From Cells to Sentences: An End-to-End Framework for Table Understanding

Deepak Vijaykeerthy
IBM Research, India

Arvind Agrawal
IBM Research, India

Abstract

Real-world tables are messy: column headers are inconsistent, cells contain errors or missing values, and crucial information is scattered across multiple tables and documents. These issues cause even state-of-the-art language models to fail at seemingly simple questions. We present a robust framework for table understanding that explicitly handles these challenges through three coordinated mechanisms: structure-aware encoders that learn invariance to common corruptions, trainable slots that compress evidence to a fixed-size representation, and grounding modules that align each slot to supporting text passages. Unlike prior work that treats tables as flat text or relies on clean schemas, our approach maintains strong performance even under schema corruption and structural perturbations. Across eight benchmarks spanning question answering, fact verification, and text generation, we achieve the best performance among methods without external tools on five tasks and remain competitive with systems using much larger models or SQL executors. Under schema corruption and row/column permutations, our method degrades by less than 2 points, while baselines drop by up to 6-22 points, confirming that explicit denoising and grounding are essential for robust table understanding.

1 INTRODUCTION

Existing methods for table understanding struggle with real-world data. Serialisation approaches (Liu et al., 2022; Zhang et al., 2024; Xie et al., 2022) flatten tables into text, destroying structural information and obscuring relevant evidence (Patnaik et al., 2024). Structure-aware methods (Herzig et al., 2020; Yang et al., 2022; Yin et al., 2020) preserve semantics through specialised encodings but assume clean tables that rarely exist in practice, with inference costs scaling quadratically with table size. Recent work explores compression strategies (Long et al., 2025; Li et al., 2025), but these methods are not trained to handle corrupted schemas and lack explicit grounding to external text. Models are brittle to column permutations and header variations (Singha et al., 2023) (e.g., "Population" becoming "Pop").

The core challenge is that current approaches conflate three distinct problems: *structure induction*, understanding what the table represents despite noise (inconsistent headers, missing values); *evidence localisation*, identifying which cells matter for a query; and *faithful reasoning*, ensuring models attend to the evidence they identify as relevant. This entanglement causes brittleness: permuting rows or columns causes 6-17 point drops, noisy headers degrade performance by 11-22 points (Table 3), and evidence spanning tables and text remains poorly handled (Chen et al., 2020b).

We address these limitations through three contributions: (1) **Corruption-aware training** learns structure-invariant representations through targeted denoisers (Stage 1). (2) **Evidence compression with external grounding** compresses tables into fixed-capacity slots (10-11× reduction) aligned with text passages via weak supervision. (3) **Faithful reasoning** through routing consistency, ensuring claimed importance matches actual attention usage, and auxiliary denoisers teaching multi-step decomposition (Stage 2b). This staged design avoids gradient interference observed in joint optimisation (≈10-point degradation).

Our 8B backbone achieves best performance on 5 of 8 benchmarks, outperforming prior work on multi-hop

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. Copyright 2026 by the author(s).

reasoning (HybridQA: 56.5 vs 53.1 EM) while running ≈3× faster. While baselines degrade by 6-22 points under schema corruption and permutations, our method maintains performance within 2 points. Our approach achieves ≈92% leave-one-dataset-out transfer, demonstrating genuine semantic learning over task-specific shortcuts.

2 RELATED WORK

Early approaches adapted language models through structural embeddings. TaBERT (Yin et al., 2020) and TAPAS (Herzig et al., 2020) introduced row/column positional encodings, while TableFormer (Yang et al., 2022) added learnable attention biases. Task-specific pretraining followed: TAPEX (Liu et al., 2022) trained on SQL execution, TabT5 (Andrejczuk et al., 2022) adapted encoder-decoders for generation. These methods proved fragile to permutations and noisy tables (Patnaik et al., 2024).

SQL synthesis approaches such as BINDER and DATER (Cheng et al., 2023; Ye et al., 2023) achieve strong results in executable environments but struggle with free-form questions. HybridQA (Chen et al., 2020b) introduced table-text reasoning, spurring graph-based methods like GRAPPA (Yu et al., 2021). CABINET (Patnaik et al., 2024) demonstrates that even strong LLMs remain sensitive to irrelevant cells and layout changes. These approaches assume clean inputs and lack explicit mechanisms for handling corruption.

Recent work has explored efficiency improvements: TNT (Long et al., 2025) utilises learnable queries for column-wise compression, while TAMO (Li et al., 2025) employs pooling for fixed-size representations. However, critical gaps remain: models are not trained to handle corrupted schemas; representations lack explicit grounding to external text; and attention is not constrained to focus on relevant information rather than distractions.

Our Approach. We address these gaps through three contributions: corruption-aware training for structure induction, evidence compression with external grounding for localisation, and routing consistency with auxiliary objectives for faithful reasoning (Section 3).

3 METHOD

Our objective is to develop a model that can robustly localise, compress, and ground tabular evidence, supporting compositional reasoning and faithful answer generation even under noise. Formally, each example consists of $(q, T, \mathcal{D}, y)$, where q is a natural-language query, T is a table with headers h_j and cells c_{ij}, $\mathcal{D}$ is an optional context, and y is the task output.

Real-world tables challenge existing methods: the "Population" column might appear as "Pop.", "inhabitants", or be missing entirely, with values embedded in text descriptions. Our framework explicitly handles such ambiguity through three stages.

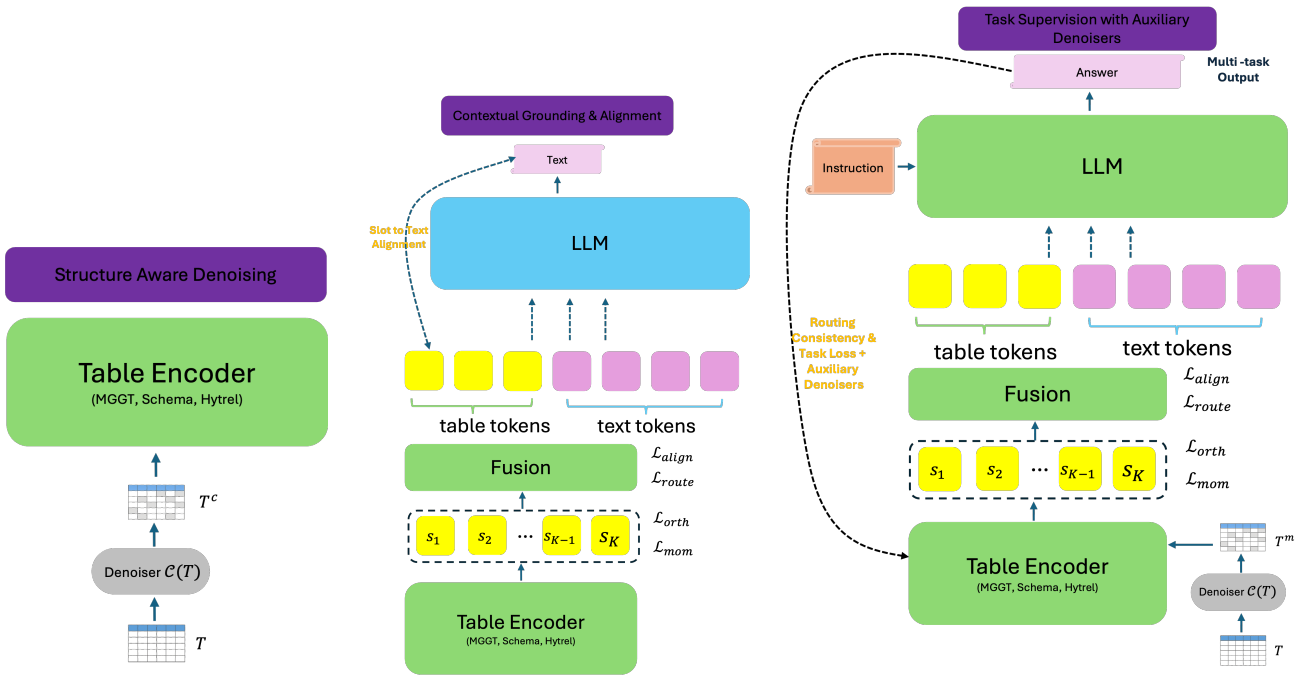
Figure 1 illustrates our three-stage approach that decouples distinct aspects of table understanding. Stage 1 learns noise-invariant representations through targeted corruption and reconstruction. Stage 2a compresses these into fixed-size slots that summarise relevant evidence and align with external text via weak supervision. Stage 2b refines the model to focus only on task-relevant information while filtering distractions, using task-specific objectives and auxiliary denoisers. This modular design separates structure understanding, evidence localisation, and reasoning, enabling robustness where monolithic approaches fail.

