A CONTEXT-ENRICHED NL-TO-SQL AGENT FOR PREMIER LEAGUE FOOTBALL DATABASE QUERYING

John Hedlund-Fay

Department of Computer Science, University of Sheffield, Sheffield, UK

ABSTRACT

Enterprise NL-to-SQL generation remains brittle in high-compliance environments where query correctness depends on external, prescriptive regulatory logic. While Retrieval-Augmented Generation (RAG) offers a solution, its efficacy in bridging this semantic gap remains under-explored. We present a 400-question benchmark in the football domain and compare a modular decomposition pipeline (RAG-R) with a monolithic agentic architecture (RAG-C). Results indicate that while RAG-C underperformed the best non-RAG baseline, ten-shot Chain-of-Thought (CoT), RAG-R achieved superior performance. Notably, RAG-R outperformed the CoT baseline by 0.116 in average Exact Set Match (EM) and showed a 0.278 EM gain for the highest-difficulty domain-specific queries ($p < 0.001$). These findings demonstrate the importance of task decomposition in prescriptive RAG systems—where correctness relies on reconciling ambiguous intent with rigid regulatory logic rather than schema knowledge alone.

KEYWORDS

NL-to-SQL, Retrieval Augment-Generation (RAG), Modular Decomposition, Benchmark Construction Domain-Specific Knowledge (DSK), Enterprise NLP

1. INTRODUCTION

The English professional football pyramid represents a high-stakes, high-compliance enterprise environment characterized by extreme semantic density. The Premier League (PL) operates under a hierarchical governance structure where database query correctness is not merely a function of accurate table joins, but of interpreting vast, external corpora of rules from the PL, the Football Association (FA), and the International Football Association Board (IFAB).

The underlying relational database (RDB) of the English professional football pyramid manages billions of data points across dozens of schemata, ranging from medical and visa status to financial regulations. Leveraging foundational operational experience within this domain, we identify that a significant semantic gap exists between natural language (NL) prompts and executable SQL.

There are two primary categories of ambiguity in this domain: logical and linguistic. While the RDB contains timestamps for player registration and loan start dates, the logic governing those dates resides in the FA Handbook (e.g., the "24-hour registration deadline" rule). Without grounding in domain-specific knowledge (DSK), standard large language models (LLMs) fail to capture these latent constraints—logic required for query correctness that is absent from the database schema. Additionally, domain-specific phrases such as "top goalscorer" fall short when

Bibhu Dash et al: CSEAI, SPNLP, EDUR - 2026
pp. 37-53, 2026. IJCI – 2026 DOI:10.5121/ijci.2026.150303

converting NL to a structured query, as they require substantial semantic unpacking that is prone to hallucination without external grounding.

A recent incident illustrates the severe consequences of a breakdown in accurate and timely access to information. Grimsby Town, in an English Football League Cup match, fielded a player who was ineligible. The player had joined on loan the day before, but was registered one minute and 59 seconds past the deadline. Grimsby Town received a fine of £20,000, attributing the non-compliance to a computer error [1]. Similarly, in 2019, Liverpool FC was fined £200,000 for a similar registration offense [2]. These failures are not due to database corruption, but to the inability of human agents or traditional business intelligence (BI) tools to synthesize real-time RDB values with static, legalistic external rules.

This environment provides an ideal testbed for evaluating Retrieval-Augmented Generation (RAG) architectures in "jargon-heavy," highly regulated enterprise settings. Enterprise-level NL-to-SQL requires multi-hop reasoning over frequently modified external corpora to yield structured answers. External regulatory texts introduce implicit temporal constraints, conditional eligibility logic, and categorical jargon that cannot be directly inferred from tables and columns alone.

1.1. Hypothesis and Contributions

The primary research hypothesis of this work is that an NL-to-SQL system augmented with a DSK RAG pipeline will achieve statistically significant improvements in Exact Set Match (EM) accuracy compared to non-augmented in-context learning (ICL) baselines.

To test this, we evaluate two competing architectural paradigms: a direct context-augmentation agent (RAG-C) and an intermediate, modular query-refinement pipeline (RAG-R). This allows us to investigate not only if external knowledge improves performance, but how the structural integration of that knowledge impacts the model's reasoning capabilities and susceptibility to hallucinations.

To investigate this hypothesis, we provide the following contributions:

- We introduce a new benchmark of NL-to-SQL prompt pairs requiring the synthesis of DSK from the association football domain. Unlike previous benchmarks, our work explicitly tests the integration of unstructured regulatory text with structured schema metadata.

- We establish a baseline upon this benchmark to measure the capabilities and limitations of unaugmented ICL techniques in resolving latent logical constraints.

- We contrast a monolithic agentic workflow (RAG-C) with a modular refinement pipeline (RAG-R). We demonstrate the quantified EM gains of the RAG-R pipeline over the best ICL baseline and reveal how direct agentic augmentation degrades performance via context-stuffing in high-jargon environments.

The remainder of the paper is organized as follows: Section 2 reviews related work in RAG and NL-to-SQL; Section 3 details the benchmark construction; Section 4 describes our architectural implementations; Section 5 presents the experimental results; and Section 6 concludes with the implications for enterprise NLP.

2. RELATED WORK

In this section, we situate our research within the evolving landscape of ICL for NL-to-SQL, the theoretical underpinnings of RAG for complex reasoning, and the architectural trade-offs between agentic and modular systems.

2.1. ICL for NL-to-SQL

Recent advancements in NL-to-SQL have shifted from fine-tuned encoder-decoder models like T5 [3] toward ICL using LLMs. In the context of NL-to-SQL, chain-of-thought (CoT) prompting has proven particularly valuable for tackling the problem of schema comprehension [4]. By thinking step-by-step, the model can first identify the relevant tables based on the user's question, then determine the necessary join paths, and finally select the correct columns and aggregation functions, constructing a plan before outputting a SQL query.

