

NEST: A Benchmark for Explicit Structural Parsing on Native Empty Templates

Chen Peng¹, Tianci He¹, Hongye Chen¹,
Jiayi Huang¹, Kangdi Hao¹, Yiping Song^{2*}

¹Jishou University, Jishou, Hunan, China

²National University of Defense Technology, Changsha, Hunan, China

chen.peng@jsu.edu.cn, songyiping@nudt.edu.cn

Abstract

Certain document automation scenarios, such as intelligent form infilling and privacy-strict parsing, require models to comprehend unpopulated templates. Yet existing benchmarks mainly use content-filled tables, leaving the template side of document understanding under-evaluated. To bridge this gap, we introduce NEST, a benchmark of 828 Native Empty Structural Templates from real-world, born-digital forms. NEST removes filled content while preserving complex hierarchical layouts and guided selection options in authentic administrative workflows. Through Structural Schema Parsing, Cell-Header Alignment, and Generative Infilling, NEST directly evaluates structural reasoning without content-based shortcuts. Experiments reveal a clear template–instance gap: state-of-the-art LLMs drop by 9.5 percentage points on average when evaluated on unpopulated templates rather than populated instances. We further show that Generative Infilling is a data-efficient tuning signal for template completion, raising infilling accuracy by 16.37 percentage points and the three-task average by 8.19 percentage points with only 662 (80%) training templates. NEST thus provides a benchmark for measuring template-level understanding and highlights a data-efficient route for improving template completion. The code, data, and results for this paper are available at <https://github.com/cpeng-research/NEST-Benchmark>.

1 Introduction

Certain document automation tasks, including intelligent form infilling and privacy-strict parsing, require LLMs to understand blank templates before user data is introduced. Agents must locate input primitives for automated data entry (Toles et al., 2026; Xie et al., 2025); privacy-sensitive document workflows further motivate minimizing

*Corresponding author.

Job Application Information Form

NO: _____

Position Applied For: _____ Desired Salary: _____ Phone Number: _____

Full Name	Identification	Date of Birth		
Gender	MS	Ethnicity		
Height	Weight	POB		
Education	University	Major		
Current Residence	Available Start Date			
Address				
Application Source	Internship: ☐ University Referral ☐ Employment Agency ☐ Social Recruitment: ☐ Talent Market ☐ Recruitment Website			
Family Members	Name	Gender	Age	Relationship
	Work Unit			
Education History	From to	Institution	Major	Degree/Educational Level
	Phone			
Language Proficiency	Standard English: ☐ Basic ☐ Good ☐ Fluent ☐ Japanese: ☐ English Levels: ☐ CET-4 ☐ CET-6 ☐ TEM-8			
Computer Proficiency				

Nested Item Lists

Figure 1: Representative unpopulated template from NEST. The form contains irregular key-value mapping, merged regions, checkable options, and nested lists, illustrating challenges beyond standard relational grids.

data sent to remote services and using local tools where possible (Li et al., 2023; Maiya, 2025). Thus, empty-template understanding is central to secure and autonomous document intelligence.

Existing table understanding benchmarks mostly use content-filled tables and QA/FV-style tasks. Filled values can provide additional anchors for interpreting table relationships: for example, a value such as “Male” may make the corresponding field easier to infer as “Gender”. Real administrative forms contain irregular key-value mappings, merged regions, nested item lists, and hybrid input primitives, as illustrated in Figure 1. Without content, these templates require genuine spatial and topological reasoning.

We introduce NEST, a benchmark of 828 Native Empty Structural Templates from authentic administrative workflows, collected as originally unpopulated templates rather than redacted from filled instances. As shown in Figure 2, NEST evaluates three complementary abilities on unpopulated forms: recovering the logical schema, grounding

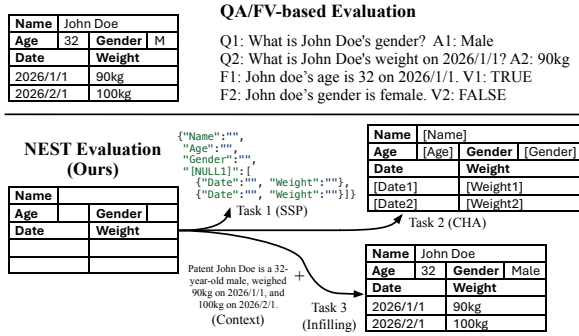


Figure 2: Template-level evaluation in NEST versus conventional filled-instance evaluation. Conventional QA/FV evaluates populated tables, while NEST evaluates the underlying empty template through SSP, CHA, and Generative Infilling.

empty fields to their governing headers, and using the template structure to fill values from context. We operationalize these abilities through Structural Schema Parsing (SSP), Cell-Header Alignment (CHA), and Generative Infilling.

Experiments show that performance on populated instances does not fully transfer to empty templates: state-of-the-art LLMs drop by 9.5 percentage points on average under the template-only setting. In other words, understanding a filled form instance is not equivalent to understanding the reusable template from which it is instantiated.

Beyond evaluation, empty templates also provide a privacy-preserving training substrate, since template-to-instance supervision can be synthesized from public templates without collecting proprietary user-filled records. We therefore use Generative Infilling as a fine-tuning objective that trains models to project plausible content into blank topological boundaries. With only 662 (80%) training templates, it raises infilling accuracy by 16.37 percentage points and improves the three-task average by 8.19 percentage points, although transfer to schema parsing and cell-header alignment is weaker and model-dependent. NEST offers both a rigorous benchmark for template understanding and a data-efficient path for improving template completion.

2 Related Work

Existing research on table understanding mainly follows three paradigms: visual structure recognition, downstream reasoning, and fine-grained structural probing. Table 1 summarizes representative benchmarks and how NEST differs from them.

2.1 Visual Structure and Information Extraction

Visual approaches treat tables and forms as document images, aiming to recover either physical layouts or semantic entities. For regular tables, early TSR benchmarks such as ICDAR 2013 and cTDA established table detection settings (Göbel et al., 2013; Gao et al., 2019), while PubTabNet, FinTabNet, and PubTables-1M scaled supervision for grid reconstruction, hierarchical headers, alignment normalization, and canonicalization (Zhong et al., 2020; Zheng et al., 2021; Smock et al., 2022). However, these datasets mainly target regular relational grids from scientific or financial documents, rather than irregular administrative templates.

A related line studies Key Information Extraction (KIE) from visually rich forms. FUNSD and XFUND evaluate entity recognition and key-value linking in filled scanned forms (Jaume et al., 2019; Xu et al., 2022), and SIBR extends this direction to more challenging real-world documents with hierarchical entity linking (Yang et al., 2023). These tasks resemble our cell-header alignment setting, but their links are grounded in populated values, making them unsuitable for testing empty-template understanding. Moreover, many real workflows operate on born-digital documents with directly extractable markup; RealHiTBench reports higher average performance for text-form table inputs than image inputs on its hierarchical-table benchmark (Wu et al., 2025a). NEST therefore focuses on high-fidelity markup and evaluates templates before value instantiation.

2.2 Markup-Based Benchmarks and Downstream Reasoning

Markup-based benchmarks usually assess table understanding through downstream QA or Fact Verification over populated tables. They have evolved from flat-table reasoning in WikiTableQuestions and TabFact (Pasupat and Liang, 2015; Chen et al., 2020) to hierarchical and numerical reasoning in HiTab and MultiHierTT (Cheng et al., 2022; Zhao et al., 2022). IM-TQA further introduced form-like tables (Zheng et al., 2023), while recent LLM-era benchmarks such as TableBench, RealHiTBench, TableEval, and MULTITAT expand scale, layout complexity, and multilingual coverage (Wu et al., 2025b,a; Zhu et al., 2025; Zhang et al., 2025).

Despite this progress, these benchmarks differ from NEST in both representation and objective.

Table 1: Evaluation scope of representative table and form benchmarks.

