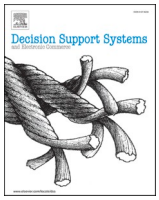
Decision Support Systems 199 (2025) 114553

Contents lists available at ScienceDirect

Decision Support Systems

journal homepage: www.elsevier.com/locate/dss



Prompting large language models based on semantic schema for text-to-Cypher transformation towards domain Q&A

Yuwei Wan [a], Zheyuan Chen [b,c], Ying Liu [a,*], Chong Chen [d], Michael Packianather [a]

[a] Department of Mechanical Engineering, School of Engineering, Cardiff University, Cardiff CF24 3AA, UK
[b] Guangzhou Institute of Industrial Intelligence, Guangzhou 511458, China
[c] Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China
[d] Guangdong Provincial Key Laboratory of Cyber-Physical System, Guangdong University of Technology, Guangzhou 510006, China

ARTICLE INFO

Keywords:
Text-to-Cypher
Prompt engineering
Semantic schema
Large language models
In-context learning
Neo4j
Cypher query

ABSTRACT

Translating natural language inquiries into executable Cypher queries (text-to-Cypher) is a persistent bottleneck for non-technical teams relying on knowledge graphs (KGs) in fast-changing industrial settings. Rule and template converters need frequent updates as schemas evolve, while supervised and fine-tuned parsers require recurring training. This study proposes a schema-guided prompting approach, namely text-to-Cypher with semantic schema (T2CSS), to align large language models (LLMs) with domain knowledge for producing accurate Cypher. T2CSS distils a domain ontology into a lightweight semantic schema and uses adaptive filtering to inject the relevant subgraph and essential Cypher rules into the prompt for constraining generation and reducing schema-agnostic errors. This design keeps the prompt focused and within context length limits while providing the necessary domain grounding. Comparative experiments demonstrate that T2CSS with GPT-4 outperformed baseline models and achieved 86 % accuracy in producing correct Cypher queries. In practice, this study reduces retraining and maintenance effort, shortens turnaround times, and broadens KG access for non-experts.

1. Introduction

In the data-driven era, organisations often need to manage and query highly interconnected information. Traditional relational databases, which store data in rigid two-dimensional tables, often struggle to efficiently represent and manage these intricate relationships [1]. Additionally, relational database management systems (RDBMS) rely on costly JOIN operations, and their performance degrades as the depth of relationships grows. Graph databases were developed to overcome these limitations by using nodes and edges as the basic unit to represent data, which allows relationships to be traversed directly [2]. This architecture provides more predictable and faster query performance for relationship-rich data [3]. Therefore, graph databases are well-suited for applications where understanding connections is critical, such as social network analysis [4], recommendation systems [5], and fraud detection [6]. A knowledge graph (KG) is a type of graph database that organises domain-specific knowledge as a network of entities and relationships [7]. As a popular graph database platform, Neo4j uses the Cypher query language to store, manage, and query KGs [8]. However, writing correct Cypher queries requires understanding the graph's structure and the query syntax. Many domain experts and casual users do not have this technical expertise [9]. As a result, non-technical users cannot easily query a KG on their own, and they must rely on technical staff. One approach to bridge this gap is text-to-Cypher, which translates a user inquiry into equivalent Cypher query. With a text-to-Cypher system, users can retrieve information from the graph by asking questions in plain language, without needing to learn the query language. However, existing text-to-query methods have inherent limitations. Rule-based systems and conventional machine learning (ML) models require laborious manual design of rules or features, and they often fail to generalize to varied or unforeseen queries. More advanced deep learning models can improve translation accuracy once trained, but they demand substantial upfront training data and tend to be rigid when the domain or schema evolves [10]. In other words, if the KG's structure changes or new concepts are introduced, these models need new annotated examples and complete retraining to adapt, which is costly and time-consuming.

To avoid these limitations, large language models (LLMs) provide a flexible and powerful solution for the text-to-Cypher task. Based on vast amounts of pre-trained knowledge, LLMs understand and generate

* Corresponding author.
E-mail address: LiuY81@Cardiff.ac.uk (Y. Liu).

https://doi.org/10.1016/j.dss.2025.114553
Received 27 July 2024; Received in revised form 30 September 2025; Accepted 4 October 2025
Available online 6 October 2025

human-like language without explicit hand-crafted rules. Moreover, LLMs can be guided using prompt engineering, where carefully designing the input prompt to include instructions or examples that lead the model to the desired output without expensive model retraining [11,12]. However, using LLMs alone in a specialised domain presents challenges. Without domain-specific context, an LLM might produce irrelevant or incorrect queries [13]. The key to accuracy is providing the model with the necessary domain knowledge. One way to achieve this is through fine-tuning the LLM on domain-specific data, but fine-tuning requires substantial resources, such as a large dataset, intensive computational demands, and extended training investment. Even parameter-efficient fine-tuning techniques, such as LoRA or QLoRA, partly mitigate this effort, they still require several hours of GPU time to adjust a model for each new domain or schema update [14,15]. Instead, in-context learning (ICL) is used to supply domain knowledge at query time, where embedding the relevant domain information directly into the LLM's prompt rather than altering the model's parameters [16]. This method enables the LLM to adapt to a new domain on the fly by eliminating the retraining overhead. By contrast, rule-based or deep learning approaches perform well once a schema is fixed, but each schema extension typically demands new annotations and a full retraining cycle [17,18].

A practical consideration when injecting domain knowledge into LLM prompts is the model's limited context length. Long prompts exceed the limit, slow down the model's response, and may lead to confusion or irrelevant outputs. Therefore, it is crucial to provide the appropriate amount of context: enough to inform the model, but not so much that the model is overloaded. We address this by using a semantic schema derived from domain ontology as a lightweight representation of the domain. The semantic schema captures the essential entities, categories, and relationships of the domain in a concise form. Unlike a full KG that contains numerous instance-level facts, the semantic schema is an abstracted outline of the domain's structure. This focused representation is compact enough to include in a prompt and gives the LLM guidance on how the domain is organised. In the context of text-to-Cypher tasks, the semantic schema serves as an ideal prompt ingredient. Given these considerations, the following research question is posed: "How can an LLM be effectively aligned with domain-specific knowledge to generate accurate Cypher queries for graph databases?". To answer this question, a schema-guided prompting approach, namely text-to-Cypher with semantic schema (T2CSS), is proposed. In T2CSS, the LLM is guided by domain knowledge through a semantic schema and a tailored prompting strategy. An adaptive information-filtering module selects the relevant subset of the semantic schema and Cypher syntax that are related to a user's query to ensure that the prompt remains both concise and relevant. Also, a unified prompt template is employed to combine the user's natural language question with the selected schema information. The main contributions are: (1) a domain semantic schema from domain ontologies is developed to cover the key concepts and relationships of the domain, (2) a mechanism is designed to dynamically filter and inject only the relevant portions of the semantic schema into the LLM's prompt for each question, (3) a unified prompt template is proposed to integrate the user's intent with the domain context. Section 2 reviews related works. Section 3 describes query language-informed meta-design. Section 4 illustrates the proposed methodology, followed by benchmarks and evaluation metrics in Section 5. A case study to demonstrate the practical application is presented in Section 6. Section 7 presents discussions, and Section 8 summarises this study.

2. Related works

2.1. Domain knowledge representation using ontologies and KGs

As a structured framework for representing and organising information within specific domains, ontology emerges as a promising solution to address the semantic communication and interoperability issues for information sharing and reuse [19]. By formalising concepts, properties and their relationships, ontologies establish systematic knowledge conceptualisations [20]. Developing an ontology typically involves stages such as specification, knowledge acquisition, conceptualisation, integration, implementation, and evaluation [21]. Several applications of ontology demonstrate its utility in knowledge representation and modelling. For instance, a steelmaking ontology was designed to build a shared resource and capability model for supporting knowledge sharing and management [22]. Ontologies play important roles in the lightweight and efficient representation of structural knowledge schema for understanding and navigating domain knowledge, particularly when prioritising conceptual clarity over instance-level details [23].

In contrast to KGs, which store detailed instance-level data, ontologies focus on abstracting domain principles into hierarchical relationships and semantic rules [24,25]. This makes ontologies more efficient and lightweight when the goal is to understand conceptual knowledge rather than specific instances. As explained in Section 1, ontologies play a key role in the T2CSS by providing a semantic schema, which is a simplified version of ontologies for representing domain concepts and relationships. The semantic schema guides LLMs in generating Cypher queries by offering essential domain knowledge without including extensive instance data. Unlike KGs, which can overwhelm LLMs with detailed information, the semantic schema is compact to ensure efficient token usage and better performance. While ontologies provide a foundational structure, KGs are used in this study to store and manage complicated and dynamic data, such as specific entities and their connections. Therefore, ontologies and KGs complement each other in this study: the ontology-derived semantic schema offers a lightweight and structural guide for LLMs to translate user intention into Cypher statements, and KGs serve as the graph database for executing those statements.

2.2. Large language models and prompt engineering

LLMs have made significant developments through deep learning paradigms and training on expansive corpora, such as processing natural language text and providing valuable information for specific tasks [26]. Prompt engineering emerges as an integral facet of leveraging the capabilities of LLMs and interacting with them, where designing and optimising prompts (the input of LLMs) to guide LLMs towards the desired output. Prompt engineering involves the description of the task, general prompting strategies, the integration of user interest modelling and the presentation of candidate items [27]. While LLMs excel at broad language tasks, they lack enough contextual knowledge and intrinsic mechanisms for specialised reasoning, which results in wrong or "hallucinated" responses [13]. Thus, a key challenge of LLMs lies in bridging the gap between the general linguistic capabilities and context-aware knowledge in specific domains, which can be remedied by specialised knowledge embedded in well-crafted prompts. Effective prompts should balance linguistic coherence with domain expertise [28]. For instance, medical applications designed prompts enriched with precise terminology and logical frameworks to reflect knowledge structures [29]. Similarly, in the text-to-Cypher task, LLMs often struggle to infer implicit relationships in the nuanced domain knowledge and adhere to strict syntactic constraints required for Cypher query generation [30].

Moreover, since an increasing number of open-source LLMs and their variants exist, the performances of LLMs on different tasks have been recognised as varying by their inherent parameters and prompting contents. Selecting an appropriate LLM as a foundational model and constructing effective prompts are two important aspects to leverage the capabilities of LLMs. Therefore, different LLMs and context-aware prompt strategies tailored to domain scenarios are worth exploring in the text-to-Cypher task.

