StructSurvey: Structured Agentic Retrieval for Automated Survey Paper Generation

Paolo Pedinotti*
Bloomberg
pedinotti.paolo@gmail.com

Enrico Santus
Bloomberg
esantus@bloomberg.net

Abstract

The rapid growth of scientific publications makes it increasingly difficult to track and synthesize research progress. While Large Language Models (LLMs) can support automated survey generation, existing methods retrieve unstructured data and require models to infer conceptual, methodological, and taxonomic relations from raw text at generation time. We introduce StructSurvey, a hierarchical multi-agent framework that shifts structural reasoning from generation to retrieval by dynamically constructing graph-based representations of entities, relations, and topical taxonomies. We evaluate StructSurvey on a new reference-grounded benchmark of ACL survey papers for reproducible long-form scientific summarization. Compared with embedding-only retrieval baselines, StructSurvey improves ROUGE-1 recall by +2.9 and ROUGE-2 recall by +1.0 on average, without reducing precision. It also improves LLM-as-a-Judge ratings for logical structure, depth, and synthesis, showing that explicit structural retrieval yields surveys closer to human-written organization and reasoning.

1 Introduction

The rapid growth of scientific publications has made it increasingly difficult for researchers to identify, interpret, and synthesize relevant developments. Scientific surveys address this challenge by organizing fragmented contributions into coherent narratives that reveal conceptual structure, methodological evolution, and emerging trends. However, writing high-quality surveys remains labor-intensive, requiring domain expertise, careful source selection, and substantial time.

This has motivated work on *automated survey paper generation* (Wang et al., 2024; Yan et al., 2025), where Large Language Models (LLMs) assist in retrieving, organizing, and synthesizing research literature. Survey generation is particularly demanding because it requires not only factual grounding, but also the ability to identify relationships among methods, tasks, datasets, and research directions across many papers.

Existing systems typically rely on **Retrieval-Augmented Generation (RAG)** (Lewis et al., 2020), retrieving semantically relevant documents and passing them to the LLM as evidence. While this improves grounding, current approaches use *unstructured retrieval*: retrieved passages provide no explicit representation of how ideas relate, how entities cluster, or how methods connect across the literature. This is a key limitation for survey writing, where quality depends not only on retrieving relevant papers but also on organizing them into coherent conceptual and methodological groupings. As a result, LLMs must reconstruct the conceptual scaffolding of a survey implicitly during generation, which is especially difficult for hierarchical documents organized into sections, subsections, and fine-grained themes.

We introduce StructSurvey, a hierarchical multi-agent framework that incorporates *structured retrieval* (Jiang et al., 2025) into survey generation. Instead of relying only on vector search, StructSurvey uses **parameterized query functions** that trigger on-demand extraction of entities, relations, and taxonomic groupings from retrieved abstracts. These outputs are merged into an evolving domain graph that captures relationships among methods, tasks, and research directions as the outline expands. Figure 1 illustrates this process. By shifting structural reasoning from generation to retrieval, StructSurvey provides explicit, interpretable context for outline formation and narrative synthesis.

To evaluate this approach, we construct a new **ACL Survey Dataset** of 33 survey papers published between 2018 and 2025, together with their full text and publicly accessible referenced papers.

*Paolo Pedinotti contributed to this work during his internship at Bloomberg.

Proceedings of the First Workshop on Structured Understanding, Retrieval, and Generation in the LLM Era (SURGeLLM 2026), pages 162–181

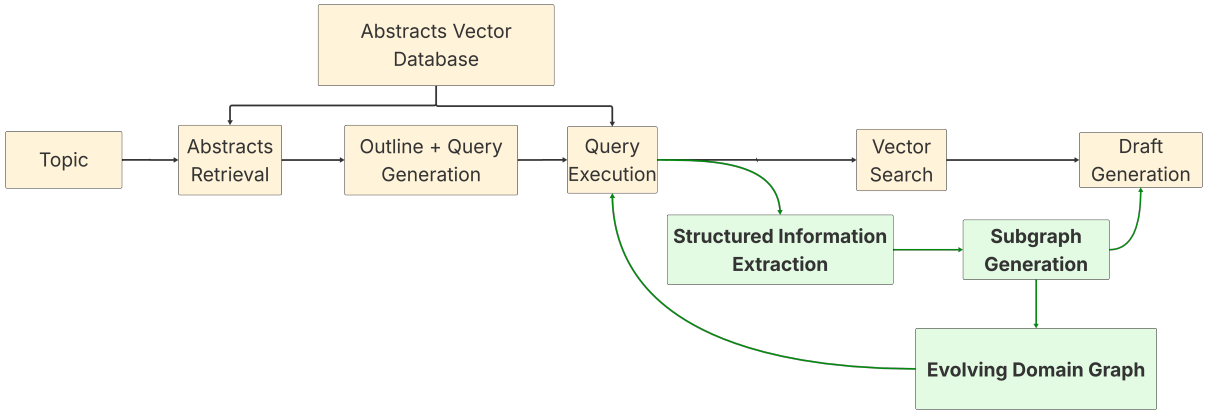

Figure 1: StructSurvey process. Unlike vector-only retrieval systems (yellow), StructSurvey extracts structured information (green), merges retrieved subgraphs into an evolving domain graph, and uses it to guide generation.

This reference-grounded benchmark enables reproducible comparison of survey-generation systems under controlled evidence conditions.

On this benchmark, StructSurvey improves over embedding-only retrieval baselines by +2.9 ROUGE-1 recall and +1.0 ROUGE-2 recall on average, while preserving comparable precision. These gains show that structured retrieval provides more relevant evidence during outline expansion and drafting. Since lexical overlap cannot fully capture survey quality, we also conduct **LLM-as-a-Judge evaluations** (Gu et al., 2024) using a robustness-oriented protocol over logical structure, depth, critical analysis, and synthesis. StructSurvey improves all dimensions except critical analysis, which remains a limitation across systems.

Our contributions are:

- **StructSurvey**, a hierarchical multi-agent framework that uses structured retrieval and evolving domain graphs to guide survey planning, retrieval, and writing;
- a *reference-grounded* benchmark comprising 33 ACL survey papers published between 2018 and 2025, all with publicly accessible references (see Appendix A);
- a reproducible **LLM-as-a-Judge evaluation protocol** designed to reduce prompt sensitivity, model bias, and reproducibility issues;
- empirical evidence that structured retrieval improves both lexical overlap and higher-level survey qualities such as logical structure, depth, and synthesis.

2 Related Work

Automated survey paper generation with LLMs is an emerging research area. Recent systems generally frame survey generation as a hierarchical planning and writing task, where LLM agents retrieve relevant literature, generate outlines, and draft sections. The most directly comparable recent systems are AutoSurvey (Wang et al., 2024), which introduced a systematic LLM-based pipeline for long-form survey writing, and SurveyForge (Yan et al., 2025), which extends this line of work with heuristic outline generation, memory-driven retrieval, and multi-dimensional evaluation.

AutoSurvey. AutoSurvey formulates survey generation as a multi-stage workflow consisting of initial retrieval and outline generation, subsection drafting, integration and refinement, and final evaluation. The system first retrieves papers relevant to the survey topic, generates and merges candidate outlines, and then drafts sections in parallel using subsection-specific retrieval. It further refines generated sections for coherence and citation correctness, and uses LLM-based evaluation to select among generated survey candidates. However, retrieval is based on embedding similarity, and the retrieved evidence is passed to downstream generation modules as unstructured textual context. As a result, reasoning about how concepts, methods, and themes relate is largely left to the generation phase, where the LLM must organize long and heterogeneous evidence without an explicit representation of conceptual or taxonomic structure.

SurveyForge. SurveyForge adopts a related hierarchical generation architecture but in-

troduces two important extensions. First, it uses both a research-paper database and a survey-outline database, allowing human-written survey structures to guide outline generation. Second, it expands outlines recursively and employs a memory-driven Scholar Navigation Agent with sub-query decomposition and temporally-aware reranking to retrieve higher-quality references for each subsection. Despite these additions, SURVEYFORGE still builds primarily on embedding-based retrieval, augmented with memory and reranking, and provides downstream agents with textual evidence rather than explicit relational or taxonomic representations.

Graph-based Retrieval. Recent work on graph-based retrieval-augmented generation (GraphRAG) investigates the use of graph-structured knowledge for reasoning and generation (Peng et al., 2025; Han et al., 2024; Edge et al., 2024). These approaches often construct or assume a graph representation over a corpus, such as a knowledge graph, citation network, or entity-relation graph. However, survey writing requires topic-specific and outline-dependent structure.

Comparison. Existing survey-generation systems largely treat the task as a tree-structured pipeline over retrieved textual evidence. Consequently, the LLM must infer relationships among concepts, methods, datasets, tasks, and research threads during generation. STRUCTSURVEY addresses this limitation by shifting structural inference into retrieval: query functions can return explicit knowledge representations, including extracted entities, relations, and lightweight taxonomies derived from retrieved abstracts. Unlike static GraphRAG settings, this structure is created on demand for each outline node and reused as the survey develops. This enables more targeted retrieval and more coherent generation, while remaining model-agnostic and compatible with previous frameworks.