Benchmark	Year	Input	Data State	Core Evaluation Capabilities			
				Content Reasoning	Structural Parsing	Cell-Header Alignment	Generative Infilling
<i>Visual Structure & Info Extraction (Section 2.1)</i>							
PubTabNet	2020	Image	Filled	–	✓	✓	–
FUNSD	2019	Image	Filled	–	–	✓	–
<i>Markup Downstream Reasoning (Section 2.2)</i>							
WikiTableQuestions	2015	HTML	Filled	✓	–	–	–
HiTab	2022	HTML	Filled	✓	–	Δ	–
IM-TQA	2023	JSON	Filled	✓	–	Δ	–
TableBench	2025	HTML/MD	Filled	✓	–	–	–
RealHiTBench	2025	Image/HTML	Filled	✓	Δ	Δ	–
<i>Markup Structural Probing (Section 2.3)</i>							
Table-Meets-LLM	2024	HTML/MD	Filled	–	○	○	–
Singha et al.	2023	HTML/MD	Filled	–	Δ	–	–
NEST (Ours)	2026	Native HTML	Empty	–	✓	✓	✓

Most remain tied to strict grid-like layouts, and even form-like resources such as IM-TQA serialize structure into abstract JSON, losing native interactional cues such as guided input placeholders and checkable options. More importantly, their tasks measure reasoning over filled values, whereas NEST evaluates whether models can recover the template structure before user content is present.

2.3 Fine-Grained Structural Probing on Markup

Recent work also probes LLMs’ structural sensitivity beyond standard QA. Table-Meets-LLM introduces SUC, a set of atomic structural tasks such as dimension detection and merged-cell recognition over multiple formats including HTML (Sui et al., 2024). Other studies use perturbations: Singha et al. (2023) analyze robustness under row/column transformations, while RealHiTBench tests whether structural variants, such as swapped sub-tables, change model predictions appropriately (Wu et al., 2025a).

These efforts are valuable but still limited in two ways. First, they evaluate structure either atomically (Sui et al., 2024) or indirectly through downstream/perturbation QA (Pasupat and Liang, 2015; Cheng et al., 2022; Zheng et al., 2023; Wu et al., 2025b,a), rather than requiring explicit reconstruction of a holistic logical topology. Second, because they operate on populated tables, their results primarily reflect filled-instance understanding; NEST complements them by evaluating template-level understanding. As shown in Table 1, NEST is the first benchmark to enforce explicit structural gener-

ation across SSP, CHA, and Generative Infilling on genuine unpopulated administrative templates.

3 Dataset Construction

3.1 Structural Complexity and Representation

Atomic Cell Types. To enable fine-grained structural evaluation, we define five atomic value cell types: Simple Input (SI), Guided Input (GI), Simple Option (SO), Guided Option (GO), and List Input (LI), as illustrated in Figure 3. Detailed definitions and examples are provided in Appendix A. We further identify Multi-Key Cells (MKC) as composite containers that pack heterogeneous atomic units, such as a guided input and a checkbox option, into a single physical boundary. During evaluation, MKCs are logically decomposed into their constituent atomic units so that each key–value target can be verified independently. Within an MKC, co-located fields that require distinct values are classified as GI; LI is reserved for genuinely repeatable fields across list rows or entries. This same-cell MKC rule takes precedence over repeated-key and numeric-suffix heuristics.

Structural Taxonomy. For the statistics reported in Table 2, we operationally assign each template to one of three structural-complexity levels according to its detected atomic cell types. Level 1 contains templates with no detected type other than SI; Level 2 contains at least one GI and no LI, SO, or GO; and Level 3 contains at least one LI, SO, or GO. We refer to these operational groups as flat tables, hierarchical tables, and composite

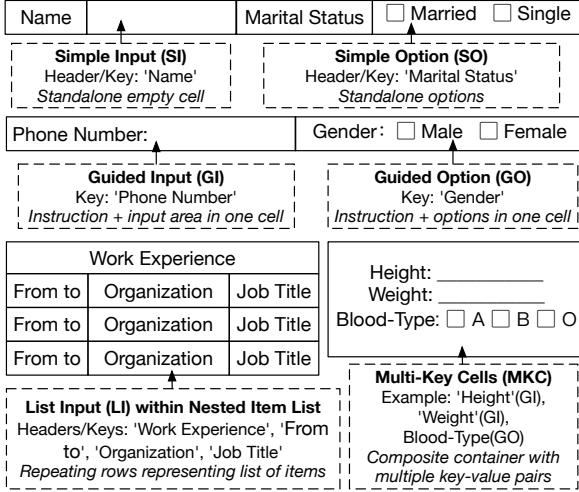


Figure 3: Taxonomy of structural units in NEST.

forms, respectively. Most NEST templates fall into Level 3, reflecting our focus on complex administrative forms. Formal definitions are provided in Appendix A.3.

Logical Representation. We represent template structure as JSON key–value schemas that capture semantic hierarchies independently of visual layout variations. This abstraction supports consistent evaluation, such as Tree Edit Distance, by treating templates with equivalent logical structures as identical even when their spatial presentations differ.

3.2 Dataset Statistics

As summarized in Table 2, NEST contains 828 curated templates, evenly split between Chinese and English and covering six real-world domains. Structurally, the benchmark is dominated by Level 3 composite forms, with List Input regions accounting for over half of its 27,305 atomic cells. This distribution reflects NEST’s focus on authentic empty administrative templates with nested, guided, and checkable structures beyond conventional relational grids.

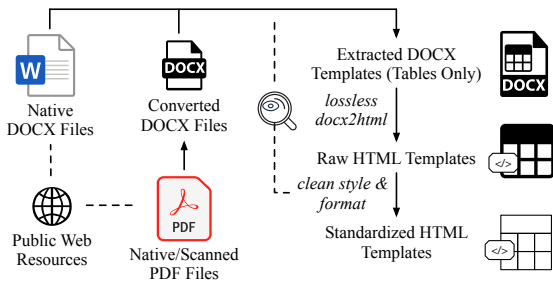


Figure 4: Data acquisition and normalization pipeline.

Table 2: NEST dataset statistics. Panel A reports domain coverage; Panel B reports structural counts and atomic cell composition.

Panel A: Domain Distribution					
Domain	Chinese		English	Total	%
Workplace/HR	205		65	270	32.6
Business	107		95	202	24.4
Finance	40		98	138	16.7
Gov./Public	20		95	115	13.9
Education	25		29	54	6.5
Healthcare	17		32	49	5.9
Total	414		414	828	100.0

Panel B: Structural Statistics					
Metric	Flat	Hierarchical	Composite	Total	%
# Templates	74	82	672	828	–
# Cells	840	1,133	25,332	27,305	–
Simple Input	840	637	5,498	6,975	25.5
Guided Input	0	496	2,200	2,696	9.9
Simple Option	0	0	2,124	2,124	7.8
Guided Option	0	0	1,122	1,122	4.1
List Input	0	0	14,388	14,388	52.7

Note: Percentages in Panel A are domain-level template shares. The % column in Panel B reports global cell-type shares; structural-level template shares for Flat, Hierarchical, and Composite are 8.9%, 9.9%, and 81.2%, respectively.

3.3 Data Acquisition and Normalization

We collect public, unfilled templates to preserve authentic administrative layouts without exposing private user data. Figure 4 summarizes our normalization pipeline: native DOCX files, PDFs, and images are unified into editable documents, target table regions are extracted into a DOCX pool, and the results are converted to topology-preserving clean HTML. All templates are manually checked against the originals, with details in Appendix B. Source provenance and release policies are summarized in Appendix B.2.

3.4 Semi-Automated Data Synthesis

We use a *schema-driven, LLM-augmented synthesis pipeline* based on a human-verified Root Schema (J_{root}), as shown in Figure 5. Following a “root-and-propagate” strategy, J_{root} serves as the SSP ground truth and seed for all synthetic artifacts. The pipeline produces two value-distinct streams: T_{filled}^A for deriving the infilling context C_{ctx}^A and alignment map M_{align}^A , and T_{filled}^B for non-leaking few-shot demonstrations. Appendix C details the workflow and five annotation artifacts.