However, as state-of-the-art (SotA) models have improved at simple question answering across many domains and datasets, multi-hop question answering (MHQA) has become an increasingly researched task. MHQA tasks involve answering NL questions which necessitate extracting and combining multiple pieces of information and doing multiple steps of reasoning [5]. The efficacy of ICL prompting techniques diminishes against these more difficult queries and has required a move beyond prompt engineering alone into methods which can retrieve and incorporate external knowledge.

2.2. RAG for Complex Reasoning

Older benchmarks such as Spider 1.0—upon which many of the pioneering studies into LLMs for NL-to-SQL were evaluated—operated under a "Closed World Assumption"; that is, while they aimed to represent real-world enterprise-level NL-to-SQL tasks involving long queries with many join operations over many databases [6], this dataset intentionally excluded questions requiring external knowledge [7].

More recent benchmarks in NL-to-SQL such as the arithmetic, commonsense, and hypothetical reasoning bench (Archer) and Spider 2.0 have expanded their scope to include tasks necessitating the incorporation of external knowledge to better test the reasoning capabilities of models [8, 9]. Of particular relevance, the Archer challenge lays out the observed gap in the research:

> Incorporating external knowledge into text-to-SQL tasks presents significant challenges in general. Firstly, models need to compare information from natural language questions with the relational database to determine if external knowledge is required. Secondly, models need to extract the most relevant knowledge from external knowledge bases. Last but not least, the process of integrating this knowledge into the text-to-SQL generation process remains largely unexplored [9].

RAG has emerged as the standard for grounding LLMs in external corpora. When compared to a fine-tuning pipeline, RAG has a reduced start-up cost and, importantly for domains with frequently-updated "living" external corpora, a fine-tuned model would have to be fine-tuned again while with RAG the updated corpora would merely have to be re-embedded while the underlying models and prompts would remain valid [10]. Further, we chose RAG over fine-tuning for explicability. Fine-tuning over PLMs acts like a black box for added DSK and is an anti-pattern for high-stakes, high-compliance enterprises where safety and auditability in AI are paramount.

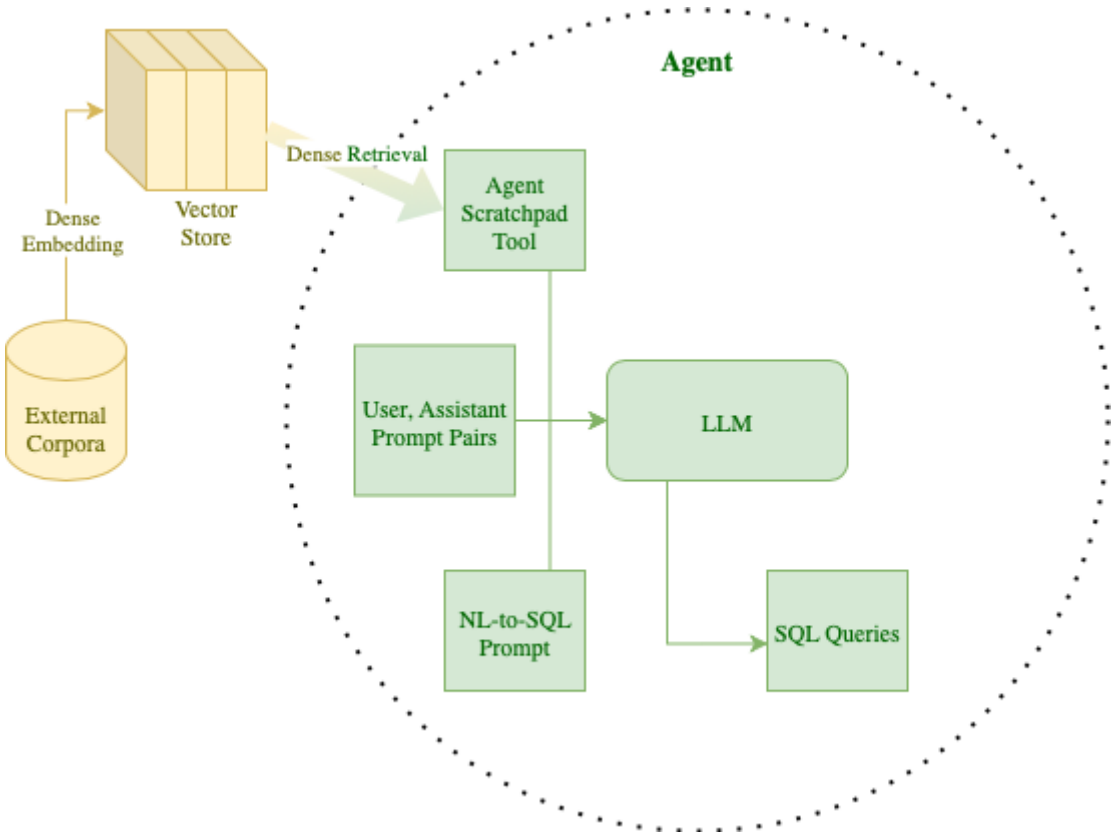

Figure 1. The one-model architecture of how RAG-C composes the parts into an agent.

2.3. Task Decomposition vs. Direct Augmentation

While RAG has been powerful in addressing limitations of hallucination and outdated knowledge in standalone LLMs, traditional RAG techniques have struggled with knowledge-intensive MHQA tasks. Approaches that have successfully tackled this drawback have included combining RAG with CoT [11] as well as query refinement [12]. Consequently, the choice between agentic workflows and modular decomposition constitutes a critical design decision within RAG.
Many SotA models have attempted to solve the parts of NL-to-SQL jointly, as an agent, rather than as a discrete pipeline. This achieves the goal of end-to-end learning, whereby a single AI agent handles the entire process from raw data to final output. However, this monolithic approach is more prone to cascading failures; if any part of the end-to-end agent fails, the entire solution fails, making diagnostics difficult. For instance, it is highly susceptible to the "Lost in the Middle" phenomenon. As observed by Liu et al. [13], LLM performance degrades significantly when critical information is buried within a long context window due the primacy and recency biases of many current LLMs.

