SchemaRAG: Dynamic Large Schema Reduction for LLM-driven Structured Information Extraction

Sin Yu Bonnie Ho* **Arlie Coles*** **Erik Larsson**

Eric Marshall **Nathan Bodenstab** **Paul Vozila**

Microsoft

Abstract

Extracting structured data from unstructured text using large language models (LLMs) becomes challenging when the target schemas are large and complex. In such cases, including the full schema in the prompt increases cost and latency, risks lost-in-the-middle performance degradation, and can exceed context length limits. We propose SchemaRAG, a retrieval-augmented generation (RAG) framework that dynamically prunes the output schema space for schema-conditioned information extraction tasks by leveraging schema metadata and few-shot examples (when available). We evaluate SchemaRAG on real-world healthcare and e-commerce datasets. Our results show that SchemaRAG can achieve up to an 8.8% increase in micro-F_1, a 47% reduction in latency, and a 48% reduction in token costs, demonstrating its practicality for large-schema extraction.

1 Introduction

Structured information extraction (IE) pairs values from unstructured text with schema-defined keys. In healthcare, this includes medical attribute extraction from unstructured clinical notes (Jiang et al., 2011; Agrawal et al., 2022) and accelerated human annotation of medical narrative data (Goel et al., 2023). Similarly, e-commerce applications focus on product attribute value extraction from unstructured text descriptions with and without supplementary multimodal data (Brinkmann et al., 2024; Zhu et al., 2020).

Further applications of structured IE exist for problems that can be cast as form-filling, where models need to populate complex, hierarchical schemas from unstructured text. These forms could have numerous rows, and each row could have complex constraints on potential values (*e.g.*, required types including strings, numbers, single-select radio buttons, multi-select checkboxes, or other custom types). In this work, we consider two such use cases: 1) populating electronic health records (EHRs) from medical narratives and 2) filling a product metadata form from e-commerce product description text.

Instruction-tuned large language models (LLMs) are well-suited for these tasks due to their broad semantic knowledge (Singhal et al., 2023) and ability to follow structured prompt instructions (Liu et al., 2024b). However, naively injecting large schemas into the prompt can exceed context limits or introduce many irrelevant rows. This leads to lost-in-the-middle effects (Liu et al., 2024a) and performance degradation as the model struggles to isolate pertinent rows. The resulting latency and inference costs can preclude real-time applications.

To mitigate these issues, we propose SchemaRAG, a retrieval-augmented generation approach that retrieves and prunes the output schema space. By leveraging schema metadata and, when available, limited labeled data, SchemaRAG dynamically selects a relevant schema subset for just-in-time in-context prompted LLM use (Figure 1). A well-selected reduced schema shrinks prompt size, decreases the chance of the LLM filling in irrelevant rows, and improves overall extraction performance.

The framework is schema-agnostic and supports arbitrary levels of hierarchy. Because the embeddings driving the retrieval can be computed offline, the system is efficient and easily integrated into existing pipelines. Crucially, SchemaRAG operates as a training-free solution, requiring no pretraining or fine-tuning. We explicitly focus on schema-conditioned extraction using hosted LLMs under real-world deployment scenarios including large, complex schemas and strict resource budgets. By utilizing prompting rather than retraining, our approach allows for rapid iteration and immediate

*Equal contribution. Corresponding authors: siho@microsoft.com, arliecoles@microsoft.com

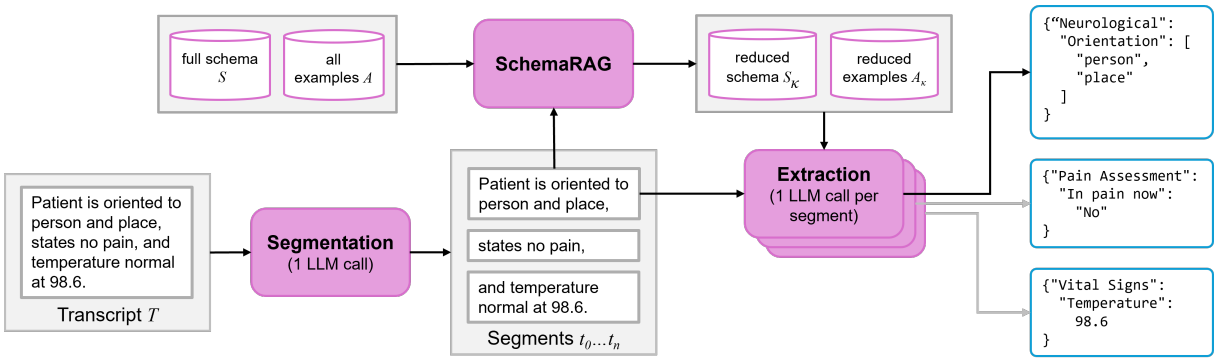

Figure 1: Overview of SchemaRAG which extracts information by segmenting unstructured text and applying schema reduction to each segment. These segments, alongside their reduced schemas and any retrieved examples, are then passed to a just-in-time prompted LLM query for extraction.

adaptability to evolving requirements.

The contributions of the paper are as follows: 1) We propose SchemaRAG—a RAG-based method to reduce the solution space for LLM-driven structured information extraction tasks with large, complex output schemas. 2) We publish experimental results showing that our method delivers up to an 8.8% relative micro-F_1 improvement on two real-world, large-schema datasets. Latency and token cost reduce by up to 47% and 48% respectively, though outcomes vary by dataset.

2 Related Work

LLMs have been widely used for IE tasks including named entity recognition (NER), relation extraction (RE), and attribute value extraction (AVE) (Dagdelen et al., 2024; Wang et al., 2023; Wei et al., 2024; Brinkmann et al., 2024; Zou et al., 2025; Wu et al., 2024). Recent retrieval-based enhancements further improve IE by augmenting prompts with relevant context. Retrieval-augmented generation (RAG) (Lewis et al., 2020) refers to techniques that augment LLM prompts with information retrieved from external knowledge sources. A common use case is the dynamic selection of relevant text-based document chunks or few-shot examples for conditioning (Ma et al., 2025; Li et al., 2024).

A parallel line of research retrieves already populated data structures (*e.g.*, tables or databases) and injects subsets into the prompt of a downstream task (Chen et al., 2024; Sui et al., 2024; Lin et al., 2023; Ye et al., 2023; Wang et al., 2024). Prior table-centric methods assume available cell values and operate over filled tables to answer questions or perform reasoning. Unlike table-centric QA over populated tables, our setting is schema-conditioned form-filling with a large, unpopulated schema to be populated from free text. Furthermore, we argue that traditional IE benchmarks are unsuitable for evaluating SchemaRAG because their rigid span-based scoring penalizes semantically correct LLM extractions that happen to differ from the dataset's specific tagging biases.