Annotation Quality. Because the human-verified Root Schema (J_{root}) serves as the ground truth for SSP and the seed for downstream artifacts, we measure annotation reliability through iterative

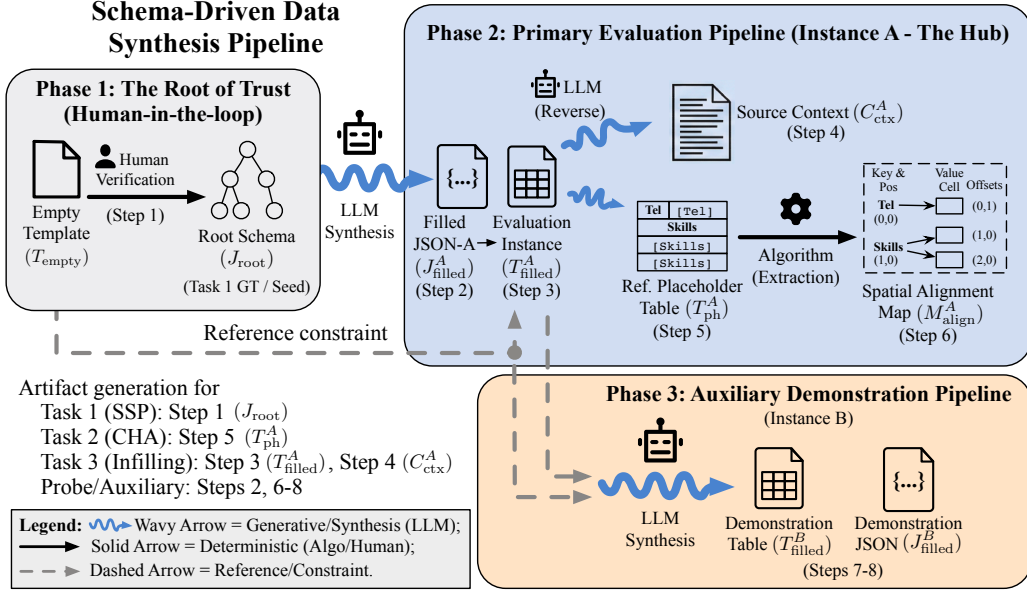


Figure 5: Schema-driven data synthesis pipeline. Starting from a human-verified root schema J_{root} , the pipeline follows a “root-and-propagate” strategy to produce benchmark artifacts for SSP, CHA, and Generative Infilling, as well as a value-distinct demonstration stream for non-leaking few-shot experiments.

inter-annotator agreement (IAA) analysis. We report the two validated post-pilot rounds and focus on node-level agreement, which directly reflects consistency over schema elements. In the final audit, node-level macro agreement reaches 83.20%, improving from 76.92% in the refinement round. Appendix B.4 details the agreement protocol, common annotation failure types, and the operational rules used to resolve structural ambiguities.

4 Evaluation Methodology

Our framework introduces three complementary tasks for evaluating template-level document understanding, ranging from schema recovery to value infilling. Unlike conventional benchmarks that evaluate populated instances, NEST evaluates models on empty templates before user values are introduced. Details are provided in Appendix D.

4.1 Task 1: Structural Schema Parsing (SSP)

Definition. This task assesses the model’s ability to convert raw HTML table skeletons into a hierarchical logical representation, independently of any user-filled values. Models must generate a single JSON object where keys represent headers (including instructional text) and values correspond to one of the five atomic types defined in our taxonomy (e.g., SI, GO). This structure must strictly preserve the header hierarchy (nesting) while re-

maining agnostic to the specific semantic content of user inputs. Detailed definitions of the permissible JSON value types are provided in Appendix D.1.

Eval: JEDIS Score. To quantify structural fidelity, we propose *JEDIS* (*JEDI Similarity*), a normalized metric derived from JEDI (JSON Edit Distance) (Hütter et al., 2022). Unlike flat text metrics (e.g., BLEU) which fail to capture topological errors, JEDIS operates directly on the JSON tree structure. Formally, the normalized per-table structural score Acc_{struct} is calculated as:

$$Acc_{struct} = JEDIS(T_{pred}, T_{ref}) = \lambda \cdot \left(1 - \frac{JEDI(T_{pred}, T_{ref})}{|T_{pred}| + |T_{ref}|} \right) \quad (1)$$

Here, $|T|$ denotes the tree size (node count). To handle imperfect JSON syntax common in LLM outputs, we introduce a *syntax-robustness factor* λ , which is set to 1.0 if the output is valid (or syntactically repairable) JSON, and 0.0 otherwise. Furthermore, we modify the standard JEDI operations by incorporating a *content-aware substitution cost*: substitution costs for string nodes utilize normalized Levenshtein distance, thereby penalizing character-level typos less severely than structural hallucinations.

4.2 Task 2: Cell-Header Alignment (CHA)

Definition. This task evaluates fine-grained spatial reasoning: can the model identify the spe-

cific governing header for every empty value cell? Given an HTML template, the model must explicitly link each empty cell to its header. For input fields, this is achieved by injecting an HTML `<input>` tag with a name attribute matching the header; for selection options (e.g., checkboxes), the model must attribute the existing tags to their corresponding header text.

Eval: Coordinate-Guided Anchoring (CGA).

To strictly validate alignment despite potential layout shifts in generated HTML, we employ a *Coordinate-Guided Anchoring (CGA)* protocol. We first anchor the expected semantic key using its ground-truth coordinates. Validity is then determined by spatial logic: (1) For separate cells, the injected placeholder must exactly match the ground truth; (2) For shared cells (e.g., GIs), the placeholder must appear strictly adjacent to the anchored key text in the correct relative direction. The final Accuracy ($\text{Acc}_{\text{align}}$) is the micro-average of valid alignments across all target cells.

Worked example. Suppose row 3, column 1 contains the label “Applicant Name” and row 3, column 2 is its blank field. CGA represents the target by the filling-cell coordinate (3, 2) together with the local key “Applicant Name.” The prediction is correct only when the corresponding input is placed at (3, 2) and linked to that key: placing it in another blank cell, or placing an “Address” field at (3, 2), is incorrect. For same-cell fields such as “Name: ____ Date: ____,” CGA additionally uses the local labels to distinguish the two fields rather than accepting either input anywhere in the row.

4.3 Task 3: Generative Infilling

Definition. This task represents the end-to-end application scenario, requiring both structural comprehension and semantic reasoning. Models are provided with an empty template, a specific instruction, and a context paragraph containing all necessary information. The objective is to populate all empty cells with precise values derived from the context, without modifying the underlying template structure.

Eval. Evaluation follows the same CGA protocol as Task 2, ensuring that values are filled in the correct spatial locations. The critical distinction is the verification target: rather than checking for a placeholder, we compare the predicted and gold semantic content at each annotation-defined physi-

cal filling cell using a deterministic source-faithful matcher. Performance is reported as Infilling Accuracy ($\text{Acc}_{\text{infill}}$).

Formalization. Let G and P denote the gold filled table and model output, and let $\mathcal{C}$ be the annotation-defined set of physical filling-cell coordinates with an observable text or form-control target. For table T and coordinate c , let $S(T, c)$ contain the normalized full-cell text when text is a target, together with the ordered values or states of the targeted HTML controls in that cell. Text normalization ignores case, whitespace, and fixed punctuation, but otherwise requires the complete cell content to match. Then

$$\text{Acc}_{\text{infill}} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \mathbf{1}[S(P, c) = S(G, c)]. \quad (2)$$

Matching is performed only at the annotated coordinate c , so content placed elsewhere receives no credit. Full-cell text is used because a filled value may share an HTML cell with labels or other static text; short fragments and partial values therefore receive no credit. When several fillable elements share one physical cell, that cell is scored once using its ordered full text and control values. This coordinate requirement reflects administrative workflows in which a value must appear in the correct field, not merely somewhere in the table.

Aggregation. CHA and Infilling are target-level rather than template-level metrics. For each model and condition, CHA micro-averages field–header alignment relations (10,460 English and 16,835 Chinese targets), whereas Infilling micro-averages annotation-defined physical filling cells with observable text or form-control content (11,881 English and 15,312 Chinese targets, each across 414 tables). Correct and total target counts are first pooled across English and Chinese for each model; Table 3 then averages the eight bilingual model-level scores. Thus, Overall CHA and Infilling are not arithmetic means of the displayed English and Chinese percentages. Their target totals differ because one physical cell can define multiple field–header alignment relations but is evaluated once as an ordered full-cell Infilling target.