2.3. Approaches for text-to-Cypher transformation

Cypher is the declarative query language for graph databases for enabling data retrieval and manipulation [9]. Text-to-Cypher bridges natural language interfaces with structured knowledge retrieval. Early rule-based systems employed pattern matching and semantic role labelling to decompose queries into Cypher. Also, ML models were constructed to determine key components. Oro et al. introduced a rule-driven semantic parsing with MANTRA language to bridge user queries with graph databases [31]. Litvin et al. mapped inflective-language phrases to Cypher queries by combining decision trees with flexible templates and addressed the variability with different natural languages [32]. However, these approaches attempted to find all possible instances. Subgraph matching or subgraph isomorphism search solves this problem by relaxing the requirement of an exact match [33]. However, these approaches still face labour-intensive design in modularity and explicit rules for precise translation and lack flexibility for diverse queries. Then, relevant research has shifted towards deep learning due to its exceptional performance in semantic generalisation and complex relationship modelling. Tran et al. introduced BERT for semantic parsing, GraphSAGE for graph-aware relation-property mapping, and transformers for query synthesis [17]. Liang et al. leveraged the capabilities of deep learning to extract the semantic features and fill in the predefined Cypher query sketch slots [18]. While effective once trained, deep learning approaches struggle with substantial upfront training efforts and suffer from rigidity when adapting to dynamic domain contexts.

LLMs have demonstrated promise by leveraging the extensive corpora of text data upon which they are trained. However, employing them in text-to-Cypher transformations is challenging, where problems arise in the nuanced understanding required beyond what LLMs currently grasp when models handle complicated queries. To address the misalignment between user inquiries and KGs, Zou et al. integrated fine-tuned LLMs with an unsupervised joint retrieval mechanism to retrieve neighbouring nodes and relations and avoid schema-agnostic query generation [34]. Recent parameter-efficient fine-tuning (PEFT) techniques, most notably LoRA [35] and QLoRA [36], adapted sub-10 B-parameter LLMs to the text-to-Cypher task with only a few million trainable weights. Studies such as GraphRAFT fine-tune a model while jointly retrieving relevant sub-graphs, achieving provably correct Cypher generation [14]. SyntheT2C shows that synthetic question-Cypher pairs can further raise accuracy when annotated data are scarce [15]. Despite these gains 70 % execution accuracy on public benchmarks, PEFT still incurs a highly computational cost of GPU training and task-specific data preparation. By contrast, an ICL-based approach was employed to bridge the gaps between text and structured knowledge representations through prompt engineering, where the corresponding entities were identified to serve as the contextual supplement in guiding LLMs for generating desired outputs [37]. Another research further iterated prompts based on the chat history and an error correction module, and allowed users to retrieve different graph databases [30]. Prompt-only approaches inject a concise schema and a handful of exemplars at inference time without any fine-tuning.

3. Query language-informed meta-design

This artifact's meta-requirements (MR1-MR4) are grounded in the formal rule families of the Cypher graph query language. In this section, the rule families are outlined and further explained, showing how each informs the artifact's design. Four meta-requirements (MR1-MR4) are grounded in Cypher's rule families, including pattern binding (MATCH), predicate filtering (WHERE, logical operators, and comparisons), projection (RETURN), and, where applicable, aggregation/grouping and path constraints. Inspired by Cypher's rule families and their role in query construction, each meta-requirement constrains a distinct design choice as follows. (i) MR1 surfaces domain relations in user terms. The system should expose and use domain-specific nodes and relationships from the semantic schema that correspond to the user's query intent. In other words, relevant domain relations mentioned in the user's own words must be reflected in the query structure, such as MATCH pattern. (ii) MR2 makes syntactic commitments explicit. The system should decide early on the necessary query structure, such as clauses and constructs to use, and make the structure explicit in the prompt. Rather than leaving the LLM to infer all query syntax implicitly, the design provides a scaffold or template of Cypher clauses aligned with the user's intent, such as MATCH/WHERE/RETURN, etc. (iii) MR3 filters to the relevant context. The system should include only information and syntax relevant to the specific query context. This entails filtering the domain knowledge (schema subgraph) and the Cypher rule base to the minimum needed subset before prompting. This mirrors how a WHERE clause filters data in a query by analogously filtering prompt content to prevent information overload. (iv) MR4 provides feedback and repairs with executability guarantees. The system should incorporate a feedback loop to refine the query or prompt if the generated Cypher is not immediately executable. In practice, this means validating the LLM's output against the schema and syntax rules, and providing corrective guidance so that the final query is syntactically correct and runnable, such as ensuring a query has the proper RETURN clause and uses valid identifiers. This requirement is informed by the need for queries to execute successfully. Each of Cypher's rule families maps onto one or more of the above meta-requirements, and these in turn inform specific modules in our artifact's design. Table 1 summarises these mappings and design implications.

Table 1
Mapping of Cypher rule families to meta-requirements and design elements in the artifcat.

Cypher rule families	Meta-requirements	Moduldes in artifact	Design implications
Graph pattern specification (e.g., MATCH clause family)	MR1, MR2	Schema mapping module, query scaffolding module	Domain concepts in the user query are identified and directly mapped to graph entities/relations. The system constructs an explicit MATCH pattern using those domain terms, ensuring user-intended relations appear in the query structure.
Conditional filtering (e.g., WHERE clause family and logical operators)	MR2, MR3	Information filtering module	Both the prompt content and the query conditions are constrained to what is relevant. The design mirrors Cypher's WHERE by filtering out unrelated schema elements and providing only pertinent conditions (e.g., numeric or temporal filters mentioned by the user) in the prompt.
Result projection (e.g., RETURN/ WITH clause family)	MR2, MR4	Validation and feedback mechanism (query verifier)	The system enforces query completeness by including a RETURN clause structure and verifying the LLM's output is executable. If the initial query is incomplete or incorrect, the design provides feedback (or iteratively repairs the prompt) to ensure a valid Cypher query that yields results.

Building on the above, we can trace the logic flow of how Cypher's theory informs our system's operation. Firstly, given a user question and the domain schema, the schema mapping module surfaces the relevant domain entities and relationships (MR1), for example, identifying that "cold-rolled coils" and "tensile strength" are key concepts to appear in the query's MATCH pattern. Next, the system selects and scaffolds the query clauses needed (MR2), committing to a skeleton Cypher structure. For instance, if the inquiry implies a filtering condition (e.g., "over standards" or a date range), the design includes a WHERE clause scaffold; if an aggregate or sorting is required, those syntax rules are chosen from the rule base. The prompt is then constructed with these clause placeholders and filled with the user's terms and schema context. Subsequently, an information filtering step prunes the prompt content to only the relevant schema triples and syntax rules (MR3), which ensure that the LLM sees a concise context focused on the user's intent. Guided by this focused prompt, the LLM generates a Cypher query, which the system then validates against the schema and syntax rules. If the query is not immediately executable, the system can leverage the LLM to refine the query using the rule base for fulfilling MR4 by guaranteeing an executable result.

4. Methodology

4.1. Overview of semantic schema-supported prompting for text-to-Cypher

In this study, a unified T2CSS prompt template incorporating domain knowledge and Cypher language rules is designed to guide LLMs for Cypher query generation. In contrast to KGs, which store instance-rich data, a semantic schema is simplified from ontologies to focus on defining the structure and domain semantics. This characteristic makes it suited for scenarios requiring schema-level understanding, such as guiding LLMs to understand the necessary domain knowledge. However, while ontologies provide foundational schemas for knowledge organisation, their static nature and limited scalability in dynamic environments necessitate integration with KGs. KGs extend ontological schemas by incorporating real-world instances and their relationships. Thus, this study employs ontologies not as replacements for KGs but as complementary tools. Specifically, lightweight semantic schemas from ontologies guide LLM prompt structuring, while KGs serve as execution layers for querying instance-level data. Fig. 1 depicts an overview of the T2CSS. It begins with the establishment of a domain semantic schema, which defines relevant concepts, properties, and their relations from multiple sources. The user inquiries in natural language that reflect the user's intentions are then collected. Also, the rule base is constructed to cover all Cypher syntax [9]. In the second stage, an information filtering mechanism is applied to select the subgraphs about specific user intentions. The contextual knowledge is then exported from the selected subgraphs, including the relevant concepts and relationships. According to user intentions, the necessary Cypher syntax is chosen. The refined prompts of LLMs are generated by merging these three parts: the user inquiries, the contextual knowledge from the subgraph, and the necessary Cypher syntax. In the last stage, LLMs interpret the refined prompts to output the desired Cypher statements, ready for execution in KGs.

4.2. Domain-specific semantic schema design

As mentioned before, the semantic schema is a promising way to illustrate principled knowledge rather than specific instantiations. It provides standardised and clear definitions that can be shared. Although the general types of things that share certain properties are modelled in domain-centric semantic schemas, these models do not contain information about specific individuals. Fig. 2 depicts the construction process of a domain semantic schema, covering domain and scope identification, requirement specification, formal design, instance creation, and evaluation. Each step serves a distinct purpose. The first step includes identifying the domain, its intended use, the contexts in which it will be

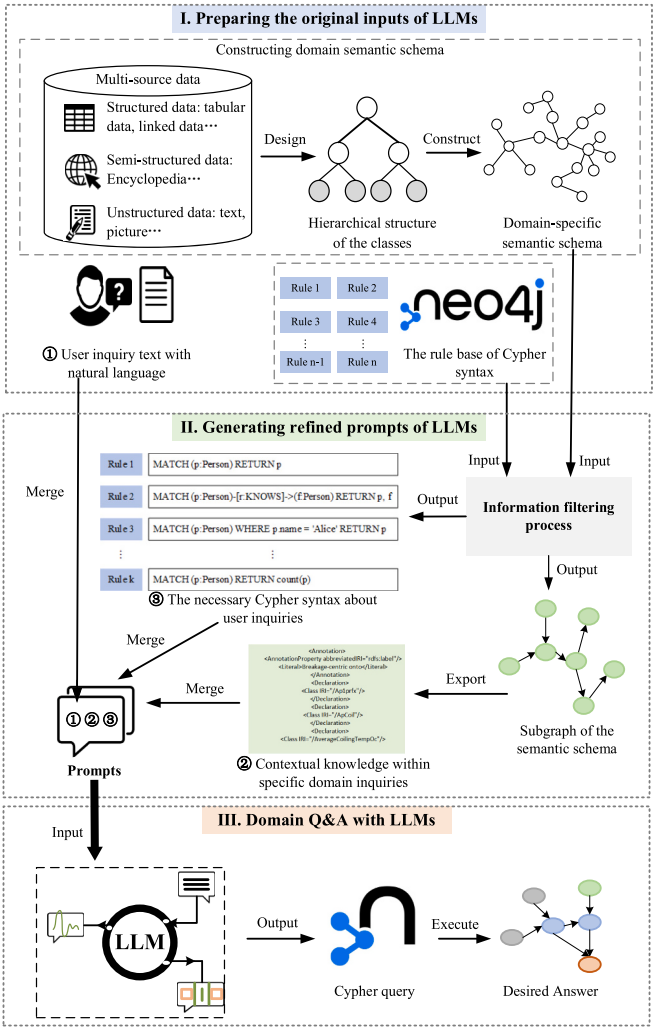


Fig. 1. An overview of T2CSS for generating Cypher languages from text by LLMs.

applied, and its maintenance strategy. Then, the requirement specification and existing reusable semantic schema are determined. The next step focuses on forming the structure of a domain semantic schema. It involves identifying key concepts and forming classes and subclasses using both top-down and bottom-up approaches. Meanwhile, the class properties are defined, such as object properties and data properties. Lastly, the general instances are created to generate the domain semantic schema, followed by evaluation to ensure error-free logic consistency and reasoning. When the semantic schema ensures logical consistency, clear reasoning, and practical use, it provides domain knowledge for LLMs in understanding task scenarios. Being lightweight, the schema is flexible and easy to adapt without being overwhelmed by too much detail. The balance between lightweight and semantic information makes the semantic schema a useful tool for applications that need context and a straightforward understanding of specific domain relationships and concepts.