By contrast, a query refinement pipeline, herein referred to as RAG-R, separates the interpretative task (applying DSK to refine the prompt) from the generative task (translating refined prompts to SQL), addressing the cascading failures common in end-to-end agents.
We evaluate these paradigms through two distinct workflows: a monolithic, tool-use agent (RAG-C, Fig. 1) and a modular, two-stage refinement pipeline (RAG-R, Fig. 2). While RAG-C seeks to solve retrieval and generation jointly, RAG-R enforces a structural separation between domain-knowledge synthesis and SQL construction. The specific implementation parameters for these architectures are detailed in Section 4.

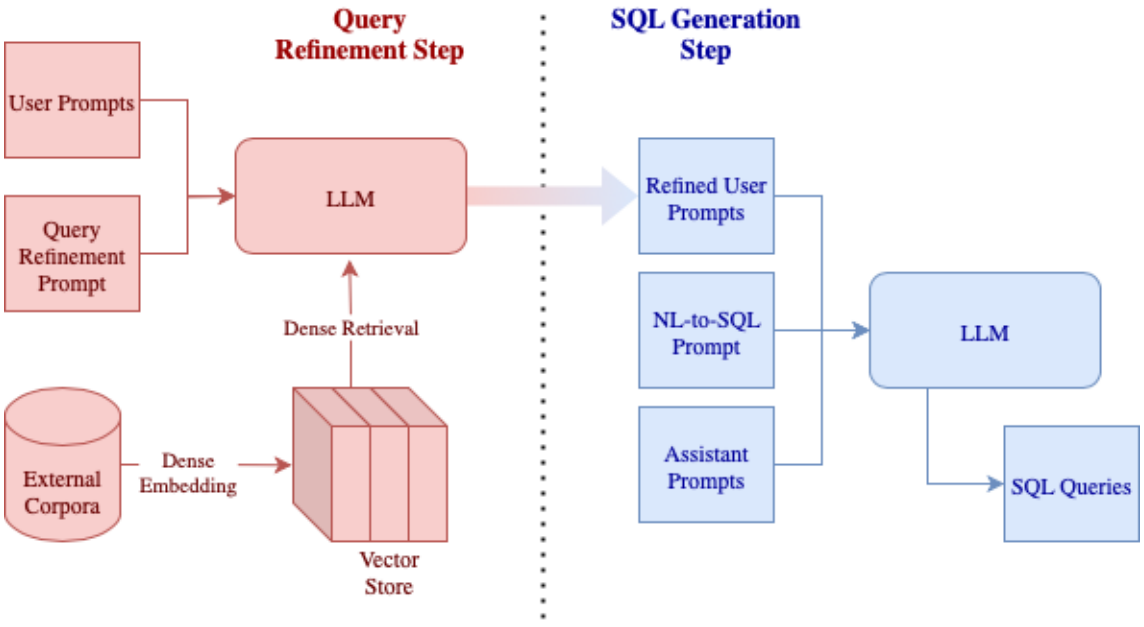

Figure 2. The two-model architecture of how the RAG-R pipeline works.

2.4. RAG over Related Domains

Existing applications of RAG in NL-to-SQL have distinct limitations. Guo et al. utilized RAG to retrieve syntactically similar SQL templates to aid generation [14]. While effective for syntax, this approach does not address semantic ambiguity regarding domain rules. While there is relatively little research on using LLMs for the football domain specifically, Schilling et al. successfully used RAG for question answering within the football domain based on individual match data from *Understat.com* [15]. This suggested that football domain context enrichment might be conducive to improving prediction accuracy. However, their approach was only able to accurately answer questions about particular football matches.

Our research bridges this gap. By embedding DSK about rules and administration, we move the task from descriptive analytics to prescriptive regulatory compliance. Yu et al. used RAG to extract encyclopedic knowledge for commonsense reasoning [16], which we mirror, applying it here to the rigid, legalistic constraints of the PL.

3. BENCHMARK CONSTRUCTION

3.1. Data Source

To create a representative subset of the proprietary relational database used by the English football pyramid, we constructed a PostgreSQL database using Transfermarkt data from 2012–2025 [17]. This dataset captures the "long tail" of enterprise data, containing records for 400 clubs, 32,000 players, and over 148,000 match appearances. By constructing this benchmark from public data rather than proprietary administrative systems, we ensure our findings are fully reproducible and available for future research. Figure 3 illustrates the core conceptual entity-relationships. To ensure complete technical transparency, the exact Data Definition Language

(DDL) schema, including specific data types and foreign key constraints, is provided in Appendix A.

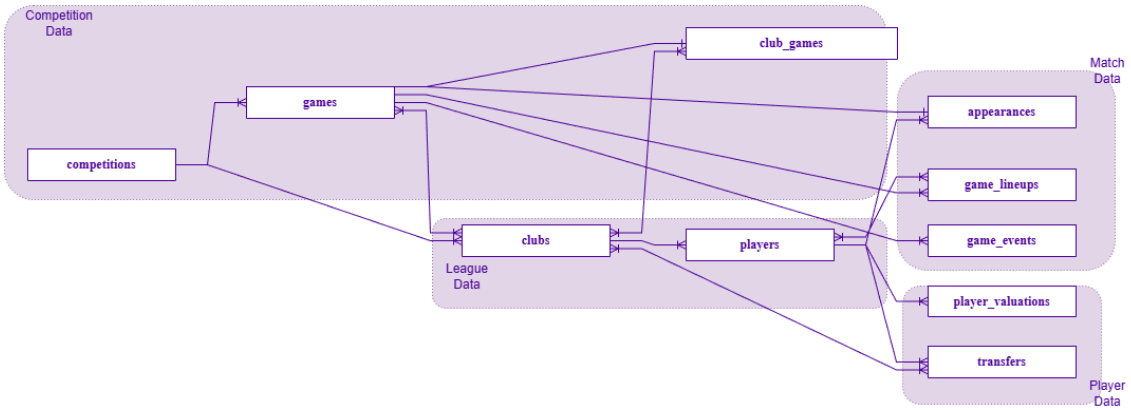

Figure 3. Conceptual entity-relationship model of the Transfermarkt dataset.
Cardinality is shown in crow's foot notation.

