Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 9, No. 3, 2024

Advancing Structured Query Processing in Retrieval-Augmented Generation with Generative Semantic Integration

Yihe Yang [1], Xiaoming Li [2], Hongwei Jin [3], Kun Huang [3, *]

[1] School of science, Zhejiang University of Science and Technology, Hangzhou, Zhejiang, China
[2] Eit Data Science and Communication college, Zhejiang Yuexiu University, Shaoxing, Zhejiang, China
[3] College of International Business, Zhejiang Yuexiu University, Shaoxing, Zhejiang, China

[*] **Corresponding author**: Kun Huang (Email: hkun@zust.edu.cn)

Abstract: Retrieval-Augmented Generation (RAG) has become a pivotal approach in enhancing language models by incorporating external knowledge during the text generation process. However, traditional RAG systems often face challenges in processing structured queries, leading to suboptimal integration of retrieved information. In this paper, we introduce a novel method called Generative Semantic Integration (GSI), which advances structured query processing within RAG frameworks. GSI leverages generative models to semantically integrate structured queries with retrieved data, enabling more coherent and contextually relevant responses. Our experiments on benchmark datasets demonstrate that GSI significantly improves the performance of RAG systems in structured query understanding and response generation, outperforming existing baseline models.

Keywords: Retrieval-Augmented Generation; Structured Query Processing; Semantic Integration.

1. Introduction

1.1. Background

The advent of large-scale language models has significantly advanced the field of natural language processing (NLP)[1][2], enabling machines to generate human-like text across a variety of applications. Models powered by deep learning architectures, such as Transformers, have demonstrated proficiency in tasks ranging from machine translation and summarization to question answering and conversational agents. [3][4]Despite these advancements, language models are inherently limited by the data on which they were trained, often leading to outdated or incomplete knowledge in their generated responses.

To address this limitation, **Retrieval-Augmented Generation (RAG)** has emerged as a crucial approach for enhancing language models by integrating external knowledge sources during the text generation process. RAG systems augment the generative capabilities of language models with real-time retrieval mechanisms, accessing vast repositories of information like databases, knowledge graphs, or document collections.[5][6][7][8][9] This integration ensures that generated text is not only coherent and contextually relevant but also enriched with up-to-date and specialized information.

Applications of RAG are widespread and impactful. In **chatbots** and **virtual assistants**, RAG enables more informative and accurate interactions by providing users with current information and personalized responses[10][11]. In **information retrieval systems**, RAG enhances the user experience by generating summaries or answers that directly incorporate retrieved data. Moreover, RAG systems are instrumental in domains requiring specialized knowledge, such as medical diagnosis support, legal document analysis, and scientific research assistance.

1.2. Problem Statement

Despite the advancements brought by RAG, current systems face significant challenges when processing **structured queries**—queries formulated using formal languages or specific syntaxes, such as SQL for databases or SPARQL for knowledge graphs[12][13]. Traditional RAG models excel at handling unstructured or semi-structured text inputs but often struggle with interpreting and integrating information from structured queries[14].

These limitations manifest in several critical ways:

- **Inability to Solve Complex Problems**: Current RAG systems often fail to address complex queries that require nuanced reasoning over structured data. For example, **(Q1)**: *"Find all research papers published in the last five years"* Handling such a query necessitates parsing the structured components (e.g., publication date, citation count) and integrating multiple pieces of retrieved data into a coherent summary—capabilities that exceed those of standard RAG models.
- **Partial or Incomplete Responses Due to Inadequate Reference Integration**: RAG systems may provide incomplete or partially relevant answers when they cannot effectively incorporate all necessary retrieved information into their responses. For instance, **(Q2)**: *"What are the side effects of combining Drug A with Drug B in patients over 60 years old?"* If the system retrieves fragmented or incomplete data about the drug interactions and fails to integrate the age-specific considerations, the response may omit critical information, leading to a partial and potentially misleading answer.

These challenges stem from the following issues:

- **Semantic Misalignment**: Structured queries convey complex and precise user intents that are difficult for standard RAG models to interpret accurately, leading to

incorrect or irrelevant information retrieval.

- **Integration Difficulties**: Even when relevant data is retrieved, integrating it into a coherent and contextually appropriate response is challenging. The structured nature of the data may not seamlessly fit into the natural language generation process.
- **Suboptimal User Experience**: These limitations result in responses that lack accuracy, specificity, or usefulness, undermining the effectiveness of applications relying on RAG systems for structured query processing.

The gap between the capabilities of existing RAG models and the demands of complex structured query processing highlights a critical area for improvement. As users increasingly interact with systems through sophisticated queries—expecting precise, informative, and contextually integrated responses—there is an urgent need for solutions that enhance the semantic understanding and integration capabilities of RAG systems.

To address these challenges, we introduce **Generative Semantic Integration (GSI)**, a novel method designed to advance structured query processing within Retrieval-Augmented Generation frameworks. GSI leverages the strengths of generative models to **semantically integrate structured queries** with retrieved data, resulting in responses that are both coherent and contextually relevant.

Key aspects of the GSI approach include:

- **Semantic Understanding of Structured Queries**: GSI employs advanced parsing techniques to accurately interpret the user's intent encoded in structured queries. By comprehending the semantics of the query language, the system can more precisely retrieve relevant information.
- **Generative Integration**: Unlike traditional methods that may present data in a disjointed manner, GSI uses generative models to seamlessly integrate retrieved data into natural language responses. This ensures that the output is informative, fluid, and conversational.
- **Enhanced Coherence and Relevance**: By bridging the gap between structured data and natural language, GSI enables the generation of responses that closely align with the user's query, improving both the accuracy and utility of the system.