4.3. Prompt design for text-to-Cypher tasks towards domain questions

The details of the proposed T2CSS prompting approach are demonstrated to guide LLMs in translating user intentions into structured Cypher queries in this section. The general prompt template is represented by $P(T, O, Q)$ to guide an LLM for Cypher generation. $T = \{t_1, t_2, \cdots, t_n\}$ denotes a set of domain inquiries within the natural language, which is defaulted to the English language in this study. O rep-

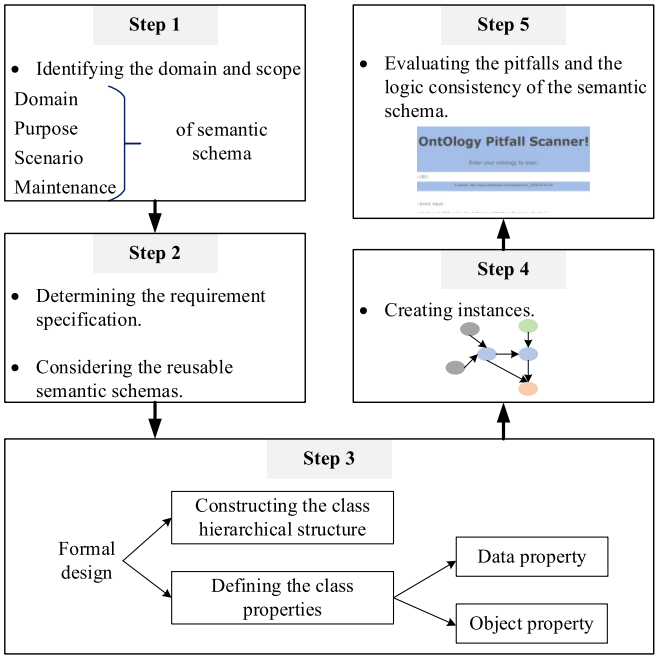


Fig. 2. The process flowchart for building a domain-specific semantic schema.

resents the semantic schema and $Q = \{q_1^*, q_2^*, \cdots, q_n^*\}$ is a set of rules of Cypher syntax. $P(T, O, Q)$ is calculated:

$$P(T, O, Q) = t + C(O) + \sum_{i=1}^{n} S\left(q_i^* | q_i^* \epsilon Q\right) \tag{1}$$

where $C(O)$ represents the semantic schema-supported concepts and relationships relevant to t, and $S(Q)$ denotes the text representations of Q. Subsequently, given a set of nodes $N = \{n_1, n_2, \cdots, n_n\}$ (refers to predefined entities, such as "line speed" or "coil tensile strength") and a set of relationships $R = \{r_1, r_2, \cdots, r_n\}$ among nodes, which consist of the semantic schema $O = \{N, R\}$, O is a series of triples (n_i, r_j, n_k), each representing a relationship between nodes within the semantic schema.

$C(O)$ represents the extraction of concepts and relationships about the user inquiries from O, which can be articulated:

$$C(O) = \sum_{(n_i, r_j, n_k) \epsilon O} \omega(n_i, r_j, n_k) \bullet e(n_i, r_j, n_k) \tag{2}$$

where $\omega(n_i, r_j, n_k)$ is a weighting function that assesses the relevance of each triple within O, and $e(n_i, r_j, n_k)$ is an extraction function that derives information from each triple. $C(O)$ is calculated by performing a weighted summation over all relevant triples in O, thereby capturing the key concepts and relationships inherent in O. Eq. (2) ensures the coverage of O and allows for differentiated weighting of triples to reflect the structural domain knowledge.

Although designing the general prompt template for T2CSS, there are two major challenges: (1) the complete domain semantic schema may arise with overloaded information for LLMs, and (2) LLMs have a limitation on context length. Overloading the prompt with too much data can waste LLM resources and be inefficient, such as introducing redundant information and generating irrelevant outputs. To address these challenges, a filtering process is proposed to restrict the inputs to be manageable for LLMs while maximising the retention of critical information. The filtering process consists of three stages, including text preprocessing, semantic schema mapping, and query structure determination. Fig. 3 details the filtering process of generating prompts for LLMs. Three parts marked by a red rectangular frame are three components of the prompts in Eq. (1), representatively, user inquiries, the semantic schema-supported information, and the necessary Cypher syntax.

The first stage aims to clean and standardise user inquiries T to extract key information and features, transforming the original text into a format for subsequent analysis and matching. It includes steps such as tokenisation, stop-word removal, and stemming. The numerical vector of each text t_i is calculated:

$$T' = \{E(t_i) | t_i \in T\} = \frac{1}{|t_i|} \sum_{w \epsilon t_i} E(w) \tag{3}$$

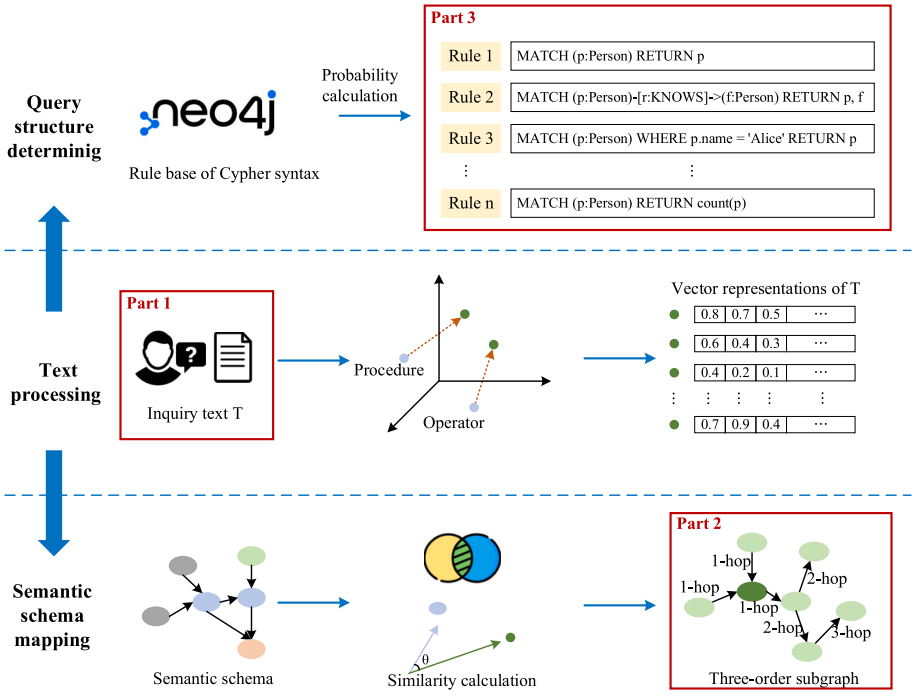


Fig. 3. A detailed filtering process regarding prompt generation of T2CSS.

where $E(t_i)$ represents the feature vector of a user inquiry t_i, $|t_i|$ is the number of words in the t_i, w denotes each word in the t_i, and $E(w)$ means the vector representations of each word w. Eq. (3) is applied to convert each word into a numerical vector, resulting in a feature representation for the next phase. The second stage is to map the processed text feature to the entities and relationships defined in the semantic schema and identify the most relevant semantic concepts by calculating their similarities to the text features. Their similarity $S(T', O)$ can be obtained from the following equation.

$$S(T', O) = \frac{T' \bullet O}{\|T'\|\|O\|} \tag{4}$$

where terms $\|T'\|$ and $\|O\|$ are the Euclidean norms of these vectors, which indicate their magnitudes or lengths in multi-dimensional space. Eq. (4) assesses the similarity between text features and concepts of O using similarity calculation functions for the accuracy and relevance of the mapping. After assessing the similarities, $C(O)$ is constructed to serve as the relevant semantic input of LLMs. To address the context length limitation and the schema information overload, $C(O)$ consists of a sequence of textual features of itself and its three-order concepts from t_i. The final phase is to generate the most relevant Cypher syntax based on the previous stages, including the text features and semantic schema mapping results. The optimal rules are determined by $q^* = argmax_{q \epsilon Q} Z(q|T')$, where $Z(q|T')$ is calculated by:

$$Z(q|T') = \frac{Z(T'|q) \bullet Z(q)}{Z(T')} \tag{5}$$

$Z(q|T')$ denotes the probability of choosing a query structure q given the text features T'. $Z(T'|q)$ represents the likelihood of observing the feature set T' given a q. $Z(q)$ is the prior probability of q, and $Z(T')$ is the marginal probability of the feature set. Eq. (5) calculates the query rules to guide the LLMs for generating Cypher queries that can be executed on KGs. Specifically, the conditional probability is utilised to predict the most suitable query rules. The user intentions, the subgraph of the semantic schema, and the selected query rules are organised in prompts to generate query statements using LLMs.

The pseudo is demonstrated to bridge unstructured queries with structured knowledge via schema-guided LLM prompting to ensure domain fidelity and computational efficiency. The input user inquiry is first parsed to extract key concepts and their semantic relationships. This process involves tokenisation, stop-word removal and vectorisation, followed by similarity matching against the domain schema to ensure contextual alignment. For each user inquiry, the domain semantic schema is then dynamically extracted into a contextual alignment by linking key concepts to predefined schema triples. Concurrently, a syntax subset is extracted from the Cypher rule base guided by schema-concept relevance. A filtering process further prunes the subgraph and rule subset to comply with the input constraints of LLMs. Also, a combined prompt structured via Eq. (1) is fed into the LLM to generate a Cypher query, which is appended to the final output. The compiled Cypher queries are validated against the semantic schema for syntactic and semantic correctness before executing in the domain KG.

Algorithm 1 Pseudo-code for Text-to-Cypher Transformation

Input: Natural language text query $T = \{t_1, t_2, \cdots, t_n\}$, Domain-specific semantic schema O, Cypher language rule base Q', Information filtering process F, Prompt template P.

Output: Structured Cypher query $M = \{M_1, M_2, \cdots, M_n\}$.