3 Method Overview

The core insight behind STRUCTSURVEY is that survey papers are inherently structured artifacts: they organize a research area into entities (e.g., models, datasets, tasks), relations among those entities, and higher-level taxonomic groupings. Instead of requiring an LLM to reconstruct this structure implicitly from unstructured text during generation, STRUCTSURVEY explicitly extracts, accumulates, and reuses structured knowledge throughout the survey-generation process.

Workflow Overview. The STRUCTSURVEY workflow consists of three stages that are applied hierarchically as the survey outline is expanded:

1. **Outline planning.** Given a survey topic, the system retrieves an initial set of candidate papers using embedding-based retrieval over a paper corpus. A planner LLM generates a first-level outline, where each section is paired with a *parameterized query function*. The query function specifies how evidence for that section should be retrieved and organized.

2. **Structured retrieval and graph construction.** Executing a query function triggers either (i) standard vector retrieval, which returns a set of relevant abstracts, or (ii) structured extraction, which produces a **local graph fragment** encoding entities, relations, or taxonomic assignments extracted from the papers. These local graph fragments are incrementally merged into an evolving domain graph that accumulates structured knowledge across outline levels.

3. **Section drafting.** Writer LLMs decompose each section-level query into finer-grained subqueries, retrieve both unstructured text and structured graph context, and draft subsections in parallel. Subsection drafts are then merged to form complete section drafts, which are finally assembled into the full survey.

This workflow operationalizes survey generation as a repeated sequence of *planning* $\rightarrow$ *structuring* $\rightarrow$ *writing*, allowing LLMs to condition generation on explicit, interpretable structure rather than raw text alone.

Query Interface. The central abstraction in STRUCTSURVEY is the **query function**. A query function is a typed retrieval instruction attached to each outline node. Unlike prior systems, which associate sections only with free-text queries, STRUCTSURVEY uses query functions to explicitly control both the retrieval strategy and the structure of the returned evidence.

Each query function returns either: (i) a set of retrieved texts (vector retrieval), or (ii) a structured,

undirected graph fragment extracted from the texts (structured retrieval).

During outline generation, the planner LLM acts as a **routing agent**, selecting the best query function for each section.

The query functions implemented in StructSurvey are:

- `single_entity_search`. This function extracts all entities of a specified type and organizes them according to a categorization criterion. **Example:** extracting datasets mentioned in the abstracts and grouping them by task type (e.g., classification, generation, parsing). **Output:** a graph whose nodes correspond to datasets, annotated with category labels and frequency counts.
- `related_entity_search`. This function extracts entities that are explicitly related to a fixed entity. **Example:** extracting downstream NLP tasks associated with the BERT model and categorizing them by linguistic level. **Output:** a graph centered on the fixed entity (e.g., BERT), with edges to related entities.
- `pair_of_related_entities_search`. This function extracts pairs of related entities and assigns each side to its own taxonomy. **Example:** extracting transformer architectures and the NLG tasks they are applied to, yielding pairs such as (*BART*, summarization), (*T5*, data-to-text generation), and (*mBART*, machine translation). **Output:** a bipartite graph linking source and target entities, with category annotations on both sides.

Structured Extraction Utilities. Each query function is implemented as a composition of LLM-based extraction utilities. Following Edge et al. (2024) and Wan et al. (2024), StructSurvey employs the following components:

- **Entity extraction**: parallel extraction of entities across documents (Prompt B.0.1).
- **Relationship-based extraction**: identification of entities related to a specified fixed entity (Prompt B.0.4).
- **Entity pair extraction**: extraction of typed source–target entity pairs (Prompt B.0.5).
- **Category discovery**: induction of a compact taxonomy (3-8 categories) covering a set of extracted entities (Prompt B.0.2).
- **Entity classification**: parallel assignment of extracted entities to the discovered categories (Prompt B.0.3).

Graph Construction and Merging. Each structured query produces a local undirected graph fragment corresponding to a specific outline node. These fragments are merged into a global domain graph by canonicalizing entity strings, merging nodes with identical canonical forms, uniting edges and category assignments, and aggregating frequency statistics across extractions. The resulting domain graph grows incrementally as the outline deepens and is reused across subsequent retrieval and generation steps.

Graph Conditioning of Generation. When drafting sections, the global domain graph or the relevant subgraph is serialized into a compact textual representation consisting of entity lists grouped by category, explicit entity–entity relations, and frequency statistics. This serialized graph context is injected directly into the second-level outline generation and the writing prompts (Prompt B.0.7 and Prompt B.0.9), where the LLM is explicitly instructed to use it to organize and structure the generated narrative.

Algorithmic Implementation. Algorithm 1 summarizes the full pipeline, combining outline planners, routing agents, query functions, extraction utilities, and writer agents into a coherent end-to-end system.

4 Experiments

4.1 Data

To evaluate StructSurvey in a realistic scientific writing setting, we construct a gold-standard dataset of 33 ACL Anthology surveys published between 2018 and 2025, listed in Appendix A. Each survey is paired with the set of papers it cites, forming a benchmark for *reference-grounded* evaluation. Under this setting, systems are evaluated only on content drawn from the same reference pool available to the human authors, preventing the use of external or future information and enabling fair comparison to human-written surveys.

Survey paper extraction. We queried the ACL Anthology to identify survey papers published between 2018 and 2025 in major ACL venues. Candidate surveys were identified using keywords indicative of surveys. This process yielded an initial

Algorithm 1 StructSurvey: Survey Generation with Structured Retrieval and Graph Reuse

Require: Survey topic T; paper vector index P; query-function library $\mathcal{Q}$; prompt set Π
Ensure: Generated survey text S

Initialize global domain graph $G \leftarrow \emptyset$
Initialize section drafts $\mathcal{D} \leftarrow \emptyset$ ▷ **Stage 1: Seed retrieval and first-level planning**
$D_{\text{top}} \leftarrow \textsc{VectorSearch}(T, P)$ ▷ Retrieve top-k papers for topic T
$\mathcal{O}^{(1)} \leftarrow \text{LLM}(\Pi^{(1)}_{\text{Outline}}, T, D_{\text{top}}, \mathcal{Q})$ ▷ Generate first-level outline with query functions
for $i = 1$ to $N^{(1)}$ **do** ▷ **Stage 2: First-level retrieval and second-level planning**
  $(O_i^{(1)}, Q_i^{(1)}) \leftarrow \mathcal{O}_i^{(1)}$ ▷ Unpack section title and assigned query
  $(C_i^{(1)}, G_i^{(1)}) \leftarrow \textsc{Execute}(Q_i^{(1)}, P, G)$ ▷ Run vector or structured retrieval
  $G \leftarrow \textsc{MergeGraph}(G, G_i^{(1)})$ ▷ Reuse and update global graph
  $G_i^{\text{ctx}} \leftarrow \textsc{SelectSubgraph}(G, O_i^{(1)}, C_i^{(1)})$
  $\mathcal{O}_i^{(2)} \leftarrow \text{LLM}(\Pi^{(2)}_{\text{Outline}}, O_i^{(1)}, C_i^{(1)}, G_i^{\text{ctx}})$ ▷ Generate subsections
  ▷ **Stage 3–4: Sub-querying and drafting**
  $\mathcal{D}_i \leftarrow \emptyset$
  for $j = 1$ to $N_i^{(2)}$ **do**
    $\mathcal{R}_{i,j} \leftarrow \text{LLM}(\Pi_{\text{Decompose}}, O_{i,j}^{(2)}, \mathcal{Q})$ ▷ Create 3–5 retrieval sub-queries
    $C_{i,j} \leftarrow \emptyset$
    for each $Q_{\text{sub}} \in \mathcal{R}_{i,j}$ **do**
      $(C_{\text{sub}}, G_{\text{sub}}) \leftarrow \textsc{Execute}(Q_{\text{sub}}, P, G)$
      $C_{i,j} \leftarrow C_{i,j} \cup C_{\text{sub}}$
      $G \leftarrow \textsc{MergeGraph}(G, G_{\text{sub}})$ ▷ Accumulate evidence and graph structure
    end for
    $G_{i,j}^{\text{ctx}} \leftarrow \textsc{SelectSubgraph}(G, O_{i,j}^{(2)}, C_{i,j})$
    $D_{i,j} \leftarrow \text{LLM}(\Pi_{\text{Write}}, O_{i,j}^{(2)}, C_{i,j}, G_{i,j}^{\text{ctx}})$ ▷ Draft subsection
    $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \{D_{i,j}\}$
  end for
  $D_i \leftarrow \textsc{MergeDrafts}(\mathcal{D}_i)$ ▷ Merge subsection drafts
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{D_i\}$
end for ▷ **Stage 5: Final assembly**
$S \leftarrow \textsc{MergeDrafts}(\mathcal{D})$ ▷ Assemble full survey
return S

pool of survey candidates spanning a broad range of NLP subfields.