These challenges highlight an underexplored area in existing RAG literature: current retrieval strategies do not address the challenge of dynamically selecting relevant portions of a complex, structured output schema to guide LLM-based extraction. Our SchemaRAG method applies retrieval to the output schema itself. SchemaRAG narrows the prompt to a relevant subset of a large, hierarchical schema by retrieving: i) a subset of candidate schema rows based on interpolated row metadata and ii) relevant schema-row-labeled text examples (when available). The retrieved rows compose the schema subset injected into the prompt; associated labeled data (if any) serve as few-shot examples.

3 Methodology

To perform structured data extraction of key-value pairs on a transcript T of unstructured text, extracted pairs must conform to some schema S. We define S as composed of m rows, where each row r_i has an identifier i, a string name n_i, and a subschema describing its possible values v_i. The value subschemas of S can be arbitrary, according to the types required by the domain (Table 1).

In a complex schema, there may exist arbitrary levels of hierarchy that form a path to any row. In this work we consider 2-level schemas, where each row may also belong to any number of categories C_i, with each category having a string name c_j. This structure is common in EHR systems, but our method is generalizable to schemas with any

Table 1: Example value subschemas for different row types.

Type	Value subschema (v_i)
Single	`{"type": "string", "enum": ["Yes", "No"]}`
Multi	`{"type": "array", "enum": ["A", "B", "C"]}`
Numeric	`{"type": "number"}`

Table 2: Example schema S containing one Single row r_i, in JSON Schema format.

```
{"type": "object",
 "properties": {
   "category (c_j)": {
      "type": "object",
      "properties": {
        "row (n_i)": {
          "type": "string",
          "enum": ["A", "B", "C"]
}}}}}
```

number of hierarchical levels.

S can be represented in string form as a JSON Schema (Table 2). In the simplest case, S and T can be used directly as in-context input to a prompted LLM query to perform the extraction. We only use the JSON Schema representation in the prompt, since LLMs may more readily fill this standard specification format than others (Brinkmann et al., 2024). When annotated examples are available, they can be included in the prompt as in-context learning material. In this work, we aim to reduce S to S_κ, a smaller set of $\kappa \ll m$ most relevant rows, in order to both improve performance on the structured extraction task and avoid the problems associated with using a very large S in context.

3.1 Schema reduction

Our schema reduction method leverages two sources of row information: the metadata of the rows themselves, and, when available, example transcripts annotated by row. For each row $r_i \in S$, we create a row embedding $e_{\text{row}i}$ by normalized weighted interpolation of embedded row metadata M_i, using a pretrained embedder H (Eq. 1). Intuitively, this is a weighted average over the embeddings of each available metadata field for the row, where the weights control the relative importance of different metadata types. The contents of M_i are configurable based on available metadata, but minimally include the row name n_i, a string concatenation of the row's category names, C_i', and (for Single or Multi rows) a string concatenation of possible values from the row's value subschema, v_i'. We treat each weight w_j as a hyperparameter to be tuned by random or other search.

$$e_{\text{row}i} = \sum_{j=1}^{|M_i|} w_j \cdot H(m_{ij}) \quad \forall m_{ij} \in M_i = \{n_i, C_i', v_i', ...\} \tag{1}$$

When a set of annotated examples A is available, we also create an example embedding $e_{\text{ex}i}$ for each example $a_i = (x_i, y_i) \in A$, where x_i is the unstructured text input and y_i is the annotated structured output. This example embedding is simply the embedding of the text input, $H(x_i)$. Both row and example embeddings are within the same vector space, enabling direct similarity comparison.

We compute a relevance score ρ_i of each embedding $e_i \in E = E_{\text{row}} \cup E_{\text{ex}}$ as its cosine similarity to the embedding of the transcript, $H(T)$. We build the reduced schema S_κ by examining the embeddings sorted by descending ρ. For each embedding e_i, if it is a row embedding, we add the corresponding row to S_κ. If it is an example embedding, we add the corresponding example's annotated rows from y_i to S_κ. When k embeddings have been examined, we stop. This k-nearest neighbors search in the embedding space may lead to more or fewer than k rows in S_κ, since multiple embeddings may map to the same row and example annotations may include multiple rows.

This process, visualized in Figure 2, yields a reduced schema $S_\kappa \subseteq S$ that is sorted by relevance to the transcript T and can be converted to JSON Schema format for use in-context when performing the extraction task. An example of S_κ and its JSON Schema representation as presented to the LLM given the transcript T is provided in Appendix B.

3.2 Transcript segmentation

In domains with long unstructured text transcripts, LLMs may struggle to locate all extractable content across extensive text. If the SchemaRAG hyperparameter k is too large, the resulting schemas may cause lost-in-the-middle effects, negating the benefits of schema reduction. Conversely, a k that is too small may fail to cover all relevant information. To counteract these issues, we experiment with segmenting the transcript into n segments, $t_1, t_2, ..., t_n$, then run SchemaRAG on each segment to obtain per-segment reduced schemas

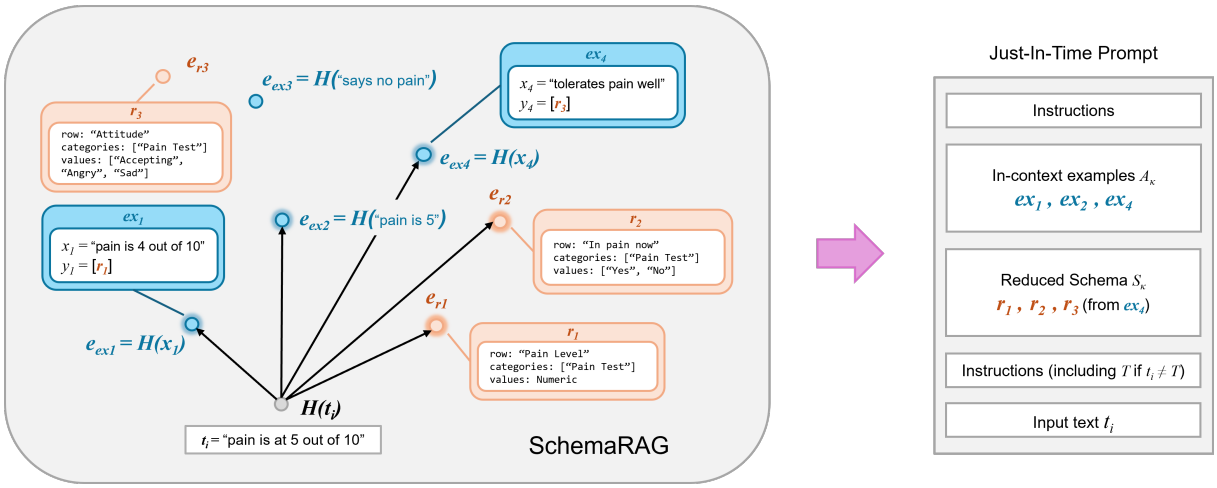

Figure 2: Schema reduction by SchemaRAG (with $k = 5$). The five closest embeddings to $H(t_i)$ are retrieved. Rows whose embeddings are found are added directly to the reduced schema, and examples whose embeddings are found have their annotated rows added to the reduced schema. The reduced schema is then converted to JSON Schema format for use in the extraction prompt, and the found examples are added as in-context learning material.