1: Initialise the structured Cypher query M as an empty string.
2: Parse the natural language text query $T = \{t_1, t_2, \cdots, t_n\}$ to identify key concepts $K = \{k_1, k_2, \cdots, k_n\}$ and their relationships R using schema O.
3: **for each** t in T
4: **for each** concept k in K do
5: identify the corresponding node or relationship in schema O
6: construct related subgraph O' of semantic schema O and the related rule subset $Q = \{q_1^*, q_2^*, \cdots, q_n^*\}$ from Q'

(*continued on next column*)

(*continued*)

7: **end**
8: Apply filtering process F to ensure T, O' and Q fit within the context length and token constraints.
9: Generate the prompt $P = T + O' + Q$.
10: Input P into LLMs.
11: Return the response M_k of LLMs
12: Append M_k to Cypher query M
13: **end**
14: Validate Cypher query M against schema O to ensure it does not exceed schema constraints
15: Return the final structured Cypher query M

5. Experimental setup

5.1. Comparative models

To demonstrate the effectiveness of the proposed T2CSS, three widely used models for text-to-query tasks are selected to serve as benchmarks, including Seq2SQL [38], TypeSQL [39], and LGESQL [40]. Moreover, a set of LLMs is chosen to compare their performances, including ChatGPT (versions 3.5 and 4.0) [41], Claude 2.0 [41], LLaMA 2 [41], and Mistral 7B [42]. Each model represents a unique blend of linguistic capabilities, training paradigms, and architectural innovations. Among them, medium-scale models are moderately parameterised LLMs (e.g., LLaMA and Mistral), and large-scale models have advanced capabilities of the commercial LLMs with large-scale parameters (e.g., Claude 2.0, ChatGPT 3.5, and 4.0). Furthermore, to ensure a comprehensive assessment reflecting the most recent research trends, we additionally evaluated ChatGLM2, an advanced open-source LLM, in combination with fine-tuning strategies such as LoRA and QLoRA. These fine-tuning methods represent state-of-the-art techniques that have emerged in 2024 and 2025, providing a valuable benchmark for our T2CSS prompting approach.

5.2. Evaluation metrics

In this section, an evaluation approach has been established to include three metrics for performance comparisons under different models and prompting strategies for the targeted tasks. These metrics provide a holistic view of performance and efficiency. The comparative analysis includes detailed assessments of how different baseline models and diverse LLMs perform across three different metrics.

(1) Logical accuracy (*LA*) measures the proportion of generated queries that correctly reflect the logical structure [43], which reflects the ability to generate logically coherent queries that align with user intents.

$$LA = \frac{C_L}{T} \tag{6}$$

where C_L is the count of generated queries with correct logic, and T is the total number of generated queries.

(2) Execution accuracy (*EA*) assesses the percentage of the correct and expected results in Cypher queries executed in a Neo4j database [43].

$$EA = \frac{C_E}{T} \tag{7}$$

where C_E is the count of generated queries yielding correct results upon execution, and T is the total number of generated queries. For each user inquiry, domain experts manually define the correct Cypher query and its expected output. Each generated Cypher query is executed in a Neo4j database containing the domain-specific KG. Then, the output of each executed Cypher query is retrieved and compared to the predefined expected output. A query is counted as correct (incrementing C_E) only if

its results match the benchmark exactly. Any mismatch classifies the query as incorrect.

(3) Average token used (*TU*) calculates the average number of tokens utilised by the LLMs for generating each Cypher query. It is an indicator of the efficiency and complexity of the generated queries.

$$TU = \frac{\sum N}{T} \tag{8}$$

where *TU* means the average tokens used, and $\sum N$ represents the sum of tokens in all, and *T* is the total number of generated queries.

6. Case study – results and discussion

6.1. Dataset description and preparation

Cold rolling is a key stage in steel manufacturing due to its ability to enhance diverse mechanical properties of steel strips, including increasing tensile and yield strengths, as well as improving hardness and surface finish. With the increasing demand for high-property steel strips in different industries, these improved characteristics of cold-rolled coils are indispensable for high-performance applications where precision and durability are crucial. Consequently, a cold-rolling case study is selected and conducted to validate the proposed T2CSS. Experimental data were provided by an electrical steel manufacturer with a reversing mill.

Fig. 4 shows the complete manufacturing workflow contributing to the production of cold-rolled coils in the steel industry. Various stages impact the quality of cold-rolled products to different extents, including hot rolling, hot-rolled coil properties, annealing, pickling, cold rolling, and quality inspection. Initially, hot-rolled coils undergo annealing to enhance ductility, reduce hardness, and improve workability, altering the coil's physical and chemical properties. Subsequently, pickling treatment is applied to metal products to remove impurities like stains, inorganic contaminants, and rust. Using emulsion, the treated steel coils are then passed through a cold rolling mill for flat deformation. These steps are repeated to achieve the desired size. After straightening, the quality of the cold-rolled products is inspected on-site by technicians. Breakage defects in the strips are manually identified and marked. Finally, the inspected and approved cold-rolled coils are cut to the required length for packing and storage. Table 2 presents details of relevant concepts and characteristics extracted from multiple sources. Five resources are regarded as the contributing factors to the strip-breakage phenomenon, including the hot-rolling process, annealing, pickling, emulsion, cold-rolling process and quality inspection. The

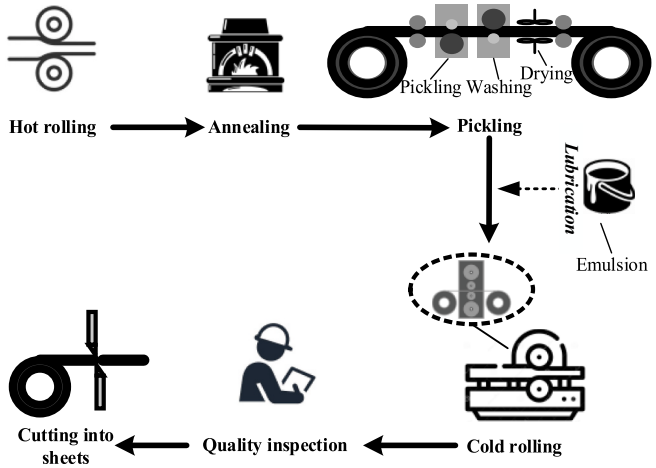


Fig. 4. An overview of the cold rolling workflow in the steel industry.

Table 2
Details of relevant concepts and characteristics extracted from multiple sources.

Sources	Concepts and variables
Hot-rolling process	Hot-rolled coil properties, such as chemical contents, quench temperature, etc.
Annealing & Pickling	Annealing temperature, Jetflow speed, etc.
Emulsion	Dirt result, pH, conductivity, chloride index, etc.
Cold-rolling process	The rolling operation, equipment, tension, measured parameters, etc.
Quality inspection	Cold-rolled coil properties, such as weight ingoing, width, weight outgoing, etc.

dataset was collected and stored from these resources, which cover a production period of six months. The historical dataset contains 1254 samples and 94 variables.

6.2. Domain semantic schema design

In this section, a steel cold-rolling semantic schema (SCRS) was designed to provide structured domain knowledge for LLMs. This SCRS was constructed from six resources, including material, man, machine, method, measurement, and environment, through specification, knowledge acquisition, conceptualisation, integration, implementation, and evaluation. OWL was selected as the encoding language due to its compatibility with diverse data formats and its ability to provide formal and comprehensive semantics for Web content [44]. The SCRS was encoded and refined using Protégé5.5.0, which is a widely adopted tool for knowledge representation. The SCRS features a class hierarchy organised around the six key concepts.

As shown in Fig. 5, the hierarchy includes detailed subclasses. For example, the "machine" class contains cold rolling mills, which deform steel into thinner gauges, alongside auxiliary equipment like shearing machines and levelling machines, as well as inspection, transport and handling devices. Moreover, relationships with the SCRS are defined by object and data properties, as detailed in Table 3. Object properties link two individuals (a subject and an object) through a predicate, whereas data properties associate a single subject with attribute data using a predicate. For instance, "Operates" connects the "man" class with the "machine" class.

The SCRS was structured as a five-layer concept hierarchy capturing multi-sourced knowledge from cold-rolling processes, where its main structure is illustrated in Fig. 6. The top layer, "Things", depicted as green rectangles, represents overarching concepts. Domain-specific subclasses, shown as yellow circles, include concepts related to cold-rolled coils. The bottom layer, with purple rhombuses, contains instances like "fault pattern" and "thickness". The schema's quality was verified using the "OOPS!" platform to confirm its logical consistency and reasoning integrity. By providing a simplified conceptual structure, the SCRS guides LLM within the proposed T2CSS to avoid the data overload that a detailed KG might introduce. In contrast, the KG serves as the data repository for Cypher query execution and result validation.

6.3. Semantic schema-supported prompt design for text-to-cypher task

The development of prompts for T2CSS was detailed in this section, which focuses on cold rolling processes in steel manufacturing. The questions about the cold-rolling process were first crafted to reflect the typical user inquiries and cover diverse aspects of the process, such as the impact of annealing temperature on steel properties and factors influencing cold-rolled steel strip strength.

Table 4 presents ten typical examples of user inquiries curated to reflect real-world challenges in steel manufacturing. These inquiries were derived from historical datasets, expert consultations, and technical documentation. Selection criteria prioritised the representativeness of core processes (e.g., annealing and pickling), quality metrics (e.

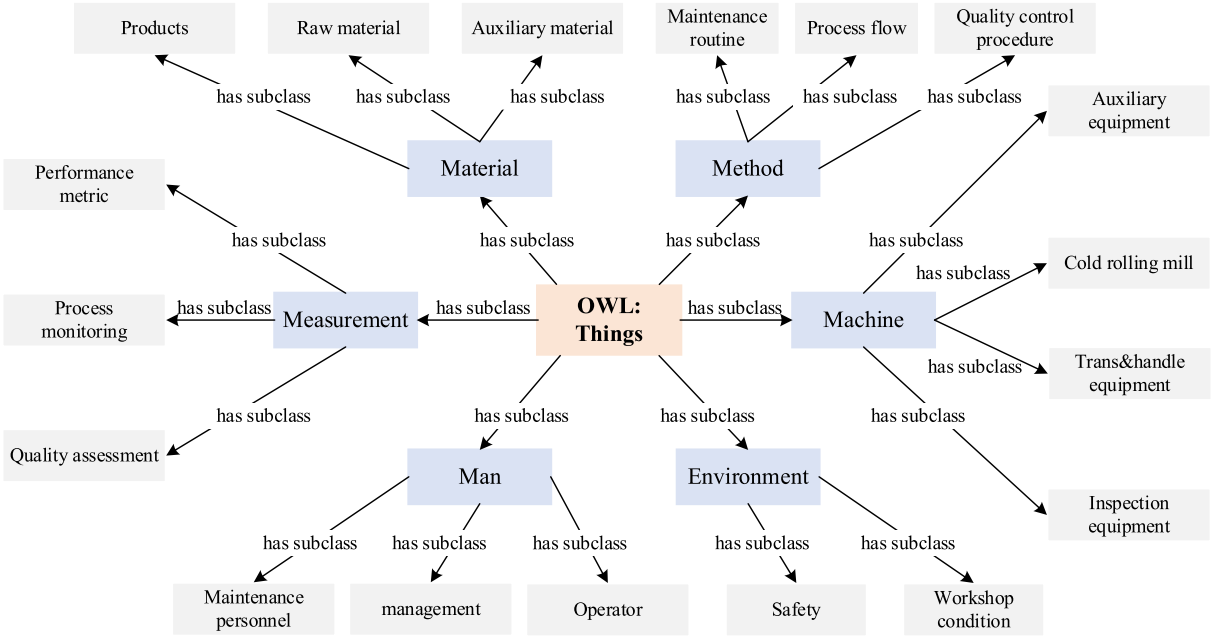


Fig. 5. The main hierarchical structure of the classes of the SCRS.