Metric scope. NEST’s automated metrics prioritize structural correctness. Infilling explicitly requires source-faithful extraction and placement of context-provided values, so paraphrased content or a correct value in the wrong field is counted as

incorrect. SSP and CHA may still under-credit semantically valid alternative hierarchies, markup, or label wording; semantic-equivalence scoring and rendered-form integrity checks are complementary dimensions rather than substitutes for structural evaluation.

5 Experiments

5.1 Experimental Setup

We evaluated a diverse suite of eight state-of-the-art LLMs from proprietary (e.g., GPT-5, GPT-4o) and open-weight (e.g., Llama-3.1, Qwen2.5) families. We additionally evaluated GPT-5.4, GPT-5.4-mini, and GPT-5.4-nano with rendered table images as a separate modality-reference block. Performance is reported using the task-specific metrics defined in Section 4: $\text{Acc}_{\text{struct}}$, $\text{Acc}_{\text{align}}$, and $\text{Acc}_{\text{infill}}$. Detailed model specifications, citations, and hyperparameters are provided in Appendix E.1.

5.2 Exp I: The Template–Instance Gap

We quantify the gap between perceived performance on populated tables versus actual structural understanding of empty templates. Table 3 presents the comparative results across all tasks.

Performance Shift from Instances to Templates.

The results reveal a clear and substantial template–instance performance gap: evaluation on empty templates yields an average drop of 9.5 percentage points compared with their filled counterparts. While top-tier models remain relatively stable, several smaller open-weight models show much larger declines, led by Mistral-7B on Alignment and Infilling, with Llama-3.1-8B also dropping sharply on Infilling. This suggests that filled-instance evaluation alone may not fully characterize a model’s ability to understand reusable form templates.

Task Sensitivity. The template–instance gap is pronounced but not uniform across tasks. Task 3 (Infilling) shows the largest and most consistent gap, with empty-template performance trailing filled-instance performance by 21.1% on average. This indicates that value generation becomes substantially more difficult when models must first ground each value in an unpopulated template. By contrast, Task 2 (Alignment) shows a smaller average drop of 1.8%, but with heterogeneous model behavior: most models decline, while Qwen2.5-7B-Instruct improves by 13.3% on empty templates.

Together, these results indicate that populated instances and empty templates test related but distinct capabilities.

Fine-Grained Atomic Analysis. To further localize where models struggle, we analyze performance across the five atomic cell types defined in Section 3, as shown in Table 4. The results show that the template–instance gap is not a uniform degradation across all cell types. CHA is mixed: GI has the largest decline (−9.8 points), whereas GO and LI improve slightly. Infilling declines for every type and is especially difficult for GI (−42.4) and GO (−33.2), whose local instructions or option semantics must be grounded before values can be placed. Thus, the gap is largest for guided structures rather than being explained only by field length or repetition.

Domain and Multi-Key Cell Effects. The six-domain breakdown in Appendix Table 10 shows SSP declines in every domain, while Infilling varies more strongly: Gov./Public and Finance have the largest gaps (−28.6% and −22.5%), consistent with their dense administrative sections, repeated fields, checkbox groups, and conditional entries. CHA is more mixed because removing values can either remove useful field-boundary cues or reduce clutter in dense layouts. Multi-Key Cell (MKC) analysis in Appendix Table 13 yields a small CHA shift (49.2% to 48.2%) but a large Infilling drop (85.0% to 59.0%). Typical MKC failures omit internal fields or confuse keys within the same cell for CHA, and produce partial overlaps, empty cells, or values in neighboring cells for Infilling.

Rendered-Image Reference Study. As an auxiliary study, we evaluate three GPT-5.4 variants on PNGs rendered from the normalized HTML; these rows are reported separately and excluded from the main averages. Image-input accuracy is lower than the strongest HTML-input results, especially on Infilling: GPT-5.4 and GPT-5.4-mini decline on templates, whereas GPT-5.4-nano remains near parity; SSP and CHA shifts are mixed and model-dependent. Because these renderings exclude native source styling, scans, handwriting, and OCR noise, they serve only as a controlled reference.

5.3 Exp II: Template-to-Instance Tuning

A key advantage of empty templates is that they support model training without exposing propri-

Table 3: Comparative performance on filled tables vs. empty templates. Reported values correspond to the task-specific metrics defined in Section 4: $\text{Acc}_{\text{struct}}$ (Task 1), $\text{Acc}_{\text{align}}$ (Task 2), and $\text{Acc}_{\text{infill}}$ (Task 3). Δ denotes the percentage-point difference between empty-template and filled-instance performance. Image-input baselines are included as reference rows and excluded from averages.

Model	Task 1: Schema Parsing			Task 2: CHA			Task 3: Infilling		
	Filled	Templ.	Δ	Filled	Templ.	Δ	Filled	Templ.	Δ
Proprietary Models									
GPT-5	60.3%	62.1%	+1.9%	95.6%	93.9%	-1.7%	98.8%	92.6%	-6.1%
GPT-4o	49.4%	50.1%	+0.6%	73.8%	72.9%	-0.9%	95.6%	83.7%	-11.8%
GPT-4o-mini	52.4%	47.3%	-5.1%	75.2%	72.6%	-2.6%	99.0%	82.2%	-16.9%
GPT-3.5-turbo	52.3%	47.5%	-4.8%	75.6%	74.5%	-1.1%	99.0%	82.1%	-16.9%
Open-Weight Models									
Llama-3.1-8B-Instruct	48.7%	35.3%	-13.4%	39.2%	36.4%	-2.8%	94.0%	61.3%	-32.7%
Qwen2.5-7B-Instruct	50.6%	39.4%	-11.3%	32.7%	46.0%	+13.3%	85.1%	58.5%	-26.7%
Gemma-2-9B-it	54.1%	51.5%	-2.6%	46.0%	38.8%	-7.3%	94.3%	70.6%	-23.7%
Mistral-7B-Instruct-v0.3	39.0%	28.9%	-10.2%	17.9%	6.7%	-11.2%	76.7%	42.3%	-34.4%
Task Average	50.9%	45.2%	-5.6%	57.0%	55.2%	-1.8%	92.8%	71.7%	-21.1%
Global Average Δ (Templ. – Filled): -9.5%									
<i>Image-input baselines (reference)</i>									
GPT-5.4 (Image)	53.7%	54.2%	+0.4%	77.6%	77.0%	-0.5%	51.0%	44.7%	-6.3%
GPT-5.4-mini (Image)	55.2%	54.7%	-0.5%	53.4%	64.9%	+11.5%	42.6%	37.3%	-5.2%
GPT-5.4-nano (Image)	42.1%	35.8%	-6.3%	43.4%	45.2%	+1.7%	15.6%	15.7%	+0.1%

Table 4: Average accuracy by atomic cell type across the eight HTML-input models in Table 3. Δ is the percentage-point difference (Template minus Filled). Task 1 is omitted because SSP is evaluated at the whole-template schema level rather than per cell. For Task 3, each physical filling cell is scored once and associated with each distinct atomic type present in that cell.

Cell Type	Filled	Templ.	Δ
Task 2: CHA			
Simple Input (SI)	63.9%	58.2%	-5.7%
Guided Input (GI)	59.8%	50.1%	-9.8%
Simple Option (SO)	60.3%	57.0%	-3.3%
Guided Option (GO)	52.9%	53.3%	+0.4%
List Input (LI)	52.9%	54.6%	+1.7%
Task 3: Infilling			
Simple Input (SI)	91.5%	68.9%	-22.6%
Guided Input (GI)	92.2%	49.8%	-42.4%
Simple Option (SO)	93.1%	76.2%	-16.9%
Guided Option (GO)	86.8%	53.6%	-33.2%
List Input (LI)	94.4%	77.0%	-17.4%

etary user-filled data. The template-to-instance supervision used here is synthesized from the empty templates through our data synthesis pipeline, rather than collected from original user-filled records. We therefore study whether template-to-instance supervision from Generative Infilling can improve not only infilling itself, but also related template-level tasks.