Referenced paper collection. For each survey, we retrieved the complete reference list via Semantic Scholar.[1] To replicate the conditions under which the original surveys were written, we retained only surveys for which at least 70% of referenced papers were publicly available through the ACL Anthology or arXiv. This constraint avoids licensing issues and yields a clean, reproducible benchmark.

Dataset characteristics. Each dataset entry includes the survey abstract, full-body text (extracted using the Mistral OCR model[2]), and the set of accessible referenced papers (with abstracts). Key corpus statistics are reported in Table 1.

4.2 Experimental Setting

We compare StructSurvey directly with SurveyForge under a controlled experimental setup. Both systems share the same multi-agent hierarchical architecture, maximum outline depth, and writing procedures. The only difference is the retrieval mechanism: SurveyForge supports only vector-based queries, whereas StructSurvey additionally allows LLMs to invoke structured query functions. This isolates the contribution of structured retrieval.

Model configuration. All LLM components run on `Gemini-2.5-Flash Lite`, accessed via the official Gemini API.[3] Embeddings for vector retrieval are generated with OpenAI `text-embedding-3-small`[4] and indexed using FAISS (Douze et al., 2025).

The decoding configuration is kept identical across both systems: temperature 1 and top-p 0.95.

Abstract-level retrieval. For both vector-based and structured retrieval, we operate exclusively on paper abstracts rather than full texts. This choice reflects a trade-off between computational cost, scal-

[1] https://www.semanticscholar.org/product/api
[2] https://mistral.ai/news/mistral-ocr
[3] https://ai.google.dev/gemini-api/docs
[4] https://platform.openai.com/docs/models/text-embedding-3-small

ability, and experimental control: abstract-level retrieval enables processing across thousands of cited papers while avoiding the substantial cost associated with long-document extraction. Importantly, both StructSurvey and the baseline system are subject to the same constraint, ensuring a fair comparison.

First-level outline generation. The system retrieves the top 20 abstracts using embedding-based similarity with the survey topic. Using Prompt B.0.6, the planner LLM produces 4-6 top-level sections, each paired with a query function. This pairing is the key decision point: SurveyForge always assigns a vector retrieval function, whereas StructSurvey may assign either a vector or a structured query function depending on the semantic cues in the retrieved context.

Second-level outline generation. For each first-level section:

- If the query function is vector-based, the system retrieves the top 15 abstracts and generates 3-5 subsections using Prompt B.0.7.
- If the query is structured, the system executes the corresponding structured extraction functions (described in Section 3) on the abstracts. The resulting structured graph is provided directly to the outlining LLM.

This stage expands each coarse section into a more fine-grained and topic-aware outline.

Content generation. After the hierarchical outline is finalized, writer LLMs decompose each section query into subqueries using Prompt B.0.8. The system retrieves evidence — either vector-based or structured — and passes the merged content to Prompt B.0.9 to produce 500-800 word drafts. Prompts for query decomposition (except for the list of available query functions) and content generation are identical across systems, ensuring a fair comparison.

Reference-grounded evaluation. Our experimental setup evaluates survey-generation systems under a reference-grounded setting, where the retrieval corpus for each survey consists exclusively of the papers cited by the gold survey available on the ACL Anthology or arXiv. This design choice isolates the core challenge addressed in this work — planning, structuring, and synthesizing a research area given relevant literature — while controlling for variability introduced by open-ended document discovery. By constraining systems to the same evidence available to the human authors, we enable fair and reproducible comparison of generated surveys against gold references. At the same time, this setting abstracts away retrieval precision errors that arise in open-corpus scenarios; incorporating citation-graph expansion or open-domain retrieval is an important direction for future work but is orthogonal to the structured-retrieval mechanisms studied here.

4.3 Evaluation

Evaluating automatically generated surveys poses unique challenges: surveys are long, require domain-level knowledge, and must be assessed along multiple qualitative dimensions. Prior work has relied primarily on LLM-as-a-Judge assessment, but such evaluation introduces reliability concerns due to prompt sensitivity, model-specific biases, and limited reproducibility (Gu et al., 2024; Li et al., 2024). We therefore combine standard lexical overlap metrics with a principled LLM-guided evaluation protocol.

Metrics. We compute ROUGE-1 and ROUGE-2 precision, recall, and F1 between system-generated surveys and the corresponding gold surveys. Although ROUGE does not capture global structure or deep semantics, it provides a stable and reproducible measure of lexical and phrasal overlap that is sensitive to topical coverage (recall) and hallucination (precision).

LLM-guided evaluation. To assess higher-level qualities not captured by ROUGE, we complement lexical metrics with LLM-as-a-Judge evaluations. Following recent analyses of evaluation robustness (Baumann et al., 2025), we adopt a principled protocol that mitigates prompt sensitivity and model-specific biases.

First, we selected a pair of evaluator models, including both proprietary and open models. For our experiments, we use `claude-sonnet-4-5-20250929` and `deepseek.r1-v1:0` via Amazon Bedrock.[5]

We then defined a set of criteria for survey quality, including outline-level criteria (involving the analysis of outlines), and full survey-level criteria. We accompany each criterion with a short and concise definition. The outline-level criteria

[5] https://aws.amazon.com/bedrock/

Average	Gold	SURVEYFORGE	STRUCTSURVEY
Words / Survey	6220.7	3954.5	4221.9
1st-Level Sect.	7.73	5.67	6.03
Refs / Survey	109.0	49.8	49.8
Subsect./Sect.	1.28	4.15	4.40

Table 1: Comparison of structural metrics across systems.

Metric	SURVEYFORGE	STRUCTSURVEY	Sig.
ROUGE-1			
Prec.	$.6354 \pm .0461$	$.6355 \pm .0378$	ns
Recall	$.4194 \pm .0393$	$\mathbf{.4480} \pm .0437$	**
F-meas.	$.5034 \pm .0313$	$\mathbf{.5241} \pm .0342$	***
ROUGE-2			
Prec.	$.1734 \pm .0233$	$.1776 \pm .0227$	ns
Recall	$.1148 \pm .0193$	$\mathbf{.1252} \pm .0188$	***
F-meas.	$.1376 \pm .0196$	$\mathbf{.1465} \pm .0193$	***

Sig.: ns: not significant; **: $p < 0.01$; ***: $p < 0.001$. Best scores per metric are bolded.

Table 2: ROUGE Performance Comparison (M±SD)

we considered are: logical structure (the extent to which the generated outline follows a logical progression), depth (the extent to which the depth of the outline is appropriate for an NLP survey paper). The full survey-level criteria are: critical analysis (The depth of engagement with the surveyed literature beyond mere description), and synthesis (The ability to identify patterns, trends, and relationships across the body of literature).

For each criterion, we fed the definition to an LLM (`claude-sonnet-4-5-20250929`) and we asked it to generate two different prompts aimed at evaluating the same criteria. This step is aimed at testing the robustness of the results to prompt variation. We asked the LLM to provide prompts in chain-of-thought-style (Wei et al., 2022), and to ask for scores from 1 to 5. For each criterion, we average the results across prompts and models.

Statistical testing. For each metric, we apply paired t-tests to assess the statistical significance of differences between STRUCTSURVEY and SURVEYFORGE.

5 Results

5.1 Analysis of Generated Texts

We analyze the structural and lexical properties of the generated surveys in comparison to gold human-written surveys. Table 1 summarizes key statistics, and representative qualitative examples

Criterion	STRUCTSURVEY	SURVEYFORGE	p-value
Logical structure	3.725	3.575	0.080
Depth	2.808	2.667	0.022*
Critical analysis	3.025	3.042	0.783
Synthesis	3.767	3.650	0.032*

Table 3: LLM-as-a-Judge evaluation scores across different criteria. Asterisks (*) indicate statistical significance at $p < 0.05$.

are provided in Appendix C. In the provided examples, we aligned each gold-standard section with the corresponding section in the outputs generated by the two systems. In the examples, we can see how the graph-generated outlines are more comprehensive and detailed, and how they align more often with the gold-standard content. Given that we imposed a limit of 500-800 words to each section draft, both automated systems produce surveys that are substantially shorter than human-authored ones — approximately one-third shorter on average.

Despite this gap, STRUCTSURVEY consistently generates longer and more detailed surveys than SURVEYFORGE (4,221.9 vs. 3,954.5 words on average). This difference suggests improved topical coverage and more effective content planning when structured retrieval is available.

We attribute this gain primarily to differences in outline construction. Although both systems are initialized with the same embedding-based retrieval results, their planners diverge in how they expand the outline. STRUCTSURVEY produces more fine-grained first-level outlines (6.03 sections on average, compared to 5.67 for SURVEYFORGE). The availability of structured query functions encourages the planner to surface a broader range of themes and to differentiate them more clearly, better reflecting the organizational patterns observed in human-written surveys.