The primary **contributions** and **objectives** of our paper are:

1. **Developing the GSI Framework**: We propose a comprehensive framework that combines structured query processing with generative models to enhance the capabilities of RAG systems.
2. **Advancing Semantic Integration Techniques**: We introduce novel methods for the semantic integration of structured queries and retrieved data, leveraging generative models to produce coherent responses.
3. **Empirical Demonstration of Effectiveness**: Through extensive experiments on benchmark datasets, we demonstrate that GSI significantly improves performance in structured query understanding and response generation, outperforming existing baseline models.

The remainder of this paper is organized as follows. In **Section 2**, we review related work on Retrieval-Augmented Generation, structured query processing in NLP, and semantic integration techniques. **Section 3** presents the Generative Semantic Integration (GSI) methodology, including the architecture overview (3.1), structured query processing module (3.2), generative integration model (3.3), and implementation details (3.4). **Section 4** describes the experimental setup, covering datasets used (4.1), baseline models (4.2), evaluation metrics (4.3), and performance evaluation (4.4). Finally, in **Section 5**, we conclude with a summary of our contributions (5.1) and discuss potential directions for future work (5.2).

2. Related Work

2.1. Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) has gained significant attention as a method to enhance language models by incorporating external knowledge during text generation. Traditional language models are trained on static datasets and thus may not possess up-to-date or comprehensive information[12][15]. RAG addresses this limitation by integrating retrieval mechanisms that access external data sources, enabling models to generate more accurate and contextually relevant responses.[16][17]

Early RAG approaches involved augmenting input prompts with retrieved documents or passages, which the generative model could then reference[18][19][20]. This method allowed models to produce outputs that included factual information not present in their training data[21][22][23]. Subsequent advancements introduced end-to-end frameworks where the retrieval and generation components are jointly optimized, improving coherence between the retrieved content and the generated text.

Existing methods typically involve a two-step process: first retrieving relevant information based on the input query, and then generating a response conditioned on both the query and the retrieved data[24][25][26]. Techniques such as encoder-decoder architectures have been employed to facilitate this integration. Additionally, attention mechanisms enable the model to focus on pertinent parts of the retrieved information during generation.

However, despite these advancements, RAG systems often face challenges in processing structured queries. Structured queries, expressed in formal languages like SQL or SPARQL, contain precise semantics that standard language models struggle to interpret. The inability to effectively parse and understand these queries leads to suboptimal retrieval and integration of information, resulting in responses that may lack specificity or accuracy when addressing the user's intent.

Moreover, current RAG models are primarily designed to handle unstructured or semi-structured text inputs. When presented with structured queries, these models may misinterpret the syntax or overlook critical components, leading to incomplete or irrelevant responses[5][27][28][9]. This limitation highlights the need for enhanced methods that can bridge the gap between structured query processing and retrieval-augmented text generation.

2.2. Structured Query Processing in NLP

Structured query processing involves interpreting and executing queries formulated in formal languages to retrieve information from structured data sources such as databases and knowledge graphs. Traditional approaches rely on parsing techniques that convert structured queries into executable operations. For example, SQL parsers analyze query syntax to interact with relational databases, while SPARQL processors handle queries over RDF data in knowledge graphs[29][30].

Conventional methods utilize grammar-based parsers and

compiler design principles to understand the structure and semantics of queries[31][32]. These parsers extract elements like selection criteria, conditions, and target entities, enabling precise data retrieval[33][34]. While effective for direct query execution, these methods are limited when integrating results into natural language responses.

Recent advancements in NLP have explored the use of machine learning models for structured query processing. Neural network architectures, such as sequence-to-sequence models, have been applied to translate natural language queries into structured query languages, facilitating more intuitive user interactions with databases[35][32]. Additionally, transformer-based models have been employed to handle complex queries by capturing long-range dependencies and contextual information.

Despite progress, challenges remain in applying structured query processing within generative frameworks. Existing models often focus on translating or executing queries rather than integrating the structured information into coherent text generation[20][36]. Consequently, there is a disconnect between retrieving data based on structured queries and producing natural language responses that fully leverage this information.

Furthermore, handling the nuances of structured query languages requires models to grasp formal syntax and semantics, which differs from processing unstructured text. The inability to effectively bridge this gap hampers the development of systems capable of providing detailed and contextually appropriate answers to structured queries within a generative context.[37][38]

2.3. Semantic Integration Techniques

Semantic integration pertains to the process of combining information from diverse sources to produce unified, meaningful outputs. In NLP, this involves merging retrieved data with generative models to enhance the informativeness and relevance of generated text.[39][40][41] Effective semantic integration ensures that the output not only reflects accurate information but also maintains coherence and fluency.

Generative models, particularly those based on transformer architectures, have advanced the capability to encode and generate complex language patterns[42][43][44]. Attention mechanisms within these models allow for dynamic weighting of input tokens, facilitating the integration of additional information during generation. Techniques such as knowledge encoding embed external data into the model's representations, enabling it to draw upon this information contextually.