Setup. We split NEST at the template level, using 662 templates for training and the remaining 166 templates for testing, with a balanced Chinese–

English distribution in both splits. For Infill-FT, open-weight student models are fine-tuned on the Task 3 (Generative Infilling) instances from the training split. This gives each training template a synthetic template-to-instance supervision signal, without requiring proprietary user-filled records or separate supervision for SSP and CHA. To examine whether the learned signal transfers beyond its training objective, we evaluate the fine-tuned models on all three tasks using a held-out test set.

Results. Table 5 shows that Infill-FT raises Generative Infilling from 57.99% to 74.36% (+16.37 percentage points), driving the overall three-task average from 42.72% to 50.91% (+8.19 percentage points). Transfer is task- and model-dependent. Across tasks, Infilling is directly optimized; CHA additionally requires local-header anchoring, while SSP requires generation of the full schema hierarchy, consistent with the smaller average changes of +5.78 and +2.41 points, respectively. Across models, Llama, Qwen, and Mistral improve overall, while Gemma declines by 3.25 points and also falls on the trained Infilling objective. Mistral’s CHA rises from 5.92% to 20.33%, whereas Gemma falls from 40.10% to 32.19%. This pattern is consistent with model-dependent task-specific over-adaptation, but does not establish the underlying mechanism.

Table 5: Base vs. fine-tuned models across three NEST tasks. Models are fine-tuned only on Task 3 using the NEST training split of 662 templates (80%, balanced Chinese/English) and evaluated on a 20% test split. Δ reports the Infill-FT–Base percentage-point difference in average score.

Model	Method	Task 1 (SSP)	Task 2 (CHA)	Task 3 (Infilling)	Avg.	Δ Avg.
Llama-3.1-8B-Instruct	Base	35.32%	34.10%	60.33%	43.25%	–
	Infill-FT	38.14%	44.22%	66.76%	49.71%	+6.46%
Qwen2.5-7B-Instruct	Base	38.15%	50.14%	55.76%	48.02%	–
	Infill-FT	44.11%	56.66%	79.47%	60.08%	+12.06%
Gemma-2-9B-it	Base	51.37%	40.10%	72.79%	54.75%	–
	Infill-FT	51.33%	32.19%	70.98%	51.50%	-3.25%
Mistral-7B-Instruct-v0.3	Base	25.57%	5.92%	43.10%	24.86%	–
	Infill-FT	26.46%	20.33%	80.22%	42.34%	+17.47%
Average	Base	37.60%	32.57%	57.99%	42.72%	–
	Infill-FT	40.01%	38.35%	74.36%	50.91%	+8.19%

Table 6: Human alignment analysis. Each cell reports Pearson’s r between per-table automated and human scores; human scores average Completeness and Correctness for Template-input Task 3 outputs.

Model	T1: SSP	T2: CHA	T3: Infilling
<i>Proprietary</i>			
GPT-5	0.20	0.38	0.66
GPT-4o	0.22	0.35	0.62
GPT-4o-mini	0.28	0.38	0.59
GPT-3.5-turbo	0.22	0.37	0.64
<i>Open-Weight</i>			
Llama-3.1-8B	0.20	0.35	0.57
Qwen2.5-7B	0.24	0.40	0.65
Gemma-2-9B	0.18	0.37	0.64
Mistral-7B-v0.3	0.22	0.40	0.55
Average	0.22	0.38	0.62

5.4 Human Alignment Analysis

To assess alignment with human judgment, annotators evaluate the visually inspectable Template-input Task 3 outputs. For model m and table j , the human score is $h_{m,j} = (C_{m,j} + R_{m,j})/2$, averaging 1–10 *Completeness* (required cells populated) and *Correctness* (accurate values and intact rendering) scores. For $\tau \in \{\text{SSP}, \text{CHA}, \text{Infilling}\}$, $s_{m,\tau,j}$ denotes, respectively, the task-specific accuracy $\text{Acc}_{\text{struct}}$, $\text{Acc}_{\text{align}}$, or $\text{Acc}_{\text{infill}}$ for model m on table j , as defined in Section 4. Table 6 reports $r_{m,\tau} = \text{Pearson}(\{h_{m,j}\}, \{s_{m,\tau,j}\})$ for each model–task pair.

Results. Task 3 correlates most strongly with human judgment (avg. $r \approx 0.62$), reflecting its focus on visible completion quality; imperfect agreement partly arises because annotators also penalize rendering defects that automated value-placement scores do not capture. SSP and CHA show lower correlations (avg. $r \approx 0.22$ and 0.38) because they

provide complementary diagnostics of schema hierarchy and cell–header relations that may be less apparent from surface inspection but still affect downstream processing.

6 Conclusion

This work introduces NEST, a benchmark for evaluating template-level understanding on native, unpopulated administrative forms with nested regions, guided inputs, and checkable options. By comparing empty templates with their populated counterparts, NEST reveals a clear template–instance performance gap: empty-template evaluation is 9.5 percentage points lower on average across tasks. We further show that Generative Infilling provides an effective template-to-instance tuning signal for template completion, with strong gains on infilling and partial, model-dependent transfer to schema parsing and cell-header alignment. Overall, NEST complements existing filled-table benchmarks and supports privacy-preserving document automation workflows that must reason about forms before sensitive values are introduced.

Limitations

Scope and representation. NEST currently covers public, born-digital administrative templates represented as normalized HTML; its findings should not be generalized to all institutions, languages, domains, or document formats. It does not claim coverage of native scans, handwriting, OCR noise, dynamic web forms, proprietary templates, or low-resource-language forms. The rendered-image study uses PNGs produced from the same normalized HTML and is a controlled modality reference rather than a native-document benchmark. Although our root schemas are human-verified, the

annotation of complex templates can still involve judgment calls, especially for dense multi-key cells and ambiguous section boundaries. Our synthesis pipeline relies on LLM-generated contexts and filled instances, which may be cleaner and more self-contained than noisy real user-information contexts; this can limit transfer to deployment inputs.

Deployment, privacy, and use boundaries. Inaccurate schema recovery or value placement can propagate errors to downstream document automation systems. NEST is therefore a diagnostic benchmark, not a deployment-safety guarantee. High-stakes use requires schema and field validation, escalation of ambiguous mappings, privacy controls, audit logs, and human approval before form submission or downstream action. Although NEST does not use real user-filled records, deployed systems may process sensitive information and require consent, access controls, retention limits, and applicable data-governance safeguards. Benchmark scores should not authorize fully autonomous processing or imply fitness for institutions, languages, or document formats outside NEST’s evaluated scope. Finally, our metrics emphasize template-level schema recovery, cell-header grounding, and exact value placement; they may under-credit semantically equivalent outputs with different valid markup or wording. Potential future directions include more robust, privacy-preserving template-to-instance tuning and broader evaluation across input conditions and structural tasks.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant Nos. 62466018 and 62576353), the Research Foundation of the Education Bureau of Hunan Province (No. 23A0387), and the Science and Technology Innovation Program of Hunan Province (No. 2025RC3125).

References

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyu Zhou, and William Yang Wang. 2020. [Tabfact: A large-scale dataset for table-based fact verification](#). In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and

Dongmei Zhang. 2022. [Hitab: A hierarchical table dataset for question answering and natural language generation](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1094–1110, Dublin, Ireland. Association for Computational Linguistics.

Liangcai Gao, Yilun Huang, Hervé Déjean, Jean-Luc Meunier, Qinqin Yan, Yu Fang, Florian Kleber, and Eva Lang. 2019. [Icdar 2019 competition on table detection and recognition \(ctdar\)](#). In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1510–1515.

Gemma Team. 2024. [Gemma 2: Improving open language models at a practical size](#). *Preprint*, arXiv:2408.00118.

Max Göbel, Tamir Hassan, Ermelinda Oro, and Giorgio Orsi. 2013. [Icdar 2013 table competition](#). In *2013 12th International Conference on Document Analysis and Recognition*, pages 1449–1453.

Thomas Hütter, Nikolaus Augsten, Christoph M. Kirsch, Michael J. Carey, and Chen Li. 2022. [Jedi: These aren’t the json documents you’re looking for...](#) In *Proceedings of the 2022 International Conference on Management of Data*, pages 1584–1597, Philadelphia PA USA. ACM.

