SheetDesigner: MLLM-Powered Spreadsheet Layout Generation with Rule-Based and Vision-Based Reflection

Qin Chen[*,1], **Yuanyi Ren**[*,1], **Xiaojun Ma**[†,2], **Mugeng Liu**[1], **Han Shi**[2], **Dongmei Zhang**[2],
[1]Peking University, [2]Microsoft,
{chenqink,yyren,lmg}@pku.edu.cn, {xiaojunma,shihan,dongmeiz}@microsoft.com

Abstract

Spreadsheets are critical to data-centric tasks, with rich, structured layouts that enable efficient information transmission. Given the time and expertise required for manual spreadsheet layout design, there is an urgent need for automated solutions. However, existing automated layout models are ill-suited to spreadsheets, as they often (1) treat components as axis-aligned rectangles with continuous coordinates, overlooking the inherently discrete, grid-based structure of spreadsheets; and (2) neglect interrelated semantics, such as data dependencies and contextual links, unique to spreadsheets. In this paper, we first formalize the spreadsheet layout generation task, supported by a seven-criterion evaluation protocol and a dataset of 3,326 spreadsheets. We then introduce **SheetDesigner**, a zero-shot and training-free framework using Multimodal Large Language Models (MLLMs) that combines rule and vision reflection for component placement and content population. SheetDesigner outperforms five baselines by at least 22.6%. We further find that through vision modality, MLLMs handle overlap and balance well but struggle with alignment, necessitates hybrid rule and visual reflection strategies. Our codes and data is available at Github.

1 Introduction

Sitting at the heart of finance, analytics, and scientific discovery, spreadsheets serve as powerful tools for organizing and analyzing data (Chan and Storey, 1996; Häcker and Ernst, 2017; Powell and Baker, 2019). They are structured in a grid of rows and columns with integrated tables and charts. Meanwhile, their effectiveness hinges on clear, well-structured layouts; otherwise, even the most rigorous analysis becomes unreadable when a chart obscures its source data, or a column cuts off text. Consequently, given the importance of well-structured layouts and the time-consuming, expertise-dependent nature of manual design, automated layout generation becomes essential.

However, existing layout generation approaches (Zhang et al., 2024; Kong et al., 2022; Gupta et al., 2021; Cheng et al., 2025) fall short of this task, as they (1) treat components as rectangles with continuous pixel coordinates, ignoring the grid-based structure of spreadsheets, where components span discrete cells and resizing them affects entire rows or columns. (2) overlook key semantic relationships, such as placing charts near their source tables, and fail to account for the global row or column resizing to fit content like long text. Consequently, their outputs must be painstakingly post-processed for spreadsheet layouts and often remain invalid or suboptimal, underscoring a significant and largely unaddressed research problem.

In this paper, we first formalize the task of spreadsheet layout generation: Given a raw sheet containing user data (e.g., tables, charts), the goal is to generate a structured layout that enhances spreadsheet usability. To quantify the evaluation of this task, we introduce an evaluation protocol that scores a candidate layout on seven complementary criteria—*fullness, compactness, compatibility, component-alignment, type-aware alignment, relation-aware alignment,* and *overlap* (see section 2). We also formulate a dataset, *SheetLayout*, comprising 3,326 real-world spreadsheets covering ten domains and thirteen frequently used topics of functions (see Table 7, Table 8).

On this foundation, we propose **SheetDesigner**, a zero-shot and training-free framework for spreadsheet layout generation, powered by Multimodal Large Language Models (MLLMs). Given a set of user data, SheetDesigner consists of two phases. (1) It initially places components on the grid in a type-aware and relation-aware manner, and sub-

[*] Equal contribution. Work done during internship at Microsoft.

[†] Corresponding author.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 18921–18939
November 4-9, 2025 ©2025 Association for Computational Linguistics

sequently applies a Dual Reflection mechanism, comprising rule-based and vision-based reflection, to refine the layouts. (2) After reflection, it populates the sheet layout with user data, inserts line breaks for lengthy entries, and generates consistent global column widths and row heights, resulting in a ready-to-use layout.

We evaluate SheetDesigner on the SheetLayout dataset against five state-of-the-art baselines, achieving a 22.6% improvement in performance. Using a 13B-parameter backbone, SheetDesigner matches or exceeds the performance of much larger architectures like LayoutPrompter (Lin et al., 2023b), leveraging GPT-4O as its backbone (OpenAI, 2024). Our ablation study highlights the contribution of each component and shows that while the vision modality of MLLMs improves *overlap* and *balance*, they struggle with *alignment*. Our further empirical analysis reveals that MLLM attention evidently fails to effectively focus on regions critical for component alignment in structured spreadsheet images, extending the observations of (Zhang et al., 2025a). This highlights the importance of the hybrid rule-based and vision-based reflection mechanisms in SheetDesigner.

In summary, our contributions are as follows:

- We present the first task formulation for spreadsheet layout generation, accompanied by a seven-criteria evaluation protocol and a novel dataset, *SheetLayout*, comprising 3,326 spreadsheets spanning 10 common domains and 13 frequent topics.
- We introduce **SheetDesigner**, a zero-shot and training-free framework that directly models spreadsheet layout generation in a two-stage process: *structure placement with Dual Reflection* and *content population with global arrangements*.
- We show that SheetDesigner achieves a 22.6% improvement over five state-of-the-art baselines, with the 13B variant matching or surpassing much larger architectures built on GPT-4o. Our ablation study and empirical analysis further validate the effectiveness of the hybrid rule- and vision-based design.

2 Preliminary

2.1 Task Formulation

In this subsection, we formally define the task of spreadsheet layout generation.

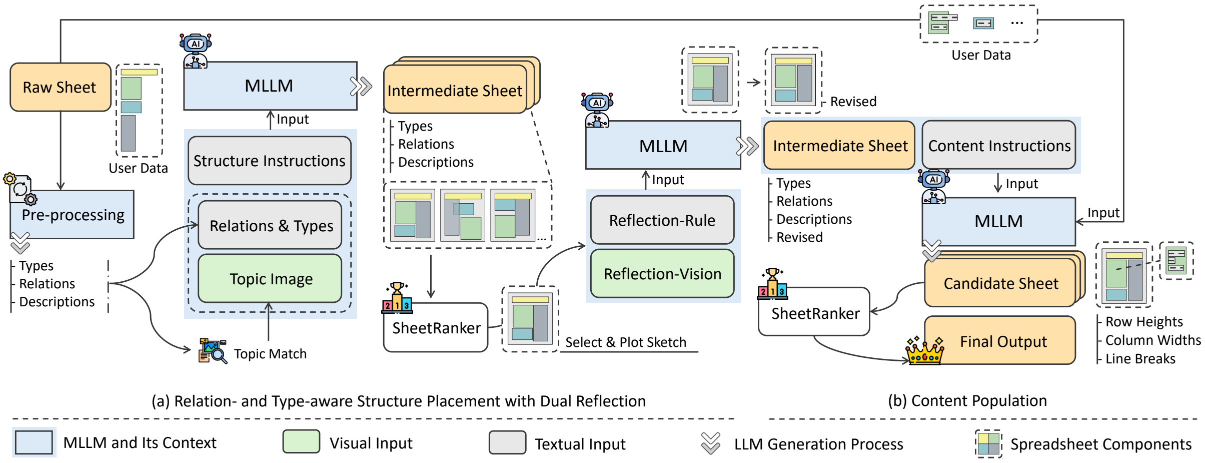
Input We denote the input raw sheet as $\mathcal{S} = [\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_N]$, where each component $\mathcal{C}_i = \{\mathcal{D}_i\}$ contains user data $\mathcal{D}_i$. $\mathcal{D}_i$, including texts, numbers, formulas, etc.

Output The goal of spreadsheet layout generation is to create a layout: $\mathcal{L} = [\tilde{\mathcal{C}}_1, \tilde{\mathcal{C}}_2, \ldots, \tilde{\mathcal{C}}_N, \mathcal{G}]$ that organizes the components while preserving their types and formatting the content appropriately. Each component in the generated layout is represented as $\tilde{\mathcal{C}}_i = \{\mathcal{P}_i, \mathcal{T}_i, \tilde{\mathcal{D}}_i\}$, where $\mathcal{P}_i$ specifies the assigned position of the component using the R1C1 format (e.g., "A1:C3"), $\mathcal{T}_i$ denotes the assigned type from five component types (e.g., "title"), and $\tilde{\mathcal{D}}_i$ denotes the formatted text with appropriate line breaks. Additionally, $\mathcal{G} = [w_1, w_2, \ldots; h_1, h_2, \ldots]$ represents the configuration of the generated layout, encompassing the column widths w_i and row heights h_i.

In this paper, we consider the following five common types of spreadsheet components divided by their semantics: (1) *title* that provides a descriptive heading for the spreadsheet. (2) *main-table* that contains the core structured data, often organized in rows and columns. (3) *meta-data* that includes supplementary information such as author names, dates, or version details. (4) *summary-data* that presents aggregated insights, such as totals, averages, or key metrics derived from the main table. (5) *chart* that visually represents data trends and relationships through graphs, bar charts, or other visual components like images or icons.

2.2 Layout Evaluation

This subsection outlines the evaluation of generated sheet layouts. Intuitively, we expect the generated layouts to (1) be compact without large empty space; (2) be well-aligned between components, and the alignment should be broadcast to components of the same *type*, or *dependent* components like table and charts that are drawn by the data in this table ; (3) be visually balanced where there is no great discrepancy for the vertical and horizontal distribution of components; (4) be compatible to the original contents; (5) avoid overlap regions. Following this intuition, we define quantitative evaluation metrics (detailed in Appendix C). For *Overlap*, a score of 0 indicates no overlap, with progressively smaller values corresponding to greater overlap. Other metrics fall within the $(0, 1]$ range, where higher scores indicate better performance.



(a) Relation- and Type-aware Structure Placement with Dual Reflection

(b) Content Population

Figure 1: SheetDesigner operates in two stages following pre-processing. (a) Components are structurally placed based on their types and relationships. Among multiple layout candidates, SheetRanker selects the best one, which is then refined through Dual-Reflection—a revision step combining rule-based (text) and vision-based (sketch-image) feedback. (b) Content is populated into the placed components, with adjustments to row heights, column widths, and line breaks to ensure proper fit. SheetRanker then selects the final output from the generated candidates.