While many traditional text-to-SQL benchmarks have utilized SQLite for its portability, we deliberately employed PostgreSQL to better align with the "enterprise-level realism" shift seen in recent frameworks like Spider 2.0 and BIRD-CRITIC [18]. PostgreSQL's strict schema enforcement presents a rigorous test for LLMs, as we retained raw, unsanitized data structures. For example, the pipeline had to reconcile natural language queries (e.g., "38th round") with specific VARCHAR labels in the database (e.g., "38. Matchday") rather than relying on simplified integer mappings. This ensures that the generated queries reflect production-grade standards, requiring the model to bridge the semantic gap in data representation rather than relying on idealized academic approximations.

3.2. Authoring the Benchmark

To evaluate different ICL prompting paradigms, we authored a benchmark of 400 (user, assistant) prompt pairs stratified across four distinct difficulty tiers. Additionally, for CoT evaluation, we authored auxiliary (user, hint) prompt pairs that provide syntactic guidance (e.g., identifying required SQL operations) while strictly excluding the DSK necessary for Difficulty 4 queries. All natural language questions, corresponding SQL queries, and semantic hints were authored by a single domain expert with professional operational experience in English football database administration.

Table 1. Examples of the benchmark difficulty stratification.

Difficulty	Description	Average Complexity	Example NL Prompt
1	Simple Retrieval	JOIN: 0.62	"List the names of players who play for Arsenal"
	Single-table lookups or simple joins. No complex aggregations.	WITH:0.00	
2	Multi-Table Aggregation	JOIN: 2.52	"Find the latest market value for the most expensive player ever sold by Tottenham Hotspur."
	Requires joining 3-4 tables, basic filtering, and sorting.	WITH: 0.17	
3	Syntactic Complexity	JOIN: 3.63	"Which team dropped the most points from a winning position in the 2024 Premier League?"
	Complex analytics requiring window functions, nested sub-queries, and multiple CTEs.	WITH: 0.92	
4	Semantic Stress Test	JOIN: 2.52	"Which club automatically qualified for the FA Community Shield in 2024"
	Queries requiring external Domain-Specific Knowledge (DSK) to map ambiguous terms to the schema.	WITH: 0.68	

Table 1 profiles the benchmark's syntactic complexity. Difficulties 1–3 exhibit a clear progression in structural difficulty, reflected by the increasing prevalence of WITH (CTEs) and JOIN operations. However, the jump to Difficulty 4 is semantic rather than syntactic. As shown in the keyword distribution (Table 2), Tier 4 queries are often less syntactically dense than Tier 3, yet remain unsolvable via standard schema-linking because they rely on external DSK. For example, while the Tier 3 "points dropped" query is structurally complex, its logic is fully contained within the schema. Conversely, the Tier 4 "Community Shield" query appears simple but requires the unstated external rule that the PL winner automatically qualifies for the trophy.

3.3. Structural Robustness and Breadth

Table 2. Benchmark SQL keyword distributions by difficulty.

Difficulty	ORDER BY	GROUP BY	Nested	DISTINCT	PARTION BY	WITH
1	22	13	2	2	0	0
2	34	25	19	19	4	15
3	84	69	84	84	31	83
4	73	78	63	63	6	61

Table 2 highlights the distribution of complex SQL operations across difficulty tiers, demonstrating the benchmark's ability to test advanced analytics, including Common Table Expressions (CTEs), window functions (PARTITION BY), and conditional logic (CASE WHEN). To ensure comprehensive enterprise realism, we incorporated a broad spectrum of SQL

keywords—such as COALESCE, CAST, UNION, LIMIT, and EXTRACT—for data cleaning and integration. As noted in Section 3.2, syntactic complexity peaks at Difficulty 3, while Difficulty 4 maintains high structural demands to ensure that the semantic stress test is not trivialized by simple SQL syntax.

4. METHODOLOGY

4.1. Large Language Model Configuration

We conducted the experiments using gpt-4o-2024-08-06. Initial proof-of-concept experiments and trial runs for debugging were performed locally on Llama 3.1 but due to memory constraints this local development approach was abandoned in favor of an OpenAI API call during RAG tuning. We selected GPT-4o from among the available GPT family of models because, at the time of experimentation, there had been no comprehensive academic surveys comparing the performance of the GPT line of LLMs since GPT-4 [19]. Recent methods on the leaderboards for major benchmarks such as Spider 2.0 and the Big Bench for Large-scale Database Grounded Text-to-SQL Evaluation (BIRD) [20] consistently show GPT-4o-based approaches outperforming GPT-4 and GPT-4-Turbo-based approaches [21, 22, 23]. The temperature parameter of LLMs controls the randomness or creativity of the generated text by influencing the probability distribution of the next word. The temperature parameter will be set to T = 0.0 for all experiments to ensure deterministic and reproducible outputs.