Guillaume Jaume, Hazim Kemal Ekenel, and Jean-Philippe Thiran. 2019. [Funsd: A dataset for form understanding in noisy scanned documents](#). In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 2, pages 1–6.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Léo Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. [Mistral 7b](#). *Preprint*, arXiv:2310.06825.

Haoran Li, Yulin Chen, Jinglong Luo, Jiecong Wang, Hao Peng, Yan Kang, Xiaojin Zhang, Qi Hu, Chunkit Chan, Zenglin Xu, Bryan Hooi, and Yangqiu Song. 2023. [Privacy in large language models: Attacks, defenses and future directions](#). *Preprint*, arXiv:2310.10383.

Llama Team, AI @ Meta. 2024. [The Llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.

Arun S. Maiya. 2025. [Onprem.llm: A privacy-conscious document intelligence toolkit](#). *Preprint*, arXiv:2505.07672.

OpenAI. 2024a. [GPT-4o mini: Advancing cost-efficient intelligence](#). Accessed: 2026-05-26.

OpenAI. 2024b. [GPT-4o System Card](#). Accessed: 2026-05-26.

- OpenAI. 2025. [GPT-5 System Card](#). Accessed: 2026-05-26.
- OpenAI. 2026a. [GPT-3.5 Turbo](#). Accessed: 2026-08-24.
- OpenAI. 2026b. [Introducing GPT-5.4](#). Accessed: 2026-08-24.
- OpenAI. 2026c. [Introducing GPT-5.4 mini and nano](#). Accessed: 2026-08-24.
- Panupong Pasupat and Percy Liang. 2015. [Compositional semantic parsing on semi-structured tables](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.
- Qwen. 2024. [Qwen2.5 technical report](#). *Preprint*, arXiv:2412.15115.
- Ananya Singha, José Cambronero, Sumit Gulwani, Vu Le, and Chris Parnin. 2023. [Tabular representation, noisy operators, and impacts on table structure understanding tasks in llms](#). *Preprint*, arXiv:2310.10358.
- Brandon Smock, Rohith Pesala, and Robin Abraham. 2022. [Pubtables-1m: Towards comprehensive table extraction from unstructured documents](#). In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4624–4632, New Orleans, LA, USA. IEEE.
- Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2024. [Table meets llm: Can large language models understand structured table data? a benchmark and empirical study](#). In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 645–654, Merida Mexico. ACM.
- Matthew Toles, Isaac Song, Rattandeep Singh, and Zhou Yu. 2026. [FormGym: Doing paperwork with agents](#). In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3771–3785, Rabat, Morocco. Association for Computational Linguistics.
- Pengzuo Wu, Yuhang Yang, Guangcheng Zhu, Chao Ye, Hong Gu, Xu Lu, Ruixuan Xiao, Bowen Bao, Yijing He, Liangyu Zha, Wentao Ye, Junbo Zhao, and Haobo Wang. 2025a. [Realhitbench: A comprehensive realistic hierarchical table benchmark for evaluating llm-based table analysis](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 7105–7137, Vienna, Austria. Association for Computational Linguistics.
- Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xinrun Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, Guanglin Niu, Tongliang Li, and Zhoujun Li. 2025b. [Tablebench: A comprehensive and complex benchmark for table question answering](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25497–25506.
- Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang, Yuhui Xu, Zekun Wang, Yiheng Xu, Junli Wang, Doyen Sahoo, Tao Yu, and Caiming Xiong. 2025. [Scaling computer-use grounding via user interface decomposition and synthesis](#). *Preprint*, arXiv:2505.13227.
- Yiheng Xu, Tengchao Lv, Lei Cui, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, and Furu Wei. 2022. [Xfund: A benchmark dataset for multilingual visually rich form understanding](#). In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3214–3224, Dublin, Ireland. Association for Computational Linguistics.
- Zhibo Yang, Rujiao Long, Pengfei Wang, Sibao Song, Humen Zhong, Wenqing Cheng, Xiang Bai, and Cong Yao. 2023. [Modeling entities as semantic points for visual information extraction in the wild](#). In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15358–15367.
- Xuanliang Zhang, Dingzirui Wang, Keyan Xu, Qingfu Zhu, and Wanxiang Che. 2025. [Multitat: Benchmarking multilingual table-and-text question answering](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 626–647, Suzhou, China. Association for Computational Linguistics.
- Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. 2022. [MultihierTT: Numerical reasoning over multi hierarchical tabular and textual data](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6588–6600, Dublin, Ireland. Association for Computational Linguistics.
- Mingyu Zheng, Yang Hao, Wenbin Jiang, Zheng Lin, Yajuan Lyu, QiaoQiao She, and Weiping Wang. 2023. [Im-tqa: A chinese table question answering dataset with implicit and multi-type table structures](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5074–5094, Toronto, Canada. Association for Computational Linguistics.
- Xinyi Zheng, Douglas Burdick, Lucian Popa, Xu Zhong, and Nancy Xin Ru Wang. 2021. [Global table extractor \(gte\): A framework for joint table identification and cell structure recognition using visual context](#). In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 697–706, Waikoloa, HI, USA. IEEE.
- Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. 2020. [Image-based table recognition: Data, model, and evaluation](#). In Andrea

Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, volume 12366, pages 564–580. Springer International Publishing, Cham.

Junnan Zhu, Jingyi Wang, Bohan Yu, Xiaoyu Wu, Junbo Li, Lei Wang, and Nan Xu. 2025. [Tableeval: A real-world benchmark for complex, multilingual, and multi-structured table question answering](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 7137–7157, Suzhou, China. Association for Computational Linguistics.

A Detailed Structural Definitions

A.1 Atomic Cell Types

We define five atomic value cell types that constitute the fundamental building blocks of complex templates. This taxonomy strictly classifies the target regions requiring user input, distinct from header or static text cells.

- *Simple Input (SI)*: Basic empty fields designated for freeform data entry. These cells physically separate the key (header) from the value (input area) and typically appear in standard grid layouts.
- *Guided Input (GI)*: “Hybrid” cells containing instructional text paired with blank input areas within the same cell boundary (e.g., “*Full Name: _____*”).
- *Simple Option (SO)*: Cells presenting standalone selection choices, such as checkboxes or radio buttons, without embedded instructional text.
- *Guided Option (GO)*: Hybrid cells combining instructional text with selection options within a single boundary (e.g., “*Payment Method: [] Credit Card [] Bank Transfer*”).
- *List Input (LI)*: Cells within structured lists where rows typically denote individual entities and columns represent their attributes. Though visually similar to SI cells, they function as repeatable elements, evaluating the model’s ability to differentiate implicit list boundaries from standalone fields.

A.2 Multi-Key Cells (MKC)

Real-world forms frequently exhibit high-density layouts that aggregate multiple independent key-value pairs within a single physical cell boundary.

We define MKC not as a distinct atomic type, but as a composite container housing a heterogeneous mix of atomic primitives. For example, a cell containing “*Gender: [] M [] F Age: _____*” is decomposed into one GO unit and one GI unit. In our evaluation, we validate each constituent unit independently.

A.3 Structural Taxonomy Definitions

The three-tier taxonomy used in Table 2 is operationally assigned from the detected atomic cell types in each template:

- *Level 1 (Flat Tables)*: Templates with no detected value-cell type other than SI.
- *Level 2 (Hierarchical Tables)*: Templates containing at least one GI and no LI, SO, or GO; SI cells may also occur.
- *Level 3 (Composite Forms)*: Templates containing at least one LI, SO, or GO; SI and GI cells may also occur.

B Data Collection Details

This appendix details the acquisition and normalization pipeline described in Section 3.3.

B.1 Source Distribution

We observed distinct distribution norms across regions: the Chinese corpus consists primarily of native DOCX files (standard in the local administrative ecosystem), whereas the English corpus was largely sourced from PDF documents common in Western institutional archives.

B.2 Source Provenance and Release Policy

The artifact provides a provenance manifest for all 828 research records, with exact source URLs where recoverable and otherwise the known source websites. Original third-party PDFs, source images, and Office documents are not redistributed. Detailed artifact scope and licensing terms are provided in the project repository.