3 Method

This section details the structure of SheetDesigner, shown in Figure 1. SheetDesigner divides the sheet layout generation into two phases: (1) *structure placement with Dual Reflection*, which assigns components to appropriate locations by considering both their type and relational context, while ensuring proper alignment. Then it refines the assignment through rule-based and vision-based reflection; and (2) *content population with global arrangements*, which fills components with user data and sets row heights and column widths based on cell content. The two-phase design leverages MLLMs' strength in handling focused tasks, rather than being distracted by diverse objectives in a single run (Wang et al., 2024; Sun et al., 2025).

Pre-processing Denote each raw sheet as $\mathcal{S} = [\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_N]$, where each component $\mathcal{C}_i = \{\mathcal{D}_i\}$ contains the corresponding data $\mathcal{D}_i$. We begin by classifying the entire spreadsheet into one of 13 topics based on its application context, yielding a general topic label $\hat{\mathcal{T}}_{\mathcal{S}}$ for the sheet. Next, for each component $\mathcal{C}_i \in \mathcal{S}$, we: (i) assign it a type $\mathcal{T}_i$, selected from five component types (e.g., "title"); (ii) generate a textual description $\hat{\mathcal{D}}_i$ based on its content $\mathcal{D}_i$ (e.g., "A main-table for different services and costs"). We then instruct large language models (LLMs) to identify pairwise relationships between components, resulting in a relation list $\mathcal{R}$ (e.g., $[(\text{Main}_1, \text{Chart}_1)]$). See Appendix J for detailed prompt examples. The processed sheet is denoted as $\hat{\mathcal{S}} = \{[\hat{\mathcal{C}}_1, \hat{\mathcal{C}}_2, \ldots, \hat{\mathcal{C}}_N], \mathcal{T}_{\mathcal{S}}, \mathcal{R}\}$, where each $\hat{\mathcal{C}}_i = \{\mathcal{T}_i, \hat{\mathcal{D}}_i\}$ includes the assigned type and generated description for the component. We employ LLMs as the pre-processing engine.

3.1 Structure Placement with Dual Reflection

3.1.1 Structure Placement

In this stage, for each preprocessed sheet $\hat{\mathcal{S}}$, we prompt the MLLMs with: (i) textual instructions guiding sheet layout generation; (ii) the relations $\mathcal{R}$ and sheet components $[\hat{\mathcal{C}}_1, \hat{\mathcal{C}}_2, \ldots, \hat{\mathcal{C}}_N]$, and (iii) an exemplar topic image $\mathcal{I}$. We generate N_1 intermediate sheet layouts per spreadsheet, score them using SheetRanker (see subsection 3.3), and select the top-performing candidates.

The instructions guide the generation by: (1) preserving alignment among components, particularly in a type-aware manner (aligning components with the same type) and a relation-aware manner (placing related components in proximity); (2) ensuring spatial fullness and balance by maximizing space usage and distributing components evenly in horizontal or vertical; (3) avoiding overlaps. We also provide examples to demonstrate each principle in practice. Furthermore, MLLMs are allowed to resize titles and charts to improve layout quality, as these have looser grid constraints than others. Other components are fixed in size. For instance, a 1×5 title can be resized to 1×6 (or 1×4) to enhance alignment without affecting its meaning, whereas resizing a 4×4 data table to 4×5 (or 4×3) would involve adding or removing data.

For each preprocessed sheet $\hat{S}$, we retrieve a topic-specific image $\mathcal{I}$, a screenshot of an exemplar spreadsheet of the same type of $\mathcal{T}_S$. We use this image as a visual exemplar to guide the layout of components. Layout conventions, such as spatial grouping, and visual emphasis, are tailored to each document's topic. For example, recipe cards prioritize vertical lists of ingredients and step-by-step instructions, whereas academic posters allocate prominent space for section headers and data visualizations.

3.1.2 Dual Reflection

Given the top-ranked intermediate sheet layout and its scores for each aspect, we refine it through a dual reflection process: (1) **Rule-based Reflection**. For each evaluation aspect, if the SheetRanker score falls below a predefined threshold (e.g., fullness < 0.5), we augment the prompt with targeted revision instructions. For instance, a low overlap score triggers guidance to explicitly avoid component overlap during the reflection step[1]. (2) **Vision-based Reflection**. We visualize the sheet layout by coloring cells based on component types for an image input. This allows the MLLM to perceive the layout from a visual perspective, enabling further refinement through multimodal understanding. For detailed information on the threshold for triggering reflection, the specific prompts for revising each aspect, and the algorithm used to generate images of layouts, please refer to Appendix D.

3.2 Content Population and Global Arrangements

After revising the intermediate sheet, we enter the content-aware stage, where the original data is populated into components with appropriate line breaks. Global sheet layout configurations—specifically, column widths and row heights—are also generated. The prompt includes the following instructions: (1) insert line breaks for lengthy content; (2) adjust column widths and row heights to fit content while minimizing empty space. We also provide examples of common font settings with corresponding row height and column width settings. The LLM generates N_2 candidate sheet layouts, which are then ranked using SheetRanker to produce the final, fully detailed sheet layout.

3.3 SheetRanker

Given a set of candidate sheets, SheetRanker assigns a score to each based on the protocol in subsection 2.2 and selects the one with the highest score. All aspects are weighted equally at 1 [2]. We adopt this uniform weighting because all aspects (except Overlap) share a common scale of $(0, 1]$, while Overlap mostly falls within $(-1, 0]$. SheetRanker serves two key functions: (i) guiding selection toward the candidate with the highest overall performance, and (ii) providing a quantitative foundation for reflecting on and refining structural placement.

4 Experiments

In this section, we conduct experiments to verify the effectiveness of the proposed SheetDesigner.

4.1 Dataset

For the evaluation of our model, we construct a dataset, *SheetLayout*, consisting of 3,326 Excel spreadsheets collected from various domains and real-world applications (See Table 7 and Table 8. The dataset encompasses diverse spreadsheet structures, reflecting practical use cases across multiple fields. We perform object detection within each spreadsheet, identifying key components. These detected objects are subsequently converted into a structured JSON format to enable standardized processing. See Appendix E for details on the collection, anonymous process, and licenses.

4.2 Baselines & Settings

We compare the proposed *SheetDesigner* with various state-of-the-art baselines, which can be broadly categorized into two groups:

- **Traditional Transformer-based models**, trained and validated on layout datasets, including BLT (Kong et al., 2022), LayoutFormer++ (Jiang et al., 2023), and Coarse-to-Fine (Jiang et al., 2022).
- **LLM-based approaches**, which leverage LLMs to enable few-shot or zero-shot layout generation, including LayoutPrompter (Lin et al., 2023b) and PosterLLaVA (Yang et al., 2024). For LayoutPrompter we adopt GPT-4o as the backbone, as the recommended `text-davinci-003` is deprecated.

[1] If no aspect falls below its threshold, this step is skipped.

[2] For aspects with horizontal and vertical sub-aspects, each sub-aspect is weighted at 0.5, maintaining a total weight of 1.

Table 1: Quantitative results on SheetLayout reported in mean scores. Scores range from 0 (poor) to 1 (optimal), except for the overlap metric (≤ 0), where values closer to 0 indicate better performance. The weighted total score assigns a weight of 0.5 to vertical and horizontal sub-aspects, and 1 to all other aspects. Relative performance is reported with respect to the best-performing model. For LLM-based methods, the underlying language model is indicated in parentheses. "C" denotes Component, "T" for Type-aware, and "R" for Relation-aware.

	Fullness	**Compatibility**		**C-Alignment**		**T-Alignment**		**R-Alignment**		**Balance**		**Overlap**	**Weighted Total**
		Horizontal	Vertical	Horizontal	Vertical	Horizontal	Vertical	Horizontal	Vertical	Horizontal	Vertical		
BLT	0.485	0.285	0.586	0.373	0.604	0.379	0.571	0.451	0.547	0.474	0.504	-0.184	2.688 (↓ 45.12%)
LayoutFormer++	0.618	0.308	0.559	0.501	0.728	0.407	0.585	0.556	0.607	0.595	0.705	-0.125	3.268 (↓ 33.27%)
Coarse-to-Fine	0.531	0.289	0.576	0.475	0.635	0.402	0.601	0.519	0.528	0.601	0.725	-0.143	3.063 (↓ 37.45%)
PosterLLaVa (LLaVA-7B)	0.653	0.376	0.608	0.404	0.712	0.430	0.642	0.609	0.684	0.610	0.732	-0.183	3.373 (↓ 31.12%)
LayoutPrompter (GPT-4o)	0.804	0.397	0.623	0.508	0.789	0.487	0.683	0.690	0.716	0.634	0.778	-0.167	3.789 (↓ 22.63%)
SheetDesigner (Vicuna-7B)	0.703	0.395	0.617	0.434	0.778	0.485	0.628	0.662	0.581	0.545	0.683	-0.103	3.504 (↓ 28.46%)
SheetDesigner (LLaVA-7B)	0.706	0.431	0.629	0.458	0.794	0.486	0.637	0.690	0.585	0.619	0.721	-0.075	3.656 (↓ 25.36%)
SheetDesigner (Vicuna-13B)	0.678	0.424	0.649	0.456	0.803	0.521	0.678	0.675	0.642	0.668	0.753	-0.056	3.756 (↓ 23.31%)
SheetDesigner (LLaVA-13B)	0.690	0.432	0.661	0.459	0.806	0.530	0.680	0.695	0.668	0.696	0.793	-0.043	3.857 (↓ 21.25%)
SheetDesigner (GPT-4o)	0.981	0.549	0.886	0.683	0.880	0.788	0.858	0.703	0.679	0.894	0.920	-0.003	4.898

Table 2: Ablation study, "w/o" denotes "without".

