

Making Mathematical Knowledge Explainable, Accessible and Interoperable Through Large Language Model Integration

Jan Range¹, Björn Schembera^{2,*} and Dominik Göddeke^{1,2}

¹Stuttgart Center for Simulation Science, University of Stuttgart, Germany

²Institute of Applied Analysis and Numerical Simulation, University of Stuttgart, Allmandring 5b, 70569 Stuttgart, Germany

Abstract

Mathematical models are central to formalizing research problems, yet their documentation often falls short of FAIR principles. Knowledge bases such as the Mathematical Model Database (MathModDB) address this gap by providing curated, semantically rich representations of mathematical models. Built on Wikibase, the same open-source infrastructure underlying Wikidata, MathModDB utilizes Semantic Web technologies to support Linked Open Data, collaborative editing, and the storage of semantically enriched metadata, making it a domain-specific knowledge graph within the broader Wikidata ecosystem. However, access to MathModDB currently requires either navigating a complex web interface or proficiency in SPARQL and Wikibase APIs, posing significant barriers for potential users. In addition, the combination of such curated knowledge bases with actual research data stored, e.g., in Dataverse repository instances, remains a challenge. To overcome these limitations, we propose integrating Large Language Models (LLMs) with MathModDB via a Model Context Protocol (MCP) server that exposes a vector-indexed schema retrieval and Steiner-tree-based join planner, combining dialogue-based natural language interaction with curated, epistemically grounded knowledge. Although instantiated on MathModDB, the architecture can be applied to other Wikibase-based systems. We demonstrate that this approach enables epistemically grounded LLM usage, improves model explainability and accessibility beyond what the standard Wikibase interface offers, and simplifies interoperability with external databases and tools, such as Dataverse data repositories. We illustrate the benefits of combining the accessibility of an LLM with the epistemic safety of a curated knowledge base through the adaptability of the MCP protocol by two use cases: linking an enzyme-kinetics dataset from a Dataverse repository to its mathematical model in MathModDB, and exploring the Stokes-Darcy model for coupled flow including federated retrieval of creator biographies from Wikidata.

Keywords

Mathematical Knowledge Bases, Large Language Models, Model Context Protocol, FAIR, XAIR, Knowledge Graphs, Vector Databases, Wikibase

1. Introduction

Mathematical knowledge is vast, ranging from classic mathematical artifacts like proofs on paper to complex numerical models on supercomputers, and it is usually scattered through different categories of data, i.e., symbolic, numeric, geometric, observational, model, or text data [1, 2, 3]. For the majority of scientific domains, mathematical models are the central means to describe their research problems in a formalized and abstract way, so researchers can make predictions, claims, and analyzes about the research problem.

To comply with the FAIR principles [4] and good scientific practice, all steps that led to a research result (i.e., the predictions, claims and analyses) must be documented. This includes, in particular, documentation of the mathematical models used, created or modified during the research process. Using mathematical knowledge bases for documenting the involved mathematical models enables exploring, researching and finding similar, alternate or complementing models and solution schemes.

Wikidata'26: Wikidata Workshop at ISWC 2026

*Corresponding author.

✉ jan.range@simtech.uni-stuttgart.de (J. Range); bjoern.schembera@mathematik.uni-stuttgart.de (B. Schembera); dominik.goeddeke@mathematik.uni-stuttgart.de (D. Göddeke)

ORCID 0000-0001-6478-1051 (J. Range); 0000-0003-2860-6621 (B. Schembera); 0000-0002-1552-497X (D. Göddeke)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Mathematical knowledge bases for mathematical models exist, and one example is the Mathematical Model Database (MathModDB) [5]. It is a curated database of mathematical models, developed by the Mathematical Research Data Initiative (MaRDI) [6]. MaRDI is embedded into the greater national research data infrastructure (NFDI) in Germany, giving MathModDB a potentially high impact in the German research landscape and beyond. MathModDB uses Wikibase [7] as its technology stack – the same open-source infrastructure underlying Wikidata – fostering a flexible, semantic, and extensible representation of data assets in line with Semantic Web standards [8]. Its support for Linked Open Data, collaborative editing, and scalability makes it a foundation for managing complex mathematical models, and positions MathModDB as a domain-specific extension of the broader Wikidata ecosystem. MathModDB with Wikibase as its technological platform is in principle explainable AI ready (XAIR) as it consists of data that are readable by both humans and machines, and which enables the storage of epistemic metadata [9] about mathematical models (e.g., descriptions, governing equations, assumptions, constraints, conditions, quantities, but also links to proofs or other theoretic grounding), going beyond mere provenance metadata. However, usage of MathModDB is either limited to a web interface for browsing and modifying the data, or requires expertise in writing API calls or SPARQL for complex, federated queries. This is often particularly discouraging for users, as it means they have to learn a new and formal query language or the technical details of the Wikibase API. For machine interaction, too, programmers must be proficient in SPARQL. Results returned are in machine-readable formats such as JSON, XML or CSV, which require further processing to make them explainable, presentable and thus fully XAIR [10].