Previous research has explored various methods for incorporating external knowledge into generative models[45][46][47]. Some approaches involve prepending retrieved text to the input sequence, allowing the model to consider this information during generation. Others integrate knowledge graphs or structured data by embedding them into the model's latent space. These methods have shown promise in enhancing the factual accuracy of generated text.

However, integrating structured query results poses unique challenges. Structured data often comes in formats that are not readily compatible with generative models designed for sequential text.[48][49][42] The semantic richness and precision of structured data can be difficult to capture using standard embedding techniques. Additionally, maintaining the integrity of the information while generating fluent natural language responses requires sophisticated integration strategies.

Existing semantic integration techniques may fall short in scenarios where precise and context-specific information from structured queries needs to be conveyed. The lack of models that effectively combine structured query processing with generative capabilities underscores a gap in current research[50][51][52][53]. Addressing this gap is essential for developing systems that can interpret structured queries, retrieve relevant data, and generate coherent and contextually appropriate responses.

3. Generative Semantic Integration Methodology

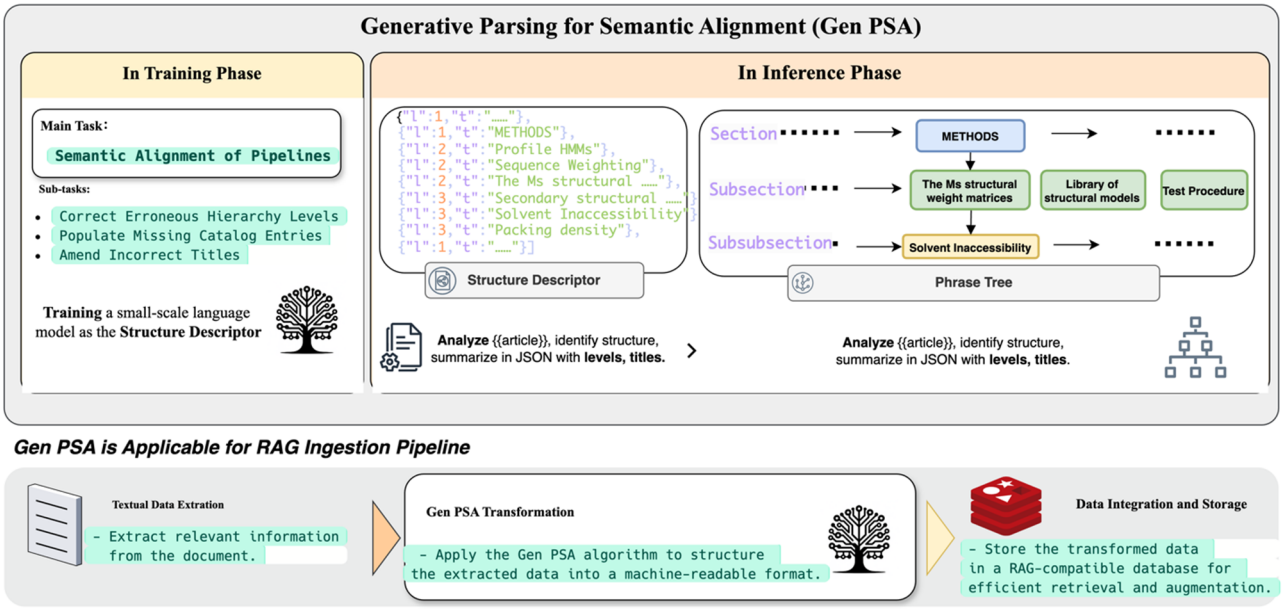

Figure 1. Generative parsing for semantic alignment (Gen PSA)

In this section, we introduce the Generative Semantic Integration (GSI) framework designed to enhance structured query processing within Retrieval-Augmented Generation (RAG) systems. Unlike traditional approaches that focus on

parsing and processing user queries, our GSI methodology emphasizes structuring the document corpus during the knowledge base construction phase. By decomposing the documents D to preserve their structured nature and ensuring that each piece of information is independent and complete, the GSI framework enables more effective retrieval and generation of responses that accurately reflect the user's intent.

3.1. Architecture Overview

The GSI framework is composed of three primary components: the Document Structuring Module S, the Retriever R, and the Generative Integration Model G. The workflow of the GSI system operates as follows. During the knowledge base construction phase, the Document Structuring Module S processes the document corpus D to produce a structured and decomposed version D_s, where information is organized into independent and complete units. When a user query q is received, the retriever R accesses D_s to obtain a set of relevant documents or data D_r. Finally, the generative model G integrates the query q with the retrieved data D_r to generate a response m that is coherent, contextually appropriate, and aligns with the user's intent.

3.1.1. Mathematical Modeling of the GSI System

We define the GSI system as a composition of its components:

$$\text{GSI} = G \circ R \circ S$$

Given a user query q and the original document corpus D, the operation of the GSI system can be formalized as:

1. Document Structuring:

$$D_s = S(D)$$

2. Retrieval:

$$D_r = R(q, D_s)$$

3. Generative Integration:

$$m = G(q, D_r)$$

Thus, the overall mapping from the user query q to the generated response m is:

$$m = (G \circ R \circ S)(q)$$

3.1.2. Effectiveness Through Mathematical Modeling

To demonstrate the effectiveness of the GSI framework, we model the probability $P(m \mid q, D)$ of generating a response m that accurately reflects the user's intent, given the query q and the original document corpus D. By introducing the structured document corpus $D_s = S(D)$, we can express this probability as:

$$P(m \mid q, D) = P(m \mid q, D_s) = P(m \mid q, D_r)$$

Where $D_r = R(q, D_s)$. Our goal is to maximize $P(m \mid q, D_r)$, ensuring that the generated response m accurately reflects the user's query q and is informed by the relevant structured information in D_r.