3.1 Stage 1: Robust Denoising Pretraining

Standard fine-tuning memorises surface patterns (exact headers, layout), causing brittleness. Given a table T, our goal is to learn embeddings $z_v \in \mathbb{R}^d$ for every table element v that are robust to structural noise. We employ structure-aware encoders tailored to table complexity: a Schema encoder (utilising 2D positional embeddings for row/column structure) for simple tables, MGGT (Multi-Granular Graph Transformer, which represents tables as directed graphs with edges for row adjacency, column membership, and relational connections, constraining attention to graph neighbourhoods) for multi-hop relational reasoning, and HyTrel (Chen et al., 2023) for entity-centric aggregation tasks. We select encoders based on task characteristics: Schema for single-table operations, MGGT for multi-hop reasoning requiring cross-table connections, and HyTrel for entity aggregation (Table 1 details assignments per task). The key insight is that different table understanding tasks require different structural inductive biases, but all benefit from corruption-aware training. We validate these assignments through cross-encoder evaluation on development sets (Appendix A), confirming that architectural intuitions align with empirical performance.

Denoising Objectives. To make encoders resilient to real-world noise, we train them with targeted denoisers. At each step, we sample a denoiser c from distribution $\pi^{(1)}$ and corrupt table T to T^c. The encoder must reconstruct the original content, learning invariance to common corruptions.

We employ five corruption-aware denoising objectives



(a) Structure-Aware Denoising (b) Contextual Grounding & Alignment (c) Task Supervision

Figure 1: Three-stage framework for robust table understanding. **(a) Stage 1:** Encoders learn invariance through targeted corruptions (header: Population→Pop; subtable: row/column masking; cell: 1.5M→1,5M; edge: relational cues; extreme: scrambled content). **(b) Stage 2a:** Table evidence is compressed into K slots aligned with external text via weak supervision, with routing consistency ensuring saliency matches attention usage. **(c) Stage 2b:** Full system optimisation with auxiliary denoisers (summary completion, DMTR for multi-step reasoning, TREaA for entity grouping, CRD for contrastive reasoning). Green indicates trainable parameters; blue indicates frozen. Unlike TNT/TAMO, our approach combines corruption-aware pretraining, external grounding, and routing regularisation for robust table understanding.

adapted for table understanding: *Header denoising* randomly permutes, drops, or paraphrases column headers, transforming "Population" to "Pop." or "inhabitants", forcing the encoder to rely on semantic relationships rather than exact header wording. *Subtable denoising* masks blocks of rows or columns, simulating filtered or truncated tables and requiring inference from context. *Cell denoising* introduces typos ("1.5M" → "1,5M"), format inconsistencies, and unit variations, teaching automatic value normalisation. *Edge denoising* perturbs structural relationships, forcing the model to infer true connections critical for multi-hop reasoning. *Extreme denoising* applies heavy corruption, stress-testing recovery on severely damaged tables. These five corruption types target common real-world table errors.

Formally, the Stage 1 objective is:

$$\mathcal{L}_{\text{stage1}} = \mathbb{E}_{c\sim\pi^{(1)}}\left[\mathcal{L}_{\text{MLM}}(T^c) + \mathcal{L}_c(T^c, T)\right],$$

where each denoiser applies its reconstruction loss $\mathcal{L}_c$, jointly optimised with masked language modelling $\mathcal{L}_{\text{MLM}}$. This ensures representations are semantically meaningful and robust to real-world corruptions.

3.2 Stage 2: Slot-Based Evidence Compression and Grounding

While Stage 1 provides robust embeddings, efficient reasoning requires compression with grounding: slots reduce computational cost while text alignment ensures predictions remain traceable. However, a 100×20 table yields 2000-element embeddings $\{z_v\}$ too numerous for efficient processing. Stage 2 compresses this evidence into a fixed set of K slots ($K \in \{8, 16, 32, 64, 128\}$) while ensuring each slot is grounded in external text. To achieve grounded compression, we maintain specialised encoders: structure-aware table encoders from Stage 1 produce $\{z_v\}$, while gte-base (Li et al., 2023) encodes text passages $\mathcal{D}$. Both of them project via learned linear layers to a 256-dimensional shared alignment space for computing $\mathcal{L}_{\text{align}}$. Crucially, only slots interface with the LLM; text embeddings serve solely for training-time align-

ment supervision, not as input to downstream models.

Stage 2a: Evidence Localisation with External Grounding We compress table evidence into slots $S = \{s_k\}_{k=1}^{K}$ through saliency-weighted attention. Each slot s_k aggregates information from relevant table elements:

$$s_k = \sum_v \alpha_{kv} z_v, \quad \text{where} \quad \alpha_{kv} = \frac{\exp(q_k^T z_v) \cdot r_v}{\sum_{v'} \exp(q_k^T z_{v'}) \cdot r_{v'}}$$

Here, q_k are learned slot queries, z_v are element embeddings from Stage 1, and r_v are saliency scores predicting element importance. Saliency scores r_v are initialised to near-uniform values and optimised end-to-end with $\mathcal{L}_{\text{route}}$ and task objectives, enabling them to evolve from uninformative priors to meaningful importance estimates. Unlike standard attention, separating element importance (r_v) from query-key matching ($q_k^T z_v$) enables explicit validation (Table 2) and guidance from text alignment. This creates K fixed-size representations regardless of table size, decoupling model complexity from input dimensions.

External Grounding via Weak Supervision. Compression alone is insufficient; predictions must be traceable to real evidence. We align each slot to relevant sentences in external passages $\mathcal{D}$ (such as Wikipedia snippets or linked documents), using weak supervision to guide attention toward relevant passages while filtering distractions. We segment $\mathcal{D}$ into sentence-level spans and embed each span d as t_d. The model learns to compute attention distributions from slots to these text embeddings.

Without gold alignments, we create weak supervision by computing both IDF-weighted token overlap and embedding similarity (using t_d) between slots and candidate text spans. For each slot-text pair, we compute $\hat{p}_{kj} = \alpha \cdot \text{IDF-overlap}(s_k, d_j) + (1-\alpha) \cdot \text{cos-sim}(s_k, t_{d_j})$ where $\alpha = 0.6$ balances lexical and semantic matching, then normalise across spans to form reference distribution $\hat{P}_k$ for each slot. Because this combines lexical and semantic cues, matching remains robust under schema corruption, even if "Population" becomes "Pop", the token overlap with text mentioning "population" provides a signal, while embedding similarity captures semantics. The alignment loss $\mathcal{L}_{\text{align}}(S, \mathcal{D}) = \sum_k \text{KL}(\hat{P}_k \| A_k)$ minimises divergence between model attention A_k and weak reference, encouraging slots to ground in verifiable evidence. Task gradients during Stage 2b refine this initial alignment. Ablations (Appendix A) confirm that slot-to-text alignment contributes 2.6 BLEU on FeTaQA.

Additionally, DocNearTable denoising forces slots to encode information predictive of external text: given slots S and masked context $\mathcal{D}_{\text{mask}}$, the model reconstructs missing words via $\mathcal{L}_{\text{DocNearTable}}$, bridging structured and unstructured data (Appendix H).

Routing Consistency. We ensure faithfulness between claimed importance and actual usage. Recall that slots aggregate elements via $s_k = \sum_v \alpha_{kv} z_v$, where α_{kv} depends on predicted saliency scores r_v. During aggregation, each element v receives total attention $\sum_k \alpha_{kv}$ across all slots. We enforce routing consistency via:

$$\mathcal{L}_{\text{route}} = \left\| r_v - \text{sg}\left(\sum_k \alpha_{kv} \right) \right\|_2^2$$

where $\text{sg}(\cdot)$ denotes stop-gradient on the attention term, preventing saliency from simply copying attention patterns. λ_{route} warms from 0.5→1.0 over 1K steps to avoid premature commitment to suboptimal patterns. This encourages consistency between claimed and actual importance, using weak supervision to guide attention toward relevant elements while allowing the task gradients to refine both saliency and attention patterns during training. This ensures claimed importance matches actual usage, enabling post-hoc validation (Table 2).

The complete Stage 2a objective combines these components with regularisation terms for stability:

$$\begin{aligned}\mathcal{L}_{\text{stage2a}} = &\mathcal{L}_{\text{DocNearTable}} + \lambda_{\text{align}} \mathcal{L}_{\text{align}} + \lambda_{\text{route}} \mathcal{L}_{\text{route}} \\ &+ \mathcal{L}_{\text{CLM}} + \gamma_{\text{mom}} \mathcal{L}_{\text{mom}} + \gamma_{\text{orth}} \mathcal{L}_{\text{orth}}\end{aligned}$$

Here, $\mathcal{L}_{\text{CLM}}$ is the standard causal language modelling loss, while $\mathcal{L}_{\text{mom}} = \|s_k^{(t)} - \text{sg}(s_k^{(t-1)})\|_2^2$ stabilizes training by penalizing large slot updates between steps (preventing oscillation), and $\mathcal{L}_{\text{orth}} = \sum_{i \neq j} (s_i^T s_j)^2$ prevents slot collapse by encouraging orthogonality (avoiding redundant representations).