To address these shortcomings and as users prefer natural-language queries over writing SPARQL, we propose a Model Context Protocol (MCP) based approach to combine the strengths of LLMs with curated knowledge bases. Although instantiated on MathModDB, the architecture is generic across Wikibase-based systems (including Wikidata) and thus offers a reusable integration pattern for the broader Wikibase and Linked Open Data ecosystem.

In this paper, we demonstrate that curated knowledge stored in the MathModDB knowledge graph supports epistemically safe use of LLMs and facilitates steps toward XAIR-compliant research outputs in a Semantic Web setting. We show that a dialogue-based, LLM-supported interface offers substantially richer access to curated mathematical knowledge than the standard Wikibase interface – in particular with respect to model explainability and navigability. We quantify our approach by schema-retrieval evaluation and illustrate its application through two use cases. In the first use case, we show that the MCP-based approach enables end-to-end interoperability between research data and mathematical knowledge: using only a natural language query, an agent links an enzyme kinetics dataset deposited in a Dataverse data repository to its corresponding mathematical model in MathModDB, identifying the underlying rate laws and mechanistic model class without any manual integration effort. In the second case, we use the Stokes Darcy model for coupled flow to demonstrate that even structurally complex models can be explored reliably: the correct coupling conditions are retrieved directly, and biographical information about the creators, spread across different knowledge bases, is retrieved without requiring the user to construct federated SPARQL queries or possess detailed knowledge of the underlying LOD database structures.

2. Related Work

2.1. Knowledge Bases for Mathematical Models

Mathematical knowledge management was already being discussed in the early 2000s [11]. Knowledge bases for mathematical models include databases, ontologies and taxonomic classifications and are discussed in this subsection.

Ontologies for mathematical models are either domain-specific [12, 13, 14, 15, 16] or primarily taxonomic in nature [17, 18], and none provides a unified, general and semantically queryable representation of mathematical relationships enriched with epistemic metadata. MathModDB addresses this gap. As part of the NFDI project MaRDI, MathModDB was initially developed as an ontology [19] for mathemat-

ical models and subsequently populated as a knowledge graph [5]. MathModDB consists of 8 classes for a rich semantic description of mathematical models in their specific research context: *Academic Discipline*, *Research Problem*, *Mathematical Model*, *Mathematical Formulation*, *Quantity*, *Quantity Kind*, *Computational Task* and *Publication*. By February 2025, MathModDB was moved from the OWL world to the Wikibase ecosystem [20], becoming a subgraph of the larger MaRDI KG [3]. The Wikibase technology stack offers open-source development momentum, a large mathematical user base [21], and the possibility to build on existing knowledge from Wikidata and other external knowledge bases. This potential is already being realized: relevant entities from Wikidata, such as publications, persons, and research institutions, have been imported into MathModDB, and direct links to Wikidata have been set for many data items in MathModDB. Mathematics-specific extensions include rendering of mathematical expressions, formula search, and the integration of data sources such as zbMath and swMath. Moreover, Wikibase supports qualifiers for relations – important in mathematics to denote special equations such as assumptions or conditions – and facilitates linking with the broader MaRDI KG and external knowledge graphs following LOD principles. As of July 2026, MathModDB includes 229 curated mathematical models and 2,153 entries comprising 26,041 statements in total.

2.2. MCP for Knowledge Graphs and Research Data

The Model Context Protocol (MCP), introduced by Anthropic in late 2024, serves as an open standard for bidirectional and typed communication between large language model (LLM) agents and external tools or data sources. MCP has rapidly emerged as the primary integration layer for agentic AI [22, 23]. In the context of the Semantic Web, MCP is increasingly regarded as a suitable framework for knowledge graphs and SPARQL endpoints, given their standardized query languages, metadata, and inherent support for federation [24].