B.3 Normalization Pipeline

All source documents were processed through the following stages:

1. *Format Unification*: PDF documents and images were first converted to editable DOCX format using WPS Office¹ to recover underlying text and table structures.

¹<https://www.wps.cn/>

2. *HTML Conversion*: Native and converted DOCX files were processed through the *word-tohtml.net* service. This step performs comprehensive HTML cleanup, removing proprietary styling tags while maintaining structural integrity.
3. *Human Verification*: The resulting HTML representations underwent rigorous manual verification. Annotators visually compared the rendered HTML output against the original source documents to ensure the complete preservation of structural semantics, including hierarchical headers, merged cells, and nested sections.

B.4 Annotation Quality Protocol

We compute node-level IAA as macro-averaged node-label F1 over extracted schema nodes; the corresponding micro F1 is also tracked for diagnostics. For the final audit, each annotator re-annotated sampled gold tables that they had not originally annotated, and we compare each re-annotation against the human-verified gold schema. In Table 7, *Tbl. Comp.* denotes the number of valid table-level annotation–reference JSON comparisons. The main disagreement types are alternative nesting choices for multi-key cells, ambiguous section-header boundaries, and inconsistent decomposition of guided input or option primitives.

Resolution protocol. We apply four operational rules when verifying the Root Schema.

1. *Target preservation and local anchoring*. Annotators first retain every independently fillable value or option group as a separate target, then attach it to the nearest explicit governing local header. A broader section title supplies nesting context but does not replace a nearer field label.
2. *Guided-input and option decomposition*. A text-entry target is SI when its label and blank occupy separate cells, and GI when the cue and blank share a cell. A selection target is SO when its choices are separate from the governing prompt, and GO when the prompt and choices share a cell. Choices answering one prompt form one option group; choices governed by different prompts remain separate targets.

Table 7: Node-level inter-annotator agreement in the two validated annotation refinement rounds. In the final audit, three annotators re-annotated sampled gold tables that they had not originally annotated; IAA compares each re-annotation against the human-verified gold schema.

Round	Breakdown	Tbl. Comp.	Node IAA
Refinement round	Overall	6	76.92%
Final audit	Auditor A	30	77.27%
	Auditor B	30	78.12%
	Auditor C	30	94.21%
	Overall	90	83.20%

3. *Dense MKC nesting*. A same-cell composite is split whenever it encodes distinct values or option groups. For example, under “Applicant Details,” the cell “Height: ____; Weight: ____; Blood Type: [] A [] B” yields three sibling targets, GI(Height), GI(Weight), and GO(Blood Type), nested under that section.
4. *Section-header boundaries*. A section header governs the smallest contiguous region that follows it or is enclosed by its merged span and contains its dependent targets; its scope ends at the next header of equal or higher visual rank or at an explicit full-width or merged separator. If wording and geometry conflict, field-level anchoring follows the nearest explicit local label, while the broader header is retained only as nesting context.

These priorities resolve conflicts in the order *independently fillable target* > *nearest local header* > *enclosing section*. Any residual disagreement is adjudicated against the original rendering before the agreed structure is frozen as J_{root} .

C Data Synthesis Details

This appendix provides a detailed breakdown of the “root-and-propagate” synthesis pipeline described in Section 3.4 and illustrated in Figure 5.

C.1 Pipeline Execution Phases

The synthesis process proceeds in three sequential phases:

Phase 1: The Root of Trust (Schema Verification). The process begins with the raw empty HTML templates T_{empty} . We convert these into an initial Hierarchical JSON. Critically, this step involves strict human intervention: expert annotators

rigorously verify and correct this JSON to ensure that it perfectly reflects the logical structure. The resulting *Root Schema* (J_{root}) serves as the ground truth for Task 1.

Phase 2: Primary Evaluation Pipeline (Instance A). Using J_{root} as the seed, the pipeline generates the core benchmarking dataset:

1. *Instance Synthesis*: The LLM hallucinates realistic values to produce a filled JSON (J_{filled}^A) and renders it into the *Evaluation Instance* (T_{filled}^A).
2. *Reverse Context Generation*: For Task 3, we employ a “reverse generation” strategy where the LLM writes a natural language source context (C_{ctx}^A) based strictly on the values in T_{filled}^A .
3. *Alignment Map Extraction*: A deterministic algorithm parses the table to extract the *Spatial Alignment Map* (M_{align}^A), recording the precise grid coordinates for every value cell.

Phase 3: Auxiliary Demonstration Pipeline (Instance B). To support few-shot prompting without data leakage, the LLM generates a parallel *Demonstration Instance* (T_{filled}^B). This instance shares the exact structure of Instance A but contains divergent data values.

C.2 Annotation Artifacts Definitions

Standardizing the outputs, our dataset provides five distinct categories of artifacts for each template:

- *Logical Schema* (J_{root}): The human-verified hierarchical tree capturing the structural logic (Task 1 Ground Truth).
- *Generative Infilling Suite* (T_{filled}^A & C_{ctx}^A): The evaluation pair for Task 3. C_{ctx}^A serves as the natural language input, while T_{filled}^A is the semantic target.
- *Spatial Alignment Map* (M_{align}^A): The core metadata structure for Task 2, providing absolute (r, c) coordinates and relative positional offsets for verification.
- *Reference Placeholder Table* (T_{ph}^A): A visual reference table containing semantic placeholders (e.g., [Name]) used as an intermediate logical ground truth.

- *Demonstration Set* (T_{filled}^B & J_{filled}^B): The auxiliary instance pair used strictly for in-context learning demonstrations (1-shot), ensuring no ground truth values are leaked to the model.

D Evaluation Task Details

This appendix supplements the methodology described in Section 4 with detailed definitions of JSON types, metric formulations, and implementation constraints.

D.1 Task 1: JSON Value Type Definitions

For the Structural Schema Parsing task, models must map template cells to specific JSON value types. Table 8 defines the five permissible atomic types and their corresponding JSON representations.

D.2 Task 2: Alignment Implementation Protocols

For the Cell-Header Alignment task, models must modify the HTML specifically as follows:

- *Input Fields* (SI, GI, LI): Insert an `<input>` tag. The name attribute must match the governing header (e.g., `<input name="Header-Text">`).
- *Selection Options* (SO, GO): Locate existing option tags (e.g., `<input type="checkbox">`) and inject the name attribute corresponding to the header into *every* tag in the group.

D.3 Coordinate-Guided Anchoring (CGA)

The CGA protocol verifies spatial validity using two rules:

- *Separate-Cell Values*: When the key and value are in distinct cells, the cell content must exactly match the expected placeholder.
- *Same-Cell Values*: When they share a cell (e.g., GI, MKC), verification enforces a directional context check. The placeholder is valid only if it appears immediately adjacent to the anchored key text in the correct relative direction (e.g., to the right of "Name:").

E Experiment I Details: Models and Analysis

This appendix supplements the “Template–Instance Gap” experiment (Section 5.2) with detailed model specifications and fine-grained structural analysis.

Table 8: JSON Value Type Definitions for Task 1.

Type	JSON Schema	Description & Example
Simple Input (SI)	""	Standalone blank cells. Key is the header text.
Guided Input (GI)	""	Instructional text paired with underscores. Key is the instruction.
Simple Option (SO)	{"choices": [...], "value": ""}	Standalone checkboxes/radios.
Guided Option (GO)	{"choices": [...], "value": ""}	Hybrid instructional text with options.
List Input (LI)	{"key": ["", ""], ...}	Arrays representing repeatable list rows.

E.1 Model Specifications

We evaluated the following models:

- **Proprietary:** GPT-5 (OpenAI, 2025), GPT-4o-2024-11-20 (OpenAI, 2024b), GPT-4o-mini (OpenAI, 2024a), GPT-3.5-turbo-0125 (OpenAI, 2026a).
- **Open-Weight:** Llama-3.1-8B-Instruct (Llama Team, AI @ Meta, 2024), Qwen2.5-7B-Instruct (Qwen, 2024), Gemma-2-9B-it (Gemma Team, 2024), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023).
- **Rendered-Image References:** GPT-5.4 (OpenAI, 2026b), GPT-5.4-mini, and GPT-5.4-nano (OpenAI, 2026c) receive PNG renderings of the normalized HTML used by the benchmark. Their rows are not included in the eight-model averages.