4.2 Baseline ICL Configurations

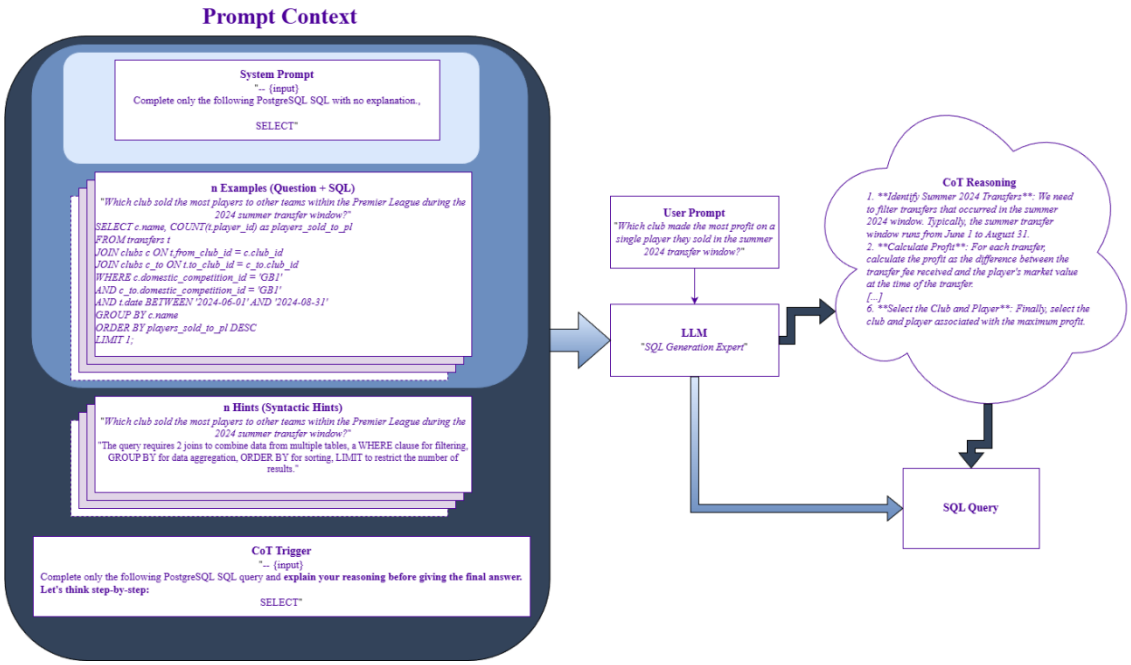

Figure 4. Differences in the ICL prompting methods visualized with a single prompt example.

To evaluate baseline performance, we utilized standard ICL methods, progressing through zero-shot, few-shot, and few-shot CoT prompting, as summarized in Fig. 4. For zero-shot prompting, a standard system prompt based on the one developed by Liu, et al. [24] was used. This system

prompt consists of table names and the corresponding column names they contain. During tuning we also evaluated this against the system prompt used by Pourezza and Rafiei for DIN-SQL [25]. This prompt includes the SQL CREATE queries used to generate the tables and examples of the data in the low-cardinality columns of each table. We did this to isolate the impact of the prompt structure itself in addition to establishing the zero-shot baseline.

For few-shot prompting, the same DIN-SQL system prompt as in zero-shot was used along with examples from the (user, assistant) pair benchmark consisting of the NL question and the correct SQL query for those examples. We employed dynamic few-shot selection using an adapted LangChain SemanticSimilarityExampleSelector. We utilized FAISS [26] to index the embeddings of the benchmark questions, retrieving the k-most semantically similar examples (excluding the target query) to populate the context window. The number of shots was tuned over $k \in \{1, 2, 5, 10\}$.

Finally, for few-shot CoT prompting, the few-shot prompt is modified in two key ways. First, we appended the corresponding syntactic hints (described in Sec. 3.2) to each retrieved few-shot example. Second, the prompt is suffixed by the phrase: "Let's think step-by-step:" from Kojima et al.'s seminal piece on prompting CoT [27], which triggers the model to utilize these hints in its reasoning process.

4.3 RAG System Implementation

To evaluate the RAG-C and RAG-R architectures, we constructed a vector knowledge base from three public corpora: the *Premier League Handbook 2025–26* [28], the *FA Handbook 2024–25* [29], and the *IFAB Laws of the Game* [30]. Documents were processed using a fixed-size chunking strategy with a 10-token overlap. Chunks were embedded using the e5-large-v2 model [31], with inputs prefixed according to E5's training objectives ("query: " and "passage: ").

```
You are a knowledgeable and precise query transformation engine
particularly related to the administration and regulation of
professional association football

Your SOLE TASK is to rewrite a user's question to be more
specific and self-contained by embedding definitions from the
provided context.

You do not answer questions; you only reformulate them.

Instructions:

Primary Directive: Your SOLE TASK is to function as a query
transformation engineer. You will rewrite the user's 'Original
Prompt' into an 'Improved Prompt' using the provided 'Context'

CRITICAL RULE: You must not, under any circumstances, answer
the user's question or provide any information beyond what is
required to rephrase the question itself. Your job is only to
rewrite the question, not to respond to it.

Integration and Specificity: Rewrite the prompt to be more
specific by finding the definitions of any rules or ambiguous
terms within the context. You must replace the ambiguous term
in the original prompt with its explicit definition from the
context.

Truthfulness: if the context does not provide a specific
definition to clarify the prompt, restate the original prompt
without modification.

Final Output: Your response must ONLY be the text of the
'Improved Prompt'. Do not include any preamble, explanations,
or conversational text.

Here is the original prompt and context for you to work with:

\nOriginal Prompt: {prompt} \nContext: {context} \nImproved
Prompt:
```

Figure 5. Query refinement prompt for the RAG-R pipeline.

A critical challenge in the RAG-R pipeline was instruction drift, where the LLM defaults to conversational question-answering rather than query transformation. To mitigate this, we developed a refinement prompt (Fig. 5) utilizing a mixture of positive directives and strict negative constraints to de-bias the input before it reaches the SQL generator.