Multiple MCP servers for knowledge graphs are currently available. The Wikidata community maintains a Wikidata MCP server¹ that provides both SPARQL execution and a vector index over Wikidata items and properties to enable semantic search. The MathModDB MCP can be understood as a domain-specific complement: where the Wikidata MCP server offers broad general-purpose knowledge, the MathModDB MCP server provides epistemically curated mathematical knowledge, grounded in formal model descriptions and governed by domain experts. Dobriy et al. [24] extend the Spider4SPARQL benchmark to federated knowledge graph question answering (KGQA) and evaluate SPARQL-MCP agents in the areas of endpoint discovery, VoID-based schema exploration, and query formulation. Their findings indicate that high-level natural-language endpoint descriptions are more effective than structured VoID metadata for guiding source selection. The optimal structuring of schema metadata for agents remains an open research question. Schema-aware NL-to-SPARQL translation has prior history in KGQA. Bio-SODA [25] constructs SPARQL candidates by solving an approximate Steiner tree problem over a schema graph – the closest structural predecessor to our join planner – but without LLM integration or MCP. Emonet et al. [26] use LLM-based RAG with schema retrieval over federated endpoints, but rely on post-hoc query correction rather than a generation-time prior. Recently, SAGA [27] proposed schema-aware grounding for agentic SPARQL generation on Wikidata. In contrast, our approach combines Steiner-tree join planning with dual-encoder retrieval within a stateless MCP interface, targeting a Wikibase-based expert-curated knowledge graph with qualifier-mediated triple patterns. In related domains, MCP has been adapted for research data infrastructure: Pan et al. [28] unify heterogeneous research cyberinfrastructure, Dasgupta et al. [29] expose a DSpace digital repository, and Kuehl et al. [30] establish a community registry of MCP servers for biomedical knowledge bases.

2.3. FAIR, XAIR, and epistemic Metadata

For around a decade, the FAIR principles have shaped best practices in research data management [4]. They address data provenance and semantics, ensuring that the origin and meaning of data are documented in a machine-readable way. However, for applications that require scientific rigor and re-

¹<https://www.wikidata.org/wiki/Wikidata:MCP>

producibility, knowing what data means is not sufficient – one must also be able to justify why it is trustworthy. The transition from FAIR to XAIR demands metadata annotations that go beyond semantic description and document the epistemic status of data and derived claims [31]: the assumptions underlying a model, the conditions under which it is valid, or the reasoning that connects a mathematical formulation to a physical phenomenon. This shift from a semantic to an epistemic perspective is operationalized through epistemic metadata [9]. MathModDB is designed around precisely this concept: it stores epistemic metadata about mathematical models – including their governing equations, coupling conditions, and formal relationships – in a curated, machine-actionable knowledge graph, making it a XAIR-compliant resource for mathematical knowledge.

3. System Architecture

As described in Sec. 2.1, MathModDB is a curated, Wikibase-based knowledge base of mathematical models. It is human- and machine-readable, carries rich epistemic metadata, and is therefore XAIR-compliant (c.f. Sec. 2.3). However, its practical utility is limited by access barriers: Users must navigate the web interface manually, write SPARQL queries, or engage Wikibase APIs directly, which requires specialist knowledge. This is a recognized shortcoming [3] and a common challenge across Wikibase-based systems, especially for federated queries spanning multiple LOD databases [32]. To overcome these barriers and extend MathModDB with dialogue-based interaction and the contextual understanding of an LLM, we introduce the MathModDB MCP server. The server architecture is not specific to MathModDB: any Wikibase instance that exposes a SPARQL endpoint and an indexable schema can be wrapped with the same approach, making this a reusable integration pattern for Wikibase-based knowledge graphs. The transfer requires only a SPARQL endpoint exposing the target graph and an indexable schema; no modifications to the underlying knowledge graph are needed.

The server comprises three components: an offline schema index built once from the MathModDB Wikibase instance, a runtime query-planning pipeline executed per request, and an MCP interface exposing this pipeline to LLM agents via network transport. In this setup, MathModDB serves as contextual grounding for LLM-based systems [33, 34], providing the semantic and epistemic trust required in scientific settings. The system architecture is illustrated in Fig. 1. The source code of the MathMod MCP server is referenced in Sec. 7, Supplementary Material.