	Fullness	**Compatibility**		**C-Alignment**		**T-Alignment**		**R-Alignment**		**Balance**		**Overlap**	**Weighted Total**
		Horizontal	Vertical	Horizontal	Vertical	Horizontal	Vertical	Horizontal	Vertical	Horizontal	Vertical		
SheetDesigner	0.981	0.549	0.886	0.683	0.880	0.788	0.858	0.703	0.679	0.894	0.920	-0.003	4.898
w/o Topic	0.973	0.537	0.859	0.653	0.856	0.736	0.821	0.683	0.673	0.876	0.897	-0.003	4.766 (↓ 2.71%)
w/o Reflection-Rule	0.956	0.530	0.876	0.641	0.820	0.744	0.793	0.642	0.664	0.823	0.856	-0.007	4.644 (↓ 5.20%)
w/o Reflection-Vision	0.941	0.544	0.879	0.672	0.867	0.788	0.854	0.698	0.674	0.852	0.882	-0.012	4.784 (↓ 2.33%)
w/o Reflection	0.925	0.520	0.853	0.622	0.805	0.739	0.781	0.628	0.631	0.804	0.802	-0.017	4.500 (↓ 8.12%)
w/o SheetRanker	0.916	0.491	0.752	0.637	0.842	0.729	0.784	0.673	0.659	0.859	0.884	-0.010	4.561 (↓ 6.88%)
w/o Vision	0.926	0.522	0.701	0.638	0.848	0.705	0.804	0.632	0.632	0.823	0.874	-0.015	4.500 (↓ 8.12%)

While the aforementioned baselines perform well in general layout generation, they are neither specifically designed nor optimized for spreadsheet layouts. Their outputs are typically pixel-based bounding boxes, like $[(x_1, y_1, x_2, y_2), \dots]$, where each box defines the top-left and bottom-right pixel coordinates of a component. To adapt these layouts for spreadsheets, we introduce a standardized procedure that maps pixel-based layouts to a grid-based structure. We assume a $\mathcal{B}_x \times \mathcal{B}_y$ pixel background, where each grid cell corresponds to a $\mathcal{C}_x \times \mathcal{C}_y$ pixel area, yielding a $\frac{\mathcal{B}_x}{\mathcal{C}_x} \times \frac{\mathcal{B}_y}{\mathcal{C}_y}$ grid. If any layout exceeds $\mathcal{B}_x$ pixels in width (or $\mathcal{B}_y$ pixels in height), we scale it down proportionally to fit within these constraints. Components that do not align perfectly with the grid (i.e., whose positions are not exact multiples of $\mathcal{C}_x$ or $\mathcal{C}_y$ pixels) are adjusted by snapping to the nearest cell.

In this study, we adopt $\mathcal{B}_x = 1000$, $\mathcal{C}_x = 50$, $\mathcal{B}_y = 500$, and $\mathcal{C}_y = 25$, reflecting typical settings in commonly used spreadsheet applications. The threshold for triggering Dual Reflection is set to a moderate value of 0.5 for all aspects except Overlap, which uses a strict threshold of 0, as any intersection significantly degrades layout usability and therefore always triggers a revision. The number of repeated runs is set to $N_1 = N_2 = 3$. To ensure a fair comparison between training-based and training-free methods, the data is split into 10% for training, 10% for validation, and 80% for testing. All reported performance metrics are based on the test set. We adopt three families of backbone models: GPT-4o (OpenAI, 2024), the Vicuna family (Zheng et al., 2023), and the LLaVA family (Liu et al., 2023). Regarding modality, GPT-4o and LLaVA are multimodal LLM with vision ability, while Vicuna models are text-only. For Vicuna, vision-related inputs are disabled. The LLaVA models utilized in this study employ their corresponding Vicuna models as the backbone LLM.

4.3 Evaluation Results

In Table 1, we compare SheetDesigner with several baselines on the dataset SheetLayout. Employing GPT-4o as its backbone, SheetDesigner demonstrates state-of-the-art performance. It surpasses the second-place LayoutPrompter, which also utilizes GPT-4o, by a notable 22.63% in total score. Variants using smaller models also remain competitive. Notably, when equipped with a 13B model (either Vicuna-13B or LLaVA-13B), SheetDesigner performs competitively or even surpasses LayoutPrompter—which relies on the much stronger GPT-4o backbone—demonstrating the effectiveness of our framework. Additionally, at comparable parameter scales, models with vision perception (LLaVA family) consistently outperform those without (Vicuna family).

We visualize some of the layouts generated by SheetDesigner, LayoutFormer++, and LayoutPrompter for comparison in Figure 2. Generally SheetDesigner achieves favorable results against the baselines. Compared to LayoutFormer++ and LayoutPrompter, SheetDesigner produces more aligned layouts. Further, SheetDesigner arranges components in a relation-aware and type-aware manner, for example, in Figure 2 (middle-right), SheetDesigner places summary tables of corresponding tables directly below them, assuring better readability and usefulness compared to the others. For a analysis on failure cases, please refer to Appendix F.

4.4 Ablation Study

To assess the contribution of each component in SheetDesigner, we conduct an ablation study (see Table 2), including a "w/o Vision" case in which both the topic image and vision-based reflection are removed, and visualize the lowest 1% of the score distributions (see Figure 3). Our results show that every component enhances performance to varying degrees: the SheetRanker yields a significant overall improvement; the topic images notably boosts balance and relation-aware alignment; rule-based reflection affects almost every metric (its smallest impact being on overlap); and vision-based reflection improves balance and reduces overlap but contributes little to alignment measures. Generally, the visual modality of MLLMs is helpful for improving balance or overlap, but struggle with alignment requirements.

Table 3: Quantitative comparison: Specified dimensions with line wraps than the AutoFit of Excel. Relative performance degradation is reported.

	Compatibility		**Deg. Ratio**
	Horizontal	Vertical	
Specified (Full)	0.549	0.886	-
AutoFit (Full)	0.538	0.853	3.07%
Specified (%5 Low)	0.523	0.882	-
AutoFit (%5 Low)	0.495	0.824	6.12%
Specified (%1 Low)	0.521	0.896	-
AutoFit (%1 Low)	0.485	0.818	8.05%

We compared our proposed ContentPopulator with Excel's built-in AutoFit function (see Table 3) across the full dataset, as well as the lowest 5% and 1% of scores. The results support our claim that explicitly specifying row and column dimensions, along with appropriate line wrapping, generally outperforms AutoFit.

4.5 Hyper-parameter Analysis on Thresholds in Dual Reflection

Table 4: Hyper-parameter Analysis on the thresholds in Dual Reflection. We report the total scores, and the average tokens costs with GPT-4o as the backbone.

	Total Score ↑	Dual Reflection Token Cost ↓
Threshold=0.3	4.604	103.5
Threshold=0.5	4.898	234.2
Threshold=0.7	4.904	616.7

The thresholds in Dual Reflection determine when a layout needs revision. We use a moderate threshold of 0.5 for the main experiments, as stated previously. Lower thresholds (e.g., 0.3) reduce computation by targeting only extreme cases, while higher thresholds (e.g., 0.7) improve quality but increase computation due to more frequent revisions. In this section, we show results for thresholds 0.3 and 0.7 in Table 4. A 0.3 threshold significantly reduces performance, while 0.7 offers little improvement but uses many more tokens. This indicates that some layouts need revision, but not all are fixable, for example, some spreadsheets are inherently difficult due to mixed object sizes. This supports our choice of a moderate threshold for efficient performance gains.

5 Why Does Vision Help Balance and Overlap but Not Alignment?

As previously stated in subsection 4.4, our findings indicate that with visual input, MLLMs excel at improving overlap, balance but struggle with alignment-related features. To further investigate this, we visualize the attention weights of LLaVA-7B using two sketch images, following the setup in (Zhang et al., 2025a). One layout contains overlapping components, while the other exhibits misaligned elements. In each case, the model is prompted to identify regions of either overlap or misalignment. To facilitate direct visual interpretation of the attention weights, we use the general prompt *"detect the spreadsheet's components"* to normalize the attention maps (Liu et al., 2025b).

As shown in Figure 4, LLaVA-7B demonstrates precise attention to overlapping regions, indicated by a highly concentrated distribution of weights. Conversely, its attention is scattered and disorganized when dealing with misalignment, failing to

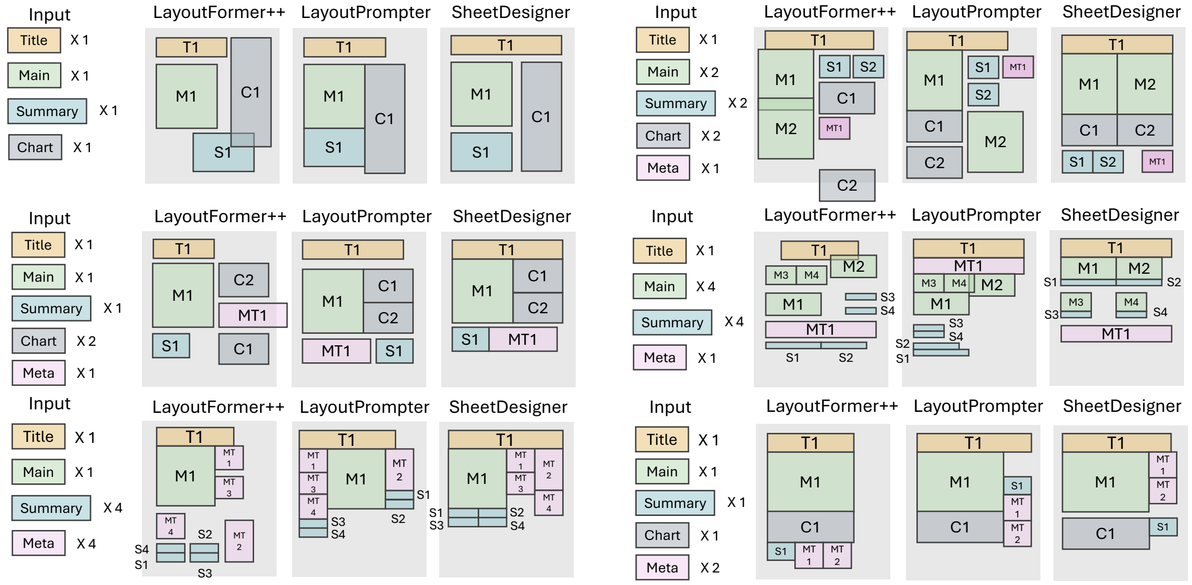


Figure 2: Qualitative comparison among SheetDesigner, LayoutFormer++ and LayoutPrompter. We denote each component with the letters, T for titles, M for main tables, S for summary tables, and MT for metadata tables.