$S_{\kappa_1}, S_{\kappa_2}, ..., S_{\kappa_n}$, and finally run n extractions, with each segment t_i and its corresponding reduced schema S_{κ_i} injected into the prompt of each extraction LLM query. Segmentation is performed by a prompted LLM call containing the transcript T and instructions describing how to segment.

4 Experiments

4.1 Datasets

Nursing. We evaluate our method on four proprietary test sets from Hospitals A–D (together the Nursing dataset), each containing between 48–50 transcripts of human-provided, de-identified nurse rounding dictations. Each transcript is human-annotated with a set of key-value pairs from per-hospital schemas S_A–S_D respectively. No row in any schema is present in any other schema. All four hospital schemas use ten row types, including Single, Multi, Numeric, and other proprietary custom types.

For each hospital, we also curate example sets, A_A–A_D, linking key-value pairs with their associated transcript provenance. We examine the effect of these example sets on our schema reduction method and on the downstream extraction task as in-context learning material. The breakdown of the size and content of Nursing can be found in Appendix A.

Amazon. For variety in domain, we also evaluate our method on a subset of the publicly available sample of the Bright Data Amazon Products dataset (Bright Data, 2025). From 1000 scraped Amazon product webpages, we sample 48 products and create a structured extraction task by assembling a transcript for each product through concatenation of the title, description, top review, and key prose attributes. We construct schema rows and annotations through this process and manually verify that annotated key–value pairs are supported by the transcript. The resulting test set contains 804 annotated row–value pairs (554 unique rows). From the full 1000-product sample, we assemble a schema S_{Am} with 1906 rows spanning four row types (Single, Multi, Numeric, Weight). All annotated rows in the test set appear in this schema. Amazon transcripts are roughly 10× longer than Nursing transcripts. Full details of the test set and schema construction process are provided in Appendix A.

4.2 Models and hyperparameters

For all the experiments, we use OpenAI's `text-embedding-ada-002` (OpenAI, 2022) as the embedder H and `gpt-4o-2024-08-06` (GPT-4o) (OpenAI, 2024) for all segmentation and extraction calls. We select GPT-4o for its balance of performance, latency and cost, and because it is currently production-stable in our deployment setting. We also assess backbone variability with `gpt-4.1-2025-04-14` (GPT-4.1) (OpenAI, 2025) and alternative embedders. GPT-4o is run with temperature 0. The SchemaRAG hyperparameter k is fixed to 60, and all row-weights w in Eq. 1 are held constant (Appendix C). We repeat each main experiment three times.

Prompts follow the structure shown in Fig. 2,

Table 3: Overall extraction accuracy. Statistical significance of macro-F_1 improvements over the full schema baseline is indicated by * ($p < 0.05$) and † ($p < 0.001$).

	Nursing		Amazon	
	micro-F_1 ±CV	macro-F_1 ±CV	micro-F_1 ±CV	macro-F_1 ±CV
Full schema	0.844 ± 0.29%	0.851 ± 24.7%	0.471 ± 3.12%	0.481 ± 47.8%
+ segmentation	0.880 ± 0.54%	0.856* ± 24.1%	0.423 ± 1.23%	0.451 ± 32.4%
SchemaRAG	**0.918** ± 0.22%	**0.899**† ± 22.7%	**0.510** ± 0.87%	**0.515** ± 34.4%
+ *oracle*	*0.952 ± 0.05%*	*0.951*† *± 13.9%*	*0.775 ± 0.35%*	*0.788*† ± 14.8%

with domain-specific tokens adjusted minimally for each dataset. When an input exceeds GPT-4o's 128k-token context window, we apply a truncation procedure that prunes schema rows until the prompt fits; the full algorithm is detailed in Appendix C.

4.3 Metrics

Accuracy. We report an F_1 score using true/false positives and false negatives defined for the structured IE task. Multi rows are evaluated per value ("unrolled"). Invalid or out-of-schema outputs are discarded, and type-coercible values are normalized (*e.g.* "4" → 4 for a Numeric row) before scoring. Scores are averaged over three runs.

Latency and token use. We report average per-transcript latency (in seconds) and average per-transcript token counts (separately for prompt/input and completion/output). For SchemaRAG, latency includes one segmentation call plus the maximum extraction-call duration (parallel execution).

Significance testing. To evaluate the statistical significance of differences in F_1, latency, and token usage, the Wilcoxon signed-rank test (Wilcoxon, 1992) is applied to per-transcript results. We treat $p < 0.05$ as statistically significant. Full metrics details can be found in Appendix D.

4.4 Baseline and oracle

We compare SchemaRAG on Nursing and Amazon against two baselines: i) a full-schema prompt and ii) a full-schema+segmentation prompt. The SchemaRAG condition represents the "maximal" form of our method, which always uses segmentation and uses examples, when available (*i.e.*, for Nursing). The full-schema baseline sorts rows by relevance only to include as many rows as fit into context, whereas SchemaRAG selects a small, targeted top-k as its retrieval set to guide extraction. To avoid understating the baseline, we retain this relevance sort. Other sorting-method comparisons and additional baselines (including a lightweight table-inspired baseline that retrieves by row name only) are provided in Appendix E. We also provide an oracle upper bound for the SchemaRAG method by constructing the reduced schema directly from the reference annotated rows.

4.5 Extraction accuracy

Nursing F_1 results across all four hospital datasets are calculated as a micro-F_1 score. We also report macro-F_1 scores by averaging per-transcript F_1 scores across the four datasets, and use them in statistical significance testing.