LLM Fusion Interface Grounded slots must be aligned with the LLM's representation space. We employ two fusion mechanisms: **Prefix fusion** projects slots via learned linear layers into the LLM's embedding space and prepends them to input sequences, making them efficient for simple tasks. **Cross-attention fusion** similarly projects slots, then injects them as external memory banks that LLM layers can query via cross-attention, enabling compositional reasoning where evidence is revisited during generation. Both approaches force all table information through the slot bottleneck while adapting between encoder and decoder representation spaces.

Stage 2b: Task Supervision with Auxiliary Robustness While Stage 1 handles input corruption,

Stage 2b addresses reasoning failures through task-specific denoisers. With grounded slots established, Stage 2b adds task-specific supervision while refining the alignment bootstrapped in Stage 2a. Beyond standard task loss $\mathcal{L}_{\text{task}}$, we adapt auxiliary denoising objectives for table understanding, sampled from $\pi^{(2)}$, that teach the model to filter noise and focus on task-relevant evidence, addressing specific reasoning failure modes: *Summary/completion denoiser* omits content-planning steps, forcing coherent generation from slots rather than copying table fragments. *DMTR* (Decomposed Multi-step Table Reasoning) teaches decomposition of complex queries into interpretable sub-steps for multi-hop reasoning. *TREaA* (Table Reasoning with Entity-aware Attention) enforces consistency across semantic groups (e.g., all "Q1 revenue" cells across years). *CRD* (Contrastive Reasoning Distillation) sharpens discrimination between genuine and spurious reasoning chains through contrastive examples (Appendix H).

The complete Stage 2b objective:

$$\mathcal{L}_{\text{stage2b}} = \mathcal{L}_{\text{task}} + \beta\, \mathbb{E}_{m\sim\pi^{(2)}}[\mathcal{L}_m] + \lambda_{\text{align}}\mathcal{L}_{\text{align}}. \quad (1)$$

Here β gradually decreases to shift focus toward task accuracy, while λ_{align} persists to maintain grounding. This corresponds to *multi-task fine-tuning*, with task loss and auxiliary denoisers optimised jointly. Crucially, $\mathcal{L}_{\text{align}}$ enables task gradients to refine Stage 2a's weak supervision, improving slot-to-text alignment and preventing drift from external evidence.

4 EXPERIMENTS

We evaluate three core claims: (1) corruption-aware training yields robustness to table perturbations, (2) slot compression maintains accuracy while reducing computational cost, and (3) our framework learns transferable table understanding. All results report the mean ± standard error across five random seeds.

Setup We evaluate across eight benchmarks spanning question answering (WikiTQ (Pasupat and Liang, 2015), HybridQA (Chen et al., 2020b), FeTaQA (Nan et al., 2022)), fact verification (TabFact (Chen et al., 2020a), FEVEROUS (Aly et al., 2021)), SQL synthesis (WikiSQL (Zhong et al., 2017)), table-to-text generation (ToTTo (Parikh et al., 2020)), and structural understanding (StructQA (Li et al., 2025)). We use Llama models at 3B and 8B scales with slot budgets $K \in \{8, 16, 32, 64, 128\}$, selecting K based on the smallest budget achieving ≤1 point loss vs K = 128 on dev-tune and <2-point mean drop under perturbations. [1] We use K=128 as the reference, as it represents near-saturation performance across tasks while remaining computationally tractable.

As baselines, we use FT-LLM (8B), sharing our architecture and training data but without corruption-aware training (Stage 1), slot compression and grounding (Stage 2), or auxiliary denoisers, which isolate our contributions. We attempted to reproduce TNT and TAMO but were unable to achieve the reported performance without public checkpoints. Therefore, we report their published results on clean data. For corruption robustness evaluation, where no prior baselines exist, we use FT-LLM as a controlled baseline, sharing our architecture and training data to isolate the contributions of corruption-aware training and grounding.

4.1 Results

Table 1 presents our main results. Our 8B model achieves the best performance on 5 of 8 benchmarks. On multi-hop table-text reasoning tasks, we outperform prior work: HybridQA achieves 56.5 vs. 53.1 EM (+3.4), while on table-text generation, we outperform tool-augmented systems using substantially larger LLMs (FeTaQA: 42.5 vs. 41.2 BLEU, +1.3), showing that our approach achieves tool-level performance without external dependencies.

For robustness comparison, CABINET (Patnaik et al., 2024), designed explicitly for noisy tables, achieves 87.3 on WikiSQL and 40.1 on FeTaQA, while our approach achieves 90.5 and 42.5, respectively.

Our external grounding contributes to multi-hop reasoning (HybridQA, StructQA): correctly answering complex queries benefits from aligning table evidence with linked external passages. Our slot-to-text alignment grounds slots in relevant passages via weak supervision, while baselines hallucinate without verifying external content. On entity-centric tasks (FEVEROUS, TabFact), corruption-aware training enables robust multi-entity verification despite lexical variations.

Compared to FT-LLM (8B), we achieve 3-8 point improvements (HybridQA: +7.1, StructQA: +7.9). Our staged training is essential: without Stage 1 denoising, permutation drops increase to 5-9 points; Stage 2a achieves efficiency while maintaining accuracy; Stage 2b provides final task-specific gains of 3-5 points. Ablating any stage results in a 5-10 point degradation in performance (Appendix A).

Slot Faithfulness & Importance Validation To validate that routing consistency produces meaningful importance rankings rather than spurious correlations, we conduct slot ablation experiments. We

[1] We report on standard test sets where available; otherwise, we partition development sets into 10% for hyper-parameter tuning (dev-tune) and 90% for evaluation (dev-eval). The slot budget K is selected on dev-tune.

Dataset (Encoder, Metric)	TFLLAMA-3B	TFLLAMA-8B	FT-LLM (8B)	Best Prior (w/ tools)	Best Prior (w/o tools)
FEVEROUS (HyTrel, Acc)	79.0 ± 2.1	**83.2** ± 1.2	76.2	—	78.5 (Long et al., 2025)
FeTaQA (MGGT, sacreBLEU)	40.0 ± 2.1	**42.5** ± 1.3	38.1	41.2 (Yang et al., 2024)	40.1 (Patnaik et al., 2024)
HybridQA (MGGT, EM)	53.2 ± 2.7	**56.5** ± 1.6	49.4	—	53.1 (Long et al., 2025)
StructQA (MGGT, Acc.)	72.4 ± 2.3	**76.0** ± 1.2	68.1	—	71.6 (Li et al., 2025)
TabFact (HyTrel, Acc.)	81.1 ± 2.3	88.0 ± 1.5	83.3	**92.0** (Yu et al., 2025)	—
ToTTo (Schema, BLEU)	49.0 ± 1.8	**51.0** ± 1	46.1	—	49.5 (Andrejczuk et al., 2022)
WikiSQL (Schema, Acc)	86.8 ± 1.7	90.5 ± 1	87.1	**92.0** (Ghosh et al., 2025)	87.3 (Patnaik et al., 2024)
WikiTQ (MGGT, Acc.)	62.2 ± 2.4	69.5 ± 1.1	63.1	**73.0** (Yu et al., 2025)	—

Table 1: Main results comparing our approach (TFLLAMA) against our baseline (FT-LLM) and prior state-of-the-art. *Best Prior* divides methods by tool usage (code/SQL/verification APIs). FT-LLM shares our architecture and training data, isolating contributions from corruption-aware training and grounding. Blank entries (—) indicate no reported results or performance below our method.

Dataset (Metric)	Clean	Top-50% masked	Random-50% masked	Bottom-50% masked
FEVEROUS (Acc)	83.2	56.2 ± 1.4 (-27.0)	73.8 ± 1.0 (-9.4)	79.4 ± 0.9 (-3.8)
HybridQA (EM)	56.5	31.5 ± 1.5 (-25.0)	48.5 ± 1.0 (-8.0)	53.5 ± 0.8 (-3.0)
WikiTQ (Acc.)	69.5	53.5 ± 1.3 (-16.0)	63.0 ± 0.9 (-6.5)	67.3 ± 0.7 (-2.2)
StructQA (Acc.)	76.0	63.0 ± 1.2 (-13.0)	70.5 ± 0.9 (-5.5)	74.0 ± 0.7 (-2.0)
TabFact (Acc.)	88.0	75.0 ± 1.1 (-13.0)	81.8 ± 0.8 (-6.2)	84.9 ± 0.6 (-3.1)
WikiSQL (Acc.)	90.5	82.5 ± 0.9 (-8.0)	87.5 ± 0.6 (-3.0)	89.0 ± 0.6 (-1.5)
ToTTo (BLEU)	51.0	45.0 ± 0.8 (-6.0)	48.9 ± 0.6 (-2.1)	49.6 ± 0.5 (-1.4)
FeTaQA (BLEU)	42.5	28.5 ± 1.0 (-14.0)	37.2 ± 0.7 (-5.3)	40.3 ± 0.6 (-2.2)

Table 2: Slot importance validation through systematic masking. Numbers in parentheses indicate the absolute drop from clean performance. The clear hierarchy Top >> Random >> Bottom validates that routing consistency learns genuine importance rankings.

mask 50% of slots under three regimes: Top (highest causal importance), Random, and Bottom (lowest). Per-example causal importance is estimated via one-pass occlusion on dev tune: for each slot k, we measure performance degradation when s_k is masked, $I_k = \mathcal{L}(y, f(\mathcal{S} \setminus \{s_k\})) - \mathcal{L}(y, f(\mathcal{S}))$, where a higher I_k indicates greater importance. Slots are ranked by I_k and partitioned accordingly. Masks are applied persistently at each slot-carrying layer: we zero-mask slot activations before LayerNorm, remove their attention keys/values, and reapply masking after the feed-forward sublayer, injecting small Gaussian noise ($\sigma = 0.01 \times \text{median}_k \|s_k\|_2$) to prevent normalisation rescue.