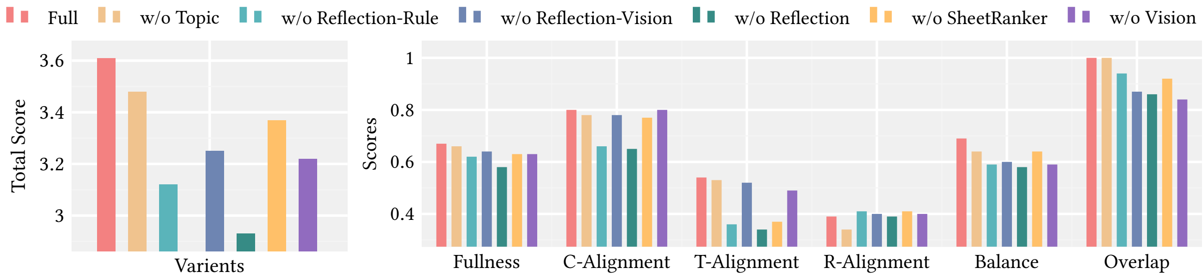


Figure 3: Ablation study for the lower 1% tail of the score distributions. An offset of 1 was added to the overlap scores for clarity. Aspects with horizontal and vertical sub-components were merged by averaging their scores.

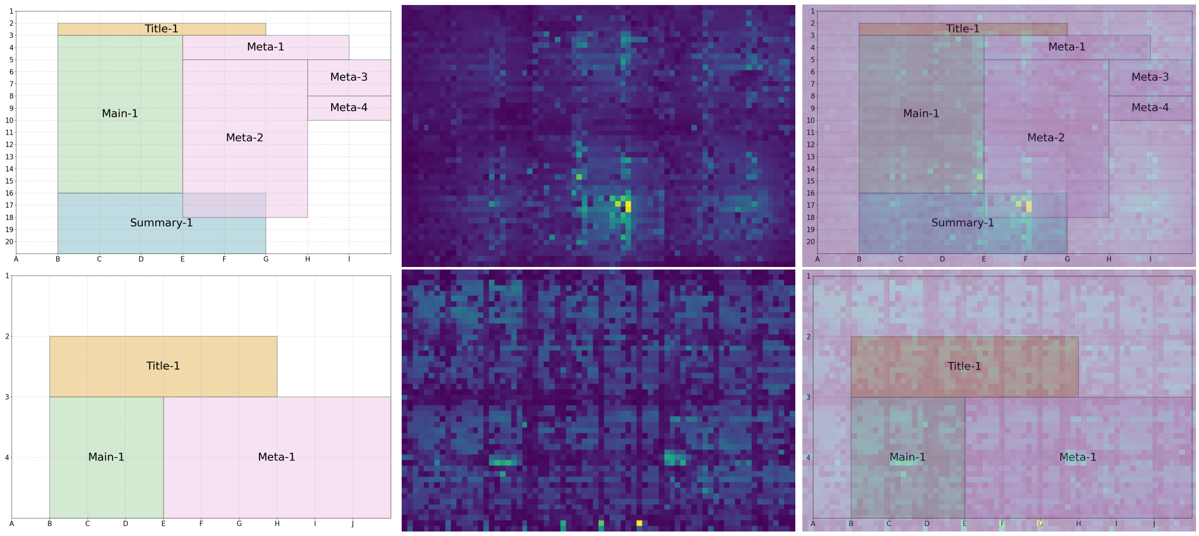


Figure 4: Visualization of attention weights on the input image. The first row shows an instance with overlapping components, where LLaVA-7B demonstrates precise attention with a concentrated weight distribution. In contrast, the second row illustrates an instance with misalignment, where the model's attention is scattered and fails to accurately capture misaligned components, such as the right border of the title block.

accurately identify misaligned components like the right border of the title block. The model's proficiency with overlap, akin to identifying a "man with a yellow backpack," likely stems from its optimization for perceiving natural objects with mixed visual features. However, alignment demands a fine-grained focus on the boundaries between paired components, an area where MLLMs currently lack sufficient optimization. This disparity in processing may explain the limited contribution of vision to alignment in Figure 3.

This analysis underscores the pivotal role of our Dual Reflection module, which leverages the complementary strengths of rule-based (textual) and vision-based (image) reasoning. In alignment tasks, textual reasoning demonstrates a clear advantage due to the explicitness of coordinate data. For instance, the alignment between "A1:A3" and "B1:B3," contrasted with the misalignment with "C2:C4," can be directly inferred from positional information. However, recognizing this relationship in images requires detailed pairwise visual reasoning. Conversely, for spatial features such as overlap or balance, visual perception provides immediate and intuitive insights, while textual analysis demands additional processing to extract the same information from raw coordinates. The Dual Reflection module's strength lies in integrating these two modalities, resulting in notable performance gains. These findings also highlight opportunities for improving MLLMs, particularly by enhancing their visual reasoning capabilities (Wang et al., 2024)—a critical need when interpreting structured formats like spreadsheet images.

6 Related Works

Traditional Pixel-oriented Layout Generation Layout generation is a widely studied topic encompassing various sub-tasks, including: (1) generation conditioned on element types (Kikuchi et al., 2021; Kong et al., 2022; Lee et al., 2020; Arroyo et al., 2021), (2) generation conditioned on both element types and sizes (Kong et al., 2022; Cheng et al., 2024), (3) generation conditioned on element relationships (Kikuchi et al., 2021; Lee et al., 2020; Cheng et al., 2024), (4) layout completion (Gupta et al., 2021), (5) layout refinement (Rahman et al., 2021), (6) content-aware generation (Hsu et al., 2023; Zheng et al., 2019; Zhang et al., 2024), (7) text-to-layout generation (Huang et al., 2021; Lin et al., 2023a), (8) layout revision (Li et al., 2024), and more. Transformers and diffusion models are popular and powerful backbones for these tasks.

LLM-driven Layout Generation Beyond the traditional methods mentioned above, recent studies (Lin et al., 2023b; Yang et al., 2024; Tang et al., 2023; Seol et al., 2024; Zhang et al., 2025b; Hsu and Peng, 2025; Tang et al., 2024) have explored leveraging Large Language Models (LLMs) for layout generation. These approaches offer benefits such as zero-shot capability, robustness, multi-task generalization, and strong generation performance, all without the need for task-specific training. Additionally, efforts are being made to enhance LLMs with vision modalities (Cheng et al., 2025), Chain-of-thought reasoning (Shi et al., 2025), and diffusion models (Liu et al., 2025a).

While considerable research has been conducted in general layout generation, the resulting methods may not directly align with the distinct characteristics of spreadsheets. Spreadsheets inherently demand strict conformity to a grid, contrasting with approaches that permit arbitrary pixel-level element positioning. Moreover, element dimensions within spreadsheets, such as row heights and column widths, function as global parameters; an adjustment to column A's width, for example, consequently alters all cells in that column. Finally, spreadsheet layouts inherently require awareness of component types, relationships, and content, which many general-purpose layout approaches do not fully incorporate.

7 Conclusion

In this paper, we formalize the task of spreadsheet layout generation, develop an evaluation protocol covering seven key aspects, and present a dataset comprising 3,326 spreadsheets. We then introduce SheetDesigner for this task, a zero-shot, and training-free framework driven by multimodal large language models. SheetDesigner adopts a two-stage strategy. SheetDesigner involves (1) structural placement with Dual Reflection and (2) content population with global arrangements. Experimental results reveal SheetDesigner's superior performance, surpassing baselines by a significant 22.63% in performance. Ablation studies reveal the impact of each component. We further conduct a further empirical analysis of MLLMs' vision capabilities. This analysis underscores the necessity of our hybrid Dual Reflection module. The study also illuminate key considerations for advancing

MLLMs in the future.

8 Limitations

This work has the following limitations: (i) The dataset is limited to a set of commonly encountered fields, potentially missing the unique requirements and challenges of less-represented or novel domains. This may impact the generalizability of our findings and highlights the need for future work to incorporate more diverse field types to more comprehensively evaluate SheetDesigner's performance. (ii) As shown in Appendix F, the model arranges elements sub-optimally in extreme cases involving a large number of components with varying sizes. This limitation is not unique to our approach and has also been reported in prior work (Arroyo et al., 2021; Lin et al., 2023b). Addressing such cases remains an open direction for future research.

References

Diego Martin Arroyo, Janis Postels, and Federico Tombari. 2021. Variational transformer networks for layout generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13642–13652.

Yolande E Chan and Veda C Storey. 1996. The use of spreadsheets in organizations: Determinants and consequences. *Information & Management*, 31(3):119–134.

Qin Chen, Yuanyi Ren, Xiaojun Ma, and Yuyang Shi. 2025a. Large language models for predictive analysis: How far are they? *arXiv preprint arXiv:2505.17149*.

Qin Chen and Guojie Song. 2025. Adaptive heterogeneous graph neural networks: Bridging heterophily and heterogeneity. *arXiv preprint arXiv:2508.06034*.

Qin Chen, Liang Wang, Bo Zheng, and Guojie Song. 2025b. Dagprompt: Pushing the limits of graph prompting with a distribution-aware graph prompt tuning approach. In *Proceedings of the ACM on Web Conference 2025*, pages 4346–4358.

Chin-Yi Cheng, Ruiqi Gao, Forrest Huang, and Yang Li. 2024. Colay: Controllable layout generation through multi-conditional latent diffusion. *arXiv preprint arXiv:2405.13045*.

Yutao Cheng, Zhao Zhang, Maoke Yang, Hui Nie, Chunyuan Li, Xinglong Wu, and Jie Shao. 2025. Graphic design with large multimodal model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 2473–2481.

Kamal Gupta, Justin Lazarow, Alessandro Achille, Larry S Davis, Vijay Mahadevan, and Abhinav Shrivastava. 2021. Layouttransformer: Layout generation and completion with self-attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1004–1014.

Joachim Häcker and Dietmar Ernst. 2017. *Financial Modeling: An Introductory Guide to Excel and VBA Applications in Finance*. Springer.

Hsiao Yuan Hsu, Xiangteng He, Yuxin Peng, Hao Kong, and Qing Zhang. 2023. Posterlayout: A new benchmark and approach for content-aware visual-textual presentation layout. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6018–6026.