This effect is even more pronounced at the subsection level. As shown in Table 1, STRUCTSURVEY produces deeper outline decompositions, introducing a larger number of subsections per topic. While human surveys often favor more compact subsection structures, the additional granularity induced by STRUCTSURVEY provides more specific prompts for downstream content generation, which appears to translate into improved performance in subsequent evaluation. Overall, these findings suggest that structured retrieval not only expands access to relevant evidence but also actively shapes the conceptual organization of generated surveys.

5.2 ROUGE Performance

We evaluate both systems using ROUGE-1 and ROUGE-2 precision, recall, and F1, with results reported in Table 2. Precision is statistically indistinguishable across systems, indicating that StructSurvey does not introduce extraneous or off-topic content despite generating longer outputs.

In contrast, StructSurvey achieves consistent and statistically significant gains in recall. ROUGE-1 recall increases from 0.419 to 0.448 (+2.86 percentage points), while ROUGE-2 recall improves from 0.115 to 0.125 (+1.04 percentage points), yielding higher F1 scores for both metrics. Paired t-tests confirm that these gains in recall and F1 are statistically significant, with medium to large effect sizes.

To test whether ROUGE gains are driven only by longer outputs, we computed, for each survey, the ROUGE F1 difference (Δ) between StructSurvey and SurveyForge for both ROUGE-1 and ROUGE-2, as well as the corresponding output-length Δ. We then ranked both quantities and computed the Spearman correlation coefficient ρ between the rankings. A value of $\rho = 1$ would indicate perfect association between longer StructSurvey outputs and larger ROUGE gains. The analysis shows that length is a contributing factor, but not the only one: the correlation is higher for ROUGE-1 ($\rho = 0.674$) and substantially lower for ROUGE-2 ($\rho = 0.394$). This suggests that as evaluation moves from unigram overlap to higher-level lexical overlap, the contribution of length decreases.

Overall, these results indicate that structured retrieval enables StructSurvey to cover a broader and more relevant set of concepts and terminology present in the reference surveys, improving topical coverage without sacrificing precision.

Added value of different graph query types. To identify which graph query types add the most value, we computed, for each survey, the ROUGE Δ between StructSurvey and SurveyForge, and correlated it with the frequency of each graph query function. The strongest correlation is observed for `related_entity_search` ($\rho = 0.380$), despite its lower average usage than `single_entity_search` (3.1 vs. 7.1 uses per paper). By comparison, `single_entity_search` has correlation 0.197, while `pair_of_related_entities_search` has correlation 0.114. This suggests that starting from an entity and exploring its relationships is a particularly useful pattern for aligning generated surveys with human-written ones.

5.3 LLM-as-a-Judge Evaluation

Table 3 reports LLM-as-a-Judge evaluation scores across the different criteria. Judge models consistently favor the outlines produced by StructSurvey in terms of both logical structure and depth, indicating clearer progression and more appropriate granularity. We also observe a statistically significant improvement in *synthesis*, suggesting that structured retrieval enables the system to more effectively integrate information across multiple sources — one of the defining challenges of survey writing.

At the same time, the results highlight a persistent limitation shared by current LLM-based survey generation systems, including StructSurvey: limited *critical analysis*. While StructSurvey remains competitive on this criterion, generated surveys still tend to emphasize descriptive coverage rather than critical evaluation of prior work.

6 Conclusion

We presented StructSurvey, a hierarchical multi-agent framework for automated survey paper generation that shifts structural reasoning from generation to retrieval through explicit extraction of entities, relations, and taxonomic groupings. By making structural information available during planning and retrieval, StructSurvey enables more coherent organization and synthesis of scientific literature.

Experiments on a new reference-grounded benchmark of 33 ACL survey papers (listed in Appendix A) show that StructSurvey improves both lexical coverage and higher-level survey qualities — such as logical structure, depth, and synthesis — without sacrificing precision. These results highlight the value of structured retrieval for long-form scientific summarization and demonstrate that explicit structural context can meaningfully guide large language models in organizing complex research areas.

7 Limitations

Computational cost. A primary limitation of StructSurvey is its higher computational cost compared to embedding-only retrieval pipelines. While vector-based queries rely solely on embed-

ding similarity search and incur no additional LLM calls beyond embedding computation—typically running in $O(\log D)$ time over an index of D documents—structured queries depend on LLM-based information extraction and are substantially more expensive.

Let E denote the number of extracted entities and b the batch size used for entity classification. Executing a single structured query requires:

- D parallel LLM calls for entity extraction,
- one LLM call for category discovery,
- $\lceil E/b \rceil$ parallel LLM calls for entity classification,

resulting in a total LLM-call cost of

$$\text{Cost}_{\text{LLM}} = D + 1 + \left\lceil \frac{E}{b} \right\rceil .$$

Structured queries produce local graph fragments that are merged into a global domain graph and reused across subsequent outline expansion and section drafting steps (Algorithm 1), which partially amortizes extraction cost. Nevertheless, overall cost still scales linearly with the number of structured queries and the choice of LLM model, making efficiency a key challenge for large-scale deployment.

Reference-grounded and abstract-only setting. Our evaluation is conducted in a reference-grounded setting, where the retrieval corpus for each survey consists exclusively of the papers cited by the gold survey, and retrieval operates at the abstract level rather than over full texts. This design isolates the core contribution of this work—structural planning, retrieval, and synthesis given relevant literature—but does not capture challenges related to open-corpus discovery, retrieval precision, or long-document processing. Extending structured retrieval to open and heterogeneous corpora is an important direction for future work.

Limited critical analysis. Despite improvements in structure, depth, and synthesis, generated surveys continue to exhibit limited critical analysis, often favoring descriptive aggregation over evaluative comparison or normative judgment. This limitation is shared by prior LLM-based survey-generation systems and reflects the difficulty of modeling expert-level critique and assessment of scientific trade-offs.

Evaluation limitations. Although we adopt a principled and robustness-oriented LLM-as-a-Judge evaluation protocol, automated evaluation of long-form scientific writing remains inherently imperfect. Metrics such as ROUGE capture lexical overlap but not conceptual correctness, while LLM-based judgments are sensitive to model capabilities and prompt design. Human expert evaluation would provide stronger validation but is costly and difficult to scale for long documents.

Together, these limitations point to promising avenues for future research, including more efficient structured retrieval, extension to open-corpus and full-text settings, stronger modeling of critical analysis, and deeper human-centered evaluation of generated surveys.

Acknowledgments

We thank David Rosenberg for his detailed and insightful feedback, which substantially improved this paper.

References

Joachim Baumann, Paul Röttger, Aleksandra Urman, Albert Wendsjö, Flor Miriam Plaza del Arco, Johannes B. Gruber, and Dirk Hovy. 2025. Large language model hacking: Quantifying the hidden risks of using llms for text annotation. *Preprint*, arXiv:2509.08825.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2025. The faiss library. *IEEE Transactions on Big Data*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*. GraphRAG constructs an LLM-derived knowledge graph to improve RAG reasoning over large corpora.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, and 1 others. 2024. A survey on llm-as-a-judge. *The Innovation*.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh M. Halappanavar, Ryan A. Rossi, Subhabrata Mukherjee, Xianfeng Tang, Qianru He, Zhigang Hua, Bo Long, Tong Zhao, Neil Shah, Amin Javari, Yinglong Xia, and Jiliang Tang. 2024. Retrieval-augmented generation with graphs (graphrag). *arXiv*, abs/2501.00309.

Pengcheng Jiang, Siru Ouyang, Yizhu Jiao, Ming Zhong, Runchu Tian, and Jiawei Han. 2025. Retrieval and structuring augmented generation with large language models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, page 6032–6042. ACM.

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 *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. 2024. Llms-as-judges: A comprehensive survey on llm-based evaluation methods. *Preprint*, arXiv:2412.05579.

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2025. Graph retrieval-augmented generation: A survey. *ACM Transactions on Information Systems*, 44(2):1–52.

Mengting Wan, Tara Safavi, Sujay Kumar Jauhar, Yujin Kim, Scott Counts, Jennifer Neville, Siddharth Suri, Chirag Shah, Ryen W White, Longqi Yang, Reid Andersen, Georg Buscher, Dhruv Joshi, and Nagu Rangan. 2024. Tnt-llm: Text mining at scale with large language models. *Preprint*, arXiv:2403.12173.