4.4. Experimental Setup and Evaluation

Few-shot and few-shot CoT prompting were tuned over the hyperparameter of number of shots k over the values $k \in \{1, 2, 5, 10\}$. These are evaluated against the standard metrics set forth by the original Spider benchmark [6]. Execution Accuracy (EX) evaluates, as a baseline, that the output SQL queries are syntactically valid and run without errors, while Exact Set Match Accuracy (EM) measures whether the set of results returned by the predicted SQL query exactly match the outputs of the gold standard query.

$$Faithfulness = \frac{C_{supported}}{C_{total}} \quad (1)$$

$$Relevancy = \frac{1}{N}\sum_{i=1}^{N} \frac{E_{g_i} \cdot E_o}{\|E_{g_i}\| \|E_o\|} \quad (2)$$

Then, we tuned the retrieval process over the hyperparameters of chunk size and retrieved chunks. Holding the chunk overlap constant at 10 tokens and the number of retrieved chunks constant at 3, the chunk size (s) will be tuned over values of $s \in \{128, 256, 512\}$. The optimal size will be selected based on the end-to-end EM accuracy of the architecture. Second, using the best chunk size, the number of retrieved chunks (k) will be tuned over values of $k \in \{3, 5, 8\}$ again selecting the value that maximizes EM accuracy on the validation set. To specifically evaluate the retrieval quality in the RAG-R pipeline, we measure faithfulness (whether generated SQL is directly implied by retrieved context [32]) and relevancy (whether retrieved context is directly related to the user query). The equations for faithfulness and relevancy are given in Eq. 1 and Eq. 2 respectively. In calculating relevancy, artificial questions are generated to reflect the intent of the response and then relevancy is calculated as the average over the sum of all cosine similarities between the embedding of the user input E_o and the embedding of each of the generated questions E_{g_i} .

$$\square^2 = \frac{(b-c)^2}{b+c} \quad (3)$$

Finally, to determine if the performance difference between RAG-R and RAG-C is statistically significant, we apply McNemar's test, particularly focusing on the marginal homogeneity of successful predictions on Difficulty 4 queries. The test statistic of McNemar's test is given in Eq. 3 where b indicates pairs where the case succeeds and the control fails and c indicates pairs where the control succeeds and the case fails.

5. RESULTS AND ANALYSIS

5.1. Baseline Performance and Analysis

We first established baselines using standard In-Context Learning (ICL) prompting. Tuning on the validation set revealed that the DIN-SQL prompt format significantly outperformed standard zero-shot prompting (0.225 vs 0.075 Avg EM). This was primarily due to its ability to resolve domain-specific nomenclature, such as mapping "Premier League" to specific competition IDs.
For few-shot experiments, we identified a trade-off between stability and reasoning. Based on the validation set results, the five-shot setting was the most robust performer, achieving the highest or joint-highest EM scores on Difficulty 1, 2, and 3. While the ten-shot setting performed better on Difficulty 4—resulting in a tied average EM—the perfect execution accuracy (1.0 Avg EX) of

the five-shot setting across all difficulties gave it the edge. It was thus selected as the primary few-shot configuration for the test set.

Finally, the ten-shot CoT configuration achieved the highest reasoning performance overall (0.4 Avg EM). Consequently, the ten-shot CoT model was selected as the foundation for evaluating our RAG architectures.

5.2. RAG Architecture Performance and Analysis

We evaluated two RAG strategies: the monolithic agent (RAG-C) and the modular refinement pipeline (RAG-R). As hypothesized in Section 2.3, the agentic approach suffered from "context stuffing." Higher k values ($k = 8$) introduced hallucinations, empirically demonstrating the "Lost in the Middle" phenomenon. While the retrieved context was often accurate, the model would hallucinate tables or columns based on the enriched text that were absent from the RDB. Future work (Section 6.1) will investigate whether specific prompt ordering can mitigate this primacy and recency bias.

The pipeline approach proved more robust. Table 3 summarizes the sensitivity analysis for the RAG-R query refinement step.

Table 3. Validation set analysis for RAG-R configurations.

Configuration	Latency	Faithfulness	Relevancy	Diff. 4 EM	Avg. EM
$k = 3, s = 128$	1.73s	0.07	0.07	0.4	0.525
$k = 3, s = 256$	4.04s	0.10	0.12	0.3	0.5
$k = 3, s = 512$	7.94s	0.17	0.15	0.4	0.475
$k = 5, s = 128$	2.40s	0.03	0.10	0.6	0.6
$k = 8, s = 128$	5.75s	0.05	0.10	0.3	0.475

While larger chunks ($s = 512$) improved intermediate Faithfulness, smaller, more focused chunks ($s = 128$) with a retrieval count of $k = 5$ yielded the highest downstream EM accuracy. Notably, the low Faithfulness scores on lower difficulties are a byproduct of the "Truthfulness" directive (Section 4.3); for non-semantic questions (Tiers 1–3), the model correctly identifies that no external DSK is required and avoids injecting unnecessary context.

5.3. Quantitative Analysis

Table 4 presents the final performance of all optimal configurations against the held-out test set ($N = 360$).

Table 4. Test Set Performance for all models.

Model	Difficulty 1		Difficulty 2		Difficulty 3		Difficulty 4		Average	
	EX	EM	EX	EM	EX	EM	EX	EM	EX	EM
Zero-shot	0.911	0.322	0.844	0.156	0.756	0.044	0.844	0.056	0.839	0.283
Five-shot	0.944	0.567	0.967	0.356	0.889	0.167	0.933	0.233	0.933	0.451
CoT	0.989	0.633	1.0	0.444	0.922	0.278	0.9	0.244	0.953	0.511
RAG-C	0.967	0.6	0.956	0.444	0.844	0.3	0.833	0.233	0.9	0.493
RAG-R	0.911	0.489	0.989	0.622	0.956	0.556	0.933	0.522	0.947	0.627

The RAG-R pipeline achieved superior EM across all difficulties significantly outperforming the baseline. Most notably, it improved performance on Difficulty 4 (Semantic Stress Test) queries by 0.278 over the CoT baseline. McNemar's test confirms this improvement is statistically significant ($\chi^2 = 15.6, p < 0.001$).

Conversely, the RAG-C agent failed to outperform the baseline. This suggests that mixing context retrieval, reasoning, and SQL generation into a single prompt overloads the model's attention, whereas RAG-R's decoupled approach allows the SQL expert to focus purely on syntax and logic using a pre-clarified prompt.