HsiaoYuan Hsu and Yuxin Peng. 2025. Postero: Structuring layout trees to enable language models in generalized content-aware layout generation. *arXiv preprint arXiv:2505.07843*.

Forrest Huang, Gang Li, Xin Zhou, John F Canny, and Yang Li. 2021. Creating user interface mock-ups from high-level text descriptions with deep-learning models. *arXiv preprint arXiv:2110.07775*.

Zhaoyun Jiang, Jiaqi Guo, Shizhao Sun, Huayu Deng, Zhongkai Wu, Vuksan Mijovic, Zijiang James Yang, Jian-Guang Lou, and Dongmei Zhang. 2023. Layoutformer++: Conditional graphic layout generation via constraint serialization and decoding space restriction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18403–18412.

Zhaoyun Jiang, Shizhao Sun, Jihua Zhu, Jian-Guang Lou, and Dongmei Zhang. 2022. Coarse-to-fine generative modeling for graphic layouts. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 1096–1103.

Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. 2021. Constrained graphic layout generation via latent optimization. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 88–96.

Xiang Kong, Lu Jiang, Huiwen Chang, Han Zhang, Yuan Hao, Haifeng Gong, and Irfan Essa. 2022. Blt: bidirectional layout transformer for controllable layout generation. In *European Conference on Computer Vision*, pages 474–490. Springer.

Talia Lavie and Noam Tractinsky. 2004. Assessing dimensions of perceived visual aesthetics of web sites. *International journal of human-computer studies*, 60(3):269–298.

Hsin-Ying Lee, Lu Jiang, Irfan Essa, Phuong B Le, Haifeng Gong, Ming-Hsuan Yang, and Weilong Yang. 2020. Neural design network: Graphic layout generation with constraints. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 491–506. Springer.

Jianan Li, Jimei Yang, Jianming Zhang, Chang Liu, Christina Wang, and Tingfa Xu. 2020. Attribute-conditioned layout gan for automatic graphic design. *IEEE Transactions on Visualization and Computer Graphics*, 27(10):4039–4048.

Tao Li, Chin-Yi Cheng, Amber Xie, Gang Li, and Yang Li. 2024. Revision matters: Generative design guided by revision edits. *arXiv preprint arXiv:2406.18559*.

Jiawei Lin, Jiaqi Guo, Shizhao Sun, Weijiang Xu, Ting Liu, Jian-Guang Lou, and Dongmei Zhang. 2023a. A parse-then-place approach for generating graphic layouts from textual descriptions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23622–23631.

Jiawei Lin, Jiaqi Guo, Shizhao Sun, Zijiang Yang, Jian-Guang Lou, and Dongmei Zhang. 2023b. Layout-prompter: awaken the design ability of large language models. *Advances in Neural Information Processing Systems*, 36:43852–43879.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916.

Wei Liu, Liuan Wang, and Jun Sun. 2025a. Efficient object placement via llm and diffusion model. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Xuyuan Liu, Yinghao Cai, Qihui Yang, and Yujun Yan. 2024. Exploring consistency in graph representations: from graph kernels to graph neural networks. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Xuyuan Liu, Lei Hsiung, Yaoqing Yang, and Yujun Yan. 2025b. Spectral insights into data-oblivious critical layers in large language models. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 4860–4877. Association for Computational Linguistics.

OpenAI. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Stephen G Powell and Kenneth R Baker. 2019. *Business analytics: The art of modeling with spreadsheets*. John Wiley & Sons.

Soliha Rahman, Vinoth Pandian Sermuga Pandian, and Matthias Jarke. 2021. Ruite: Refining ui layout aesthetics using transformer encoder. In *Companion Proceedings of the 26th International Conference on Intelligent User Interfaces*, pages 81–83.

Yuanyi Ren, Haoran Ye, Hanjun Fang, Xin Zhang, and Guojie Song. 2024. ValueBench: Towards comprehensively evaluating value orientations and understanding of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2015–2040. Association for Computational Linguistics.

Jaejung Seol, Seojun Kim, and Jaejun Yoo. 2024. Posterllama: Bridging design ability of language model to content-aware layout generation. In *European Conference on Computer Vision*, pages 451–468. Springer.

Hengyu Shi, Junhao Su, Huansheng Ning, Xiaoming Wei, and Jialin Gao. 2025. Layoutcot: Unleashing the deep reasoning potential of large language models for layout generation. *arXiv preprint arXiv:2504.10829*.

Yiliu Sun, Yanfang Zhang, Zicheng Zhao, Sheng Wan, Dacheng Tao, and Chen Gong. 2025. Fast-slow-thinking: Complex task solving with large language models. *arXiv preprint arXiv:2504.08690*.

Hongbo Tang, Shuai Zhao, Jing Luo, Yihang Su, and Jinjian Yang. 2024. Layoutkag: Enhancing layout generation in large language models through knowledge-augmented generation. In *2024 3rd International Conference on Artificial Intelligence, Human-Computer Interaction and Robotics (AIHCIR)*, pages 292–299. IEEE.

Zecheng Tang, Chenfei Wu, Juntao Li, and Nan Duan. 2023. Layoutnuwa: Revealing the hidden layout expertise of large language models. *arXiv preprint arXiv:2309.09506*.

Yiqi Wang, Wentao Chen, Xiaotian Han, Xudong Lin, Haiteng Zhao, Yongfei Liu, Bohan Zhai, Jianbo Yuan, Quanzeng You, and Hongxia Yang. 2024. Exploring the reasoning abilities of multimodal large language models (mllms): A comprehensive survey on emerging trends in multimodal reasoning. *arXiv preprint arXiv:2401.06805*.

Tao Yang, Yingmin Luo, Zhongang Qi, Yang Wu, Ying Shan, and Chang Wen Chen. 2024. Posterllava: Constructing a unified multi-modal layout generator with llm. *arXiv preprint arXiv:2406.02884*.

Jiahao Zhang, Ryota Yoshihashi, Shunsuke Kitada, Atsuki Osanai, and Yuta Nakashima. 2024. Vascar: Content-aware layout generation via visual-aware self-correction. *arXiv preprint arXiv:2412.04237*.

Jiarui Zhang, Mahyar Khayatkhoei, Prateek Chhikara, and Filip Ilievski. 2025a. Mllms know where to look: Training-free perception of small visual details with multimodal llms. *arXiv preprint arXiv:2502.17422*.

Peirong Zhang, Jiaxin Zhang, Jiahuan Cao, Hongliang Li, and Lianwen Jin. 2025b. Smaller but better: Unifying layout generation with smaller large language models. *International Journal of Computer Vision*, pages 1–27.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Xinru Zheng, Xiaotian Qiao, Ying Cao, and Rynson WH Lau. 2019. Content-aware generative modeling of graphic design layouts. *ACM Transactions on Graphics (TOG)*, 38(4):1–15.

A Potential Risks

This work focuses on automating the generation of spreadsheet layouts through SheetDesigner. While automation offers clear efficiency and usability benefits, it also introduces certain risks. In particular, over-reliance on automatically generated layouts may lead to reduced user control or unintended formatting choices that do not align with domain-specific expectations. Additionally, if deployed in high-stakes environments (e.g., finance or healthcare), errors in layout generation could affect data interpretation and decision-making (Chen et al., 2025a). It is therefore important to incorporate mechanisms for human oversight, validation, and customization to mitigate these risks and ensure responsible deployment.

B Declaration of Generative AI Tools

Generative AI tools were used solely for the purpose of language polishing. All ideas, experiments, analyses, and writing were originally developed by the authors.

C Evaluation of Spreadsheet Layouts

In this section we detail the seven aspects of evaluation metrics. All metrics, except for *Overlap*, are scaled from $(0, 1]$, where higher scores indicate better performance. For *Overlap*, a score of 0 signifies no overlaps, while increasingly negative scores reflect a greater degree of overlap.

Fullness This aspect evaluates the spatial utilization of generated layouts (Hsu et al., 2023). We first identify the top-left and bottom-right corners to determine the background region size R_{bg}. Next, we mark the areas occupied by layout components, denoted as R_{ft}. Note that when calculating area sizes, the row and column dimensions G are taken into account. The *fullness* metric is defined as:

$$S_{\text{full}}([\tilde{C}], G) = \begin{cases} 1, & \text{if } \frac{\text{size}(R_{\text{ft}})}{\text{size}(R_{\text{bg}})} \geq \theta_{\text{full}}, \\ \frac{\text{size}(R_{\text{ft}})}{\text{size}(R_{\text{bg}})}, & \text{otherwise.} \end{cases} \tag{1}$$

where size$(\cdot)$ represents the two-dimensional area measurement, considering both row and column dimensions, θ_{full} is a threshold value. This metric encourages the generation of compact and practically useful spreadsheet layouts while allowing sufficient space for line breaks and separation where necessary by assigning full scores for fullness greater than θ_{full}.

Compatibility This aspect evaluates how well the provided row heights and column widths accommodate the corresponding text within each cell (Hsu et al., 2023). To achieve this, we first approximate the average pixel dimensions of the text, assuming a width of $\mathcal{W}_{\text{text}}$ pixels per character and a height of $\mathcal{H}_{\text{text}}$ pixels per line. We then convert the row heights and column widths into pixel units via . The compatibility scores in both the horizontal and vertical directions are defined as the normalized average compatibility scores across all cells:

$$\begin{aligned} S_{\text{compt\_h}} &= \frac{1}{1 + \frac{1}{M} \left| \sum_{i=1}^{M} \frac{S_h w_i}{\mathcal{W}_{\text{text}} l_i + P_h} - 1 \right|}, \\ S_{\text{compt\_v}} &= \frac{1}{1 + \frac{1}{M} \left| \sum_{i=1}^{M} \frac{h_i}{\mathcal{H}_i n_i + P_v} - 1 \right|}. \end{aligned} \tag{2}$$

Here, M represents the total number of data-containing cells. The variables w_i and h_i denote the width and height of the i-th cell, respectively. The term l_i corresponds to the number of characters in the cell's content, while n_i represents the number of text lines in the cell. S_h denotes the factor translating spreadsheet cell width to pixels. P_h, P_v denotes the padding space for horizontal and vertical. We apply the shifted reciprocal transformation $f(x) = \frac{1}{1+|x-1|}$ to normalize the scores. This ensures that cells that are either too wide or too narrow for the given text receive lower scores, while optimal compatibility results in a score approaching 1.