Across both Nursing and Amazon, SchemaRAG outperforms all baselines in micro-F_1 (Table 3), with relative gains of 8.8% on Nursing and 8.3% on Amazon. These improvements persist with a GPT-4.1 backbone and other embedders on Nursing (Appendix F). On Amazon, segmentation alone improves recall (0.367 → 0.641) but degrades precision (0.661 → 0.316). When combined with SchemaRAG, we observe a recovery in precision (0.316 → 0.467), resulting in a more balanced performance profile and the highest F_1. The extraction accuracy on Amazon is consistently lower compared to Nursing, likely due to several dataset-specific factors we discuss in Section 6. Finally, while oracle results significantly outperform SchemaRAG on both datasets, they establish a high theoretical ceiling for schema reduction and validate the potential for further refinements to this approach.

4.6 Latency and token use

Computing latency. In general, SchemaRAG reduces or maintains latency (Table 4). Compared to the baseline, SchemaRAG introduces three differences impacting latency: i) the added latency of segmentation; ii) parallel extraction of n segments, with total latency bounded by the longest segment's extraction time; and iii) the substantially reduced size of the schema used in extraction prompts.

Table 4: Average per-transcript latency and token use. Statistical significance of each condition over the previous condition is indicated by † ($p < 0.001$).

	Lat. (s) ± CV	Tok. in ± CV	Tok. out ± CV	Total tok.
	Nursing			
Full-schema	11.3 ± 61.4%	77,382 ± 27.8%	**212** ± 134.5%	77,594
SchemaRAG	**6.0**† ± 68.9%	**38,707**† ± 137.1%	421† ± 142.2%	**39,128**
+ *oracle*	*4.0*† ± *82.9%*	*31,721*† ± *169.1%*	*415*† ± *142.6%*	*32,136*
	Amazon			
Full-schema	**16.4** ± 27.6%	**122,095** ± 2.3%	**147** ± 72.0%	**122,243**
SchemaRAG	17.7 ± 35.9%	170,929† ± 44.7%	2,786† ± 43.2%	173,714
+ *oracle*	*13.0*† ± *38.7%*	*107,004*† ± *87.6%*	*3,766*† ± *51.5%*	*110,771*

Despite segmentation's additional latency, SchemaRAG is 47% faster than the baseline on Nursing, owing to the far smaller number of extraction input tokens (Table 4) and the reduction in longest-parallel-extraction output tokens (here reducing from 212 to 52). Our Amazon experiments show slightly higher latency for SchemaRAG relative to the baseline, though the effect is not significant. SchemaRAG's $O(T)$ segmentation call (Appendix G) dominates latency due to Amazon's larger transcripts. Thus, despite reduced schema use and extraction parallelization, SchemaRAG yields no reduction in latency on this dataset. This difference demonstrates the impact of scaling transcript size on SchemaRAG: while F_1's trajectory is unaffected by an increasing transcript length, latency may plateau or increase.

Computing token use. SchemaRAG reduces token cost for Nursing but raises it for Amazon (Table 4). Although OpenAI charges 4× more for output tokens than input tokens (OpenAI, 2024), input tokens dominate token cost in our experiments, exceeding output volume by two to three orders of magnitude. On Nursing, SchemaRAG uses roughly twice the output tokens of the baseline, nearly reaching the oracle count, while halving input tokens. This suggests the baseline undergenerates, likely due to lost-in-the-middle effects. SchemaRAG's mitigation yields a 50% reduction in total token use and 48% in token cost.

On Amazon, SchemaRAG's segmentation generates more parallel extraction calls due to larger average transcript sizes. Because each per-segment prompt includes the full transcript for context (Figure 2), transcript size T becomes a decisive factor for token complexity, which can be modeled as $O(T^2+TS_{\kappa i}+TA_i)$ (Appendix G). Consequently, SchemaRAG's input tokens exceed the baseline by about 40%. SchemaRAG also yields more output tokens, approaching the oracle condition. This suggests that the baseline also undergenerates and that SchemaRAG mitigates this issue.

Nevertheless, given SchemaRAG's parallelization of extraction and reduced schema size, we anticipate SchemaRAG could remain competitive in latency and cost for datasets with moderately sized T.

4.7 Ablations

Since SchemaRAG is a flexible, modular method, we conduct ablations to assess the impact of different components on extraction performance. We analyze the effects of using i) supplementary examples, ii) row embeddings and example embeddings, and iii) segmentation scope. Guidance from labeled examples, especially when used in both retrieval and prompting, consistently improves performance. Combining row and example embeddings yields the best macro-F_1 (0.899 vs. 0.888 row-only and 0.798 example-only). Segmentation primarily helps retrieval (Nursing: 0.898 vs. 0.774 baseline; Amazon: 0.554 vs. 0.484 baseline).

We also examine the effect of the hyperparameter k. Performance generally improves as k increases until saturating, with k=60 providing a strong operating point in our experiments. SchemaRAG also attains consistently higher recall@k than the full-schema+segmentation baseline, approaching the upper bound faster (*e.g.*, recall@5: 0.800 vs. 0.749 baseline; recall@60: 0.991 vs 0.981 baseline). Full ablation results are in Appendix H.

5 Conclusion

SchemaRAG is a retrieval-augmented generation method for structured information extraction that dynamically selects a relevant subset of a complex schema based on unstructured input text. Evaluations on two real-world, large-schema datasets show that SchemaRAG improves accuracy over full-schema baselines and remains competitive in latency. While its impact on token cost depends on transcript size, oracle experiments show that better accuracy can be achieved while using fewer tokens. This validates the promise of more precise subschema selection as a method, positioning SchemaRAG as a practical step toward efficient and accurate LLM-based structured information extraction.

6 Limitations

We observe consistently lower extraction accuracy on the Amazon dataset compared to the Nursing dataset. This difference likely reflects several dataset-specific factors. The Nursing dataset uses schemas designed *a priori* by hospitals to support real EHR workflows, paired with naturally occurring nurse dictations and labeled in-domain examples. In contrast, the Amazon dataset was manually constructed from noisier source data. Preparing the Amazon dataset for structured information extraction required substantial cleaning and transformation (Appendix A.2), and the resulting schema rows and gold labels are therefore likely imperfect. Moreover, unlike the Nursing dataset, we do not have access to additional labeled examples to assess the effect of label quality or data scale, which ablation results on Nursing suggest could be beneficial (Appendix H.1). Finally, Amazon input transcripts were created by concatenating product descriptions and related metadata, resulting in longer, more diverse, and multi-topic inputs than nurse dictations. While SchemaRAG's segmentation strategy was held constant across datasets for consistency, it was not exhaustively optimized for this setting. These factors likely contribute to the reduced performance observed on the Amazon dataset.

Ethical Considerations

Our clinical experiments use de-identified, automatically transcribed nursing speech; transcription and labeling errors may propagate to extracted structures. We suggest a human-in-the-loop deployment model requiring clinician review and robust audit trails. With respect to data privacy, we do not release any clinical data or prompts that could reveal protected information; e-commerce data is derived from a publicly available sample and is used only for internal evaluation.