5.4. Qualitative Analysis

To evaluate the practical efficacy of the RAG-R pipeline, we conducted a case study on a Difficulty 4 query: "Which Premier League clubs are closest to the maximum limit of non-homegrown players in their 2024 squad?" The CoT baseline correctly identified that "non-homegrown" relates to foreign players but failed to account for the specific limit (Rule A.1.245). It simply sorted clubs by raw foreign player counts, failing to bridge the semantic gap between the user's intent and the regulatory constant.

The failure of the RAG-C agent compared to the success of RAG-R, however, reveals a more fundamental divergence in architectural robustness:

- The RAG-C agent retrieved Rule A.1.272 (the legal criteria for homegrown status). As shown in the AgentExecutor logs (Appendix B), the model attempted to calculate "homegrown" status from first principles using player ages and academy dates. It became "distracted" by the depth of the retrieved knowledge and ignored the existing foreigners_number column in the database in favor of a complex, impossible calculation.

- The RAG-R transformation engine identified the need for a specific limit and retrieved Rule A.1.245. It passed a refined prompt to the SQL generator: *"Which Premier League clubs are closest to the maximum limit of non-homegrown players considering each club is required to have a minimum of eight registered homegrown players?"* As shown in Fig. 6, this pre-clarification allowed the CoT model to bridge the semantic gap by integrating the regulatory constant directly into the logic.

```
"To determine which Premier League clubs are closest to the
maximum limit of non-homegrown players in their 2024 squad, we
need to consider the following:

1. **Premier League Rules**: Each club is required to have a
minimum of eight registered homegrown players in its squad
list. This means that the maximum number of non-homegrown
players allowed is 17 (since the total squad size is 25).

2. **Data Requirements**: We need to identify clubs that are
part of the Premier League and check their number of foreign
players (non-homegrown players).

3. **Filtering and Sorting**: We will filter clubs that are in
the Premier League and sort them by the number of non-homegrown
players, in descending order, to find those closest to the
limit of 17.

4. **SQL Query**: We will join the 'clubs' table with the
'competitions' table to filter for Premier League clubs and
then order by the number of foreign players.

Here is the SQL query to achieve this:

```sql SELECT c.name, c.foreigners number FROM clubs c JOIN
competitions comp ON c.domestic competition id =
comp.competition id WHERE comp.name = 'premier-league' ORDER BY
(17 - c.foreigners number) ASC; ```

**Explanation**: - We join the 'clubs' table with the
'competitions' table to filter for clubs that are part of the
Premier League. - We select the club name and the number of
foreign players ('foreigners number'). - We order the results
by the difference between the maximum allowed non-homegrown
players (17) and the current number of foreign players. This
helps us identify clubs that are closest to the limit of 17
non-homegrown players."
```

Figure 6. RAG-R output demonstrating successful integration of the Rule A.1.245 regulatory constant (17-player limit) into the generated SQL logic.

This case study confirms that for enterprise-level SQL generation, isolating domain-specific knowledge retrieval into a dedicated refinement stage is significantly more robust than direct context-augmentation in a single-turn agentic architecture.

6. CONCLUSIONS

This research addressed the critical semantic gap in enterprise NL-to-SQL generation. Publicly available LLMs and conventional ICL prompt engineering techniques alone were insufficient to tackle enterprise-level NL-to-SQL. Further, the nature of the corpora in the management and administration of professional football as annually-updated living documents ill-suits them to fine-tuning.

By constructing a novel 400-question benchmark grounded in the real-world complexity of the PL, we rigorously evaluated two retrieval-augmented strategies. Our findings provide strong empirical evidence that decoupling reasoning from generation is essential for domain-specific accuracy.
```

The direct augmentation agent (RAG-C) failed to outperform the baseline, confirming that "stuffing" complex regulatory text into the generation context exacerbates the "Lost in the Middle" phenomenon, leading to hallucinations. The two-stage refinement pipeline (RAG-R) achieved a 0.278 improvement in Exact Match (EM) accuracy on the most difficult queries ($p < 0.001$). By isolating the retrieval and disambiguation of domain knowledge (DSK) from the SQL generation step, RAG-R effectively bridges the gap between ambiguous user intent and rigid database schemata.

These results suggest that in legalistic enterprise environments, the agentic trend of end-to-end solving may be less effective than modular pipelines that enforce a structural separation of concerns.

6.1. Limitation and Future Work

While these results are promising, several avenues remain for future investigation.

- The current benchmark relies on a single expert annotator. Future iterations will employ cross-validation with multiple domain experts and expand the dataset to include synthetic, anonymized data from the full proprietary RDB schema.

- Our evaluation relied on Execution Accuracy (EX) and Exact Set Match (EM). However, matching result sets does not guarantee logical alignment. Future work will incorporate semantic equivalence metrics, following the methodology of Zhong et al. [33], to verify that the generated SQL logic matches the gold standard even when multiple distinct queries could yield the same output set. This will help distinguish between queries that are functionally identical and those that are merely incidentally correct.

- To address the minor performance degradation observed in RAG-R on low-complexity queries, we will explore adaptive retrieval strategies. By using confidence scoring to bypass the refinement step for simple questions, we can ensure the pipeline is only invoked when semantic ambiguity is detected.

- The failure of RAG-C points to a significant research gap regarding LLM primacy and recency biases. Future inquiry is needed to determine if a dynamic reordering strategy—placing prioritized information at the specific "sweet spots" of an LLM's context window—can mitigate these biases without necessitating modular decoupling.

- Ultimately, this methodology provides a blueprint for prescriptive RAG—a system capable not just of retrieving facts, but of adhering to the living, ruling documents of an enterprise.

ACKNOWLEDGEMENTS

I wish to extend my profound gratitude to my dissertation supervisor, Dr. Xi Wang, for his guidance, unwavering support, and insightful feedback, all of which were vital to this project. I'd also like to thank my former software development manager and teammates for the opportunity to learn from them while doing interesting work in a convivial environment. Finally, I would like to thank the Football Administration and Premier League for allowing me to work with them on their relational databases; the expertise I gained was invaluable to this project.