Table 2 demonstrates a clear hierarchy across all datasets: removing high-importance slots causes substantially larger drops than removing low-importance slots. On complex reasoning tasks requiring multi-hop evidence aggregation (FEVEROUS, HybridQA), top-slot removal causes drops of 25-27 points, while bottom-slot removal degrades by only 1-4 points. Even on simpler tasks (WikiSQL, ToTTo), the importance ranking holds with ≈5x larger drops for top versus bottom slots.

The systematic hierarchy (Top >> Random >> Bottom) provides evidence for two claims. First, the systematic performance hierarchy indicates that slots capture task-relevant information rather than encoding random patterns. Second, routing consistency produces saliency scores that correlate with task-level causal importance, as evidenced by the clear separation between masking regimes.

Ablation: Slot Budget & Encoder Specialisation Figures 2 and 3 show how slot budget K affects performance. We chose $K \in \{8, 16, 32, 64, 128\}$ to span from aggressive compression (K=8: 100:1 for large tables) to minimal compression (K=128). Parameter K directly controls the compression-accuracy tradeoff, with K=64 compressing median tables by 10-11× (e.g., 648 elements → 64 slots).

Performance patterns reveal task-specific optimal configurations. Simple single-table tasks (WikiSQL) saturate at K=16-32 with negligible gains beyond. Compositional tasks (FEVEROUS, HybridQA, StructQA) requiring distributed evidence benefit from larger budgets (K=64-128) to capture multi-hop dependencies and complex relationships, suggesting our slot mechanism successfully identifies and preserves task-relevant information while discarding redundancy.

Encoder specialisation emerges consistently across experiments. Schema encoders excel on WikiSQL (90.5 Acc) and ToTTo (51.0 BLEU), where columnar structure dominates. MGGT achieves best results on HybridQA (56.5 EM) and WikiTQ (69.5 Acc), leveraging graph structure for cross-table reasoning. HyTrel leads on FEVEROUS (83.2 Acc) and TabFact (88.0 Acc), where hypergraph connections capture entity-level consistency crucial for fact verification. These

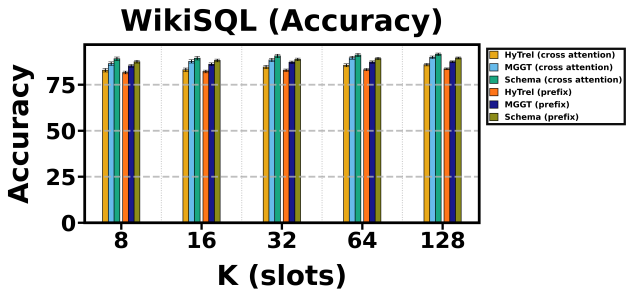


Figure 2: WikiSQL Accuracy vs slot budget K. Performance saturates at K=32.

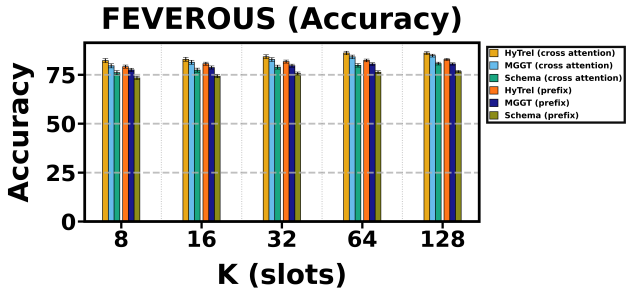


Figure 3: FEVEROUS Accuracy score vs slot budget K. Multi-hop tasks benefit from higher K.

specialisation patterns persist across slot budgets, though performance gaps narrow at high K as additional capacity compensates for architectural differences. Cross-encoder evaluation (Appendix A) validates these assignments empirically.

Robustness to Permutation and Schema Corruption Table 3 shows that our framework remains stable under structural perturbations that preserve semantics but often break existing systems. Using the same architecture, training data, and protocols as FT-LLM, we isolate the effect of corruption-aware training and slot compression. Under row/column shuffling, our accuracy drops by <2 points across all tasks, while FT-LLM degrades by 6–17 points, with particularly severe drops on WikiSQL (–16.6) and WikiTQ (–16.8). Schema anonymisation shows even starker contrasts: FT-LLM drops 11–22 points, while we maintain similar stability.

These gains are consistent with our design, which combines corruption-aware training (including permutation and header denoisers) with slot-based compression. The consistent margins across tasks suggest that our staged approach provides robustness benefits that generalise across different table understanding scenarios.

Component Ablations Stage 2a component contributions on FeTaQA: DocNearTable (-3.6 BLEU), slot-to-text alignment (-2.6 BLEU), routing consistency (-1.3 BLEU). Individual removals sum to -7.5 BLEU, while joint removal yields -8.2 BLEU. While individual contributions may appear modest, their combined effect is substantial: our full system outperforms FT-LLM by 7.1 points on HybridQA and 4.4 points on FeTaQA, despite sharing the same architecture and training data. These gains emerge from synergistic effects: corruption-aware training, slot compression, and grounding work in coordination rather than redundantly. Denoiser importance varies by task: Cell denoising (-2.8 Acc) handles value normalisation for numerical comparisons, while edge denoising (-2.1 Acc) preserves relational structure. Task-specific patterns emerge: WikiSQL exhibits a larger header impact (queries reference column names), HybridQA benefits from subtable denoising (evidence spans partial tables), and TabFact displays stronger edge effects (entity relationships drive verification).

	Row/Column Permutation		Schema Anonymization	
Dataset	TFLLAMA	FT-LLM	TFLLAMA	FT-LLM
FEVEROUS	82.6 (-0.6)	66.3 (-9.9)	83.0 (-0.2)	62.1 (-14.1)
FeTaQA	41.6 (-0.9)	31.2 (-6.9)	41.3 (-1.2)	27.0 (-11.1)
HybridQA	56.1 (-0.4)	37.4 (-12.0)	56.0 (-0.5)	31.4 (-18.0)
StructQA	75.4 (-0.6)	56.1 (-12.0)	75.0 (-1.0)	52.1 (-16.0)
TabFact	86.1 (-1.9)	68.6 (-14.7)	86.4 (-1.6)	65.3 (-18.0)
ToTTo	49.6 (-1.4)	40.2 (-5.9)	49.7 (-1.3)	34.2 (-11.9)
WikiSQL	89.3 (-1.2)	70.5 (-16.6)	89.8 (-0.7)	69.7 (-17.4)
WikiTQ	68.2 (-1.3)	46.3 (-16.8)	68.0 (-1.5)	41.3 (-21.8)

Table 3: Robustness under structural perturbations. Numbers in parentheses show degradation from clean performance. TFLLAMA maintains <2 point drops while FT-LLM degrades by up to 6-22 points. Row/column permutations are random shuffles (5 per table, mean reported). Schema anonymisation replaces column headers with generic labels (Col1, Col2, etc.).

Training Strategy Validation To validate that staged training is essential, we compare our approach with alternatives that have identical training budgets. *Joint* trains all three stage objectives simultaneously with balanced sampling. *NoS1* omits Stage 1 corruption-aware pretraining, beginning directly with Stage 2a. Table 4 shows representative results across three task categories. Joint training consistently degrades performance by approximately 10 points, confirming that simultaneous optimisation causes gradient interference. Omitting Stage 1 pretraining results in substantial drops, particularly on tasks that are robustness-critical, such as HybridQA. These patterns hold across all eight benchmarks (Appendix A), validating that staged training with convergence-based transitions is essential for learning robust table representations.