Component Alignment Alignment lies in the core of assessing a layout's practical usage and beauty (Li et al., 2020). We begin by measuring general alignment between components and then extend the evaluation to type-aware and relation-aware alignment. Given a list of components, we identify frequently occurring positions and assess alignment. Deviations from these positions contribute to an alignment violation score $S_{\text{vio\_h}}$ and $S_{\text{vio\_v}}$. Formally, we detect the top-k most frequent positions in both directions and check whether each component aligns with them. A perfect match results in no violation; otherwise, the violation score ($S_{\text{vio\_h}}$ or $S_{\text{vio\_v}}$) increases by one. We then normalize the final alignment scores via the reciprocal transformation $f(x) = \frac{1}{1+x}$, ensuring that perfect alignment results in a score of 1, while greater misalignment lowers the score:

$$S_{\text{align\_h}} = \frac{1}{1 + \frac{1}{N} S_{\text{vio\_h}}}, \; S_{\text{align\_v}} = \frac{1}{1 + \frac{1}{N} S_{\text{vio\_v}}}. \tag{3}$$

Type-aware Alignment Type-aware alignment focus on measuring how components of the same type aligns with each other. We classify the components by their type, and calculate the alignment scores within each group. The final score of type-aware alignment is calculated by averaging the scores between the types.

Relation-aware Alignment Relation-aware alignment evaluates how referenced components correspond to each other. For instance, in a main table and its summary table, proper alignment ensures visual hierarchy and perceptual integrity. We detect the groups of related components and measure the alignment within each group. The final score of relation-aware alignment is calculated by averaging the scores between the groups.

Balance Balance assesses whether components are evenly distributed to maintain visual equilibrium in a spreadsheet (Lavie and Tractinsky, 2004). A well-balanced layout should be achieved both vertically and horizontally, avoiding excessive weight on one side, such as clustering components on the left while leaving the right sparsely populated. Technically, the spreadsheet is divided vertically and hierarchically into two parts[3]. The *fullness* metric is then applied to each part, and the final scores are computed as:

$$\begin{aligned} S_{\text{balance\_h}} &= 1 - \frac{\left| S_{\text{full}}([\tilde{C}]_{\text{left}}, G) - S_{\text{full}}([\tilde{C}]_{\text{right}}, G) \right|}{S_{\text{full}}([\tilde{C}]_{\text{left}}, G) + S_{\text{full}}([\tilde{C}]_{\text{right}}, G)} \\ S_{\text{balance\_v}} &= 1 - \frac{\left| S_{\text{full}}([\tilde{C}]_{\text{upper}}, G) - S_{\text{full}}([\tilde{C}]_{\text{down}}, G) \right|}{S_{\text{full}}([\tilde{C}]_{\text{upper}}, G) + S_{\text{full}}([\tilde{C}]_{\text{down}}, G)}, \end{aligned} \tag{4}$$

where $[\tilde{C}]_{\text{name}}$ represents the list of components belonging to the corresponding part. For a well-balanced layout, the balance score is 1. The more imbalanced the layout, the lower the score.

Overlap Overlap assesses whether components occupy previously assigned areas (Li et al., 2020). We iterate through all components, marking the background to compute the overlap score, where each pair of collision increases the overlap count C_{overlap} by 2. The final score is given by:

$$S_{\text{overlap}} = -\frac{C^2_{\text{overlap}}}{N} \tag{5}$$

We impose a strong penalty on overlap by applying a quadratic term, C^2_{overlap}, since even minor overlaps severely undermine practical utility. A perfectly non-overlapping layout receives a score of 0, while overlapping components are penalized with increasingly negative values. Theoretically, the scores fall within the range $[-N(N-1)^2, 0]$, where every pair of components overlaps. In practice, however, the scores typically lie within $(-1, 0)$ in most cases.

Settings We set $\theta_{\text{full}} = 0.8$ for *fullness*. For *compatibility*, we approximate the $\mathcal{W}_{\text{text}} = 12$ and $\mathcal{H}_{\text{text}} = 15$ for the default settings of Calibri with a font size of 12 in English in this paper. For other fonts and languages, the corresponding constants can be adjusted accordingly. We set the padding terms $P_h = 40$ and $P_v = 10$. The translating factor S_h is set to 7.

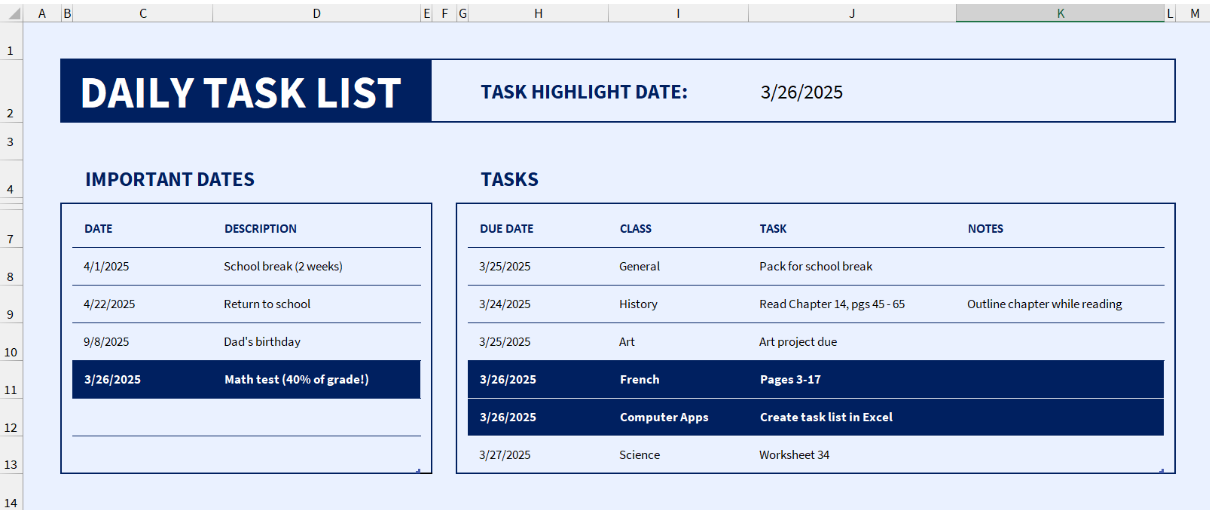
We do not use metrics like Fréchet Inception Distance to assess similarity to real layouts. While real layouts can serve as useful references, they are not the only valid designs. Effective spreadsheet layouts can be arranged in many different ways. There is no definitive ground truth for what a layout should be. Instead, we evaluate generated layouts using the seven criteria described above. A layout does not need to resemble existing ones to be useful. If it scores well on these criteria, it can still be effective for practical applications. This evaluation approach also encourages diversity in the generated layouts.

D Additional Methodology Details

D.1 Exemplar Image $\mathcal{I}_S$

During structural placement, we provide an exemplar image $\mathcal{I}_S$ to the MLLMs as a reference for topic-aware component arrangement. This exemplar is selected based on the topic or application context, as the topic significantly influences layout structures. For example, given the topic "Check List," layouts from domains such as finance, education, IT, or healthcare tend to share common

[3] Components spanning the midpoint are proportionally allocated to both parts



DAILY TASK LIST

TASK HIGHLIGHT DATE: 3/26/2025

IMPORTANT DATES

DATE	DESCRIPTION
4/1/2025	School break (2 weeks)
4/22/2025	Return to school
9/8/2025	Dad's birthday
3/26/2025	Math test (40% of grade!)

TASKS

DUE DATE	CLASS	TASK	NOTES
3/25/2025	General	Pack for school break	
3/24/2025	History	Read Chapter 14, pgs 45 - 65	Outline chapter while reading
3/25/2025	Art	Art project due	
3/26/2025	French	Pages 3-17	
3/26/2025	Computer Apps	Create task list in Excel	
3/27/2025	Science	Worksheet 34	

Figure 5: An exemplar image of topic "To-do Lists and Calendars".

structural patterns. We provide an example in Figure 5

In our implementation, we curate 5–10 exemplar images for each of the 13 general spreadsheet topics in Table 8. For each structural placement task, one exemplar from the same topic is randomly selected. Importantly, all exemplars are excluded from the SheetLayout test set, ensuring there is no risk of information leakage.

D.2 Sketch Image Generation for Layouts

For the sketch image of generated layouts, we first detect the maximum grid size of the layout and define the background. Then, for each component we color the corresponding cells with corresponding texts. Different types of elements are colored with different colors. This sketch image provide clear visual information of the fullness, alignment (and its variants), balance, and overlap. We provide the algorithm in 2, and an example of the images in Figure 6.

Algorithm 1 PlotComponent

```
Require: $K$ component, $\mathcal{C}$ canvas, $\mathcal{S}$ style map
$L \leftarrow K.\text{layout.location}$
$(c_s, e_c) \leftarrow \text{PhraseLocation}(L)$
$(r_s, c_s) \leftarrow \text{CellToIndex}(c_s)$
$(r_e, c_e) \leftarrow \text{CellToIndex}(e_c)$
$\Delta c \leftarrow c_e - c_s + 1$
$\Delta r \leftarrow r_e - r_s + 1$
$\mathit{fill} \leftarrow \mathcal{S}[\,K.\text{type}\,]$
$\text{DrawRectangle}(\mathcal{C}, c_s, r_s, \Delta c, \Delta r, \mathit{fill})$
$\text{DrawText}(\mathcal{C}, K.\text{id}, c_s + \frac{\Delta c}{2}, r_s + \frac{\Delta r}{2})$
```

Algorithm 2 PlotLayout

```
Require: $\Lambda$ layout data
$\mathcal{C} \leftarrow \text{InitCanvas}()$
$\mathcal{S} \leftarrow \text{DefineStyles}()$
$(R_{\max}, C_{\max}) \leftarrow \text{ComputeGridSize}(\Lambda)$
$\text{ConfigureCanvas}(\mathcal{C}, R_{\max}, C_{\max})$
for all $\mathcal{K} \in \Lambda$ do
    $\text{PLOTCOMPONENT}(K, \mathcal{C}, \mathcal{S})$
end for
```

D.3 Thresholds and Instructions in Dual Reflection

We present the reflection-triggering thresholds in Table 5. If any aspect score from SheetRanker falls below its corresponding threshold, the relevant instruction from Table 6 is appended to the reflection prompt. All scores, except *Overlap*, are on a unified scale of $(0, 1)$; thus, we adopt a moderate threshold of 0.5. For *Overlap*, where any intersection degrades layout usability, a strict threshold of 0 is used—any overlap triggers revision.