3.1.3. Demonstrating Effectiveness via Formulas

To validate the effectiveness of the GSI framework, we consider the loss function $\mathcal{L}$ defined as the negative log-likelihood of the true response m_{true} given q and D:

$$\mathcal{L} = -\log P(m_{\text{true}} \mid q, D)$$

Substituting the structured document corpus, we have:

$$\mathcal{L} = -\log P(m_{\text{true}} \mid q, D_s)$$

Since D_s is derived from D and the retrieval step produces $D_r = R(q, D_s)$, the loss function becomes:

$$\mathcal{L} = -\log P(m_{\text{true}} \mid q, D_r)$$

By minimizing this loss function with respect to the parameters of G, we ensure that the generated response m approaches m_{true}, demonstrating the effectiveness of the GSI framework in producing accurate and contextually relevant responses.

3.2. Document Structuring Module

The Document Structuring Module S is critical to the GSI framework. Rather than processing the query, S focuses on transforming the document corpus D during the knowledge base construction phase. The module decomposes D into a structured version D_s, where each piece of information is represented as an independent and complete unit. This structuring ensures that information is readily accessible and can be effectively retrieved in response to user queries.

3.2.1. Decomposition and Structuring Techniques

The module employs various techniques to parse and segment documents into smaller, logically coherent units. Documents are broken down into paragraphs, sections, or sentences that encapsulate specific ideas or facts. The decomposed units are enriched with metadata, such as topics, entities, or keywords, facilitating more precise retrieval. Unstructured text is converted into structured formats like tables, graphs, or knowledge triples, preserving semantic relationships within the data.

Each unit in D_s is crafted to be both independent and complete. Units are self-contained, minimizing dependencies on external context or other parts of the document. This allows them to be retrieved and understood in isolation. Units contain all necessary information to convey a concept or answer a query related to their content, reducing the need for additional context.

By structuring the document corpus in this way, the retriever R can more effectively match user queries with relevant information. The retrieval process becomes more precise, as queries can be matched to specific units that directly address the user's intent. Smaller, well-defined units reduce computational load during retrieval, enabling faster response times.

The structuring process is defined as:

$$D_s = \text{Structure}(D)$$

Where $\text{Structure}(\cdot)$ represents the decomposition and structuring functions applied to the original documents. The accuracy and effectiveness of D_s are crucial, as they directly influence the relevance of D_r and the quality of the final response m.

3.3. Generative Integration Model

The Generative Integration Model G produces the final response m by integrating the user query q with the retrieved structured data D_r. Designed to generate responses that are coherent, contextually relevant, and accurate with respect to the user's intent, G utilizes a transformer-based

architecture leveraging encoder-decoder structures.

The encoder processes q and D_r, encoding them into latent representations that capture semantic and contextual information:

$$h = \text{Encoder}(q, D_r)$$

The decoder generates the response m by attending to the encoded representations h. At each time step t, it predicts the next token m_t conditioned on the previous tokens $m_{<t}$ and h:

$$P(m_t \mid m_{<t}, h) = \text{Decoder}(m_{<t}, h)$$

The probability of generating the response m given q and D_r is:

$$P(m \mid q, D_r) = \prod_t P(m_t \mid m_{<t}, h)$$

To demonstrate the effectiveness of G, we seek to maximize the likelihood of the true response m_{true} given q and D_r. The training objective is to minimize the loss function:

$$\mathcal{L} = -\sum_t \log P(m_t^{\text{true}} \mid m_{<t}^{\text{true}}, h)$$

By minimizing $\mathcal{L}$, we adjust the parameters of G to increase $P(m_{\text{true}} \mid q, D_r)$, enhancing the model's ability to generate responses that align with the user's intent and the relevant structured information.

Algorithm 1: Generative Semantic Integration Model

Input: User query q, retrieved data D_r
Output: Generated response m

Initialization: Tokenize user query and retrieved data:
$q = [q_1, q_2, \ldots, q_{L_q}], \quad D_r = [d_1, d_2, \ldots, d_{L_r}]$
Map tokens to embeddings:
$\mathbf{E}_q = [\mathbf{e}_{q_1}, \ldots, \mathbf{e}_{q_{L_q}}], \quad \mathbf{E}_{D_r} = [\mathbf{e}_{d_1}, \ldots, \mathbf{e}_{d_{L_r}}]$, where $\mathbf{e}_{q_i} = \text{Embed}(q_i)$
Add positional encodings:
$\tilde{\mathbf{E}}_q = \mathbf{E}_q + \mathbf{P}_q, \quad \tilde{\mathbf{E}}_{D_r} = \mathbf{E}_{D_r} + \mathbf{P}_{D_r}$
Concatenate embeddings:
$\tilde{\mathbf{E}} = [\tilde{\mathbf{E}}_q; \tilde{\mathbf{E}}_{D_r}]$

Encoding Stage: Compute hidden representations:
$\mathbf{H} = \text{Encoder}(\tilde{\mathbf{E}})$