Table 4: Training strategy validation on representative tasks. All methods use identical training budgets. Full results across all 8 benchmarks in Appendix A

Dataset	Staged	Joint	NoS1
FEVEROUS (Acc)	83.2	73.8 (-9.4)	74.4 (-8.8)
HybridQA (EM)	56.5	47.3 (-9.2)	50.3 (-6.2)
WikiTQ (Acc.)	69.5	59.0 (-10.5)	63.0 (-6.5)
Mean drop	—	-9.7	-7.2

4.2 Additional Results

Data Efficiency Figure 4 shows TFLLAMA-8B reaches 90% peak accuracy using 50-75% of training data, while FT-LLM requires the full dataset. At low-data regimes (10-25%), we maintain 30-60% peak performance versus FT-LLM's 15-37%.

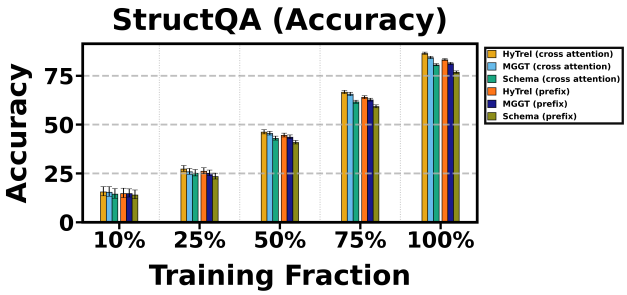


Figure 4: Data efficiency on StructQA. Our approach achieves 90% accuracy at 50% data, whereas FT-LLM requires full data.

Computational Efficiency Table 5 shows that our slot-based compression achieves substantial speedups over serialisation approaches that scale quadratically. The largest WikiTQ tables (2,688 tokens) require 15.2GB and often timeout for serialisation, while our K=32 uses <1GB for table processing and completes in 130ms. This reflects our $O(NK)$ vs $O(N^2)$ complexity advantage where $K \ll N$. With K=16-32, we achieve ≥90% of peak performance while substantially reducing memory usage, enabling deployment on consumer-grade GPUs. This compression-accuracy tradeoff is controllable via parameter K, unlike serialisation methods, where efficiency requires aggressive pruning that risks losing essential evidence.

Leave-One-Dataset-Out Transfer Table 6 evaluates generalisation by training on seven benchmarks and evaluating on the eighth one held out during fine-tuning. TFLLAMA-8B achieves 87.8-94.8% of its supervised performance (92.3% mean), substantially outperforming FT-LLM. The high transfer rates reflect shared fundamental operations: locating cells, handling missing values, and extracting information. Corruption-aware training explicitly learns these invariances, enabling generalisation; our held-out performance often exceeds FT-LLM's supervised results (e.g., 71.5% vs 68.1% on StructQA). Transfer effectiveness correlates with task complexity: simpler extraction tasks (WikiSQL: 94.7%, FEVEROUS: 94.8%) transfer better than tasks requiring complex multi-hop reasoning (WikiTQ: 87.8%), suggesting basic operations generalise well while complex reasoning benefits from task-specific optimisation.

Method	Tokens/sec	Memory (GB)	Latency (ms)	Speedup
FT-LLM (full, avg)	142	2.3	420	1.0x
FT-LLM (full, max)	38	15.2	1580	0.3x
TFLLAMA (K=16)	512	<1	95	3.6x
TFLLAMA (K=32)	456	<1	130	3.2x
TFLLAMA (K=64)	387	<1	180	2.7x

Table 5: Computational metrics on WikiTQ. Memory measures table processing overhead beyond base model weights. Full serialisation degrades severely on large tables due to quadratic attention complexity, while our slot-based approach maintains consistent overhead regardless of table size.

Held-Out Dataset	Held-Out (Ours)	Supervised (Ours)	FT-LLM LOO	Transfer Rate
FEVEROUS (Acc)	78.9	83.2	68.1	94.8%
FeTaQA (BLEU)	39.5	42.5	32.7	92.9%
HybridQA (EM)	51.2	56.5	35.1	90.6%
StructQA (Acc.)	71.5	76.0	57.4	94.1%
TabFact (Acc.)	78.1	88.0	74.3	88.8%
ToTTo (BLEU)	48.3	51.0	41.0	94.7%
WikiSQL (Acc.)	85.7	90.5	76.1	94.7%
WikiTQ (Acc.)	61.0	69.5	48.7	87.8%

Table 6: Leave-one-dataset-out multi-task transfer. Transfer rate measures the percentage of supervised performance retained when training on 7 datasets and evaluating on the held-out 8th without fine-tuning. Our approach maintains 92.3% of supervised performance, demonstrating generalisation.

Training Configurations. All strategies receive identical total gradient updates to ensure fair comparison. *Joint* trains all three stage objectives simultaneously, sampling uniformly across losses, so that each receives equal effective updates, as in staged training. *NoS1* omits Stage 1 corruption-aware pretraining entirely. *Alternating* cycles through stages in round-robin fashion, training each objective for 500 steps before switching.

The consistent degradation hierarchy validates our architectural design. Joint training results in approximately 8-11 point drops across all tasks, demonstrating that gradient interference between objectives hinders effective learning. Stage 1 pretraining proves critical (NoS1 drops 6-9 points), particularly for tasks requiring robustness to corruption. Alternating shows intermediate performance, indicating that while round-robin stage cycling improves over joint training, explicit convergence-based transitions enable superior

Table 7: Complete training strategy ablation across all benchmarks. Numbers in parentheses indicate the performance drop relative to the Staged version.

Dataset	Staged	Alternating	NoS1	Joint
FEVEROUS (Acc.)	83.2	76.9 (-6.3)	74.4 (-8.8)	73.8 (-9.4)
FeTaQA (BLEU)	42.5	36.6 (-5.9)	36.9 (-5.6)	34.3 (-8.2)
HybridQA (EM)	56.5	51.1 (-5.4)	50.3 (-6.2)	47.3 (-9.2)
StructQA (Acc.)	76.0	70.9 (-5.1)	70.0 (-6.0)	67.0 (-9.0)
TabFact (Acc.)	88.0	82.2 (-5.8)	80.6 (-7.4)	79.3 (-8.7)
ToTTo (BLEU)	51.0	46.9 (-4.1)	46.0 (-5.0)	43.3 (-7.7)
WikiSQL (Acc.)	90.5	86.0 (-4.5)	85.0 (-5.5)	82.4 (-8.1)
WikiTQ (Acc.)	69.5	64.5 (-5.0)	63.0 (-6.5)	59.0 (-10.5)
Mean drop	—	-5.3	-6.4	-8.9

optimisation by allowing each component to stabilise before integration.

Limitations Error analysis reveals four primary limitations: (1) Extreme compositional queries requiring 4+ reasoning steps exceed slot capacity in ∼12% of cases even at K=128; (2) heavily nested tables with multi-level headers (8% of failures) challenge our flat slot representation, causing structures like "Q1 → Revenue → Product" to collapse into single slots; (3) sparse evidence patterns where 3-4 relevant rows amongst 50+ lack clear grouping markers confound our saliency mechanism; (4) encoder selection currently requires understanding task characteristics. However, this reflects our framework's flexibility: all three encoder types (Schema, MGGT, HyTrel) work within our framework, with performance differences validating architectural intuitions. Consistent robustness patterns across encoders (Table 3) confirm our contributions apply independently of encoder architecture. Despite these limitations, we achieve 45-52% accuracy on these complex failure modes versus 30-38% for baselines, maintaining a 13-15 point advantage. Failures are typically partial rather than catastrophic in nature.

5 CONCLUSION

We presented a framework for robust table understanding that explicitly decouples structure induction, evidence localisation, and reasoning through corruption-aware encoder training, slot-based compression with external grounding, and staged optimisation. Our approach achieves 10-11× compression while maintaining performance with substantial speedups. It degrades by only <2 points under structural perturbations that cause 6-22 point drops in baselines, stemming from learning invariance to surface variations rather than memorising layout patterns. Strong leave-one-dataset-out transfer demonstrates that our framework learns genuine table semantics. Our 8B backbone outperforms tool-augmented systems using substantially larger models on multi-hop table-text reasoning, showing that corruption-aware training enables learned representations to achieve tool-level performance. Our staged training recipe applies to any encoder architecture, demonstrating that explicit handling of corruption, compression, and grounding is crucial for robust and efficient table understanding.

References

R. Aly, Z. Guo, M. S. Schlichtkrull, J. Thorne, A. Vlachos, C. Christodoulopoulos, O. Cocarascu, and A. Mittal. FEVEROUS: Fact extraction and VERification over unstructured and structured information. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

E. Andrejczuk, J. Eisenschlos, F. Piccinno, S. Krichene, and Y. Altun. Table-to-text generation and pretraining with TabT5. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6758–6766, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.503.

P. Chen, S. Sarkar, L. Lausen, B. Srinivasan, S. Zha, R. Huang, and G. Karypis. HyTrel: Hypergraph-enhanced tabular data representation learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

W. Chen, H. Wang, J. Chen, Y. Zhang, H. Wang, S. Li, X. Zhou, and W. Y. Wang. TabFact: A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, April 2020a.