Yidong Wang, Qi Guo, Wenjin Yao, Hongbo Zhang, Xin Zhang, Zhen Wu, Meishan Zhang, Xinyu Dai, Min Zhang, Qingsong Wen, Wei Ye, Shikun Zhang, and Yue Zhang. 2024. Autosurvey: Large language models can automatically write surveys. In *Advances in Neural Information Processing Systems*, volume 37, pages 115119–115145. Curran Associates, Inc.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Xiangchao Yan, Shiyang Feng, Jiakang Yuan, Renqiu Xia, Bin Wang, Lei Bai, and Bo Zhang. 2025. SURVEYFORGE : On the outline heuristics, memory-driven generation, and multi-dimensional evaluation for automated survey writing. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12444–12465, Vienna, Austria. Association for Computational Linguistics.

A Surveys Dataset Papers

This appendix lists the survey papers used to construct the reference-grounded evaluation dataset. Entries are listed in alphabetical order by the surname of the first author. The papers are documented for dataset reproducibility purposes and do not constitute additional paper references.

1. Briakou, E., Agrawal, S., Zhang, K., Tetreault, J., and Carpuat, M. (2021). *A Review of Human Evaluation for Style Transfer*. Proceedings of the First Workshop on Natural Language Generation, Evaluation, and Metrics (GEM), 58–67.

2. Chen, Y.-P., Nishida, N., Nakayama, H., and Matsumoto, Y. (2024). *Recent Trends in Personalized Dialogue Generation: A Review of Datasets, Methodologies, and Evaluations*. Proceedings of LREC–COLING 2024, 13650–13665.

3. Chu, C., and Wang, R. (2018). *A Survey of Domain Adaptation for Neural Machine Translation*. Proceedings of the 27th International Conference on Computational Linguistics, 1304–1319.

4. Cui, W., Yu, D., Jiao, X., Meng, Z., Zhang, G., Wang, Q., Guo, S. Y., and King, I. (2025). *Recent Advances in Speech Language Models: A Survey*. Proceedings of ACL 2025 (Long Papers), 13943–13970.

5. Dong, Q., Li, L., Dai, D., Zheng, C., Ma, J., Li, R., Xia, H., Xu, J., Wu, Z., Chang, B., Sun, X., Li, L., and Sui, Z. (2024). *A Survey on In-Context Learning*. Proceedings of EMNLP 2024, 1107–1128.

6. Dong, Z., Zhou, Z., Yang, C., Shao, J., and Qiao, Y. (2024). *Attacks, Defenses and Evaluations for LLM Conversation Safety: A Survey*. Proceedings of NAACL 2024, 6734–6747.

7. Geng, J., Cai, F., Wang, Y., Koeppl, H., Nakov, P., and Gurevych, I. (2024). *A Survey of Confidence Estimation and Calibration in Large Language Models*. Proceedings of NAACL 2024, 6577–6595.

8. Iacob, R. C. A., Brad, F., Apostol, E.-S., Truică, C.-O., Hosu, I. A., and Rebedea, T. (2020).

Neural Approaches for Natural Language Interfaces to Databases: A Survey. Proceedings of COLING 2020, 381–395.

9. Kim, M., Kim, M., Kim, H., Kwak, B.-W., Kang, S., Yu, Y., Yeo, J., and Lee, D. (2024). *Pearl: A Review-driven Persona-Knowledge Grounded Conversational Recommendation Dataset*. Findings of ACL 2024, 1105–1120.

10. Lee, H.-Y., Li, S.-W., and Vu, T. (2022). *Meta Learning for Natural Language Processing: A Survey*. Proceedings of NAACL 2022, 666–684.

11. Li, M., Zhao, Y., Zhang, W., Li, S., Xie, W., Ng, S.-K., Chua, T.-S., and Deng, Y. (2025). *Knowledge Boundary of Large Language Models: A Survey*. Proceedings of ACL 2025 (Long Papers), 5131–5157.

12. Li, Z., Qu, L., and Haffari, G. (2020). *Context Dependent Semantic Parsing: A Survey*. Proceedings of COLING 2020, 2509–2521.

13. Lu, P., Qiu, L., Yu, W., Welleck, S., and Chang, K.-W. (2023). *A Survey of Deep Learning for Mathematical Reasoning*. Proceedings of ACL 2023 (Long Papers), 14605–14631.

14. Qin, L., Pan, W., Chen, Q., Liao, L., Yu, Z., Zhang, Y., Che, W., and Li, M. (2023). *End-to-end Task-oriented Dialogue: A Survey of Tasks, Methods, and Future Directions*. Proceedings of EMNLP 2023, 5925–5941.

15. Ramisch, C., Walsh, A., Blanchard, T., and Taslimipoor, S. (2023). *A Survey of MWE Identification Experiments: The Devil is in the Details*. Proceedings of the MWE Workshop 2023, 106–120.

16. Sadeddine, Z., Opitz, J., and Suchanek, F. (2024). *A Survey of Meaning Representations: From Theory to Practical Utility*. Proceedings of NAACL 2024, 2877–2892.

17. Shen, X., Vakulenko, S., del Tredici, M., Barlacchi, G., Byrne, B., and de Gispert, A. (2023). *Neural Ranking with Weak Supervision for Open-Domain Question Answering: A Survey*. Findings of EACL 2023, 1736–1750.

18. Urlana, A., Mishra, P., Roy, T., and Mishra, R. (2024). *Controllable Text Summarization: Unraveling Challenges, Approaches, and Prospects*. Findings of ACL 2024, 1603–1623.

19. Wang, L., Liu, S., Xu, M., Song, L., Shi, S., and Tu, Z. (2023). *A Survey on Zero Pronoun Translation*. Proceedings of ACL 2023 (Long Papers), 3325–3339.

20. Wang, X., Wang, H., and Yang, D. (2022). *Measure and Improve Robustness in NLP Models: A Survey*. Proceedings of NAACL 2022, 4569–4586.

21. Wang, Y., Wang, M., Manzoor, M. A., Liu, F., Georgiev, G. N., Das, R. J., and Nakov, P. (2024). *Factuality of Large Language Models: A Survey*. Proceedings of EMNLP 2024, 19519–19529.

22. Xia, P., Wu, S., and Van Durme, B. (2020). *Which *BERT? A Survey Organizing Contextualized Encoders*. Proceedings of EMNLP 2020, 7516–7533.

23. Xia, Z., Xu, J., Zhang, Y., and Liu, H. (2025). *A Survey of Uncertainty Estimation Methods on Large Language Models*. Findings of ACL 2025, 21381–21396.

24. Yan, Y., Su, J., He, J., Fu, F., Zheng, X., Lyu, Y., Wang, K., Wang, S., Wen, Q., and Hu, X. (2025). *A Survey of Mathematical Reasoning in the Era of Multimodal Large Language Models: Benchmark, Method & Challenges*. Findings of ACL 2025, 11798–11827.

25. Ye, H., Zhang, N., Chen, H., and Chen, H. (2022). *Generative Knowledge Graph Construction: A Review*. Proceedings of EMNLP 2022, 1–17.

26. Yu, X., Chatterjee, T., Asai, A., Hu, J, and Choi, E. (2022). *Beyond Counting Datasets: A Survey of Multilingual Dataset Construction and Necessary Resources*. Findings of EMNLP 2022, 3725–3743.

27. Youssef, P., Koraş, O., Li, M., Schlötterer, J., and Seifert, C. (2023). *Give Me the Facts! A Survey on Factual Knowledge Probing in Pre-trained Language Models*. Findings of EMNLP 2023, 15588–15605.

28. Zhang, Z., Fang, M., Chen, L., Namazi-Rad, M.-R., and Wang, J. (2023). *How Do Large Language Models Capture the Ever-changing World Knowledge? A Review of Recent Advances*. Proceedings of EMNLP 2023, 8289–8311.

29. Zhang, Z., Yu, W., Yu, M., Guo, Z., and Jiang, M. (2023). *A Survey of Multi-task Learning in Natural Language Processing: Regarding Task Relatedness and Training Methods*. Proceedings of EACL 2023, 943–956.

30. Zhang, Q., Chen, S., Xu, D., Cao, Q., Chen, X., Cohn, T., and Fang, M. (2023). *A Survey for Efficient Open Domain Question Answering*. Proceedings of ACL 2023 (Long Papers), 14447–14465.

31. Zhao, R., Chen, H., Wang, W., Jiao, F., Do, X. L., Qin, C., Ding, B., Guo, X., Li, M., Li, X., and Joty, S. (2023). *Retrieving Multimodal Information for Augmented Generation: A Survey*. Findings of EMNLP 2023, 4736–4756.

32. Zhou, J., and Bhat, S. (2021). *Paraphrase Generation: A Survey of the State of the Art*. Proceedings of EMNLP 2021, 5075–5086.

33. Zhou, Y., Ringeval, F., and Portet, F. (2023). *A Survey of Evaluation Methods of Generated Medical Textual Reports*. Proceedings of the 5th Clinical Natural Language Processing Workshop, 447–459.