Table 3
Description of examples of object and data properties of the SCRS classes.

Properties		Description Examples
Object properties	Operates	A cold rolling machine operator would operate the cold rolling mill.
	Uses	Maintenance personnel repair machines by maintenance plans and procedures.
	Has_Input	Hot-rolled coils are fed into the cold-rolling mill.
	Has_Output	A cold rolling mill produces cold-rolled coils.
	Has_Observation	A quality inspector checks for breakage in cold-rolled steel products.
	Has_Procedure	Auxiliary equipment has procedures for annealing and pickling processes.
	Has_Property	Mechanical measurements can reveal the properties of the hot-rolled coils.
	Has_Parameter	Process monitoring methods can be used to obtain machine parameters during the cold rolling process.
	Is_Measured_By	The properties of cold-rolled coils are determined through various inspection standards and approaches.
	Is_Deployed_To	Pickling procedures could be applied to pickling machines.
Data properties	Has_Pattern	The cold-rolled coils can be identified into two patterns: strip breakage and normal.
	Has_Attribute	Describing textual attributes, formatted as strings, such as the ID of a steel coil.
	Has_Value	Indicating numeric attributes that have specific values, like the parameters involved in the pickling processes

g., tensile strength and surface roughness), and complexity for balancing single-hop and multi-hop relationships to test semantic and syntactic alignment. Integrating a semantic schema into the prompts is critical for the proposed T2CSS and involves embedding essential entities and relationships. For instance, to design the prompt of Q1 in Table 4, the key concepts and their properties are identified, including cold-rolled coils, tensile strength, and six-month production. The subgraph of the SCRS is selected to contain key concepts and their three-order neighbours. After that, all triples included in the subgraph serve as domain knowledge to support the designed prompts and guide the LLMs for understanding domain nuances required, such as <cold rolling mill, produce, cold rolled coil>. To effectively construct and execute a Cypher query that retrieves cold-rolled coils with tensile strength over standards within six months, a clear understanding of the necessary Cypher language syntax is crucial. According to the rule families of Cypher language in Table 1, it is essential to incorporate Cypher syntax within the prompts. As shown in Table 5, the representations of key Cypher syntax needed for Q1 are utilised to prompt LLMs, including the use of clauses such as MATCH, WHERE, and RETURN, alongside functions such as AVG, MAX, or COUNT.

To provide prompts to LLMs for generating correct Cypher queries of Q1, the inputs should be structured to include the user inquiry, the domain knowledge from the semantic schema, and the necessary Cypher syntax. Fig. 7 depicts an interactive dialogue process between a user and an LLM, which focuses on the LLM-driven query formulation phase. The process begins with a predefined instructional prompt that contextualises the LLM as a domain-specific expert in steel manufacturing for generating syntactically and semantically accurate queries. A sample user inquiry, "*Which cold-rolled coils had tensile strength over standards for six-month production?*" is then integrated with domain semantic schemas, including key entities and their relationships, alongside the essential rules of Cypher syntax. The framework further demonstrated contextual adaptability: the LLM appends a cautionary note advising the user to calibrate the property name (e.g., production_date) and temporal parameters (e.g., 2023-01-01 to 2023-06-30) for aligning with their specific graph schema and temporal constraints. The schema-aware customisation highlights the framework's robustness in balancing automation with user-specific data requirements.

6.4. Comparative analysis across different models

6.4.1. Accuracy comparison across different approaches

To evaluate the effectiveness of T2CSS, we benchmarked three conventional text-to-query approaches (Seq2SQL, TypeSQL, LGESQL), one recently fine-tuned small-scale LLM (Qwen-1.5B), and five prompt-based T2CSS using diverse LLMs of varying size. These models are further classified into four groups: non-LLM models, fine-tuned small-scale LLMs, medium-scale LLMs and large-scale LLMs. Group 1 (non-LLM models) includes the conventional NLP methods, which serve as benchmarks to evaluate the effectiveness of LLM-based approaches, including Seq2SQL, TypeSQL, and LGESQL. Group 2 is the fine-tuned

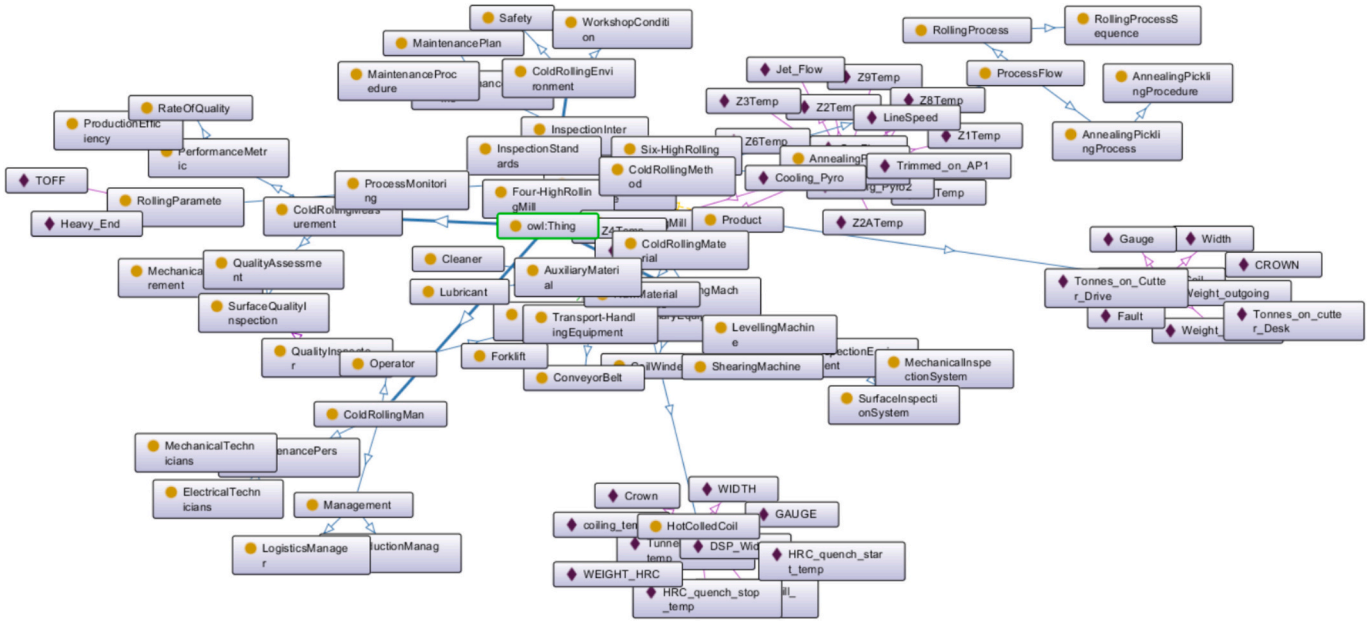


Fig. 6. The main structure of the lightweight SCRS.

Table 4
Examples of the specific questions of the cold rolling process for user inquiries.

Inquiries	Descriptions
Q1	Which cold-rolled coils had tensile strength over the standards for six-month production?
Q2	What are the main factors influencing the strength of cold-rolled steel strips?
Q3	What role does the emulsion play in the cold rolling process?
Q4	Can the properties of hot-rolled coils influence the final quality of cold-rolled coils?
Q5	Is there some correlation between the surface roughness of cold-rolled coils and annealing temperature?
Q6	How do variations in jet flow speed during pickling influence the steel quality?
Q7	Which stage has the longest processing time in the cold rolling production line?
Q8	Which temperature control procedure is critical in the cold rolling process?
Q9	What are the key indicators of mechanical measurement in quality control of cold-rolled coils?
Q10	Which batches of cold-rolled coils had strip breakages recently?

Table 5
The representations of the key Cypher syntax elements for Q1.

Cypher elements	Explanations
MATCH	The 'MATCH' clause is utilised to specify patterns in a graph, essentially denoting the structural form in which data is queried.
WHERE	The 'WHERE' clause applies conditions that filter the results of the 'MATCH' clause, ensuring only entities that meet specified criteria are included in the query result.
RETURN	The 'RETURN' clause defines the data to be returned from a query, which could be nodes, relationships, properties, or combinations thereof.
Property access	To access properties of nodes or relationships, the syntax involves the entity followed by a '.', then the property name.
Comparison operations	Operations, such as '>', '<', '≥', '≤', '=', are used in the 'WHERE' clause to compare property values against specified conditions.
Logical operations	'AND', 'OR', and 'NOT' operators are employed within 'WHERE' clauses to combine multiple conditions for more precise data filtering.
Functions	Cypher provides functions like 'date()' for converting strings to date values, used in date comparisons within the 'WHERE' clause.

Qwen-1.5B (LoRA) model, which sits between groups 1 and 3 in parameter count but represents the recent studies in parameter-efficient tuning. Group 3 comprises medium-scale LLMs with T2CSS, such as Mistral 7B and LLaMA 2. Group 4 includes large-scale commercial LLMs based on T2CSS, namely Claude 2.0, ChatGPT 3.5, and ChatGPT 4.0. Specifically, for group 2, Qwen-1.5 B is a 1.5-billion-parameter bilingual model released by the Qwen team in 2024 as a lightweight alternative to their 7 B and 14 B checkpoints. A LoRA adapter (rank $= 32$, $\alpha = 16$) is applied on a corpus of 11,042 questions (Cypher pairs). Among them, 3042 are annotated from the cold-rolling KG, and 8000 synthetic pairs are generated with the SyntheT2C recipe. Training was carried out for four epochs at a learning rate of 5×10^{-4} on a single NVIDIA GTX 2080 Ti (11 GB VRAM) using 4-bit quantisation and gradient checkpointing. The run completed in approximately six hours, with peak GPU memory just under 10 GB. By structuring the experimental groups in this manner, we aim to demonstrate the benefits of employing LLMs with T2CSS for text-to-Cypher and provide a clear comparison of T2CSS performance across different model scales.

Figs. 8 and 9 compare the LA and EA of the proposed T2CSS with different LLMs and the baseline models. Group 1 (rule-based and ML models) showed moderate performance, with LGESQL showing the best performance among the group at 68 % LA / 66 % EA; their reliance on fixed patterns limits adaptation to dynamic queries. In Group 2, Qwen-1.5 B (LoRA) rose to 75 % LA / 73 % EA, which is substantially ahead of Group 1. This illustrates the benefit of lightweight fine-tuning, but it is still below the prompt-based T2CSS using LLMs. Group 3 (medium-scale prompt-only LLMs) edged higher, with LLaMA 2 reaching 76 % LA / 76 % EA, confirming that even modest-sized models gain from schema-guided prompting. Group 4 (large-scale commercial LLMs) yielded the best results: ChatGPT 4.0 attained 86 % for both metrics, outperforming Mistral 7 B (Group 3) by 10 %.