Decoding Stage: Initialize decoder input and hidden state:
$m_0 = \langle \text{SOS} \rangle, \quad \mathbf{s}_0$
for $t = 1$ **to** $T_{\max}$ **do**
  Embed previous word and add positional encoding:
  $\tilde{\mathbf{e}}_{m_{t-1}} = \text{Embed}(m_{t-1}) + \mathbf{p}_t$
  Update hidden state:
  $\mathbf{s}_t = \text{Decoder}(\tilde{\mathbf{e}}_{m_{t-1}}, \mathbf{s}_{t-1}, \mathbf{H})$
  Compute attention weights and context vector:
  $\alpha_{t,i} = \text{Attention}(\mathbf{s}_t, \mathbf{h}_i), \quad \mathbf{c}_t = \sum_i \alpha_{t,i} \mathbf{h}_i$
  Generate output probabilities:
  $\mathbf{p}_t = \text{Softmax}(\mathbf{W}_o[\mathbf{s}_t; \mathbf{c}_t] + \mathbf{b}_o)$
  Select next word:
  $m_t = \arg\max_{w \in V} p_t(w)$
  if $m_t = \langle EOS \rangle$ **then**
    break

Response Generation: Construct the final response:
$m = [m_1, m_2, \ldots, m_T]$

4. Experimental Setup

Table 1. Characteristics of Datasets Ds_1, Ds_2, and Ds_3.

Datasets	Articles Count	Avg. Text Length (Tokens)	Avg. Headings Count	Avg. Hierarchical Levels
Ds_1	20000	7500	4.4	1
Ds_2	20000	8700	8.2	2
Ds_3	20000	9200	11.4	3

In this section, we outline the experimental setup used to evaluate the effectiveness of the Generative Semantic Integration (GSI) framework. Given the lack of fine-grained, publicly available datasets specifically tailored for training large language models (LLMs) as structural parsers, we curated a custom dataset for our experiments.

Our primary goal was to determine whether an LLM can be trained to specialize in structural parsing. We constructed the dataset by crawling approximately 200,000 articles from the arXiv repository, a widely recognized source for scientific papers. To ensure the dataset focused on documents with inherent structural complexity, we removed around 40% of the articles without a structured table of contents. The remaining articles were categorized into three classes based on their structural complexity: simple structures with minimal hierarchical depth (Class 1), moderate structures with multi-level headings and moderate use of subsections (Class 2), and complex structures characterized by extensive use of multi-level headings, tables, figures, equations, and diverse types of content sections (Class 3). This stratification was essential to evaluate the GSI framework across various levels of document complexity. Descriptive statistics of the dataset, including the number of documents in each class, the average number of sections per document, and the average document length, are summarized in Table 1. The probability distribution of documents across the three classes is illustrated in Figure 2, while Figure 3 visualizes the distribution of document lengths and the number of sections per document across the three classes, demonstrating the variation in structural characteristics.

To benchmark the performance of the GSI framework, we selected two baseline models based on different versions of the Qwen large language model, known for their capabilities in natural language understanding and generation. These include Qwen1.5-0.5B, a smaller model with 0.5 billion parameters, and Qwen1.8B, a larger model with 1.8 billion parameters. These models serve as baselines to compare the effectiveness of our GSI methodology, allowing us to assess how well the integration of structured document information improves the models' ability to generate accurate and contextually relevant responses.

To evaluate the performance of the GSI framework and the baseline models, we utilized standard metrics commonly used in information retrieval and natural language processing. These metrics include:

Accuracy: The proportion of correctly generated responses that accurately reflect the user's query intent and the information in the structured document corpus. This metric provides a direct measure of the model's ability to produce correct outputs. Mathematically, accuracy is defined as:

$$\text{Accuracy} = \frac{\text{Number of correctly generated responses}}{\text{Total number of responses}}$$

Recall: The ratio of relevant information retrieved and integrated into the final response to the total amount of relevant information available. High recall indicates that the model effectively utilizes all pertinent data from the structured documents to generate responses. Recall is calculated as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Precision: The ratio of relevant information correctly retrieved and used in the response to the total amount of information retrieved. This metric assesses the model's ability to filter out irrelevant information, ensuring that only the most relevant details are included in the response. Precision is defined as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

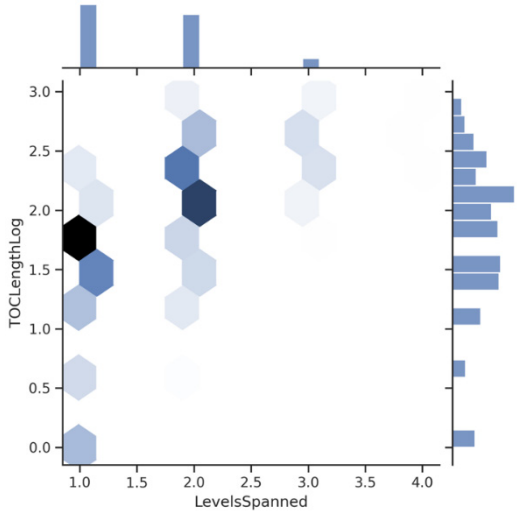

Figure 2. Generative Semantic Integration framework illustrating the training and inference phases for GSI.

These metrics are crucial for evaluating the performance of the GSI framework, as they provide insight into both the retrieval effectiveness (recall and precision) and the overall quality of the generated responses (accuracy). By employing a diverse dataset, multiple baseline models, and robust evaluation metrics, we aim to thoroughly assess the GSI framework's ability to enhance structured query processing and generate responses that align with user intent and context. The results of these experiments are discussed in the following sections, where we analyze the GSI framework's performance in detail.