B Prompts

B.0.1 Entity Extraction Prompt

System Prompt:

```
System Prompt:
You are an expert at extracting specific
    entities from academic abstracts.

Your task: Extract all instances of "{query}"
    from the given abstract.

For each entity you find:
1. Identify the specific name/type of the
    entity
2. Describe its role or use in the abstract
    (1-2 sentences)

Format your response EXACTLY as follows (one
    entity per block):

ENTITY: [entity name]
DESCRIPTION: [brief description of its role]

ENTITY: [entity name]
DESCRIPTION: [brief description of its role]

If you find NO entities matching the query,
    respond with:
NO_ENTITIES_FOUND

Important:
- Be specific about entity names
- Only extract entities that clearly match
    the query type
- Describe how the entity is used in THIS
    specific abstract

User Message:
Paper: {paper_title}

Abstract:
{abstract}

Extract all instances of: {query}
```

User Prompt:

```
Paper: {paper_title}

Abstract:
{abstract}

Extract all instances of: {query}
```

B.0.2 Category Discovery Prompt

System Prompt:

```
   You are an expert at categorizing entities
       based on specific criteria.

You have been given a list of entities
    extracted from academic papers.

Categorization Criterion: {categorize_by}

Your task: Analyze the entities and propose
    3-8 clear, mutually exclusive categories
    that cover most entities.

Format your response EXACTLY as follows (one
    category per line):

CATEGORY: [category name]
CATEGORY: [category name]
CATEGORY: [category name]
...

Important:
- Categories should be specific and meaningful
- Categories should align with the
    categorization criterion
- Aim for 3-8 categories (not too few, not
    too many)
- Use clear, descriptive names
```

User Prompt:

```
   Here are the extracted entities:

{entities_text}

Total entities: {entity_count}
```

```
Propose categories based on: {categorize_by}
```

B.0.3 Entity Classification Prompt

System Prompt

```
You are an expert at classifying entities into
    predefined categories.

Available categories:
{categories_text}

Your task: Classify each entity into ONE of
    the categories above.

Format your response EXACTLY as follows (one
    classification per line):

ENTITY: [entity name]
CATEGORY: [category name]

ENTITY: [entity name]
CATEGORY: [category name]

Important:
- Each entity must be assigned to exactly ONE
    category
- Use the exact category names from the list
    above
- If an entity doesn't fit well, assign it to
    "Other"
- Base classification on the entity's
    description
```

User Prompt:

```
Classify these entities:

{entities_text}

Assign each entity to one of the categories
    listed above.
```

B.0.4 Relationship-Based Entity Extraction

System Prompt:

```
You are an expert at extracting entity
    relationships from academic abstracts.

Your task: Find all instances of
    "{find_entities}" that are related to
    "{fixed_entity}" in the given abstract.

For each related entity you find:
1. Identify the specific name/type of the
    entity
2. Describe its relationship with
    "{fixed_entity}" in detail (2-3 sentences)
3. Include the paper context to clarify where
    this information comes from

Format your response EXACTLY as follows (one
    entity per block):

ENTITY: [entity name]
DESCRIPTION: In the paper "[paper_title]",
    [detailed description of how this entity
    relates to {fixed_entity}, including
    specific context and use case]

ENTITY: [entity name]
DESCRIPTION: In the paper "[paper_title]",
    [detailed description of the relationship]

If you find NO entities matching the
    criteria, respond with:
NO_ENTITIES_FOUND

Important:
- Be specific about entity names
- Only extract entities that have a clear
    relationship with "{fixed_entity}"
- Descriptions must be detailed and explain
    the relationship clearly
- Always start descriptions with "In the
    paper [paper_title]," to provide context
- Focus on HOW the entity relates to the
    fixed entity, not just what it is
```

User Prompt:

```
Paper: {paper_title}

Abstract:
{abstract}

Fixed Entity: {fixed_entity}
Find Entities: {find_entities}

Extract all instances of "{find_entities}"
    that are related to "{fixed_entity}".
```

B.0.5 Entity Pair Extraction Prompt

System Prompt:

```
You are an expert at extracting entity
    relationships from academic abstracts.

Your task: Find pairs of entities where:
- Entity A is: "{entity_type_a}"
- Entity B is: "{entity_type_b}"
- There is a clear relationship or connection
    between them in the paper

For each pair you find:
1. Identify the specific name/type of Entity A
2. Identify the specific name/type of Entity B
3. Describe their relationship in detail (2-3
    sentences)
4. Include the paper context to clarify where
    this information comes from

Format your response EXACTLY as follows (one
    pair per block):

ENTITY_A: [entity A name]
ENTITY_B: [entity B name]
DESCRIPTION: In the paper "[paper_title]",
    [detailed description of how these
    entities relate, their connection, and
    the context of their use]

ENTITY_A: [entity A name]
ENTITY_B: [entity B name]
DESCRIPTION: In the paper "[paper_title]",
    [detailed description of the relationship]

If you find NO pairs matching the criteria,
    respond with:
NO_PAIRS_FOUND
```

```
Important:
- Be specific about entity names
- Only extract pairs that have a clear
    relationship
- Descriptions must explain HOW the entities
    relate to each other
- Always start descriptions with "In the
    paper [paper_title]," to provide context
- Focus on the CONNECTION between the two
    entities
```

User Prompt:

```
    Paper: {paper_title}

Abstract:
{abstract}

Entity Type A: {entity_type_a}
Entity Type B: {entity_type_b}

Extract all pairs where Entity A relates to
    Entity B.
```

B.0.6 First-Level Outline Generation Prompt

System Prompt (full version):

```
You are an expert survey paper writer. Your
    task is to create a first-level outline
    for a survey paper.

Survey Title: {survey_title}
Survey Abstract: {survey_abstract}
Topic: {topic}

Here are relevant papers from the literature
    to help you understand the field:

{abstracts_text}

Generate a first-level outline with 4-6 main
    sections. For each section:
1. Provide a clear, descriptive title
2. Choose a query type (vector or graph) that
    best suits the section's needs
3. Generate the appropriate query based on
    the chosen type

QUERY TYPES:
You can generate two types of queries:

1. VECTOR QUERY: A traditional semantic
    search query for retrieving relevant
    papers
   Format:
   TYPE: vector
   QUERY: [semantic search query text]

2. GRAPH QUERY: A structured query for
    finding and categorizing
    entities/relationships
   Three patterns available:

   PATTERN 1 - Find and categorize entities of
       a given type:
   TYPE: graph_pattern_1
   QUERY: [entity type or property to find]
   CATEGORIZE_BY: [categorization criterion]

   PATTERN 2 - Find entities related to a
       fixed entity and categorize them:
   TYPE: graph_pattern_2
   FIXED_ENTITY: [specific entity name, e.g.,
       "BERT model" or "question answering
       task"]
   FIND_ENTITIES: [type or property of related
       entities to find]
   CATEGORIZE_BY: [categorization criterion
       for found entities]

   PATTERN 3 - Find pairs of entity types and
       categorize both:
   TYPE: graph_pattern_3
   ENTITY_TYPE_A: [type or property of first
       entity set]
   ENTITY_TYPE_B: [type or property of second
       entity set]
   CATEGORIZE_A_BY: [categorization criterion
       for A entities]
   CATEGORIZE_B_BY: [categorization criterion
       for B entities]

EXAMPLES:

Example 1 - Vector Query:
SECTION: Introduction to Neural Machine
    Translation
TYPE: vector
QUERY: neural machine translation
    architectures and approaches

Example 2 - Graph Pattern 1:
SECTION: Attention Mechanisms in Transformers
TYPE: graph_pattern_1
QUERY: attention mechanisms used in
    transformer models
CATEGORIZE_BY: attention pattern type
    (self-attention, cross-attention, sparse
    attention, linear attention)

Example 3 - Graph Pattern 2:
SECTION: Applications of BERT
TYPE: graph_pattern_2
FIXED_ENTITY: BERT pre-trained model
FIND_ENTITIES: downstream NLP tasks and
    applications
CATEGORIZE_BY: task category (sequence
    classification, token classification,
    question answering, generation)

Example 4 - Graph Pattern 3:
SECTION: Evaluation Metrics and Model
    Architectures
TYPE: graph_pattern_3
ENTITY_TYPE_A: evaluation metrics for
    language generation
ENTITY_TYPE_B: neural language model
    architectures
CATEGORIZE_A_BY: aspect measured (fluency,
    coherence, factuality, diversity)
CATEGORIZE_B_BY: architecture family
    (transformer-based, RNN-based,
    retrieval-augmented)

Example 5 - Vector Query:
SECTION: Challenges in Low-Resource Languages
TYPE: vector
QUERY: challenges and methods for NLP in
    low-resource languages
```

```
  Format your response EXACTLY as follows (each
      section on separate lines):

  SECTION: [Section Title]
  TYPE: [vector or graph_pattern_1 or
      graph_pattern_2 or graph_pattern_3]
  [Query fields based on type - see formats
      above]

  SECTION: [Next Section Title]
  TYPE: [type]
  [Query fields]
  ...

  Important: Start with an Introduction section
      and end with a Conclusion/Future
      Directions section.
```