EA trails LA in every group, which underscores the added difficulty of producing queries that both parse and run against a domain KG. The gap narrows in Groups 3 and 4. This pattern suggests that the semantic schema in T2CSS helps medium and large models align Cypher syntax with graph structure. Model architecture still matters. Claude 2.0 performs LLaMA 2 even though it has more parameters, indicating that the training strategy can limit domain-specific reasoning. LLaMA 2 matches

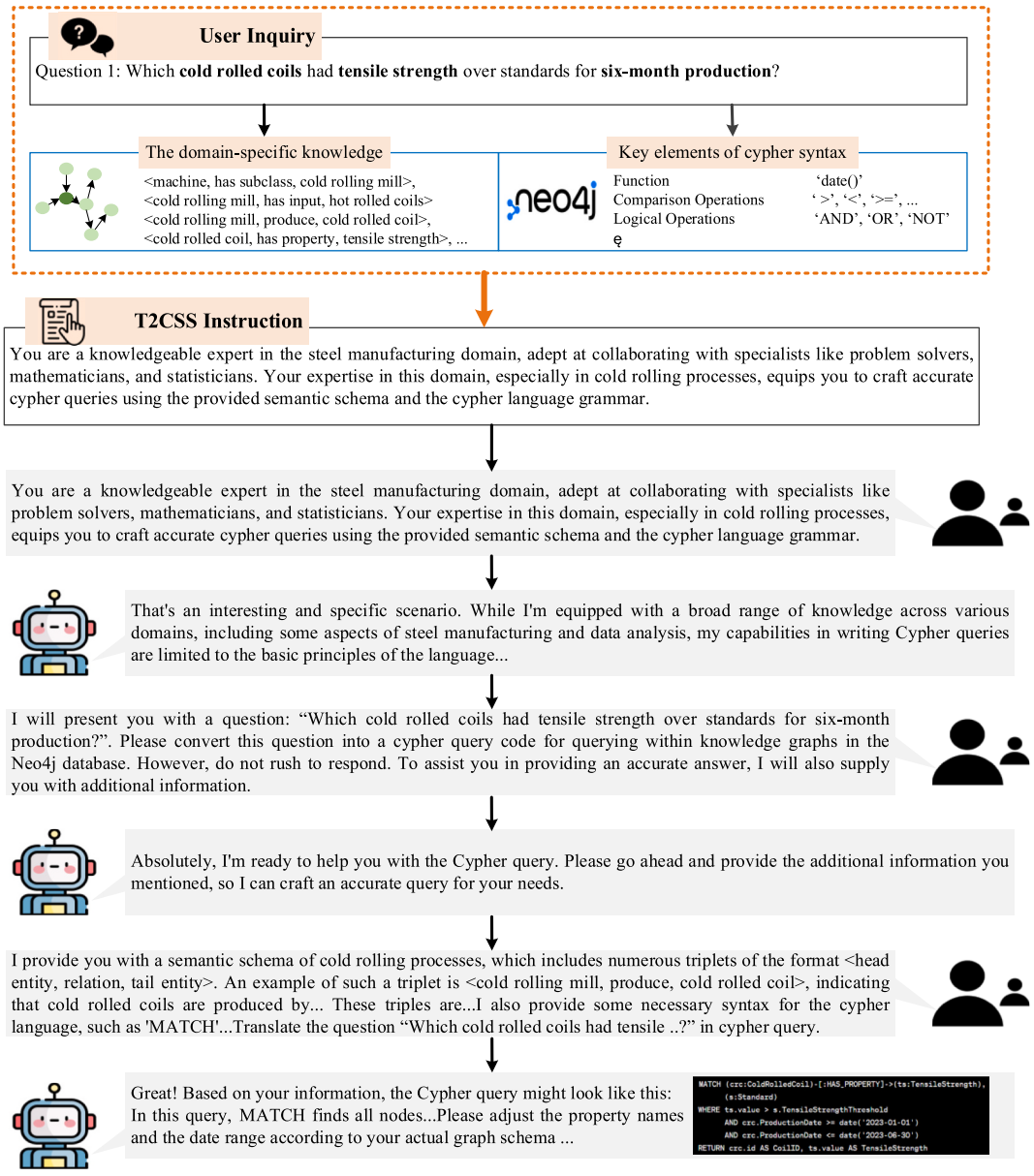


Fig. 7. An illustrative visualisation of a prompting session.

Claude's execution accuracy with roughly half the parameters, which illustrates the efficiency of a well-integrated schema prompt. The progression from 0.58 EA for Seq2SQL, 0.73 EA for Qwen-1.5 B with LoRA, to 0.86 EA for ChatGPT 4.0 with T2CSS supports two practical conclusions. First, parameter-efficient fine-tuning offers a meaningful accuracy boost when computer resources are limited. Second, schema-guided prompt engineering can close much of the remaining gap without the time, data, and energy that continual tuning demands.

6.4.2. *Validation of the T2CSS model through different inputs*

Following the comparison groups in Section 6.4.1, we further validated the effectiveness of the designed prompts from T2CSS by comparing the performances of five models (groups 3 and 4 in Section 6.4.1) under different input conditions. The proposed T2CSS approach integrates three key components to construct the prompts for LLMs: user inquiries, necessary Cypher syntax related to the inquiries, and semantic schema relevant to the inquiries. The inputs were divided into four strategies. Strategy 1 is to input only user inquiries, which aims to evaluate the performances of the models when no additional structural or semantic information is provided. Strategy 2 incorporated user inquiries and the necessary Cypher syntax to assess the impact of necessary Cypher language rules. Strategy 3 included user inquiries and a semantic schema to assess the impact of domain knowledge. Strategy 4 combined user inquiries, Cypher syntax, and semantic schema, aiming to showcase the full potential of the T2CSS approach. The experimental design of four strategies assesses how different input combinations affect model performance and demonstrates the potential of incorporating necessary Cypher syntax and semantic schema into the prompts.

Table 6 presents experimental results across different input conditions, which provide the detailed effectiveness of each input component in text-to-Cypher tasks. Under Strategy 1 (user inquiries only), both groups show limited accuracy. Medium models in Group 3 stay below 0.55 EA, while the larger models in Group 4 remain under 0.60 EA, indicating that pre-training alone does not substitute for structural guidance. In strategies 2 and 3, adding Cypher syntax and semantic schema, respectively, both improved LA and EA across all groups. It highlights the importance of Cypher syntax and schema-driven contextualisation. Semantic schema (Strategy 3) yields larger gains than

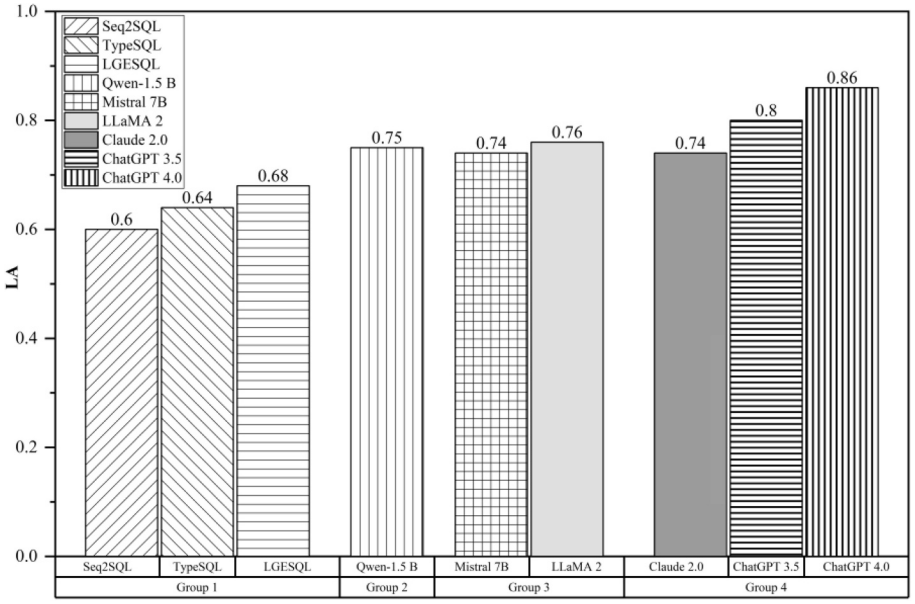


Fig. 8. Performance of the proposed T2CSS in comparison to baselines on LA.

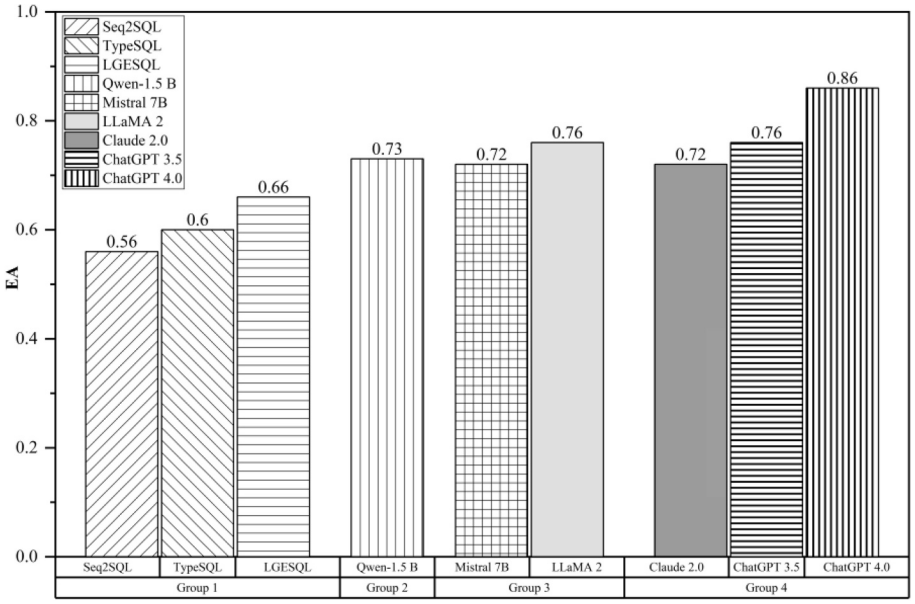


Fig. 9. Performance of the proposed T2CSS in comparison to baselines on EA.

Cypher syntax alone (Strategy 2), confirming the value of domain knowledge. Strategy 4 produced the best scores: Group 3 models rose into the mid-0.70 EA range, while ChatGPT 4.0 reached 0.86 EA. Large-scale models in Group 4 consistently outperformed the medium-scale models in Group 3, reflecting the added benefit of parameter count when both groups received the full T2CSS prompt. Overall, the T2CSS prompting approach enhanced text-to-Cypher accuracy by integrating domain semantics, syntactic rules, and user intention, which advocates for hybrid prompting strategies in text-to-Cypher tasks requiring both logical coherence and execution fidelity.