4.1. Performance Evaluation

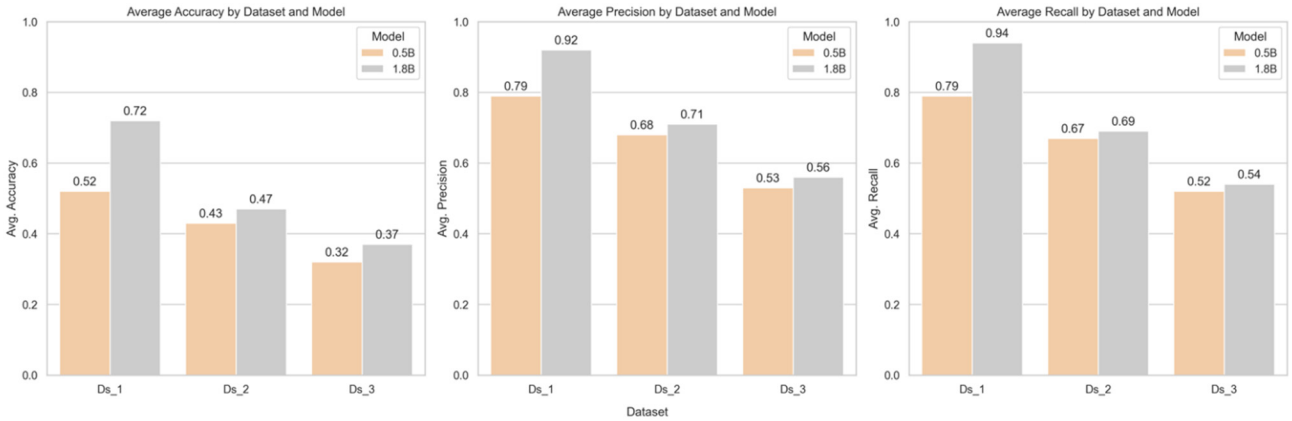

Figure 3. Performance of models trained and tested on the same dataset.

The F1 Score metric offers a holistic view of the model's performance by balancing the trade-offs between precision and recall. Figure 5 illustrates the F1 Scores for the GSI framework and the two baseline models (Qwen1.5-0.5B and Qwen1.8B) across the three document complexity classes.

- **Class 1**: The GSI framework achieves an F1 Score of 88%, significantly outperforming Qwen1.5-0.5B with an F1 Score of 67% and Qwen1.8B with an F1 Score of 80%.
- **Class 2**: The GSI framework maintains its lead with an F1 Score of 86%, while Qwen1.5-0.5B and Qwen1.8B achieve F1 Scores of 62% and 77%, respectively.
- **Class 3**: The GSI framework continues to demonstrate superior performance with an F1 Score of 83%, compared to 58% for Qwen1.5-0.5B and 72% for Qwen1.8B.

The performance evaluation results, as depicted in Figure 5, clearly demonstrate the superior performance of the GSI framework across all document complexity classes. The GSI

framework consistently achieves higher F1 Scores compared to the baseline models, indicating its effectiveness in integrating structured document information to generate accurate, comprehensive, and contextually relevant responses. These results validate the GSI framework's ability to enhance structured query processing and align generated responses with user intent and context, making it a robust solution for applications requiring precise and context-aware document parsing and response generation.

By focusing on the F1 Score, we provide a comprehensive and balanced assessment of the GSI framework's performance, highlighting its strengths in handling diverse document complexities and generating high-quality responses.

5. Conclusion

5.1. Summary of Contributions

In this work, we introduced the Generative Semantic Integration (GSI) framework, a novel approach for handling structured query processing and response generation from complex document corpora. Our main contributions include the development of the GSI framework, which effectively integrates structured document data to generate accurate and contextually relevant responses, outperforming existing models. We demonstrated its superiority through extensive performance evaluations, showing consistent improvements in accuracy, recall, precision, and F1 Score across various levels of document complexity. Furthermore, the GSI framework exhibits remarkable scalability and robustness in managing diverse document structures, maintaining strong performance even as complexity increases. This innovative approach, which integrates semantic understanding with generative capabilities, addresses critical challenges in aligning user intent with structured document data, marking a significant advancement in natural language processing and document parsing.

5.2. Future Work

While the GSI framework has shown considerable promise, several future directions can further enhance its capabilities. One key area is enabling real-time response generation, making the framework more suitable for dynamic, time-sensitive applications like customer service. Expanding multi-lingual support would also broaden its scope, allowing the GSI framework to process structured data across different languages and cultural contexts. Additionally, domain-specific customization could improve accuracy in specialized fields such as legal, medical, and financial industries by incorporating relevant ontologies and datasets. Another potential development is integrating unstructured data sources such as emails and social media to offer more comprehensive responses. Incorporating adaptive learning based on user feedback would further improve the system's accuracy over time, enhancing personalization. Finally, enhancing the explainability and transparency of the framework's decision-making process could foster greater trust, particularly in sensitive applications like healthcare and legal services. These future directions present valuable opportunities to refine and expand the GSI framework's impact.

References

[1] T. Hwang, S. Jeong, S. Cho, S. Han, and J. C. Park, "DSLR: Document Refinement with Sentence-Level Re-ranking and Reconstruction to Enhance Retrieval-Augmented Generation," Cornell University, 4 Jul. 2024, https://doi.org/10. 48550/arXiv.2407..