These thresholds are tunable hyper-parameters. Lower values (e.g., 0.3) reduce computation by filtering only extreme cases, while higher values (e.g., 0.7) enhance quality at the cost of increased processing due to more frequent revisions.

E Dataset Details

In this section, we summarize the dataset statistics. The 3,326 spreadsheets span 10 domains (see Table 7) and cover 13 common topics grouped by the application context (see Table 8).

Datasets were acquired from a variety of public platforms. These include official government open

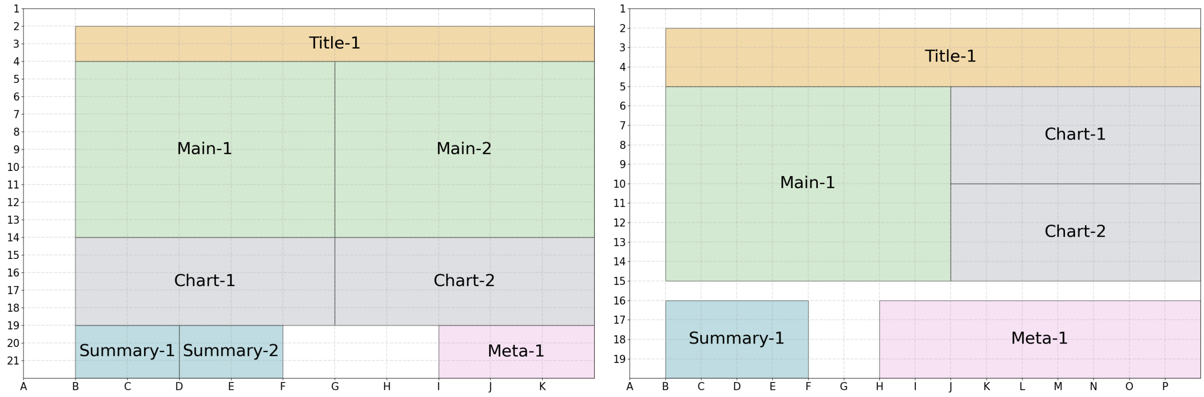


Figure 6: Examples of the sketch images generated from layouts.

Table 5: Thresholds for triggering specific instructions in Dual Reflection.

Aspect	Threshold
Fullness	0.5
Overlap	0.0
Alignment	0.5
T-Alignment	0.5
R-Alignment	0.5
Balance	0.5

data websites (e.g., data.gov) that share public reports or tools in spreadsheet formats; repositories for open science data, such as Zenodo [4], where researchers deposit supplementary materials; and digital libraries like the Internet Archive, which preserve publicly accessible documents. All pre-existing cell content within these publicly sourced spreadsheets was subsequently anonymized using offline LLMs, which replaced original data with synthetically generated content that is semantically coherent with the original data's type and structure. The specific terms and numerical values were excluded from the input and the output cell content was generated entirely by privacy-secure LLMs, ensuring no sensitive data was retained. This process preserves the structural and contextual integrity of the dataset while mitigating privacy risks.

For every acquired spreadsheet, its specific licensing and terms of use were meticulously verified to ensure suitability for academic research and inclusion in this layout-focused dataset. Preference was given to content in the public domain or under permissive open licenses, such as Creative Commons (e.g., CC0, CC BY).

In addition to these publicly sourced materials, the dataset incorporates spreadsheet layouts originating from Microsoft 365 Create [5] published online. The incorporation of these anonymized layouts for academic research was authorized under a formal agreement with Microsoft. This agreement required the thorough anonymization of all original cell content prior to inclusion in the dataset.

Table 7: Statistics of the dataset on domain distribution.

Domain	#Sheets
Business and Finance	445
Marketing and Sales	397
Engineering and Manufacturing	386
Sports and Entertainment	320
Healthcare and Medical	330
Education and Research	355
Personal and Daily Life	302
Technology and IT	290
Agriculture and Food	260
Hospitality and Tourism	241
Total	3326

[4] https://zenodo.org

[5] https://create.microsoft.com/en-us

Table 6: Specific instructions in Dual Reflection.

Aspect	Instruction
Fullness	This spreadsheet is with much empty space. Consider redistribute the elements to minimize empty space.
Overlap	This spreadsheet has overlapping components. Consider moving the components to avoid overlapping
Alignment	The horizontal alignment of components is not good. Consider align the top of the components
Alignment	The vertical alignment of components is not good. Consider align the left of the components
T-Alignment	The type-specific horizontal alignment of components is not good. Consider align the top of the components according to their types
T-Alignment	The type-specific vertical alignment of components is not good. Consider align the left of the components according to their types
R-Alignment	The relation-specific horizontal alignment of components is not good. Consider align the top of the components according to their relations
R-Alignment	The relation-specific vertical alignment of components is not good. Consider align the left of the components according to their relations
Balance	The horizontal balance of components is not good. Consider distribute the components horizontally
Balance	The vertical balance of components is not good. Consider distribute the components vertically

Table 8: Statistics of the dataset on topic distribution.

Topic	#Sheets
Financial Management and Forecasting	499
Data and Task Logs	394
Staff Scheduling and Shift Management	301
Performance and KPI Dashboards	298
Event Scheduling and Planning	288
Inventory and Asset Management	269
Report and Publication Tracking	267
Maintenance Scheduling	155
Marketing Campaign Tracking	151
Project Scheduling	150
To-do Lists and Calendars	180
Travel Itinerary and Planning	165
Goal and Habit Tracking	159
Total	3326

F Case Study

F.1 Failure Cases

Parallel to the general cases revealed in Figure 2, in this sub-section we analyze the failure cases of SheetDesigner (with GPT-4o). We find that there are two frequent issues in instances with low scores: (1) some easy-to-get alignment score is not achieved; (2) in cases of extremely enormous components with different sizes, these elements are arranged non-optimally. We provide two examples in Figure 7. In the upper figure, there are some easy steps like expanding the title `T1` to the right border of `MT1` (Note that title components is allowed to resize), and moving `MT2` down to align the downsize border of `M1` will increase the score of alignment. In the lower figure with seven metadata tables of varying shapes, it fails to provide proper arrangement, leading to a somewhat messy layout, whereas still of certain organization. This analysis reveals the future objectives for improving the SheetDesigner.

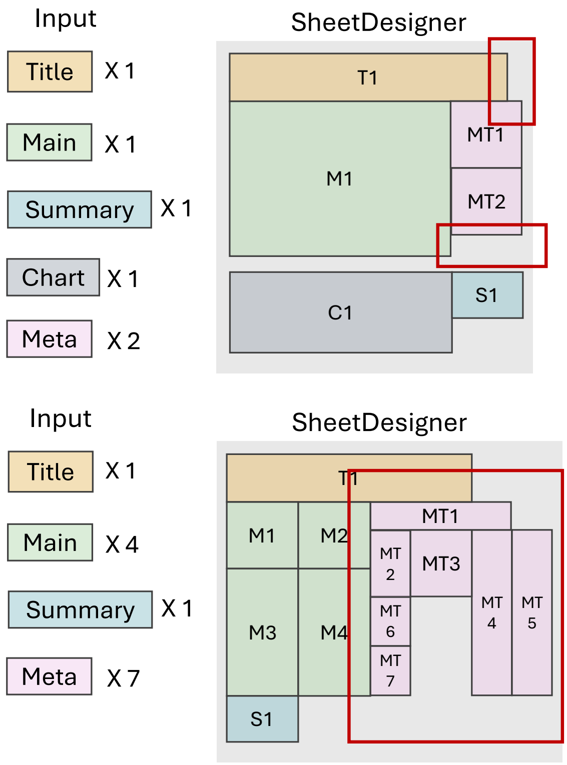


Figure 7: Fail Case study of SheetDesigner

G Details of Ablation Study

Comprehensive details of the ablation study are presented in Table 9 and Table 10, which report the ablation scores corresponding to the lower 5% and 1% tails of the score distributions, respectively.

H Additional Experiments

In this section we provide some additional experiments.

H.1 Choice of Exemplar Images

We provide the ablation on the choice of exemplar images in Table 11. Table 2 demonstrates that topic-guided exemplar retrieval outperforms methods that do not leverage topic exemplars, achieving a 2.71% performance improvement. We also provide an additional comparison below, introducing a variant that uses purely random exemplars without regard to topic.

Table 11: Ablation on the choice of exemplar images

	Score
SheetDesigner (Topic Exemplar)	4.898
SheetDesigner (Random Exemplar)	4.780
SheetDesigner (No Exemplar)	4.766

This demonstrates that even a simple topic-based selection strategy offers clear benefits. Although more fine-grained semantic retrieval could further improve performance, it primarily enhances content-level alignment rather than layout structure, making it less directly relevant to the layout generation task. Additionally, such approaches are often computationally intensive and tailored to specific tasks. We therefore consider them more appropriate for future work.

H.2 Traditional Methods with R1C1 Format

Additionally, we conducted an experiment equipping LayoutPrompter with the R1C1 coordinate system in Table 12. While this improved its overall performance, a significant gap remained between SheetDesigner and LayoutPrompter, highlighting the effectiveness of our design beyond the form of coordinate system.

H.3 Traditional Transformer-based Methods with More Training Data

Below, we present the results of an experiment conducted under a label-rich setting, using a 60%-20%-20% train-validation-test split. With more labels available, traditional methods such as LayoutFormer++ and Coarse-to-Fine achieve substantial performance gains; however, they still perform significantly below SheetDesigner.

H.4 Statistics of Experimental Results

We provide the standard deviation of some experimental results of Table 1 in Table 14.

I Experimental Environment Details

We use GPT-4o via the official OpenAI API[6], while all other models are run locally on a server equipped with an AMD EPYC 7V13 64-Core Processor, 866 GB of RAM, and four NVIDIA A100 GPUs with a total of 320 GB GPU memory. The experiments required approximately 1,200 GPU hours in total. The cost associated with GPT-4o API usage is estimated at approximately $2,892.