References

Monica Agrawal, Stefan Hegselmann, Hunter Lang, Yoon Kim, and David Sontag. 2022. Large language models are few-shot clinical information extractors. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1998–2022, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Bright Data. 2025. Amazon Products. Accessed: 2025-04-24.

Alexander Brinkmann, Roee Shraga, and Christian Bizer. 2024. ExtractGPT: Exploring the potential of large language models for product attribute value extraction. In *Information Integration and Web Intelligence: 26th International Conference, IiWAS 2024, Bratislava, Slovak Republic, December 2–4, 2024, Proceedings, Part I*, page 38–52, Berlin, Heidelberg. Springer-Verlag.

Si-An Chen, Lesly Miculicich, Julian Martin Eisenschlos, Zifeng Wang, Zilong Wang, Yanfei Chen, Yasuhisa Fujii, Hsuan-Tien Lin, Chen-Yu Lee, and Tomas Pfister. 2024. TableRAG: Million-Token Table Understanding with Language Models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

John Dagdelen, Alexander Dunn, Sanghoon Lee, Nicholas Walker, Andrew S. Rosen, Gerbrand Ceder, Kristin A. Persson, and Anubhav Jain. 2024. Structured information extraction from scientific text with large language models. *Nature Communications*, 15(1):1418.

Akshay Goel, Almog Gueta, Omry Gilon, Chang Liu, Sofia Erell, Lan Huong Nguyen, Xiaohong Hao, Bolous Jaber, Shashir Reddy, Rupesh Kartha, Jean Steiner, Itay Laish, and Amir Feder. 2023. LLMs Accelerate Annotation for Medical Information Extraction. In *Proceedings of the 3rd Machine Learning for Health Symposium*, volume 225 of *Proceedings of Machine Learning Research*, pages 82–100. PMLR.

Min Jiang, Yukun Chen, Mei Liu, S Trent Rosenbloom, Subramani Mani, Joshua C Denny, and Hua Xu. 2011. A study of machine-learning-based approaches to extract clinical entities and their assertions from discharge summaries. *Journal of the American Medical Informatics Association*, 18(5):601–606.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020.

Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Yihao Li, Ru Zhang, and Jianyi Liu. 2024. An enhanced prompt-based llm reasoning scheme via knowledge graph-integrated collaboration. In *Artificial Neural Networks and Machine Learning – ICANN 2024*, pages 251–265, Cham. Springer Nature Switzerland.

Weizhe Lin, Rexhina Blloshmi, Bill Byrne, Adria de Gispert, and Gonzalo Iglesias. 2023. An inner table retriever for robust table question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9909–9926, Toronto, Canada. Association for Computational Linguistics.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Yu Liu, Duantengchuan Li, Kaili Wang, Zhuoran Xiong, Fobo Shi, Jian Wang, Bing Li, and Bo Hang. 2024b. Are LLMs good at structured outputs? A benchmark for evaluating structured output capabilities in LLMs. *Information Processing & Management*, 61(5):103809.

Chuangtao Ma, Sriom Chakrabarti, Arijit Khan, and Bálint Molnár. 2025. Knowledge Graph-based Retrieval-Augmented Generation for Schema Matching. *Preprint*, arXiv:2501.08686.

OpenAI. 2022. New and improved embedding model. Accessed: 2026-02-12.

OpenAI. 2024. API Pricing. Accessed: 2026-02-12.

OpenAI. 2024. Gpt-4o: Large multimodal model. v. gpt-4o-2024-08-06. Accessed: 2026-02-12.

OpenAI. 2025. Gpt-4.1 technical report. v. gpt-4.1-2025-04-14. Accessed: 2026-02-12.

Samuel Sanford Shapiro and Martin B Wilk. 1965. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3-4):591–611.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Abubakr Babiker, Nathanael Schärli, Aakanksha Chowdhery, Philip Mansfield, Dina Demner-Fushman, and 13 others. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

Yuan Sui, Jiaru Zou, Mengyu Zhou, Xinyi He, Lun Du, Shi Han, and Dongmei Zhang. 2024. TAP4LLM: Table Provider on Sampling, Augmenting, and Packing Semi-structured Data for Large Language Model Reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10306–10323, Miami, Florida, USA. Association for Computational Linguistics.

Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuansen Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, Jihua Kang, Jingsheng Yang, Siyuan Li, and Chunsai Du. 2023. InstructUIE: Multi-task Instruction Tuning for Unified Information Extraction. *Preprint*, arXiv:2304.08085.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and 1 others. 2024. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *arXiv preprint arXiv:2401.04398*.

Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, Yong Jiang, and Wenjuan Han. 2024. ChatIE: Zero-Shot Information Extraction via Chatting with ChatGPT. *Preprint*, arXiv:2302.10205.

Frank Wilcoxon. 1992. Individual comparisons by ranking methods. In *Breakthroughs in statistics: Methodology and distribution*, pages 196–202. Springer.

Haolun Wu, Ye Yuan, Liana Mikaelyan, Alexander Meulemans, Xue Liu, James Hensman, and Bhaskar Mitra. 2024. Learning to Extract Structured Entities Using Language Models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6817–6834, Miami, Florida, USA. Association for Computational Linguistics.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*, pages 174–184.

Tiangang Zhu, Yue Wang, Haoran Li, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2020. Multimodal Joint Attribute Prediction and Value Extraction for E-commerce Product. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2129–2139, Online. Association for Computational Linguistics.

Yi Zou, Mengying Shi, Zhongjie Chen, Zhu Deng, Zongxiong Lei, Zihan Zeng, Shiming Yang, Hongxiang Tong, Lei Xiao, and Wenwen Zhou. 2025. ESGReveal: An LLM-based approach for extracting structured data from ESG reports. *Journal of Cleaner Production*, 489:144572.

A Datasets

A.1 Nursing

Table 5 and Figure 3 give a breakdown of the size and content of the Nursing dataset.

Table 5: Nursing dataset: total and unique row counts for schemas, test sets, and example sets.

Hospital	Sch.	Test		Examples	
		tot.	un.	tot.	un.
A	1064	739	133	1816	149
B	1118	308	108	881	210
C	601	741	169	2176	170
D	1261	551	166	1424	192

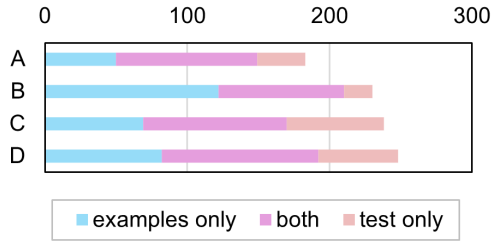

Figure 3: Counts of unique rows that appear in each Hospital's test set, example set, or both.