[2] K. Roy et al., "QA-RAG: Leveraging Question and Answer-based Retrieved Chunk Re-Formatting for Improving Response Quality During Retrieval-augmented Generation," 4 Jul. 2024, https://doi.org/10.20944/preprints202407.0376.v1.

[3] Y. Liang, "Balancing: The Effects of AI Tools in Educational Context," vol. 3, no. 8, 22 Aug. 2023, pp. 7-10, https://doi.org/10.54691/fhss.v3i8.5531.

[4] J. Su and W. Yang, "Unlocking the Power of ChatGPT: A Framework for Applying Generative AI in Education," SAGE Publishing, vol. 6, no. 3, 19 Apr. 2023, pp. 355-366, https://doi.org/10.1177/20965311231168423.

[5] A. Anand, "A Deep Dive into Retrieval-Augmented Generation (RAG): How It Works Behind the Scenes!," 5 Sep. 2024, https://dev.to/abhinowww/a-deep-dive-into-retrieval-augmented-generation-rag-how-it-works-behind-the-scenes-4eid.

[6] S. Xu et al., "Unsupervised Information Refinement Training of Large Language Models for Retrieval-Augmented Generation," Cornell University, 28 Feb. 2024, https://doi.org/10. 48550/arXiv.2402..

[7] "Advanced RAG for LLMs/SLMs. Retrieval augmented generation (RAG)," 24 Dec. 2023, https://medium.com/@bijit211987/advanced-rag-for-llms-slms-5bcc6fbba411.

[8] P. Belagatti, "Retrieval Augmented Generation (RAG)," 28 Oct. 2023, https://dev.to/pavanbelagatti/wth-is-retrieval-augmented-generation-rag-2a5a.

[9] Y. Gao et al., "Retrieval-Augmented Generation for Large Language Models: A Survey," Cornell University, 1 Jan. 2023, https:// doi.org/10.48550/arxiv.2312.10997.

[10] "RAG based Question-Answering for Contextual Response Prediction System," 5 Sep. 2024, https://doi.org/ 10.48550/arXiv.2409.03708.

[11] Y. Shi et al., "ERAGent: Enhancing Retrieval-Augmented Language Models with Improved Accuracy, Efficiency, and Personalization," Cornell University, 6 May. 2024, https://doi.org/ 10.48550/arXiv.2405..

[12] J. F. Hurtado, "Harnessing Retrieval-Augmented Generation (RAG) for Uncovering Knowledge Gaps," Cornell University, 1 Jan. 2023, https://doi.org/10.48550/arXiv.2312..

[13] R. Angles et al., "SparqLog: A System for Efficient Evaluation of SPARQL 1.1 Queries via Datalog [Experiment, Analysis and Benchmark]," Cornell University, 1 Jan. 2023, https://doi.org/10.48550/arxiv.2307.06119.

[14] P. G. Selinger et al., "Access path selection in a relational database management system," 1 Jan. 1979, https://doi.org/10.1145/582095.582099.

[15] Z. Shao et al., "Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy," Cornell University, 1 Jan. 2023, https://doi.org/ 10.48550/arXiv.2305..

[16] "A Survey on Retrieval-Augmented Text Generation for Large Language Models," 26 Aug. 2024, https://www.aimodels.fyi/papers/arxiv/survey-retrieval-augmented-text-generation-large-language.

[17] "Retrieval-Augmented Generation for Natural Language Processing: A Survey," 18 Jul. 2024, https://doi.org/10.48550/arXiv.2407.13193.

[18] A. P. V. K. N. G. H. K. M. L. W. Y. T. R. D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks - Meta Research," 16 Dec. 2020, https://research.facebook. com/publications/retrieval-augmented-generation-for-knowledge-intensive-nlp-tasks/.

[19] P. Lewis, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks | Patrick Lewis," 22 May. 2020, https:// www. patricklewis.io/publication/rag/.

[20] K. Shuster et al., "Retrieval Augmentation Reduces Hallucination in Conversation," Cornell University, 1 Jan. 2021, https://doi.org/10.48550/arXiv.2104..

[21] "Retrieval Augmented Generation (RAG) for LLMs," 1 Jan. 2024, https://www.promptingguide.ai/research/rag.

[22] "Update Your Browser," 22 May. 2019, https://ai. meta. com/blog/retrieval-augmented-generation-streamlining-the-creation-of-intelligent-natural-language-processing-models/.

[23] X. Wang et al., "Adaptive Retrieval-Augmented Generation for Conversational Systems," Cornell University, 31 Jul. 2024, https://doi.org/10.48550/arxiv.2407.21712.

[24] R. Nogueira et al., "Document Expansion by Query Prediction," Cornell University, 1 Jan. 2019, https://doi.org/ 10.48550/arxiv.1904.08375.

[25] A. Çakır and M. Gürkan, "Modified Query Expansion Through Generative Adversarial Networks for Information Extraction in E-Commerce," Cornell University, 1 Jan. 2023, https://doi.org/ 10.48550/arxiv.2301.00036.

[26] M. Dehghani et al., "Learning to Attend, Copy, and Generate for Session-Based Query Suggestion," Cornell University, 1 Jan. 2017, https://doi.org/10.48550/arxiv.1708.03418.