For GPT-4o, we set a maximum token limit of 16,384 per invocation, with top-p set to 0.95 and a temperature of 0.7. Structured output is enabled. For Vicuna and LLaVA models, we follow the hyperparameter settings provided in their official implementations. The threshold values for Dual LoRA are selected based on a balance between performance and average token cost, as detailed below

We conduct a token cost analysis running SheetDesigner on different models in Table 15 [7]. We calculate the total token costs using the official OpenAI API response for GPT-4o, while the token length for Vicuna/LLaVA models is determined by the specific tokenizer used. Based on current pricing, a single run with GPT-4o costs approximately 0.0029$, totaling around 0.0719$ per instance for a complete execution.

J Detailed Prompts

In this section we provide the detailed prompts for the SheetDesigner, where {...} denotes the placeholder to fill in the corresponding data. For all the prompts the input data includes the different stage of the developing spreadsheet layout, from raw spreadsheet data to the revised layout to be populated. For the prompt of Dual Reflection, there are additional inputs of the specific instructions

[6] https://openai.com

[7] Note that revision is conditionally triggered; runs without revision are marked with a cost of 0 in this procedure

Table 9: Ablation study on 5%-low scores.

	Fullness	Compatibility		C-Alignment		T-Alignment		R-Alignment		Balance		Overlap	Weighted Total
		Horizontal	Vertical	Horizontal	Vertical	Horizontal	Vertical	Horizontal	Vertical	Horizontal	Vertical		
SheetDesigner	0.887	0.523	0.882	0.742	0.861	0.391	0.494	0.434	0.440	0.815	0.885	-0.057	4.064
w/o Topic	0.869	0.510	0.863	0.727	0.844	0.369	0.467	0.411	0.408	0.798	0.855	-0.056	3.939 (↓ 3.06%)
w/o Reflection-Rule	0.776	0.518	0.885	0.685	0.780	0.377	0.380	0.361	0.372	0.734	0.852	-0.068	3.680 (↓ 9.44%)
w/o Reflection-Vision	0.783	0.525	0.866	0.724	0.830	0.394	0.466	0.421	0.432	0.741	0.860	-0.098	3.814 (↓ 6.13%)
w/o Reflection	0.755	0.522	0.867	0.656	0.765	0.357	0.365	0.355	0.356	0.719	0.839	-0.083	3.572 (↓ 12.08%)
w/o SheetRanker	0.785	0.460	0.865	0.724	0.812	0.403	0.367	0.425	0.413	0.760	0.829	-0.068	3.746 (↓ 7.81%)
w/o Vision	0.753	0.503	0.856	0.717	0.818	0.360	0.423	0.409	0.419	0.734	0.836	-0.102	3.688 (↓ 9.23%)

Table 10: Ablation study on 1%-low scores.

	Fullness	Compatibility		C-Alignment		T-Alignment		R-Alignment		Balance		Overlap	Weighted Total
		Horizontal	Vertical	Horizontal	Vertical	Horizontal	Vertical	Horizontal	Vertical	Horizontal	Vertical		
SheetDesigner	0.876	0.521	0.896	0.770	0.879	0.253	0.263	0.385	0.360	0.845	0.867	-0.287	3.608
w/o Topic	0.867	0.521	0.893	0.756	0.858	0.255	0.256	0.338	0.333	0.800	0.795	-0.293	3.477 (↓ 3.66%)
w/o Reflection-Rule	0.829	0.525	0.887	0.611	0.756	0.187	0.189	0.301	0.345	0.720	0.746	-0.347	3.115 (↓ 13.66%)
w/o Reflection-Vision	0.812	0.523	0.890	0.759	0.854	0.241	0.267	0.384	0.352	0.709	0.723	-0.417	3.246 (↓ 10.05%)
w/o Reflection	0.785	0.519	0.883	0.604	0.744	0.175	0.182	0.287	0.338	0.702	0.719	-0.429	2.933 (↓ 18.73%)
w/o SheetRanker	0.830	0.503	0.880	0.759	0.831	0.249	0.232	0.380	0.372	0.780	0.815	-0.363	3.368 (↓ 6.68%)
w/o Vision	0.833	0.516	0.893	0.811	0.848	0.224	0.182	0.396	0.390	0.693	0.712	-0.445	3.220 (↓ 10.75%)

Table 12: Results of traditional methods with R1C1 format.

	Fullness	Compatibility	C-Alignment	T-Alignment	R-Alignment	Balance	Overlap	Weighted Total
SheetDesigner	0.981	0.718	0.782	0.823	0.691	0.907	-0.003	4.898
LayoutPrompter	0.804	0.510	0.649	0.585	0.703	0.706	-0.167	3.789
LayoutPrompter(R1C1)	0.812	0.509	0.672	0.604	0.713	0.693	-0.142	3.861

Table 13: Results of traditional transformer-based methods with 60%-20%-20% train-validation-test split

	Fullness	Compatibility	C-Alignment	T-Alignment	R-Alignment	Balance	Overlap	Weighted Total
SheetDesigner	0.978	0.721	0.769	0.831	0.689	0.901	-0.004	4.885
LayoutPrompter	0.803	0.513	0.653	0.594	0.699	0.712	-0.158	3.816
Coarse-to-Fine	0.583	0.462	0.573	0.51	0.583	0.652	-0.132	3.231
LayourFormer++	0.631	0.464	0.63	0.504	0.621	0.69	-0.116	3.424

Table 14: Statistics of experimental results.

	Fullness	Compatibility	C-Alignment	T-Alignment	R-Alignment	Balance	Overlap	Weighted Total
SheetDesigner	0.978±0.03	0.721±0.08	0.769±0.12	0.831±0.13	0.689±0.16	0.901±0.08	-0.004±0.13	4.885 ±0.28

Table 15: Token cost analysis of SheetDesigner, reporting average token cost per instance, for a single run, and for a full run with three repeats.

	Pre-Process	Structure	Revise	Content	Total (Single)	Total
Vicuna-7B	278.8	1301.2	310.2	1080.5	2970.7	7734.1
Vicuna-13B	283.1	1339.6	262.3	1154.5	3039.5	8027.7
GPT-4o	280.5	1165.7	234.2	958.4	2638.9	6887.3

which are triggered by rules, the full set of specific instructions are in Table 6.

Prompts for Pre-processing

## **Task**
You will receive a list of spreadsheet components, each accompanied by comments, descriptions, and detailed data. Your task is to identify pairs of components that have a logical relationship. For example, if `summary_table_1` summarizes data from `main_table_1`, you should extract and present this relationship as: `(main_table_1, summary_table_1)`
## **Hints**
(1) Relationships can be based on dependencies, references, or summarization within the spreadsheet structure. (2) If `component_A` describes, summarizes, or illustrates data derived from `component_B`, then `A` and `B` are related. (3) Organize the results in list of lists, where the inner list should be a 2-component list like `[A, B]`.
## **Spreadsheet Components**
{...}

Prompts for Structure Placement

## **Task**
I will provide you with a spreadsheet skeleton with multiple elements including title, main-table, meta-data, summary-table, and charts in JSON. The task is to place the elements by setting their position in the spreadsheet in a good structure.
## **Instructions**
There are some hints to place the elements:
- The location of elements should be provided via the "location" attribute, which should be a list of two strings indicating the left-top and bottom down corner of the element. Example: ["A1", "C3"].
- The elements placed should align with each other. You can also maintain some symmetry. - Specially, maintain a type-aware alignment between element groups. For example, the metadata tables should be aligned with each other. - Specially, maintain a relation-aware alignment between elements. For example, the chart demonstrating certain main-table should be aligned with that main-table.
- Avoid overlapping the elements.
- The spreadsheet is a 2D grid, so don't place the elements wholly horizontally or vertically. Arrange them in a compound manner.
- When placing the elements, you can leave some space as margins between them. But, avoid leaving too much space empty in the whole spreadsheet.
- Place the elements considering the relationship between them, for example, the summary-table should be placed below the main-table.
- You can change the size of the components following these rules: - Title: can be arbitrarily resized. - Main-table: you can add empty rows (or namely, changing the height of the table) to make it look good. But, the width should be the same as the given width. - Meta-data, summary data: not re-sizable.
- The title should be placed at the top of the spreadsheet, spanning all active columns where there are components.
- Do not duplicate the components, each type of components should be placed under the corresponding lists.
## **Spreadsheet Skeleton Set**
{...}

Prompts for Dual Reflection

## **Task**
I will provide you with a spreadsheet layout with multiple elements including title, main-table, meta-data, summary-table, and charts in JSON. Your task is to revise the structure following the instructions. I will first provide you with the general instructions, then is the specific instructions I want you to follow. You will need to revise the structure of the spreadsheet accordingly.
## **General Instructions**
<Instructions for structure placement>
## **Specific Instructions**
{...}
## **Spreadsheet layout**
{...}

Prompts for Content Population

## Task
I will provide you with a spreadsheet with multiple elements including title, main-table, meta-data, summary-table, and charts in JSON. These components include their location in the spreadsheet, as well as their content. Your task is to generate the proper line heights and column widths for the spreadsheet, as well as adding line breaks to the content. The fundamental goal is to (1) make the cells compatible with the content, and (2) make the spreadsheet visually appealing.

## Instructions
There are some hints: - Carefully consider the content of each cell and adjust the row height and column width accordingly. - For a cell with lengthy content, you can either wrap the text for a line break, or increase the column width. - Try assigning line heights and column widths and line breaks to make (1) the spreadsheet is balanced vertically and horizontally, and (2) the neither too compact nor too much empty spaces.. - You may assume a default font setting of Calibri 11 and excel standard column width and row heights, where 1 character of text are compatible with 0.65 of column width.

## Spreadsheet Skeleton Set with Contents
{...}

K Future Works

Future work can extend this research in several promising directions:

- Graph-based Representation Learning: We plan to leverage the heterogeneous graph structure of spreadsheet components (Chen and Song, 2025) and apply advanced graph learning techniques (Liu et al., 2024; Chen et al., 2025b). This will enable a more comprehensive modeling of component relationships and facilitate more powerful representation learning.

- Expanded Ethical Analysis: We aim to broaden the ethical analysis to explore other critical dimensions of AI safety, such as the problem of value and preference alignment between humans and AI systems (Ren et al., 2024).