6.4.3. Computational resource and token utilisation analysis

Following our previous experimental results in Section 6.4.2, it is evident that ChatGPT 4.0 with T2CSS outperformed other models in terms of LA and EA. However, given that ChatGPT 4.0 is a commercial, non-open-source model, it is crucial to analyse the token utilisation of various LLMs, where token efficiency directly impacts the cost and performance. Lower token counts generally indicate more efficient query generation, faster processing times and reduced computational costs. In this context, LLMs were categorised into two sets: set 1 (medium-scale LLMs) and set 2 (large-scale LLMs).

With the same prompts from the proposed T2CSS, the mean token usage of the five LLMs was calculated and compared in Fig. 10. Set 1 exhibited a higher token usage and indicated that it requires more tokens to generate queries, which may lead to higher computational costs and longer processing times. Compared to set 1, set 2 showed better token efficiency, especially ChatGPT 3.5, making them more suitable for applications where computational efficiency and processing speed are critical. By analysing the token utilisation, a clear understanding of the trade-offs between accuracy and computational resources was provided. For instance, ChatGPT 4.0 with the proposed T2CSS offered better accuracy than ChatGPT 3.5 based on T2CSS, but it comes with higher

Table 6
Results of diverse models under different input conditions.

Strategies	Models		Inputs			LA	EA
			User inquiries	Cypher syntax	Semantic schema		
Strategy 1	Group 3	Mistral 7B	√			0.44	0.42
		LLaMA 2	√			0.54	0.54
	Group 4	Claude 2.0	√			0.46	0.4
		ChatGPT 3.5	√			0.48	0.44
		ChatGPT 4.0	√			0.58	0.56
Strategy 2	Group 4	Mistral 7B	√	√		0.64	0.58
		LLaMA 2	√	√		0.68	0.66
		Claude 2.0	√	√		0.66	0.6
	Group 4	ChatGPT 3.5	√	√		0.68	0.64
		ChatGPT 4.0	√	√		0.76	0.74
Strategy 3	Group 3	Mistral 7B	√		√	0.62	0.6
		LLaMA 2	√		√	0.68	0.62
	Group 4	Claude 2.0	√		√	0.64	0.62
		ChatGPT 3.5	√		√	0.68	0.64
		ChatGPT 4.0	√		√	0.78	0.78
Strategy 4	Group 3	Mistral 7B	√	√	√	0.74	0.72
		LLaMA 2	√	√	√	0.76	0.76
	Group 4	Claude 2.0	√	√	√	0.74	0.72
		ChatGPT 3.5	√	√	√	0.8	0.76
		ChatGPT 4.0	√	√	√	**0.86**	**0.86**

Bold values represent the maximum value in each column.

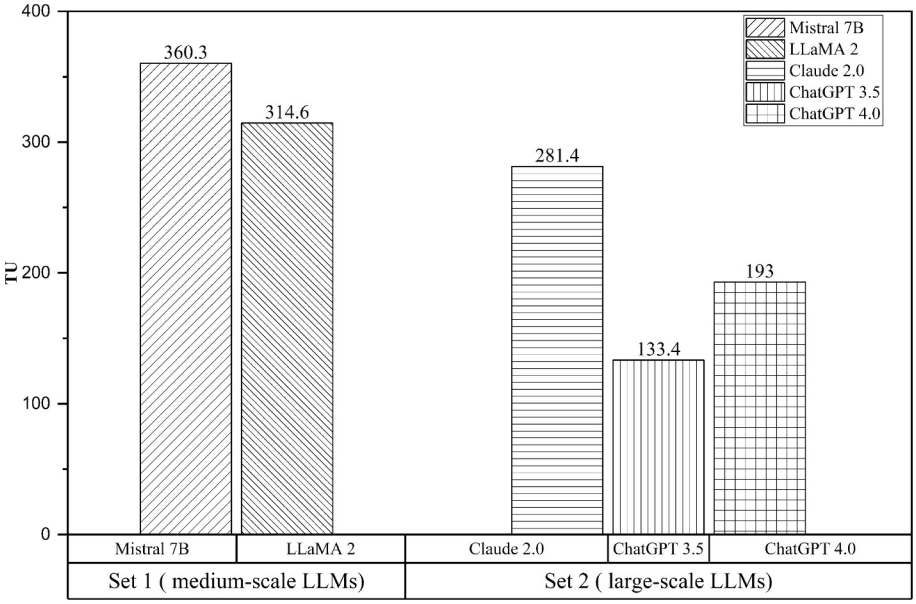


Fig. 10. Performances of diverse LLMs with the proposed T2CSS on TU.

computational costs. In this context, the token usage analysis helps in making informed decisions about the deployment of LLMs in various practical applications, balancing performance, cost, and computational feasibility.

7. Discussions

The findings of this study extend the principles of prompt engineering into an applied domain context, which illustrates how a semantic schema can guide an LLM in generating graph queries. By injecting a lightweight domain schema into the prompt, the T2CSS approach bridges the gap between unstructured natural language questions and the structured graph query language Cypher. It should be noted that this is not a new fundamental theory of prompting, but rather a practical extension of schema-guided prompt design to the text-to-Cypher task. The approach demonstrates that even without model retraining, an LLM can be aligned with domain-specific knowledge through in-context learning. When the domain ontology is distilled into a concise semantic schema, the model uses those structured relationships as contextual priors for query construction. This yields a form of structured knowledge grounding in the prompt, which reduces ambiguity in the model's understanding of user intent. Furthermore, the ability to regenerate the prompt rapidly with an updated schema (instead of retraining the model for each schema change) suggests a scalable alternative to traditional fine-tuning. Maintaining supervised learning across ten schema revisions would require ten retraining rounds, whereas the T2CSS prompt is regenerated in seconds by simply merging the updated schema fragment. The dynamic filtering of prompt content in T2CSS also exemplifies a resource-efficient strategy through ensuring that only relevant schema triples and Cypher syntax are included. In this regard, it offers a promising solution to token limitations by prioritising schema-level relationships (e.g., "machine-produce-coil" in Fig. 6) over instance-level data for resource-aware LLM deployment. In theory, this highlights how in-context learning can be

augmented with a structured semantic component to improve accuracy without exceeding context length limits. In summary, the theoretical significance lies in showing that schema-guided prompting is a viable and efficient approach to grounding LLMs in specialised domain knowledge for query generation.

In practical terms, the proposed T2CSS approach offers several benefits for real-world applications, especially in industrial and enterprise settings. The first one is democratized access for non-experts. The method allows domain specialists and other non-technical users to query KGs directly using natural language without requiring expertise in Cypher or database query languages. In other words, as users can obtain answers from data by simply asking questions in plain language, it lowers the barrier to accessing complex graph databases. Therefore, organisations can eliminate lengthy technical training phases and dependency on database experts for faster decision-making and problem-solving. Also, it is helpful for knowledge sharing and reuse. By making querying more intuitive, T2CSS facilitates broader knowledge sharing across teams. Practitioners in different roles can retrieve information from the knowledge graph on their own, which promotes collaboration and reuse of data insights. The natural language interface serves as a user-friendly layer for enterprise knowledge management, so information is no longer siloed with IT specialists. In this regard, this improves human-computer interaction with graph data and empowers users to explore data on demand for evidence-based decision processes. Another benefit lies in cost efficiency and flexibility. The prompt-based approach reduces the need for extensive model training or costly development of custom query tools for each domain. Employing large commercial LLMs with T2CSS yields the highest accuracy, but the framework also works with smaller open-source models at lower cost, albeit with some accuracy trade-off. In practice, organisations can balance performance and cost according to their resources. For instance, a company could choose an open-source LLM for routine queries to save on costs, and reserve a commercial model for cases requiring the utmost accuracy. The ability to update the domain schema without retraining the model further contributes to cost and time efficiency for rapidly adapting to evolving data. Additionally, T2CSS enhances scalability in industrial deployments. Because the semantic schema is a compact representation of domain knowledge, the prompt remains concise even as the underlying database grows. This schema-level prompting strategy reduces token usage per query, which lowers computational overhead and latency. In other words, queries can be generated and executed faster, making real-time or near-real-time querying feasible for time-sensitive applications. The approach also scales to new domains by simply swapping in a new schema and context, rather than rebuilding an entirely new system, which is advantageous for organisations that maintain multiple domain-specific KGs. Overall, these practical benefits mean that the T2CSS approach can accelerate data-driven decision-making and broaden the adoption of graph databases in industry.

Despite its promise, T2CSS has several limitations. Firstly, the T2CSS method relies on a manually crafted semantic schema tailored to a particular domain, which means the solution does not generalize out of the box to other domains. Secondly, the current implementation and evaluation are limited to English-language queries. Posing inquiries in other languages remains untested. Thirdly, maintaining and updating the semantic schema can be labour-intensive. In rapidly evolving domains, keeping the schema up-to-date with new concepts and relationships would demand continuous effort. Finally, the highest performance in our experiments was achieved with a large proprietary model (GPT-4), which may be expensive or inaccessible for some organisations. Future research should address the above limitations and explore new extensions of the schema-guided prompt approach. One important direction is to develop techniques for automating or evolving the semantic schema. Another vital extension is multilingual support. Additionally, the general framework could be adapted to other query languages and database types beyond Neo4j. Finally, future direction can focus on achieving comparable performance with lower resource requirements, such as exploring knowledge distillation or hybrid approaches that combine the strengths of large models with the efficiency of smaller ones.

8. Conclusions

This study addresses the critical challenge of translating unstructured natural language into structured Cypher queries by ICL with LLM prompting. The proposed T2CSS demonstrates that schema-guided ICL enhances both syntactic precision and semantic alignment (86 % logical and execution accuracies) in text-to-Cypher tasks and outperforms baselines. T2CSS extends the application of schema-guided prompt engineering and the deployment of LLMs. Also, it empowers industries like steel manufacturing to simplify decision-making through timely and context-aware knowledge retrieval. Limitations include, but are not limited to, reliance on manually curated schemas and costs associated with commercial LLMs. Future work will explore incremental learning, multilingual and multimodal support, and computational cost optimisation.

CRediT authorship contribution statement

Yuwei Wan: Writing – original draft, Validation, Methodology, Investigation, Formal analysis, Conceptualization. **Zheyuan Chen:** Writing – original draft, Validation, Methodology, Investigation. **Ying Liu:** Writing – review & editing, Supervision, Methodology, Conceptualization. **Chong Chen:** Validation, Methodology, Investigation. **Michael Packianather:** Writing – review & editing, Supervision.

Declaration of competing interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

We confirm that we have given due consideration to the protection of intellectual property associated with this work and that there are no impediments to publication, including the timing of publication, with respect to intellectual property. In so doing we confirm that we have followed the regulations of our institutions concerning intellectual property.

We understand that the Corresponding Author, Prof Ying Liu, is the sole contact for the Editorial process (including Editorial Manager and direct communications with the office). Prof Liu is responsible for communicating with the other authors about progress, submissions of revisions and final approval of proofs. We confirm that we have provided a current, correct email address which is accessible by the Corresponding Author and which has been configured to accept email from LiuY81@cardiff.ac.uk.