W. Chen, H. Zha, Z. Chen, W. Xiong, H. Wang, and W. Wang. HybridQA: A dataset of multi-hop question answering over tabular and textual data. *Findings of EMNLP 2020*, 2020b.

Z. Cheng, T. Xie, P. Shi, C. Li, R. Nadkarni, Y. Hu, C. Xiong, D. Radev, M. Ostendorf, L. Zettlemoyer, N. A. Smith, and T. Yu. Binding language models in symbolic languages. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

P. Ghosh, A. Jain, and P. Yenigalla. SQLGenie: A practical LLM based system for reliable and efficient SQL generation. In G. Rehm and Y. Li, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*

(Volume 6: Industry Track), pages 1004–1012, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-288-6. doi: 10.18653/v1/2025.acl-industry.71.

J. Herzig, P. K. Nowak, T. Müller, F. Piccinno, and J. M. Eisenschlos. TaPas: Weakly supervised table parsing via pre-training. In D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4320–4333. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.ACL-MAIN.398.

L. Li, C. Ye, W. Ye, Y. Sun, Z. Jiang, H. Wang, G. Chen, and J. Zhao. Injecting learnable table features into LLMs, 2025. URL `https://openreview.net/forum?id=Mi45HjlVRj`.

Z. Li, X. Zhang, Y. Zhang, D. Long, P. Xie, and M. Zhang. Towards general text embeddings with multi-stage contrastive learning. *CoRR*, abs/2308.03281, 2023. doi: 10.48550/ARXIV.2308.03281. URL `https://doi.org/10.48550/arXiv.2308.03281`.

Q. Liu, B. Chen, J. Guo, M. Ziyadi, Z. Lin, W. Chen, and J. Lou. TAPEX: table pre-training via learning a neural SQL executor. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

L. Long, X. Gu, X. Sun, W. Ye, H. Wang, S. Wu, G. Chen, and J. Zhao. Bridging the semantic gap between text and table: A case study on NL2SQL. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL `https://openreview.net/forum?id=qmsX2R19p9`.

L. Nan, C. Hsieh, Z. Mao, X. V. Lin, N. Verma, R. Zhang, W. Kryściński, H. Schoelkopf, R. Kong, X. Tang, M. Mutuma, B. Rosand, I. Trindade, R. Bandaru, J. Cunningham, C. Xiong, and D. Radev. FeTaQA: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10:35–49, 2022.

A. P. Parikh, X. Wang, S. Gehrmann, M. Faruqui, B. Dhingra, D. Yang, and D. Das. ToTTo: A controlled table-to-text generation dataset. In B. Webber, T. Cohn, Y. He, and Y. Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 1173–1186. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.EMNLP-MAIN.89.

P. Pasupat and P. Liang. Compositional semantic parsing on semi-structured tables. In C. Zong and M. Strube, editors, *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, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1142.

S. Patnaik, H. Changwal, M. Aggarwal, S. Bhatia, Y. Kumar, and B. Krishnamurthy. CABINET: content relevance-based noise reduction for table question answering. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

A. Singha, J. Cambronero, S. Gulwani, V. Le, and C. Parnin. Tabular representation, noisy operators, and impacts on table structure understanding tasks in llms. *CoRR*, abs/2310.10358, 2023. doi: 10.48550/ARXIV.2310.10358. URL `https://doi.org/10.48550/arXiv.2310.10358`.

T. Xie, C. H. Wu, P. Shi, R. Zhong, T. Scholak, M. Yasunaga, C.-S. Wu, M. Zhong, P. Yin, S. I. Wang, V. Zhong, B. Wang, C. Li, C. Boyle, A. Ni, Z. Yao, D. Radev, C. Xiong, L. Kong, R. Zhang, N. A. Smith, L. Zettlemoyer, and T. Yu. UnifiedSKG: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 602–631, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.39.

J. Yang, A. Gupta, S. Upadhyay, L. He, R. Goel, and S. Paul. TableFormer: Robust transformer modeling for table-text encoding. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 528–537. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.ACL-LONG.40.

Z. Yang, S. Wang, Y. Yan, P. Liu, and D. Yu. Enhancing free-form table question answering models by distilling relevant-cell-based rationales. In S. Maosong, L. Jiye, H. Xianpei, L. Zhiyuan, and H. Yulan, editors, *Proceedings of the 23rd Chinese National Conference on Computational Linguistics (Volume 1: Main Conference)*, pages 973–985, Taiyuan, China, July 2024. Chinese Information Processing Society of China.

Y. Ye, B. Hui, M. Yang, B. Li, F. Huang, and Y. Li. Large language models are versatile decomposers:

Decomposing evidence and questions for table-based reasoning. In H. Chen, W. E. Duh, H. Huang, M. P. Kato, J. Mothe, and B. Poblete, editors, *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, pages 174–184. ACM, 2023. doi: 10.1145/3539618.3591708.

P. Yin, G. Neubig, W. Yih, and S. Riedel. TaBERT: Pretraining for joint understanding of textual and tabular data. In D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8413–8426. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.ACL-MAIN.745.

P. Yu, G. Chen, and J. Wang. Table-Critic: A multi-agent framework for collaborative criticism and refinement in table reasoning. In W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17432–17451, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.853.

T. Yu, C. Wu, X. V. Lin, B. Wang, Y. C. Tan, X. Yang, D. R. Radev, R. Socher, and C. Xiong. GraPPa: Grammar-augmented pre-training for table semantic parsing. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

T. Zhang, X. Yue, Y. Li, and H. Sun. TableLlama: Towards open large generalist models for tables. In K. Duh, H. Gómez-Adorno, and S. Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 6024–6044. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.NAACL-LONG.335. URL `https://doi.org/10.18653/v1/2024.naacl-long.335`.

V. Zhong, C. Xiong, and R. Socher. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103, 2017. URL `http://arxiv.org/abs/1709.00103`.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [No]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Not Applicable]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [No]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [No]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes] All benchmark datasets and pretrained models used in this work are publicly available under permissive research licenses
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

From Cells to Sentences: An End-to-End Framework for Table Understanding

A Detailed Experimental Analysis

Summary. We ablate components (App. A.1) and other strategies to isolate contributions cited in the main results.

A.1 Component Ablations

To understand the contribution of each component, we systematically remove individual losses and denoisers, measuring the resulting performance degradation.

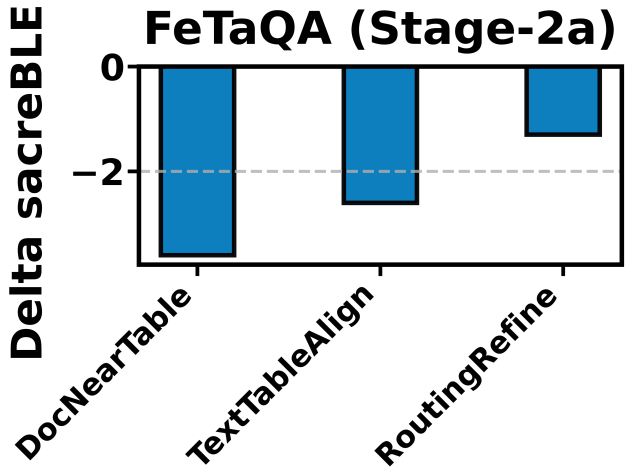


Figure 5: Removing Stage 2a components from FeTaQA. Each loss addresses distinct failure modes.

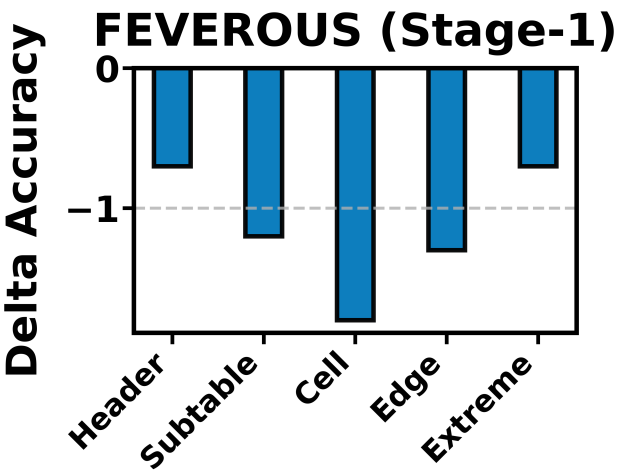


Figure 6: Removing Stage 1 denoisers from FEVEROUS. Cell, Edge & Sub-table denoising proves most critical.

Figure 5 shows the impact of removing Stage 2a losses on FeTaQA. DocNearTable contributes -3.6 BLEU, slot-to-text alignment -2.6 BLEU, and routing consistency -1.3 BLEU when removed individually. The individual component removals sum to -7.5 BLEU, while joint removal yields -8.2 BLEU, indicating modest interactions between components. Each primarily addresses distinct aspects while contributing to overall robustness.