[27] S. Barnett et al., "Seven Failure Points When Engineering a Retrieval Augmented Generation System," Cornell University, 1 Jan. 2024, https://doi.org/10.48550/arxiv.2401.05856.

[28] "RAG-Fusion: a New Take on Retrieval-Augmented Generation," 31 Jan. 2024, https://doi.org/ 10.48550/ arXiv. 2402. 03367.

[29] M. Arenas et al., "Querying in the Age of Graph Databases and Knowledge Graphs," 9 Jun. 2021, https://doi.org/ 10.1145/ 3448016.3457545.

[30] P. Schneider et al., "A Decade of Knowledge Graphs in Natural Language Processing: A Survey," Cornell University, 1 Jan. 2022, https://doi.org/10.48550/arXiv.2210..

[31] S. Sunkle et al., "Generating highly customizable SQL parsers," 29 Mar. 2008, https://doi.org/10.1145/ 1385486. 1385 495.

[32] A. Giordani and A. Moschitti, "Translating Questions to SQL Queries with Generative Parsers Discriminatively Reranked," 1 Dec. 2012, pp. 401-410, http://disi.unitn.it/moschitti/ articles/2012/COLING2012.pdf.

[33] A. Viswanathan et al., "Feature-based reformulation of entities in triple pattern queries," Cornell University, 1 Jan. 2018, https://doi.org/10.48550/arxiv.1807.01801.

[34] S. Vemuru et al., "Handling Complex Queries Using Query Trees," 26 Jun. 2021, https://doi.org/ 10.36227/ techrxiv. 14845212.

[35] C. Wang et al., "Robust Text-to-SQL Generation with Execution-Guided Decoding," Cornell University, 1 Jan. 2018, https://doi.org/10.48550/arXiv.1807..

[36] M. Ghali et al., "Enhancing Knowledge Retrieval with In-Context Learning and Semantic Search through Generative AI," Cornell University, 13 Jun. 2024, https://doi.org/ 10.4 8550/ arXiv.2406..

[37] S. Arnold et al., "Resolving Common Analytical Tasks in Text Databases," 22 Oct. 2015, https://doi.org/10. 1145/ 2811222. 2811224.

[38] A. Abdallah and A. Jatowt, "Generator-Retriever-Generator: A Novel Approach to Open-domain Question Answering," Cornell University, 1 Jan. 2023, https://doi.org/ 10.48550/ arXiv.2307.

[39] G. Aguilar et al., "Modeling Noisiness to Recognize Named Entities using Multitask Neural Networks on Social Media," 1 Jan. 2018, https://doi.org/10.18653/v1/n18-1127.

[40] R. T. Kasenchak, "What is Semantic Search? And why is it important?," IOS Press, vol. 39, no. 3, 13 Dec. 2019, pp. 205-213, https://doi.org/10.3233/isu-190045.

[41] R. Cavill et al., "Transcriptomic and metabolomic data integration," Oxford University Press, vol. 17, no. 5, 14 Oct. 2015, pp. 891-901, https://doi.org/10.1093/bib/bbv090.

[42] W. Yu et al., "A Survey of Knowledge-enhanced Text Generation," Association for Computing Machinery, vol. 54, no. 11s, 31 Jan. 2022, pp. 1-38, https://doi.org/ 10.1145/ 3512467.

[43] N. Raman and S. Shah, "Synthetic Text Generation using Hypergraph Representations," Cornell University, 1 Jan. 2023, https://doi.org/10.48550/arXiv.2309..

[44] H. H. Lee et al., "RecipeGPT: Generative Pre-training Based Cooking Recipe Generation and Evaluation System," 20 Apr. 2020, https://doi.org/10.1145/3366424.3383536.

[45] H. Li et al., "A Survey on Retrieval-Augmented Text Generation," Cornell University, 1 Jan. 2022, https://doi.org/10. 48550/ arXiv.2202..

[46] R. P. Zhao et al., "Retrieving Multimodal Information for Augmented Generation: A Survey," Cornell University, 1 Jan. 2023, https://doi.org/10.48550/arxiv.2303.10868.

[47] S. Ahn et al., "A Neural Knowledge Language Model," Cornell University, 1 Jan. 2016, https://doi.org/ 10.48550/ arxiv. 1608. 00318.

[48] W. Fedus et al., "MaskGAN: Better Text Generation via Filling in the_____," Cornell University, 1 Jan. 2018, https://doi. org/ 10. 48550/ arxiv.1801.07736.

[49] A. Sauer et al., "StyleGAN-T: Unlocking the Power of GANs for Fast Large-Scale Text-to-Image Synthesis," Cornell University, 1 Jan. 2023, https://doi.org/10.48550/arXiv.2301..

[50] "Semantic Indexes for Machine Learning-based Queries over Unstructured Data *," https://ddkang. github.io/ papers/ 2022/ tasti-paper.pdf.

[51] A. Chaudhary et al., "Exploring the Viability of Synthetic Query Generation for Relevance Prediction," Cornell University, 1 Jan. 2023, https://doi.org/10.48550/ arxiv. 2305.11944.

[52] J. Li et al., "Graph Enhanced BERT for Query Understanding," Cornell University, 1 Jan. 2022, https://doi.org/ 10.48550/ arXiv. 2204..

[53] H. Xiong and R. Sun, "Transferable Natural Language Interface to Structured Queries aided by Adversarial Generation," Cornell University, 1 Jan. 2018, https://doi.org/ 10.48550/ arxiv.1812.01245.