A.2 Amazon

The Amazon dataset used in this work (the "Subset") is a subset of the publicly available sample of the Bright Data Amazon Products dataset (Bright Data, 2025) (the "Public Sample"). While the Public Sample is clearly labeled as publicly downloadable, it does not come with a named license. Therefore, out of an abundance of caution and wish to abide by any existing copyright restrictions, we do not release our Subset derived from the Public Sample. We instead describe our process for internally creating the Subset here.

The Public Sample consists of scraped data from 1000 Amazon product webpages, including various prose features and a set of annotated key-value pairs for each product. To transform this data into a schema and test set suitable for structured information extraction, we follow the steps outlined below.

Schema preparation. In the Public Sample, each product has a list of annotated groups $G = \{g_0, ..., g_n\}$ and a set of annotated key-value pairs. We form the schema S_{Am} by creating a distinct row for each unique g_0–key combination, taken from across all 1000 products. Possible values for each of these rows are collected from every instance of the g_0–key combination in the Public Sample. Each row's categories C are the set of all groups from the products where the row's g_0–key combination is present.

After assembling possible values for every row in this way, closer attention is needed. We discard all rows with metadata not in English. For rows with values containing commas, semicolons, or other obviously delimited lists of choices, we split each delimited string into its own possible value and schematize the row as a Multi row. Rows that, on manual inspection, indicate numeric values or weights are schematized as Numeric or Weight rows, respectively. All remaining rows are schematized as Single rows.

Single rows for which the above process yielded only one possible option are either "hydrated" when reasonably completable (*e.g.*, a Single row only containing a "yes" option is also supplied a "no" option) or discarded.

Test set preparation. We randomly sample a maximum of two products per annotated "department" key, resulting in a total of 48 products in the Subset. For each of these products, we prepare a transcript by concatenating the product title, description, top review, product dimensions, weight, manufacturer, and bullet point prose features into a single string.

We also manually review the annotated key-value pairs (excluding "department") to ensure they can be substantiated by the transcript content. Key-value pairs that cannot be reasonably inferred from the transcript, such as serial numbers or model names that are never mentioned, are discarded from the annotations. When an option from the schema is clearly available and applicable to the transcript, we add it to the annotations. For example, if the transcript mentions a product is "red and blue," and there exist "red" and "blue" Multi options in the appropriate schema row, but the original annotated value was "red" alone, we add "blue" to the values for that annotated key-value pair. Annotations that are plainly inconsistent with the transcript are corrected. For example, if the transcript mentions a quantity of "3" and the original annotated value was "2," we correct the annotation to "3." We also remove any duplicate key-value pairs from the annotations.

B Schema representation

An example of a reduced schema S_κ that SchemaRAG might produce given the transcript T *"pain is at 5 out of 10"*:

```
{
  "reduced_rows": [
    {
      "id": "0003",
      "type": "Number",
      "row": "Pain Level",
      "category": ["Pain Test"]
    },
    {
      "id": "0001",
      "type": "Single",
      "row": "In pain now",
      "category": ["Pain Test"],
      "values": ["Yes", "No"]
    }

}
```

A corresponding example of that reduced schema in JSON Schema format, which we present to the LLM for filling:

```
{
  "type": "object",
  "properties": {
    "Pain Test": {
      "type": "object",
      "properties": {
        "Pain Level": {
          "type": "number"
        },
        "In pain now": {
          "type": "string",
          "enum": ["Yes", "No"]
        }
      }
    }
  }
}
```

C Models and hyperparameters

C.1 Row weights

All weights w in Eq. 1 are fixed throughout and were tuned by grid search on preliminary proprietary held-out test sets drawn from Hospitals A–D; the values of w and metadata to which they apply are proprietary, but include weighting a row's name, category(ies), and possible value(s).

C.2 Prompt truncation method

To ensure that the prompt length stays within the input token limit, we apply a truncation strategy that dynamically adjusts the number of schema rows to be included in the prompt in the full schema experiments (with no examples). The procedure is as follows:

1. First, construct the full prompt using all schema rows and compute the total token count, including the instruction prompt and the schema (in JSON Schema format as in Table 2). If the token count is already within the limit, return the prompt as-is.

2. Otherwise, compute a relevance score ρ for each row (*i.e.*, cosine similarity to the input), and sort the rows in descending order of relevance so that the least relevant rows can be removed from the end of the list during truncation.

3. Given the sorted list of rows, initialize two pointers: `left` at 0 and `right` at the total number of rows in the schema.

4. While `left` $\leq$ `right`, compute the midpoint index `mid`.

5. Construct a temporary prompt using only the first `mid` rows and compute the total token count, including the instruction prompt and schema (in JSON Schema format as in Table 2).

6. If the total token count is within the limit, update `left = mid + 1` to try including more rows.

7. If the token count exceeds the limit, update `right = mid - 1` to reduce the number of rows.

8. Repeat until the maximum valid number of matches is found. The final prompt is then constructed using this schema subset.

D Metrics

D.1 Accuracy computation

We compute F_1 from true positives (TP), false positives (FP), and false negatives (FN) defined at the

row–value level. For Multi rows, each predicted value is treated independently ("unrolled") to avoid crediting extra unattested values. LLM responses that cannot be mapped to any schema row or valid value type are marked invalid and excluded from scoring. When a predicted value is type-coercible (*e.g.*, the string "4" for a Numeric row), we normalize it before comparison. All F_1 results are averaged over three independent runs, with coefficients of variation computed across runs.

D.2 Latency and token accounting

We measure per-transcript latency and token usage separately for (i) prompt/input tokens and (ii) completion/output tokens. For the baseline (no segmentation), latency is simply the extraction call duration. For SchemaRAG, we measure (a) the segmentation call and (b) the n extraction calls; because extraction calls run in parallel, the effective extraction latency is the maximum of these n durations. Final reported latency and token counts are averaged over three runs for each experimental condition, also with coefficients of variation.

D.3 Significance testing

To assess whether differences in F_1, latency, or token usage are statistically reliable, we apply the Wilcoxon signed-rank test (Wilcoxon, 1992) to per-transcript metrics, averaged over three runs. This non-parametric test is chosen because Shapiro–Wilk tests (Shapiro and Wilk, 1965) indicate that per-transcript distributions significantly deviate from normality, often exhibiting skewness or heavy tails. Unless otherwise stated, we treat $p < 0.05$ as statistically significant.