Figure 6 reveals the importance of task-specific denoisers. For FEVEROUS, cell denoising (-2.8 Acc) handles value normalisation essential for numerical comparisons. Edge denoising (-2.1 Acc) preserves the relational structure required for multi-hop reasoning.

Task-specific patterns emerge across benchmarks: WikiSQL depends heavily on header denoising (queries reference column names), HybridQA requires subtable denoising (evidence spans partial tables), and TabFact needs edge denoising (entity relationships drive verification). These patterns validate our design of diverse, complementary types of corruption.

A.2 Encoder Selection Validation

To validate encoder assignments, we evaluated all three encoders (Schema, MGGT, HyTrel) on each task's development set. Table 8 shows representative results. We select the best-performing encoder for each task:

Schema for single-table operations (WikiSQL, ToTTo), MGGT for multi-hop reasoning (HybridQA, WikiTQ, FeTaQA, StructQA), and HyTrel for entity-centric aggregation (FEVEROUS, TabFact). The performance gaps validate our architectural intuitions: flat encoders excel when a columnar structure suffices, graph encoders when multi-hop connections are critical, and hypergraph encoders when entity-level consistency is crucial. Importantly, all encoders maintain robustness under perturbations (Table 3), confirming our contributions are orthogonal to encoder choice.

Table 8: Encoder selection validation on development sets. Best encoder (bold) used for final results. All three encoders work, validating the framework's generality.

Task (Metric)	Schema	MGGT	HyTrel
WikiSQL (Acc)	**90.5**	88.2	87.9
HybridQA (EM)	51.1	**56.5**	53.3
FEVEROUS (Acc)	78.4	80.1	**83.2**

A.3 Mechanistic Understanding of Component Contributions

Our ablations reveal how components address distinct failure modes. Stage 1 corruption-aware training provides robustness to layout perturbations, maintaining <2 point drops under schema corruption where baselines degrade 6–22 points (Table 3). Stage 2a slot compression enables computational efficiency while grounding aligns predictions with verifiable text passages. Routing consistency ensures internal faithfulness, with direct validation showing 2.7× improvement in saliency–attention correlation (Appendix D).

The staged design prevents gradient interference: joint training degrades by ≈10 points (Table 7) because corruption invariance and task-specific reasoning require different optimisation dynamics. This separation enables each component to converge independently before integration, a capability that monolithic approaches cannot achieve.

A.4 Stage 2b Auxiliary Denoiser Ablations

To validate the contribution of Stage 2b auxiliary denoisers (introduced in Section 3.2), we systematically remove each denoiser and measure performance degradation. These denoisers address specific reasoning failure modes: Summary/Completion forces coherent generation from slots, DMTR teaches multi-step decomposition, TREaA enforces entity-level consistency, and CRD sharpens discrimination between genuine and spurious reasoning chains.

Figure 7 demonstrates that auxiliary denoisers are critical for complex generation tasks, with Summary/Completion providing the largest contribution. Figure 8 shows that even simple extraction tasks benefit from compositional reasoning denoisers like DMTR, though contributions are smaller (-0.3 to -0.8 Acc). These patterns validate our design: Stage 2b denoisers maintain reasoning quality while Stage 1 denoisers provide robustness to corruption.

Task-specific importance patterns emerge: FeTaQA (generation) relies heavily on Summary (-1.8 sacreBLEU), while WikiSQL (extraction) depends more on DMTR (-0.8 Acc.) for handling compositional WHERE clauses. Across all tasks, individual denoiser removals sum to approximately the joint removal impact, indicating each addresses distinct failure modes with minimal overlap.

B Complexity Analysis

Summary. We provide a structured comparison between full serialisation and our staged approach, linking theory to the measured speedups used in the main text.

Theoretical Complexity. Traditional serialisation approaches suffer from quadratic complexity. For a table with N elements, self-attention requires $O(N^2)$ operations and memory. In contrast, our approach decomposes

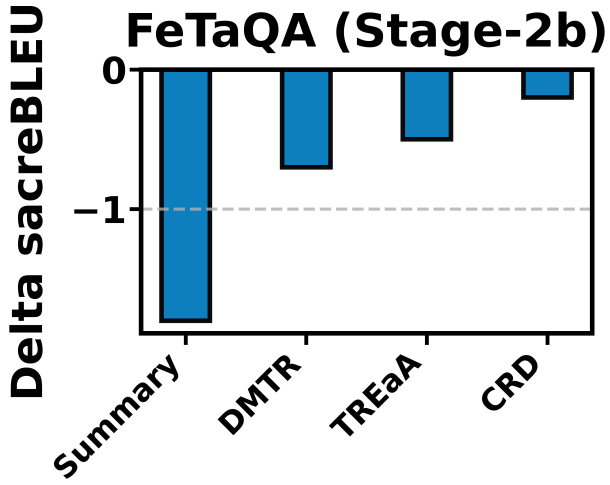


Figure 7: Removing Stage 2b auxiliary denoisers from FeTaQA. The summary denoiser provides the largest contribution (-1.8 sacreBLEU), which is essential for coherent long-form generation. DMTR and TREaA contribute -0.7 and -0.5 sacreBLEU, respectively, supporting multi-hop reasoning.

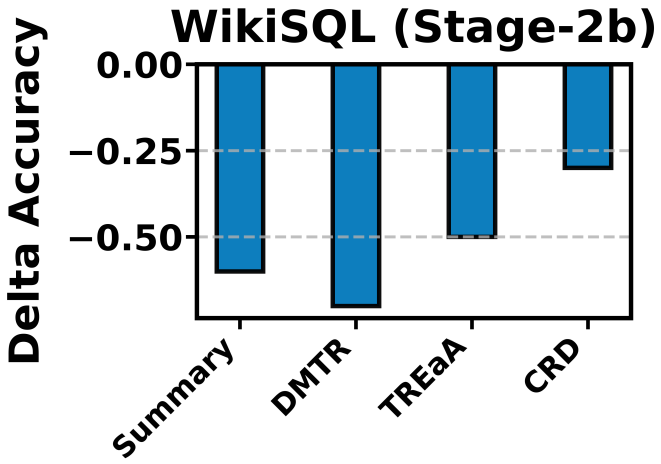


Figure 8: Removing Stage 2b auxiliary denoisers from WikiSQL. Even on simple SQL synthesis tasks, auxiliary denoisers provide modest but consistent improvements. DMTR contributes -0.8 Acc for handling compositional queries, while Summary provides smaller gains.

computation into three efficient stages:

$$\text{Serialization:} \quad O(N^2) \text{ attention} + O(N \cdot d) \text{ memory} \tag{2}$$

$$\text{Ours:} \quad \underbrace{O(N \cdot \text{deg})}_{\text{Stage 1}} + \underbrace{O(N \cdot K)}_{\text{Compression}} + \underbrace{O(K^2)}_{\text{Fusion}} \tag{3}$$

where deg is the average attention degree per element, bounded by the table structure. For MGGT and HyTrel, graph constraints ensure deg $= O(1)$, yielding linear complexity. For Schema encoder with structure-aware attention, deg $= O(\sqrt{N})$ for square tables, yielding sub-quadratic $O(N^{1.5})$ complexity. With $K \ll N$ (e.g. $K = 32$ vs $N = 500$–2000), the total complexity is dominated by Stage 1, giving $O(N)$ to $O(N^{1.5})$ depending on encoder choice—substantially better than serialisation's $O(N^2)$. This yields empirical speedups of $\approx 3\times$, validating these theoretical gains.

Compression Efficiency. Our slot mechanism achieves substantial compression without sacrificing accuracy. With $K = 64$, we compress median-sized tables by 10–11× (648 to 64 tokens for FEVEROUS, 709 to 64 for WikiTQ), with compression ratios reaching 39–128× for the largest tables in each dataset. This compression maintains performance within 1-2 points of K=128 across tasks. The compression is effective because tables contain substantial redundancy, and our saliency mechanism identifies and preserves only task-relevant information while discarding irrelevant cells.

C Encoder Architectures

Summary. We give architectural detail for the three encoders referenced in the method: **Schema**, **MGGT**, and **HyTrel**.

Schema Encoder. Designed for simple tables, this encoder represents tables as flat sequences with multi-headed self-attention. It incorporates 2D positional embeddings that encode both row and column positions, enabling the model to understand tabular structure while processing cells sequentially. Type-specific embeddings distinguish between headers, rows, and cells, supporting direct question-to-column mapping, which is essential for single-table reasoning.

MGGT (Multi-Granular Graph Transformer). For complex relational scenarios, MGGT models each token as a node in a directed graph with edges defined by:

- Row adjacency (cells in the same row)
- Column membership (cells in the same column)
- Shared attributes (cells with similar values)
- Foreign key relationships (for multi-table scenarios)

Attention is constrained by this graph structure, enabling efficient multi-hop reasoning that scales linearly with the average node degree, rather than quadratically with the table size.