Data availability

The authors do not have permission to share data.

References

[1] C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen, D. Wilkins, A comparison of a graph database and a relational database: a data provenance perspective, in: Proceedings of the 48th Annual ACM Southeast Conference, 2010, pp. 1–6.
[2] Y. Unal, H. Oguztuzun, Migration of data from relational database to graph database, in: Proceedings of the 8th International Conference on Information Systems and Technologies, 2018, pp. 1–5.
[3] M. Besta, et al., Demystifying Graph Databases: Analysis and Taxonomy of Data Organization, System Designs, and Graph Queries 56(2), 2023, pp. 1–40.
[4] A. Ariadi, T. Shi, H. Ma, A.S. Da Silva, S. Hartmann, A graph database supported GA-based approach to social network analysis, in: in 2021 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2021, pp. 01–08.
[5] S. Sen, A. Mehta, R. Ganguli, S. Sen, Recommendation of influenced products using association rule mining: Neo4j as a case study, SN Comput. Sci. 2 (2021) 1–17.

[6] R. Ojino, R. Ndolo, Knowledge graph for fraud detection: case of fraudulent transactions detection in Kenyan SACCOs, in: International Conference on Artificial Intelligence: Towards Sustainable Intelligence, Springer, 2023, pp. 178–186.
[7] L. Nizzoli, M. Avvenuti, M. Tesconi, S. Cresci, Geo-semantic-parsing: AI-powered geoparsing by traversing semantic knowledge graphs, Decis. Support. Syst. 136 (2020) 113346.
[8] B. Liu, X. Wang, P. Liu, S. Li, Q. Fu, Y. Chai, UniKG: A unified interoperable knowledge graph database system, in: in 2021 IEEE 37th International Conference on Data Engineering (ICDE), IEEE, 2021, pp. 2681–2684.
[9] N. Francis, et al., Cypher: an evolving query language for property graphs, in: Proceedings of the 2018 International Conference on Management of Data, 2018, pp. 1433–1445.
[10] H.S. Dar, M.I. Lali, M.U. Din, K.M. Malik, S.A.C. Bukhari, Frameworks for Querying Databases Using Natural Language: A Literature Review, arXiv preprint arXiv: 1909.01822, 2019.
[11] J.Z. Pan, et al., Large Language Models and Knowledge Graphs: Opportunities and Challenges, arXiv preprint arXiv:2308.06374, 2023.
[12] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, X. Wu, Unifying Large Language Models and Knowledge Graphs: A Roadmap, arXiv preprint arXiv:2306.08302, 2023.
[13] K. Singhal, et al., Large language models encode clinical knowledge, Nature 620 (7972) (2023) 172–180.
[14] A. Clemedtson, B. Shi, GraphRAFT: Retrieval Augmented Fine-Tuning for Knowledge Graphs on Graph Databases, arXiv preprint arXiv:2504.05478, 2025.
[15] Z. Zhong, L. Zhong, Z. Sun, Q. Jin, Z. Qin, X. Zhang, Synthet2c: Generating Synthetic Data for Fine-tuning Large Language Models on the text2cypher Task, arXiv preprint arXiv:2406.10710, 2024.
[16] N. Mihindukulasooriya, S. Tiwari, C.F. Enguix, K. Lata, Text2kgbench: a benchmark for ontology-driven knowledge graph generation from text, in: International Semantic Web Conference, Springer, 2023, pp. 247–265.
[17] Q.-B.-H. Tran, A.A. Waheed, S.-T. Chung, Robust text-to-cypher using combination of BERT, GraphSAGE, and transformer (CoBGT) model, Appl. Sci. 14 (17) (2024) 7881.
[18] Y.-L. Liang, C.-Y. Chang, S.-J. Wu, KEI-CQL: a keyword extraction and infilling framework for text to cypher query language translation, Int. J. Des. Anal. Tools Integr. Circ. Syst. 13 (1) (2024).
[19] Z. Ming, G. Sharma, J.K. Allen, F. Mistree, An ontology for representing knowledge of decision interactions in decision-based design, Comput. Ind. 114 (2020) 103145.
[20] L. Otero-Cerdeira, F.J. Rodríguez-Martínez, A. Gómez-Rodríguez, Ontology matching: a literature review, Expert Syst. Appl. 42 (2) (2015) 949–971.
[21] M. Fernández-López, A. Gómez-Pérez, N. Juristo, Methontology: from Ontological Art towards Ontological Engineering, 1997.
[22] Q. Cao, S. Beden, A. Beckmann, A core reference ontology for steelmaking process knowledge modelling and information management, Comput. Ind. 135 (2022) 103574.
[23] A. Konys, An ontology-based knowledge modelling for a sustainability assessment domain, Sustainability 10 (2) (2018) 300.
[24] L. Ehrlinger, W. Wöß, Towards a definition of knowledge graphs, in: SEMANTiCS (Posters, Demos, SuCCESS) 48(1–4), 2016, p. 2.
[25] F.N. Al-Aswadi, H.Y. Chan, K.H. Gan, From ontology to knowledge graph trend: ontology as foundation layer for knowledge graph, in: Iberoamerican Knowledge Graphs and Semantic Web Conference, Springer, 2022, pp. 330–340.
[26] F. Maibaum, J. Kriebel, J.N. Foege, Selecting textual analysis tools to classify sustainability information in corporate reporting, Decis. Support. Syst. 183 (2024) 114269.
[27] W. Wang, V.W. Zheng, H. Yu, C. Miao, A survey of zero-shot learning: settings, methods, and applications, ACM Trans. Intell. Syst. Technol. 10 (2) (2019) 1–37.
[28] J. He, N. Lin, Q. Bai, H. Liang, D. Zhou, A. Yang, Towards fair decision: a novel representation method for debiasing pre-trained models, Decis. Supp. Syst. 181 (2024) 114208.
[29] A. Ahmed, X. Zeng, R. Xi, M. Hou, S.A. Shah, MED-prompt: a novel prompt engineering framework for medicine prediction on free-text clinical notes, J. King Saud Univ. 36 (2024) 101933.
[30] M. Hornsteiner, M. Kreussel, C. Steindl, F. Ebner, P. Empl, S. Schönig, Real-time text-to-cypher query generation with large language models for graph databases, Fut. Internet 16 (12) (2024) 438.
[31] E. Oro, M. Ruffolo, A Natural Language Interface for Querying RDF and Graph Databases, Consiglio Nazionale delle Ricerche Istituto di Calcoloe Reti and Alte Prestazioni, 2015.
[32] A. Litvin, V.Y. Velychko, V. Kaverynskyi, Tree-based semantic analysis method for natural language phrase to formal query conversion, Radio Electron. Comp. Sci. Control 2 (2021) 105–113.
[33] C. Sun, A Natural Language Interface for Querying Graph Databases, Massachusetts Institute of Technology, 2018.
[34] Y. Zou, Y. Wang, D. Liu, Q2Cypher: Converting natural language questions to cypher with fine-tuned large language models, in: in 2024 5th International Conference on Artificial Intelligence and Computer Engineering (ICAICE), IEEE, 2024, pp. 783–788.
[35] E.J. Hu, et al., Lora: low-rank adaptation of large language models, ICLR 1 (2) (2022) 3.
[36] T. Dettmers, A. Pagnoni, A. Holtzman, L. Zettlemoyer, Qlora: efficient finetuning of quantized llms, Adv. Neural Inf. Proces. Syst. 36 (2023) 10088–10115.
[37] G. Feng, et al., Robust nl-to-cypher translation for kbqa: Harnessing large language model with chain of prompts, in: China Conference on Knowledge Graph and Semantic Computing, Springer, 2023, pp. 317–326.
[38] V. Zhong, C. Xiong, R. Socher, Seq2sql: Generating Structured Queries From Natural Language Using Reinforcement Learning, arXiv preprint arXiv: 1709.00103, 2017.
[39] T. Yu, Z. Li, Z. Zhang, R. Zhang, D. Radev, Typesql: Knowledge-based Type-aware Neural Text-to-sql Generation, arXiv preprint arXiv:1804.09769, 2018.
[40] R. Cao, L. Chen, Z. Chen, Y. Zhao, S. Zhu, K. Yu, LGESQL: Line Graph Enhanced Text-to-SQL Model With Mixed Local and Non-local Relations, arXiv preprint arXiv:2106.01093, 2021.
[41] Y. Chang, et al., A Survey on Evaluation of Large Language Models, ACM Transactions on Intelligent Systems and Technology, 2023.
[42] A.Q. Jiang, et al., Mistral 7B, arXiv preprint arXiv:2310.06825, 2023.
[43] A. Guo, X. Li, G. Xiao, Z. Tan, X. Zhao, SpCQL: a semantic parsing dataset for converting natural language into Cypher, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022, pp. 3973–3977.
[44] D.L. McGuinness, F. Van Harmelen, OWL web ontology language overview, W3C Recommend. 10 (10) (2004) 2004.

Yuwei Wan received Ph.D. degree from School of Engineering, Cardiff University, UK in 2025. She has published over 10 papers in international journals and conferences. Her research interests mainly focus on data mining, machine learning, knowledge engineering, intelligent manufacturing, decision-support systems, and large language models.

Zheyuan Chen received PhD degree from the School of Engineering, Cardiff University, UK in 2023. Dr. Chen is conducting post-doctoral research at Guangzhou Institute of Industrial Intelligence. He has published over 10 papers in international journals and conferences. His research interests mainly focus on intelligent manufacturing, knowledge engineering and large language models.

Ying Liu is currently Professor and Chair in Intelligent Manufacturing and the Group Leader for High-value Manufacturing with the Department of Mechanical Engineering at the School of Engineering in Cardiff University, Cardiff, UK. His research interests focus primarily on engineering informatics, digital and intelligent (smart) manufacturing, AI and machine learning for engineering, design methodology and process and advanced ICT in engineering design and manufacturing, in which areas he has published over 250 scholarly articles, one edited book and more than 25 journal special issues.

Chong Chen received PhD degree from the School of Engineering, Cardiff University, UK in 2021. He is currently a lecturer with Guangdong Provincial Key Laboratory of Cyber-Physical System, Guang-dong University of Technology. He has published over 40 papers in international journals and conferences. His research interests mainly focus on intelligent manufacturing, artificial intelligence and industrial big data.

Michael Packianather is a Senior Lecturer at Cardiff University. In 2005 he was appointed as the Director of Postgraduate Research Studies at the MEC. His research interests include Industry 4.0, data mining and machine learning, deep neural networks and AI, robotics and human-robot collaboration, and intelligent optimisation. Dr. Michael Packianather has several journal and conference publications. He is a Full Member of the EPSRC College, an Expert Reviewer for European Commission FET PROACT Research Programme, and an Expert Reviewer for SMART4.0 Programme. He is an Associate Editor of the IEEE Systems Engineering Journal and IMechE Proceedings Part I.