E.2 Source-Filtered Robustness Analysis

To test whether the main finding depends on templates from commercial source sites, we reaggregated the existing predictions over 250 templates whose recorded sources are government or selected public institutions. Table 9 uses the same task-specific metrics as Table 3; image-input reference models are excluded from this HTML-input robustness analysis. The average Template-minus-Filled gaps remain negative for all three tasks (SSP: -2.9; CHA: -7.3; Infilling: -30.4 percentage points), showing that the central template-instance gap persists under this conservative source filter.

E.3 Domain Analysis

Table 10 reports the Filled-to-Template comparison across all six NEST domains. SSP declines consistently across domains, suggesting that the global schema-recovery gap is not driven by a single source category. Infilling is more domain-sensitive: Gov./Public and Finance show the largest drops,

whereas Business has the smallest. CHA changes direction across domains, with small gains for Business and Education but declines elsewhere.

E.4 Language and Qualitative Analysis

Table 11 separates the Filled-to-Template comparison by language. Both languages show lower SSP and Infilling accuracy on templates, with the larger decline in Infilling. CHA decreases by 5.0 points in English but is nearly unchanged in Chinese (+0.2 points). This contrast is sensitive to corpus composition: the Chinese split is concentrated in Work/HR and Business, where domain-level CHA is stable or positive, whereas the English split contains more Finance and Gov./Public forms, where CHA declines. It should therefore not be interpreted as an intrinsic language advantage.

Qualitatively, removing values can occasionally reduce interference in dense List Input layouts. For Qwen’s CHA predictions, an English bill-of-sale table (#55) changes from 0/44 to 44/44 correct targets and a Chinese inventory table (#115) from 0/70 to 70/70. These examples illustrate one source of the small LI gain in Table 4, while the aggregate results show that the effect is not uniform across structures.

Table 11: Language-specific accuracy averaged over eight HTML-input models. Δ denotes Template minus Filled.

Task	English			Chinese		
	Filled	Templ.	Δ	Filled	Templ.	Δ
SSP	34.4%	30.6%	-3.8%	67.3%	59.9%	-7.4%
CHA	58.3%	53.2%	-5.0%	56.2%	56.4%	+0.2%
Infilling	90.6%	65.9%	-24.6%	94.6%	76.1%	-18.4%

E.5 Task 1 Schema Error Analysis

Task 1 cannot be placed in the cell-type accuracy table because SSP evaluates a whole-template JSON schema. We therefore classify its erroneous predictions separately in Table 12. The categories are non-exclusive: one prediction may omit keys, add spurious keys, and misnest others. The dominant er-

Table 9: Source-filtered robustness analysis over 250 government and public-institutional templates. Reported values correspond to the task-specific metrics defined in Section 4. Δ denotes the percentage-point difference between empty-template and filled-instance performance. Image-input reference models are excluded.

Model	Task 1: Schema Parsing			Task 2: CHA			Task 3: Infilling		
	Filled	Templ.	Δ	Filled	Templ.	Δ	Filled	Templ.	Δ
Proprietary Models									
GPT-5	30.8%	31.3%	+0.4%	92.2%	91.4%	-0.8%	98.9%	92.8%	-6.1%
GPT-4o	23.1%	22.4%	-0.7%	63.0%	61.5%	-1.5%	93.7%	70.1%	-23.6%
GPT-4o-mini	23.9%	20.7%	-3.3%	67.0%	65.0%	-2.0%	98.2%	71.7%	-26.5%
GPT-3.5-turbo	24.0%	21.0%	-2.9%	68.0%	65.1%	-3.0%	98.4%	71.1%	-27.3%
Open-Weight Models									
Llama-3.1-8B-Instruct	23.2%	20.0%	-3.2%	46.9%	36.3%	-10.7%	92.8%	47.7%	-45.1%
Qwen2.5-7B-Instruct	23.1%	15.9%	-7.2%	33.1%	39.7%	+6.6%	82.8%	49.2%	-33.6%
Gemma-2-9B-it	25.7%	25.6%	-0.1%	44.3%	31.1%	-13.2%	88.8%	54.2%	-34.6%
Mistral-7B-Instruct-v0.3	21.6%	15.4%	-6.3%	40.9%	7.0%	-33.9%	80.6%	34.5%	-46.1%
Task Average	24.5%	21.5%	-2.9%	56.9%	49.6%	-7.3%	91.8%	61.4%	-30.4%
Global Average Δ (Templ. – Filled): -13.5%									

Table 10: Six-domain performance across the eight HTML-input models in Table 3. Domain counts are numbers of templates; Δ is the percentage-point difference (Template minus Filled).

Domain	n	Task 1: SSP			Task 2: CHA			Task 3: Infilling		
		Filled	Templ.	Δ	Filled	Templ.	Δ	Filled	Templ.	Δ
Work/HR	270	58.0%	51.4%	-6.5%	56.9%	56.0%	-0.9%	91.4%	69.2%	-22.2%
Business	202	59.4%	53.4%	-5.9%	55.7%	56.3%	+0.6%	95.2%	78.2%	-17.0%
Finance	138	47.7%	41.6%	-6.2%	57.4%	52.6%	-4.8%	91.5%	69.0%	-22.5%
Gov./Public	115	26.7%	22.6%	-4.1%	56.3%	50.5%	-5.9%	92.6%	63.9%	-28.6%
Education	54	50.0%	44.9%	-5.0%	55.3%	55.4%	+0.1%	94.2%	74.9%	-19.3%
Healthcare	49	43.3%	41.4%	-1.9%	63.7%	59.2%	-4.5%	93.0%	75.4%	-17.6%

rors concern schema coverage and hierarchy, while invalid JSON is uncommon.

Table 12: Task 1 schema-error rates across eight HTML-input models. Categories are non-exclusive; Δ denotes Template minus Filled.

Schema Error	Filled	Templ.	Δ
Invalid/non-dict JSON	4.3%	4.7%	+0.3%
Omitted gold keys	91.7%	90.8%	-0.9%
Extra keys	93.1%	91.9%	-1.2%
Misnested keys	71.5%	66.1%	-5.4%
Container type mismatch	23.4%	20.5%	-2.8%
List cardinality mismatch	0.7%	2.7%	+2.1%
Value/content mismatch	7.1%	1.7%	-5.4%

E.6 Multi-Key Cell Analysis

Table 13: Multi-Key Cell analysis across eight HTML-input models: localized Task 1 schema-error rates and Tasks 2–3 region accuracy. Δ denotes Template minus Filled.

Measure	Filled	Templ.	Δ
<i>Task 1: SSP (localized schema-error rates)</i>			
Omitted gold keys	43.2%	44.3%	+1.1%
Misnested keys	42.5%	39.4%	-3.1%
Extra keys	30.8%	21.4%	-9.5%
Container type mismatch	5.0%	4.1%	-0.9%
Value/content mismatch	2.1%	0.5%	-1.6%
<i>Tasks 2–3: MKC region accuracy</i>			
Task 2: CHA	49.2%	48.2%	-1.0%
Task 3: Infilling	85.0%	59.0%	-26.0%

An MKC is a physical cell or merged region containing multiple guided input or guided option fields. For Task 1, we instead map schema errors back to MKC constituents and report the share of paired MKC regions affected by each non-exclusive error category. For Task 2, an MKC’s score is the fraction of its alignment targets handled correctly. For Task 3, an MKC’s score is the fraction of its filling targets handled correctly. The reported results average these region scores over the same MKC regions in the Filled and Template conditions.

For Task 1, omitted, misnested, and extra keys are the most frequent schema errors localized to MKCs. MKCs remain difficult for CHA in both conditions, but their Filled-to-Template difference is small. Their Infilling accuracy instead drops by 26.0 points on templates, mirroring the task-level pattern. Within failed MKC regions, common CHA errors omit one or more internal fields or confuse keys that share a physical cell. Infilling errors most often show only partial text overlap, leave the target cell empty, or place a value in a neighboring cell.