HyTrel (Chen et al., 2023). For entity disambiguation and aggregation tasks, we leverage the existing HyTrel encoder, where hyperedges connect semantically related elements (e.g., all "GDP" cells across years, or all "Total" rows across tables). This hypergraph structure enables entity-centric analysis and ensures group consistency when performing operations on clustered data.

D Routing Consistency

Our routing consistency mechanism ensures that the model's predicted importance scores align with actual attention patterns. Slots aggregate table elements using saliency-weighted attention, where a learned importance modulates each element's contribution score $r_v \in (0, 1)$.

To prevent the model from claiming certain elements are essential while ignoring them in practice, we compute the actual attention usage (without saliency gating) and penalise discrepancies through a routing loss. This dual supervision ensures importance scores are both task-relevant and faithful to actual model behaviour.

During the 1,000-step warmup period, we gradually increase the routing loss weight from 0.5 to 1.0 and anneal the attention temperature from 2.0 to 1.0, preventing early commitment to poor attention patterns.

D.1 Direct Routing Consistency Validation

Our routing loss is defined over *table elements* v:

$$\mathcal{L}_{\text{route}} = \left\| r_v - \text{sg}\left(\sum_k \alpha_{kv} \right) \right\|_2^2,$$

where r_v is the predicted saliency and $\sum_k \alpha_{kv}$ is the total attention element v receives across slots. To validate its effect, we compute element-wise Pearson correlation, as well as top-k overlap, between saliency and attention vectors on the FEVEROUS validation set.

With $\mathcal{L}_{\text{route}}$, Pearson correlation rises to $\rho = 0.52 \pm 0.03$ from 0.19 ± 0.04 without the loss; Precision@1 (top element agreement) increases from 0.54 to 0.75, and Precision@3 from 0.61 to 0.81.

These results show that task supervision alone induces weak overall alignment ($\rho = 0.19$), though it identifies highly important elements with moderate success ($P@1 = 0.54$). $\mathcal{L}_{\text{route}}$ substantially improves both correlation metrics and top-k consistency. The stronger correlation indicates that routing primarily enforces rank-order agreement rather than exact magnitude matching, which is appropriate for importance-based selection. The moderate correlations ($\rho = 0.52$) reflect our design intent: routing loss provides regularisation toward consistency rather than strict enforcement, allowing task supervision to override the weak alignment signal when beneficial. This validates that $\mathcal{L}_{\text{route}}$ fulfils its intended purpose of guiding the model toward consistent importance attribution.

E Denoiser Distributions $\pi^{(1)}$ and $\pi^{(2)}$

Stage 1: $\pi^{(1)}$ (structure denoisers)

At step t in Stage 1, sample a corruption $c \sim \pi_t^{(1)}$:

Denoiser c	Prob. (first 50% steps)	Prob. (last 50% steps)
Header (permute/drop/paraphrase)	0.25	0.30
Cell (typos, units, numeric formats)	0.25	0.25
Subtable (row/col block masking)	0.20	0.20
Edge (graph/relational perturb.)	0.20	0.15
Extreme (compound heavy noise)	0.10	0.10

Magnitudes are uniformly sampled from task-independent ranges.

Stage 2b: $\pi^{(2)}$ (auxiliary robustness)

Sample $m \sim \pi_t^{(2)}$ per batch:

Aux. denoiser m	Prob. (first 50% steps)	Prob. (last 50% steps)
DMTR (decomposed multi-step reasoning)	0.35	0.25
TREaA (entity-aware attention consistency)	0.25	0.30
CRD (contrastive reasoning distillation)	0.25	0.30
Summary/Completion (content plan masking)	0.15	0.15

Auxiliary weight β decays cosine from $0.6 \rightarrow 0.2$ over Stage 2b; λ_{align} remains active.

F Auxiliary Denoisers: Implementations and Losses

Our framework incorporates a family of auxiliary denoisers designed to promote robustness, encourage diverse reasoning behaviours, and unify supervision across table–text modalities. All denoisers conform to a standardised interface, allowing them to be seamlessly integrated into staged training. Below, we summarise their objectives and training signals.

DocNearTable The auxiliary loss focuses on document spans near tables. It applies causal language modelling over nearby textual passages, conditioning on masked table inputs. This encourages the model to leverage local textual context when reasoning about tables, yielding standard next-token prediction losses.

DMTR (Decomposed Multi-step Table Reasoning). This denoiser constructs arithmetic or program-style questions over multiple numeric cells, requiring the model to combine evidence across several elements. Supervision is given through an answer sequence trained under a language modelling loss, with UL2-style masking applied to relevant table tokens. The objective strengthens the model's capacity for compositional reasoning and multi-hop aggregation, which is especially useful for SQL-like tasks, without relying on explicit intermediate program annotations.

TREaA (Entity-aware Attention). This denoiser generates prompts that highlight entity-centric structures, such as all cells belonging to a row or semantically linked groups. Training encourages the model to recover consistent groupings across these entities, thereby promoting stable attention patterns over related table elements. Explicit regularisation of attention coherence remains part of Stage 2a, while TREaA contributes a complementary denoising signal expressed as textual supervision.

CRD (Contrastive Reasoning Distillation). CRD sharpens discrimination between genuine and spurious reasoning chains by injecting valid and adversarial/spurious evidence and training the model to prefer the former. Concretely, we form paired contexts (S^+, S^-) by perturbing rows, links, or linked sentences to create plausible

but incorrect variants; the objective encourages a higher likelihood for S^+ than for S^-, reducing reliance on shortcut cues and stabilising multi-hop aggregation under structural noise.

Summary/Completion Denoisers. For longer tables, the summary denoiser exposes only a prefix of the rows and requires the model to produce a textual synopsis of the masked portion. In contrast, the completion denoiser randomly masks a large portion of cells, tasking the model with reconstructing their content. Both combine sequence-level language modelling with masked-token supervision, training the system to generalise beyond partially observed structures, an ability directly relevant to slot-based compression.

G Complete Per-Task Configurations

Encoder and slot budgets (K). We explore $K \in \{8, 16, 32, 64, 128\}$ and choose the smallest K satisfying (i) $\leq$1 point loss vs K=128 on dev-tune and (ii) $<$ 2-point mean drop under perturbations.

Table 9: Detailed hyperparameters for each dataset.

Dataset	Encoder	d	Layers	Heads	K	Fusion
WikiSQL	Schema	768	12	12	16	Prefix
HybridQA	MGGT	768	12	12	64	Cross-4
FeTaQA	MGGT	768	12	12	64	Cross-4
FEVEROUS	HyTrel	768	12	12	64	Cross-4
TabFact	HyTrel	768	12	12	128	Cross-4
ToTTo	Schema	768	12	12	32	Prefix
WikiTQ	MGGT	768	12	12	64	Cross-4
StructQA	MGGT	768	12	12	64	Cross-4

Note: Cross-4 indicates cross-attention in the top 4 LLM layers. All tasks use batch size 256–512 and train for 1–1.5K steps in Stage 2b.

H Training Data and Dataset Construction

Summary. This section provides complete details on the datasets used for each training stage, addressing reproducibility requirements for our three-stage training pipeline.

H.1 Stage 1: Corruption-Aware Pretraining Data

For Stage 1 denoising pretraining, we use a corpus of **3 million tables** collected from:

- Wikipedia tables (extracted from English Wikipedia dump, March 2024)
- Web tables from Common Crawl (filtered for quality and English language)

These tables undergo the five corruption types described in Section 3.1 (header, subtable, cell, edge, extreme) to learn robust representations. No task-specific supervision is used at this stage; only reconstruction objectives are employed.

H.2 Stage 2a: Grounding and Alignment Data

Stage 2a uses a filtered subset of **1 million tables** from the Stage 1 corpus, augmented with grounding supervision. For each table, we:

1. Retrieve statements from the same Wikipedia page or pages linking to the table
2. Filter statements containing entities (Wikipedia URLs, numbers, or dates) that match table content
3. Create weak supervision signals by including matched entities as comma-separated plain text to guide slot-text alignment

This process creates approximately 3–5 grounding statements per table, enabling the weak supervision mechanism described in Section 3.2.

H.3 Stage 2b: Multi-Task Training Data

Stage 2b employs task-specific supervision through multi-task training on:

- **Seven benchmark datasets** from the TableLlama collection (Zhang et al., 2024): WikiTQ, HybridQA, FeTaQA, TabFact, FEVEROUS, ToTTo, WikiSQL
- **StructQA synthetic examples**: Following TAMO (Li et al., 2025), we generate 50K synthetic examples by:
  1. Sampling tables from Stage 2a corpus
  2. Generating compositional questions using templates
  3. Creating gold answers through programmatic execution