure 9: Example of Layout Refinement for a Figure-only Slide

| Retriever | ROUGE-1 | ROUGE-2 | ROUGE-L |
|---|---|---|---|
| BM25 | 28.67 | 11.27 | 22.49 |
| Embedding-based | 28.40 | 11.68 | 21.97 |

Table 3: F1 comparison of sparse and dense retrievers using LLaMA3-8B.