E Baselines

E.1 Baseline sorting controls and results

The baseline used in this work sorts the rows by relevance before it estimates how many rows are able to fit in the prompt, then truncates those that do not fit. Unlike SchemaRAG, which uses a small, targeted k, the baseline aims to completely fill the prompt up with as many rows from the full schema as possible, stopping only when the context limit is reached.

The relevance sort is a pragmatic measure that is especially necessary for the Amazon dataset, whose schema is very large. (An average of 86% of its rows must be truncated for final extraction prompts to fit in the context.) If we were to truncate this schema without sorting on relevance, the chances of any correct row surviving truncation is low, because the schema comes by default in an arbitrary order. This would lead to artificially low performance and would not be an interesting condition. To ensure consistency in method, the same sorting approach for Nursing is applied, even though its full schemas fit in context.

Table 6: Comparison of schema sorting methods on macro-F_1 performance (run 3×).

Sorting Method	**macro-F_1**
As-is, arbitrary order	0.762
By JSON keys (alphabetical on hierarchy)	0.772
By relevance (proposed method)	**0.851***

* Statistically significant ($p < 0.001$) vs. other methods.

To compare the relevance sorting approach with other approaches, we ran additional full-schema experiments on Nursing. As shown in Table 6, the relevance-based sorting approach provides a more competitive baseline. Notably, SchemaRAG's relative gains would appear even more significant if compared against a baseline lacking this sorting method.

E.2 Additional baseline

We report an additional baseline here that is inspired from tabular, rather than text-based, RAG strategies. This method uses only each schema row's embedded name for retrieval, which is a lightweight grounding strategy somewhat analogous to TableRAG's "ReadSchema" approach (Chen et al., 2024) and minimally exploits the structured nature of the schema by using only one element of its metadata. We report micro-F_1 (averaged over three runs) on the Nursing and Amazon datasets (Table 7).

Table 7: Comparison of additional baselines.

Method	**Nursing**	**Amazon**
Full-schema	0.844	0.471
Row-name-only	0.800	0.341
SchemaRAG	**0.918**	**0.510**

SchemaRAG outperforms this lightweight table-inspired baseline as well as the full-schema, information-extraction-inspired baseline.

F Other backbone models

SchemaRAG is fully modular and requires no fine-tuning, so applying it to other LLMs is straightforward. To assess backbone variability, we replicate the Nursing experiments with `gpt-4.1-2025-04-14` (GPT-4.1) and report macro-F_1 (averaged over three runs). Results show that even with a more powerful and capable LLM, SchemaRAG significantly outperforms the full-schema baseline (Table 8). Even as context windows expand in state-of-the-art models, SchemaRAG offers continued value by steering model attention toward the most relevant schema elements. This improves extraction accuracy and mitigates "lost-in-the-middle" effects that can still arise even with larger context capacities.

Table 8: Nursing macro-F_1 (3 runs averaged) with GPT-4o vs. GPT-4.1. Asterisks indicate significance over the corresponding full-schema baseline ($p < 0.005$).

Method	GPT-4o	GPT-4.1
Full-schema	0.851	0.868
SchemaRAG	**0.899***	**0.900***

Because retrieval relies on semantic similarity, the choice of embedder can influence performance. We run a single-pass (x1) check on the Nursing dataset to assess sensitivity to different off-the-shelf embeddings; results are reported as micro-F_1. Across embedders, SchemaRAG consistently outperforms the full-schema baseline, indicating that while the encoder choice can shift absolute accuracy, the method's relative gains are robust (Table 9).

Table 9: Nursing micro-F_1 (single run) across `text-embedding-*` embedders.

Embedder	Full-schema	SchemaRAG
`-ada-002`	0.847	**0.915**
`-3-small`	0.853	**0.902**
`-3-large`	0.849	**0.904**

G Token complexity

We model the token complexity of SchemaRAG in this way.

For all API LLM-driven methods, the worst-case upper bound on the number of tokens used in a single LLM call (prompt and completion) is the token limit imposed by the service, as the LLM may or may not follow instructions to respond in the form of the provided schema and could exceed it. In this analysis, we assume a typical case to involve a well-behaved LLM that follows instructions to respond in the desired format.

SchemaRAG consists of two steps: segmentation and extraction. For this analysis, we assume that the transcript text T to be segmented contains non-duplicated, non-contradictory extractable facts that are well-distributed throughout the text, so that a well-behaved LLM segmenter will yield segments of roughly equal length. In segmentation, the input prompt contains the full transcript, yielding complexity $O(T)$, and the segmented output is expected to be of roughly equal size, also $O(T)$. Input and output together yield a complexity of $O(T)$ (with the doubling of the single term from input and output an ignorable constant).

In extraction, we assume that the hyperparameter k is set such that it will exhaust neither the number of available schema rows nor the number of examples in a full example set A. Each segment t_i's extraction prompt contains a number of dynamically-sized items: t_i, the full transcript T (provided for context), the reduced schema $S_{\kappa i}$, and any retrieved examples A_i. We can model this complexity as $O(t_i + T + S_{\kappa i} + A_i)$, which reduces to $O(T + S_{\kappa i} + A_i)$ (because t_i is related to T by an ignorable constant).

Segmentation yields n segments, each with an extraction prompt. Given our expectation of segments of roughly equal length and roughly equal extractable fact density, we can also suppose that SchemaRAG would yield n reduced schemas with a roughly equal number of rows, and n retrieved example sets with a roughly equal number of examples, also assuming that the available full available example set A uniformly covers the test set rows.

On these assumptions, we can model all n input extractions as having complexity $O(nT + nS_{\kappa i} + nA_i)$. Since n is a linear factor of T, this can be rewritten as $O(T^2 + TS_{\kappa i} + TA_i)$.

Extraction output complexity is $O(nS_{\kappa i})$, since in the worst case, the entire reduced schema could be filled in each segment's extraction. This can be rewritten as $O(TS_{\kappa i})$.

Total extraction complexity for a transcript, across all segments, is therefore $O(T^2 + TS_{\kappa i} + TA_i)$ (as the doubling of $O(TS_{\kappa i})$ in input and output complexity is an ignorable constant).

Finally, total complexity of SchemaRAG in-

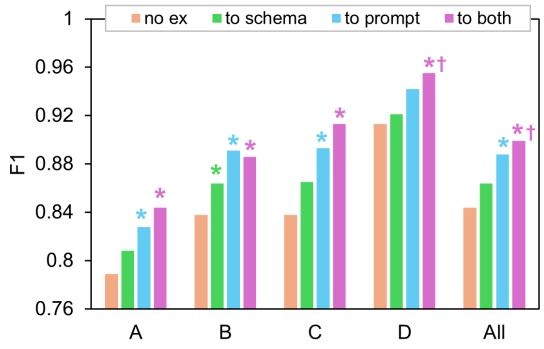

Figure 4: Effect of example use method on SchemaRAG F_1 on the Nursing dataset. * indicates significance over the no-example baseline, and † over the "to-prompt" method, at $p < 0.01$.

cluding segmentation and extraction is $O(T^2 + TS_{\kappa i} + TA_i + T)$, which can be simplified to $O(T^2 + TS_{\kappa i} + TA_i)$ (because $T \leq T^2$).