User Prompt:

```
Generate the first-level outline for this
    survey on: {topic}
```

B.0.7 Second-level Outline Generation Prompt

System Prompt (Full Version):

```
You are an expert survey paper writer. Your
    task is to create subsections for a
    specific section of a survey paper.

  Survey Title: {survey_title}
  Survey Topic: {topic}
  Section Title: {section_title}
  Section Focus: {section_query}

  Here is context from the literature for this
      section:

  {context_text}

  Generate 3-5 subsections for this section.
      For each subsection:
  1. Provide a clear, descriptive title that
      fits within the scope of the main section
  2. Choose a query type (vector or graph) that
      best suits the subsection's needs
  3. Generate the appropriate query based on
      the chosen type

  QUERY TYPES:
  You can generate two types of queries:

  1. VECTOR QUERY: A traditional semantic
      search query for retrieving relevant
      papers
     Format:
     TYPE: vector
     QUERY: [semantic search query text]

  2. GRAPH QUERY: A structured query for
      finding and categorizing
      entities/relationships
     Three patterns available:

     PATTERN 1 - Find and categorize entities of
         a given type:
     TYPE: graph_pattern_1
     QUERY: [entity type or property to find]
     CATEGORIZE_BY: [categorization criterion]

     PATTERN 2 - Find entities related to a
         fixed entity and categorize them:
     TYPE: graph_pattern_2
     FIXED_ENTITY: [specific entity name, e.g.,
         "BERT model" or "question answering
         task"]
     FIND_ENTITIES: [type or property of related
         entities to find]
     CATEGORIZE_BY: [categorization criterion
         for found entities]

     PATTERN 3 - Find pairs of entity types and
         categorize both:
     TYPE: graph_pattern_3
     ENTITY_TYPE_A: [type or property of first
         entity set]
     ENTITY_TYPE_B: [type or property of second
         entity set]
     CATEGORIZE_A_BY: [categorization criterion
         for A entities]
     CATEGORIZE_B_BY: [categorization criterion
         for B entities]

  EXAMPLES:

  Example 1 - Vector Query:
  SECTION: Core Concepts and Terminology
  TYPE: vector
  QUERY: fundamental concepts and terminology
      in the field

  Example 2 - Graph Pattern 1:
  SECTION: Model Architectures
  TYPE: graph_pattern_1
  QUERY: neural network architectures for this
      task
  CATEGORIZE_BY: architectural design
      (encoder-only, decoder-only,
      encoder-decoder, hybrid)

  Example 3 - Graph Pattern 2:
  SECTION: BERT Variants and Extensions
  TYPE: graph_pattern_2
  FIXED_ENTITY: BERT model
  FIND_ENTITIES: model variants and extensions
  CATEGORIZE_BY: modification type
      (domain-specific, multilingual,
      efficient, task-specific)

  Example 4 - Graph Pattern 3:
  SECTION: Training Techniques and Datasets
  TYPE: graph_pattern_3
  ENTITY_TYPE_A: training techniques and
      optimization methods
  ENTITY_TYPE_B: benchmark datasets
  CATEGORIZE_A_BY: technique category
      (supervised, self-supervised,
      semi-supervised, reinforcement learning)
  CATEGORIZE_B_BY: dataset scale and domain
      (small-scale academic, large-scale web,
      domain-specific)

  Example 5 - Vector Query:
  SECTION: Recent Developments
  TYPE: vector
  QUERY: recent advances and novel approaches
      in the subfield
```

```
Format your response EXACTLY as follows (each
    subsection on separate lines):

SECTION: [Subsection Title]
TYPE: [vector or graph_pattern_1 or
    graph_pattern_2 or graph_pattern_3]
[Query fields based on type - see formats
    above]

SECTION: [Next Subsection Title]
TYPE: [type]
[Query fields]
...

Important: The subsections should logically
    break down the main section topic.
```

User Prompt:

```
Generate subsections for section:
    {section_title}
```

B.0.8 Query Decomposition Prompt

System Prompt (Full Version):

```
You are an expert at decomposing search
    queries. Your task is to create specific
    sub-queries for retrieving relevant
    papers.

Section: {section_title}
Main Query: {section_query}

Subsections:
{subsections_text}

Generate 3-5 specific search queries that
    would help retrieve relevant papers for
    this section and its subsections.
Each query should be focused and specific.
    You can choose between vector and graph
    query types.

QUERY TYPES:
You can generate two types of queries:

1. VECTOR QUERY: A traditional semantic
    search query for retrieving relevant
    papers
   Format:
   TYPE: vector
   QUERY: [semantic search query text]

2. GRAPH QUERY: A structured query for
    finding and categorizing
    entities/relationships
   Three patterns available:

   PATTERN 1 - Find and categorize entities of
       a given type:
   TYPE: graph_pattern_1
   QUERY: [entity type or property to find]
   CATEGORIZE_BY: [categorization criterion]

   PATTERN 2 - Find entities related to a
       fixed entity and categorize them:
   TYPE: graph_pattern_2
   FIXED_ENTITY: [specific entity name, e.g.,
       "BERT model" or "question answering
       task"]
   FIND_ENTITIES: [type or property of related
       entities to find]
   CATEGORIZE_BY: [categorization criterion
       for found entities]

   PATTERN 3 - Find pairs of entity types and
       categorize both:
   TYPE: graph_pattern_3
   ENTITY_TYPE_A: [type or property of first
       entity set]
   ENTITY_TYPE_B: [type or property of second
       entity set]
   CATEGORIZE_A_BY: [categorization criterion
       for A entities]
   CATEGORIZE_B_BY: [categorization criterion
       for B entities]

EXAMPLES:

Example 1 - Vector Query:
TYPE: vector
QUERY: recent advances in attention
    mechanisms for transformers

Example 2 - Graph Pattern 1:
TYPE: graph_pattern_1
QUERY: pre-training objectives for language
    models
CATEGORIZE_BY: objective type (masked
    language modeling, next sentence
    prediction, causal language modeling,
    denoising)

Example 3 - Graph Pattern 2:
TYPE: graph_pattern_2
FIXED_ENTITY: GPT-3 model
FIND_ENTITIES: applications and use cases
CATEGORIZE_BY: application domain (text
    generation, code generation, question
    answering, translation)

Example 4 - Graph Pattern 3:
TYPE: graph_pattern_3
ENTITY_TYPE_A: loss functions
ENTITY_TYPE_B: model architectures
CATEGORIZE_A_BY: loss type (cross-entropy,
    contrastive, ranking, generative)
CATEGORIZE_B_BY: model size (small <100M
    params, medium 100M-1B, large >1B)

Format your response as follows:

TYPE: [vector or graph_pattern_1 or
    graph_pattern_2 or graph_pattern_3]
[Query fields based on type]

TYPE: [type]
[Query fields]
...
```

User Prompt:

```
Generate sub-queries for section:
    {section_title}
```

B.0.9 Section Writing Prompt

System Prompt:

```
You are an expert survey paper writer. Your
```

```
    task is to write a comprehensive section
    for a survey paper.

Survey Title: {survey_title}
Survey Topic: {topic}
Section Title: {section_title}
Section Focus: {section_query}

Subsections to cover:
{subsections_text}

Here are relevant papers from the literature
     to reference:

{docs_text}

{graph_context}

Write a comprehensive section that:
1. Covers all subsections naturally (don't
     use subsection titles as headers)
2. Synthesizes information from multiple
     papers
3. If structured entity analysis is provided
     above, USE IT to organize and structure
     your discussion
   - Mention the categories and how they
       relate to each other
   - Use the entity counts to show prevalence
       of different approaches
   - Reference specific entities when
       discussing techniques
4. Provides clear explanations and comparisons
5. Uses proper academic writing style
6. Is approximately 500-800 words
7. IMPORTANT: Cite papers by their ACTUAL
     TITLES when referencing them
   - Good examples:
     * "The work 'Retrieval-Augmented
         Generation' introduces..."
     * "As shown in 'BERT: Pre-training of
         Deep Bidirectional Transformers',
         ..."
     * "Recent work on multimodal learning
         ('CLIP', 'DALL-E') has
         demonstrated..."
   - BAD examples (DO NOT USE):
     * "Paper 1 shows..."
     * "The first paper introduces..."
     * "As mentioned in [1]..."
8. When multiple papers address the same
     topic, mention their titles together
9. Use the paper titles as natural references
     in your writing

CRITICAL FORMATTING INSTRUCTIONS:
- DO NOT include the section title as a
     heading (e.g., "## {section_title}") in
     your output
- Start directly with the section content
- The section title will be added
     automatically by the system
- You MUST use the actual paper titles in
     your citations, not generic labels like
     "Paper 1" or numbered references
- If structured analysis is provided,
     integrate it naturally into your writing
     to organize the discussion
```