REFERENCES

[1] Colman, J. Grimsby town fined over ineligible player in manchester united win, Sep 2025.
[2] Dobson, M. Liverpool fined for fielding ineligible player Pedro Chirivella in Carabao Cupe-player, Oct. 2019.
[3] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.
[4] Kim, S., Joo, S. J., Kim, D., Jang, J., Ye, S., Shin, J., and Seo, M. The cot collection: Improving zero-shot and few-shot learning of language models via chain-of-thought fine-tuning, 2023.
[5] Mavi, V., Jangra, A., and Jatowt, A. Multi-hop question answering, 2024.
[6] Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., Zhang, Z., and Radev, D. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task, 2019.
[7] Pan, J. Z., Razniewski, S., Kalo, J.-C., Singhania, S., Chen, J., Dietze, S., Jabeen, H., Omeliyanenko, J., Zhang, W., Lissandrini, M., Biswas, R., de Melo, G., Bonifati, A., Vakaj, E., Dragoni, M., and Graux, D. Large language models and knowledge graphs: Opportunities and challenges, 2023.
[8] Zheng, D., Lapata, M., and Pan, J. Archer: A human-labeled text-to-SQL dataset with arithmetic, commonsense and hypothetical reasoning. In Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers) (St. Julian's, Malta, Mar. 2024), Y. Graham and M. Purver, Eds., Association for Computational Linguistics, pp. 94–111.
[9] Lei, F., Chen, J., Ye, Y., Cao, R., Shin, D., Su, H., Suo, Z., Gao, H., Hu, W., Yin, P., Zhong, V., Xiong, C., Sun, R., Liu, Q., Wang, S., and Yu, T. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows, 2025.
[10] Balaguer, A., Benara, V., de Freitas Cunha, R. L., de M. Estevão Filho, R., Hendry, T., Holstein, D., Marsman, J., Mecklenburg, N., Malvar, S., Nunes, L. O., Padilha, R., Sharp, M., Silva, B., Sharma, S., Aski, V., and Chandra, R. Rag vs fine-tuning: Pipelines, tradeoffs, and a case study on agriculture, 2024.
[11] Trivedi, H., Balasubramanian, N., Khot, T., and Sabharwal, A. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions, 2023.
[12] Chan, C.-M., Xu, C., Yuan, R., Luo, H., Xue, W., Guo, Y., and Fu, J. Rq-rag: Learning to refine queries for retrieval augmented generation, 2024.
[13] Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts, 2023.
[14] Guo, C., Tian, Z., Tang, J., Li, S., Wen, Z., Wang, K., and Wang, T. Retrieval-augmented gpt-3.5-based text-to-sql framework with sample-aware prompting and dynamic revision chain, 2023.
[15] Schilling, A., Anurathan, J., Mühlberger, J., Gerschner, F., Rössle, M., Theissler, A., and Klaiber, M. *Querying Football Matches for Event Data: Towards Using Large Language Models*. 09 2024, pp. 216–227.
[16] Yu, W., Zhu, C., Zhang, Z., Wang, S., Zhang, Z., Fang, Y., and Jiang, M. Retrieval augmentation for commonsense reasoning: A unified approach, 2022.
[17] Cariboo, D. Football data from transfermarkt, Aug 2024. [Online]. Available: https://www.kaggle.com/datasets/davidcariboo/player-scores.
[18] Li, J., Li, X., Qu, G., Jacobsson, P., Qin, B., Hui, B., Si, S., Huo, N., Xu, X., Zhang, Y., Tang, Z., Li, Y., Widjaja, F., Zhu, X., Zhou, F., Huang, Y., Papakonstantinou, Y., Ozcan, F., Ma, C., and Cheng, R. SWE-SQL: Illuminating LLM Pathways to Solve User SQL Issues in Real-World Applications, 2025.
[19] Kalyan, K.S. A Survey of GPT-3 Family Large Language Models Including ChatGPT and GPT-4, 2023
[20] Li, J., Hui, B., Qu, G., Yang, J., Li, B., Li, B., Wang, B., Qin, B., Cao, R., Geng, R., Huo, N., Zhou, X., Ma, C., Li, G., Chang, K. C. C., Huang, F., Cheng, R., and Li, Y. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls, 2023.
[21] Shkapenyuk, V., Srivastava, D., Johnson, T., and Ghane, P. Automatic metadata extraction for text-to-sql, 2025.
[22] Deng, M., Ramachandran, A., Xu, C., Hu, L., Yao, Z., Datta, A., and Zhang, H. Reforce: A text-to-sql agent with self-refinement, consensus enforcement, and column exploration, 2025.
[23] Wang, B., Ren, C., Yang, J., Liang, X., Bai, J., Chai, L., Yan, Z., Zhang, Q.-W., Yin, D., Sun, X., and Li, Z. Mac-sql: A multi-agent collaborative framework for text-to-sql, 2025.

[24] Liu, A., Hu, X., Wen, L., and Yu, P. S. A comprehensive evaluation of chatgpt's zero-shot text-to-sql capability, 2023.
[25] Pourreza, M., and Rafiei, D. Din-sql: Decomposed in-context learning of text-to-sql with self-correction, 2023.
[26] Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.-E., Lomeli, M., Hosseini, L., and Jégou, H. The faiss library, 2025.
[27] Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners, 2023.
[28] The Premier League. *The PL Handbook 2025/2026*. The Premier League, Jul 2025.
[29] The Football Association. *The FA Handbook 2024/2025*. The Football Association, 2024.
[30] The International Football Association Board. *The IFAB Laws of the Game*. The International Football Association Board, 2025.
[31] Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Majumder, R., and Wei, F. Text embeddings by weakly-supervised contrastive pre-training, 2024.
[32] Yu, H., Gan, A., Zhang, K., Tong, S., Liu, Q., and Liu, Z. *Evaluation of-Retrieval-Augmented Generation: A Survey*. Springer Nature Singapore, 2025, p. 102–120.
[33] Zhong, R., Yu, T., and Klein, D. Semantic evaluation for text-to-sql with distilled test suites, 2020