H Ablation studies

H.1 Effect of example method

An advantage of SchemaRAG is its openness to the use of any extra annotated data. When a set of annotated examples A is available, it could be incorporated in three ways: i) as example embeddings that drive the schema reduction process ("to-schema"), ii) as in-context learning material that is injected into the prompt ("to-prompt"), or iii) both ("to-both"). We examine the effect of these example methods on the extraction performance of SchemaRAG on the Nursing dataset, breaking down by hospital (Figure 4).

Across all four hospitals, the "to-both" method significantly improves extraction accuracy over the no-example baseline. This demonstrates that incorporating guidance from examples at both the retrieval and inference stages is effective across different schema structures.

The gain from the "to-schema" method alone is not statistically significant. In contrast, the "to-prompt" method yields a statistically significant improvement alone, indicating that few-shot supervision at inference time plays a central role in guiding accurate extraction. The performance of the "to-both" method also significantly exceeds that of the "to-prompt" method. This suggests that inclusion of example embeddings in retrieval, in addition to in-prompt example text, yields complementary information that helps improve alignment between the reduced schema and relevant prompt content.

H.2 Effect of row and example embeddings

We examine how row embeddings and example embeddings, used independently and together during the retrieval stage, affect extraction performance on the Nursing dataset. All pairwise differences between conditions are statistically significant (Table 10).

Using only example embeddings yields the lowest performance. This can be partly attributed to the fact that each per-hospital example set does not cover all rows present in the per-hospital test set. Some test rows have no corresponding examples (see Figure 3), placing a fundamental limit on row recall. Additionally, the example set contains schema rows that are not part of the test set, which introduces noise. By contrast, row-level embeddings provide complete coverage and ensure that all schema entries are potentially retrievable.

The superior performance of combining both embeddings highlights their complementary nature: row embeddings offer broad coverage, while example embeddings provide context-rich signals when available.

H.3 Effect of segmentation

We evaluate the role of segmentation in SchemaRAG by comparing three conditions: i) no segmentation at all, ii) segmentation used only for scoping the RAG process (with the full transcript and the union over all segments' resulting reduced schemas as the S_κ provided to a single LLM call at extraction), iii) and full segmentation used both in RAG and prompt construction over per-segment extraction calls (see Table 11).

On the Nursing dataset, both segmentation conditions significantly outperform the no-segmentation baseline. On the Amazon dataset, segmentation for scoping RAG significantly outperforms no segmentation, though its advantage over full segmentation used both in RAG and prompt construction is not statistically significant. These results suggest that segmentation benefits SchemaRAG primarily by guiding schema selection during the retrieval phase. Additional gains from segment-level extraction itself depend on the domain and dataset characteristics.

Table 10: Effect of row and example embedding availability in SchemaRAG retrieval ($k = 60$) on average per-transcript extraction performance on the Nursing dataset. All pairwise differences are statistically significant, indicated by † ($p < 0.001$).

Row	Example	macro-F_1
yes	no	0.888†
no	yes	0.798†
yes	yes	**0.899**†

Table 11: Effect of segmentation scope on extraction performance of SchemaRAG. Statistical significance vs. no-segmentation SchemaRAG baseline is indicated by * ($p < 0.05$) and † ($p < 0.001$).

Segmentation scope	macro-F_1	
	Nursing	Amazon
Baseline	0.774	0.484
RAG-only	0.898†	**0.554***
RAG + prompt	**0.899**†	0.515

H.4 Effect of k

We examine the effect of the hyperparameter k on the extraction performance of SchemaRAG on the Nursing dataset. k is the number of embeddings consulted during the schema reduction process and roughly correlates with κ, the final number of rows in the reduced schema. We vary k to produce reduced schemas of size κ between 1 and 1100 with and without segmentation, and with and without examples, running each condition only once (Figure 5).

In general, increasing k improves performance to a point, as the relevant rows appear in the reduced schema; but performance saturates and declines as k increases, as the reduced schema becomes too large and the LLM is unable to focus on the relevant rows. This effect holds over all segmentation and example conditions, though using segmentation and examples shift the optimal k to a lower value.

H.5 Retrieval upper bound via Precision/Recall@k

We compare SchemaRAG to a full-schema baseline that uses segmentation to eliminate the effect of segmentation on top-k counting. Each segment of a transcript yields its own SchemaRAG-retrieved set of rows; we take the top k of each segment, then combine these and compare to the gold-standard annotations for the entire transcript.

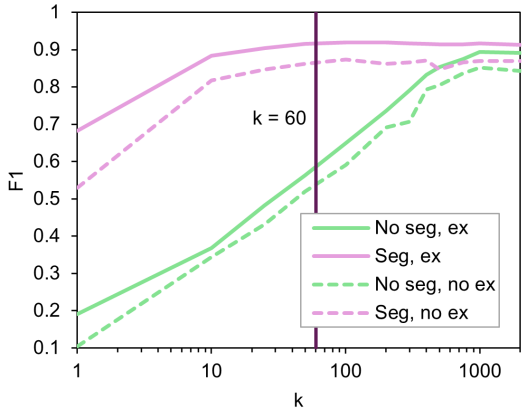

Figure 5: Effect of k on SchemaRAG F_1 on Nursing.

Table 12: Comparison of Precision and Recall between the Full-schema baseline and SchemaRAG at various levels (@k).

Method	@k	Precision	Recall
Full-schema + seg.	1	0.724	0.431
	5	0.275	0.749
	10	0.161	0.857
	30	0.060	0.948
	60	0.031	0.981
SchemaRAG	1	0.837	0.503
	5	0.288	0.800
	10	0.163	0.886
	30	0.060	0.959
	60	0.031	0.991

SchemaRAG achieves consistently higher recall@k than the full-schema+segmentation baseline at equivalent k values (Table 12). This suggests that SchemaRAG yields more of the annotated rows than the naïve baseline and approaches the upper bound of retrieval faster with increasing k.

Similarly, SchemaRAG exhibits higher precision@k at low k (Table 12), suggesting that the annotated rows it finds are more densely packed near the top of the retrieval. (At high k, the baseline and SchemaRAG have similar precisions because there are more cases of fewer than k annotated rows over each transcript).