User Prompt:

```
Write the section: {section_title}
```

C Examples of Outputs from the Systems

Gold Standard	SURVEYFORGE	STRUCTSURVEY
Example: "Factuality of Large Language Models: A Survey"		
1 Introduction	**1 Introduction**	**1 Introduction and Problem Definition**
	○ Characterizing Factuality and Hallucinations	○ Rise of LLMs and Imperative of Factuality
	○ Importance of Factuality in Applications	○ Defining and Characterizing Hallucinations
	○ Overview of the Factuality Problem	○ Scope and Significance of Research
2 Background	**2 Understanding Factuality Issues**	**3 Causes and Contributing Factors**
○ Hallucination vs. Factuality	○ Data and Training Artifacts	○ Knowledge Gaps and Incomplete Understanding
○ Trustworthiness/Reliability vs. Factuality	○ Knowledge Representation and Retrieval Deficiencies	○ Biases and Distortions in Training Data
	○ Outdated Information and Temporal Drift	○ Model Architecture and Training Objectives
3 Evaluating Factuality	**3 Evaluating LLM Factuality**	**2 Understanding and Quantifying Factuality**
○ 3.1 Datasets and Metrics	○ Benchmarking LLM Factuality	○ Taxonomy of Factuality Errors
○ 3.2 Other Metrics	○ Human vs. Automated Assessment	○ Automated Metrics and Frameworks
	○ Fine-grained Evaluation Methods	**5 Benchmarking and Evaluation**
		○ Domain-Specific and Specialized Benchmarks
4 Improving Factuality	**4 Mitigating LLM Factuality Issues**	**4 Mitigation Techniques**
○ 4.1 Pre-training	○ Retrieval-Augmented Generation	○ Retrieval-Augmented Generation (RAG)
○ 4.2 Tuning and RLHF	○ Fine-tuning and Knowledge Editing	○ Fine-tuning and Preference Optimization
- Retrieval Augmentation	○ Verification, Rectification, and Debating	○ Prompting and Generation Strategies
○ 4.3 Inference	○ Context and Citation Enhancement	○ Model Editing and Multi-Agent Verification
- 4.3.1 Decoding Strategy		
- 4.3.2 ICL and Self-reasoning		
○ 4.4 Automatic Fact Checkers		
5 Factuality of Multimodal LLMs	**5 Challenges and Future Directions**	**6 Challenges in Automated Evaluation**
○ Existence, Attribute, and Relationship Factuality	○ Obstacles to Automated Fact-Checking	○ Ambiguity and Subjectivity in Assessment
○ Evaluation and Mitigation	○ Domain-Specific Factuality in LLMs	○ Atomic vs. Holistic Factuality
	○ Real-Time Verification	○ Dependence on External Knowledge
6 Challenges and Future Directions	**6 Conclusion**	**7 Future Directions and Open Challenges**
○ Learning Distributions vs. Facts	○ Current State of Research	○ Dynamic Knowledge Updates
○ RAG Bottlenecks (Latency/Multi-hop)	○ Key Challenges and Limitations	○ Bridging Fluency and Factual Accuracy
○ Improving Retrieval and Fact-checker Accuracy	○ Role of Benchmarking	○ Hallucination Snowball Effects
7 Conclusion		

Table 4: Comparison of survey outlines for "Factuality of Large Language Models: A Survey". STRUCTSURVEY introduces a taxonomy of factuality errors, distinguishes domain-specific and specialized benchmarks, and covers a broader set of mitigation techniques and future directions than SURVEYFORGE.

Gold Standard	SURVEYFORGE	STRUCTSURVEY
Example: "Context Dependent Semantic Parsing: A Survey"		
1 Introduction ○ CISP vs. CDSP ○ Challenges (Reference, Ellipsis)	**1 Introduction** ○ Rise of Context-Dependent Parsing ○ Challenges in CDSP	**1 Introduction** ○ Evolution of Semantic Parsing ○ Motivations for Context Dependence
2 Background ○ Meaning Representations (MRs) ○ Symbolic vs. Neural Approaches - SEQ2SEQ, SEQ2TREE, RNNs ○ **2.2 Evaluation** - Exact/Set Match Accuracy	**2 Foundations of Semantic Parsing** ○ Core Task and MRs ○ Traditional vs. Modern Approaches **5 Evaluation and Benchmarks** ○ Semantic Parsing Metrics	**2 Foundational Concepts** ○ Knowledge Representation ○ Algorithmic Approaches **5.4 Evaluation Metrics** ○ Benchmarking and Frameworks
3 Context Dependent Semantic Parsing ○ Definition of Local Context ○ **3.1 Symbolic Approaches** ○ **3.2 Neural Approaches** - Context-aware Encoders/Decoders - Copy Mechanisms & Pointers ○ **3.3 Neural-Symbolic Approaches**	**3 CDSP Methods** ○ Dialogue History Awareness ○ Temporal and Sequential Context ○ Interactive and Iterative Parsing	**3 Contextual Information** ○ Dialogue and Document Context ○ User-Specific and External Knowledge **4 Methods for CDSP** ○ Encoder-Decoder Architectures ○ Pointer Networks and Copying ○ Structure-Aware Models
4 Datasets and Resources ○ 4.1 Scenarios (Single vs. Multi-party) ○ 4.2 Context and Annotations	**4 Datasets and Tasks** ○ Cross-Domain and Text-to-SQL ○ Temporal and Interactive Datasets	**5 Datasets and Evaluation** ○ Overview of CDSP Datasets ○ Conversational SQL Datasets ○ Semantic Parse Correction
5 Challenges and Future Directions ○ Far-side Pragmatics/Implicatures ○ Causal Structure Discovery ○ Low-resource CDSP	**6 Challenges and Future Directions** ○ Ambiguity and Complexity of Context ○ Role of Interaction and Feedback ○ Limited Supervision	**7 Challenges and Future Directions** ○ Data Scarcity and Generalization ○ Ambiguity and Knowledge Integration ○ Multimodal and Situated Context
		6 Applications and Case Studies ○ Text-to-SQL and Code Generation ○ Conversational AI and QA

Table 5: Comparison of survey outlines for "Context Dependent Semantic Parsing: A Survey". The graph-based method captures context types, method categories such as encoder–decoder architectures, pointer mechanisms, and structure-aware models, and includes data scarcity among future challenges.

Gold Standard	SURVEYFORGE	STRUCTSURVEY
Example: "Paraphrase Generation: A Survey of the State of the Art"		
1 Introduction	**1 Introduction** ○ Motivation and Significance ○ Evolution of Techniques	**1 Introduction** ○ Defining Paraphrase Generation ○ Historical Evolution
4 Traditional Approaches ○ Rule-Based Approaches ○ Thesaurus-Based Approaches ○ SMT-Based Approach	**2 Traditional and Rule-Based** ○ Lexical Substitution ○ Syntactic Transformation ○ Template-Based Paraphrasing	**2 Traditional and Rule-Based** ○ Word Reordering and Substitution ○ Template-Based Systems ○ Early Data-Driven Approaches
5 Neural Approaches ○ 5.1 Encoder-Decoder Architecture - Encoding Side - Decoding Side	**3 Neural Network Models** ○ Sequence-to-Sequence Models ○ Transformer-based Generation ○ Adversarial and RL Approaches	**3 Neural Paraphrase Models** ○ Encoder-Decoder Architectures ○ Attention and Transformer Models ○ RNNs and Specialized Architectures
5.2 Improvements (Advanced) ○ A. Model-focused (Attention, VAE) ○ B. Attribute-focused (Syntax, Multi) - Explicit vs. Implicit Control	**4 Advanced Techniques** ○ Diverse Paraphrase Generation ○ Controlled Paraphrase Generation ○ Unsupervised and Weakly Supervised	**4 Advanced Enhancements** ○ Latent Representations for Diversity ○ Controlling Linguistic Properties ○ Adversarial and RL Improvements
2 Datasets / 3 Evaluation ○ 2 Datasets (PPDB, MSCOCO, etc.) ○ 3 Evaluation (BLEU, METEOR, ROUGE)	**4.5 Evaluation Metrics** ○ Challenges in Generation Evaluation	**5 Datasets, Eval, and Apps** ○ Overview of PG Datasets ○ Common Evaluation Metrics ○ Relationship: Datasets vs. Metrics
6 SOTA / 7 Conclusion ○ Pretrained Language Models ○ Controllable Generation ○ Stylistic Paraphrasing	**5 Applications / 6 Conclusion** ○ QA and Dialogue Systems ○ Data Augmentation ○ Bridging the Unsupervised Gap	**6 Conclusion** ○ Challenges and Open Problems ○ Emerging Trends ○ Unsupervised and Cross-Lingual

Table 6: Comparison of survey outlines for "Paraphrase Generation: A Survey of the State of the Art". The graph-based output includes a dedicated paraphrase-generation dataset section aligned with the gold survey and distinguishes it from evaluation.