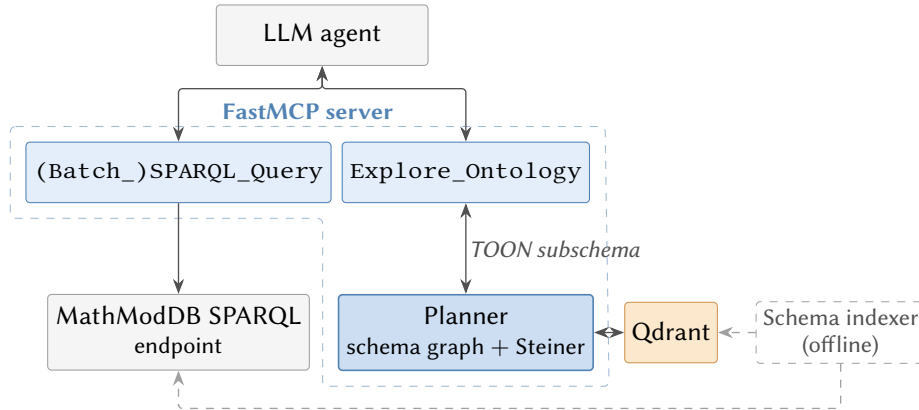


Figure 1: Architecture of the MathModDB MCP server. The LLM agent interacts with three tools exposed by the FastMCP server (dashed outline): *Explore_Ontology* delegates to the Planner, which retrieves and re-ranks schema candidates from the offline-built Qdrant index and returns a TOON-encoded subschema. *SPARQL_Query* and *Batch_SPARQL_Query* are direct passthroughs to the MathModDB endpoint. Solid arrows denote runtime flow and dashed arrows denote the offline path, in which the Schema indexer harvests the endpoint and populates Qdrant.

3.1. Schema Indexer

During initialization, the server queries the MathModDB SPARQL endpoint at <https://query.portal.mardi4nfdi.de/sparql> to collect all entities marked with the MathModDB corpus identifier (Q6534265). For each class, the server retrieves its Wikibase identifier, English label and description, associated object and data properties, and observed usage counts. For each object property, empirical subject-to-object class connections and their frequencies are recorded through separate SPARQL queries. Qualifier properties are extracted along with their usage paths, represented as subject-class and main-property pairs. The resulting structure is serialized to a YAML cache and reused across server restarts. This process produces four typed entity sets: *classes*, *object properties*, *data properties*, and *qualifier properties*. Each set contains a Wikibase Q/P identifier, textual metadata, and relational metadata for downstream graph construction.

For semantic retrieval, each entity is represented as a retrieval string derived from its label and description, optionally enriched with an LLM-generated summary. These strings are embedded using two complementary encoders: A dense OpenAI *text-embedding-3-large*² model for high-recall candidate generation and a ColBERT v2 [35] late-interaction model for fine-grained re-ranking. The resulting vectors are stored in four typed Qdrant collections, one for each entity type, on disk alongside the complete entity payload. These collections persist across server restarts and are rebuilt only when an explicit refresh flag is set. For the reported experiments, entity descriptions were enriched with LLM-generated summaries, which augments each entity’s retrieval string with a GPT-generated paragraph optimized for vector search.

3.2. Query Executioner

During query execution, the server executes the complete query-planning pipeline described in Section 4 as a single synchronous operation. The pipeline processes a natural-language input, performs dual-encoder retrieval across all four Qdrant collections, constructs and conditions the schema graph, computes the Steiner subgraph, and attaches qualifier and data-property constraints to generate a TOON-encoded output payload. All intermediate states are maintained in memory for the duration of the request, with no per-request persistence.

3.3. MCP Server Interface

The MCP server component is implemented in Python using FastMCP and is served over streamable HTTP, ensuring compatibility with any MCP-capable LLM client. Three tools are registered: `Explore_Ontology`, which accepts a short natural-language query and returns a TOON-encoded payload containing ranked schema candidates and the Steiner-derived join blueprint, and must be invoked prior to any SPARQL execution. `SPARQL_Query`, which executes a single SPARQL SELECT query against the MathModDB endpoint with preconfigured `wd :` and `wdt :` prefix bindings, subject to a strict result limit of 100 per query. Finally, `Batch_SPARQL_Query`, which executes a named dictionary of SPARQL queries in a single call, returning results keyed by the caller-supplied names. The recommended agent workflow is schema-first: the LLM first invokes `Explore_Ontology` to obtain valid entity identifiers and a join blueprint before issuing targeted data retrieval queries. The TOON-encoded subschema returned by `Explore_Ontology` acts as a generation-time prior, fixing the candidate class and predicate space before any query is written rather than filtering results afterwards.

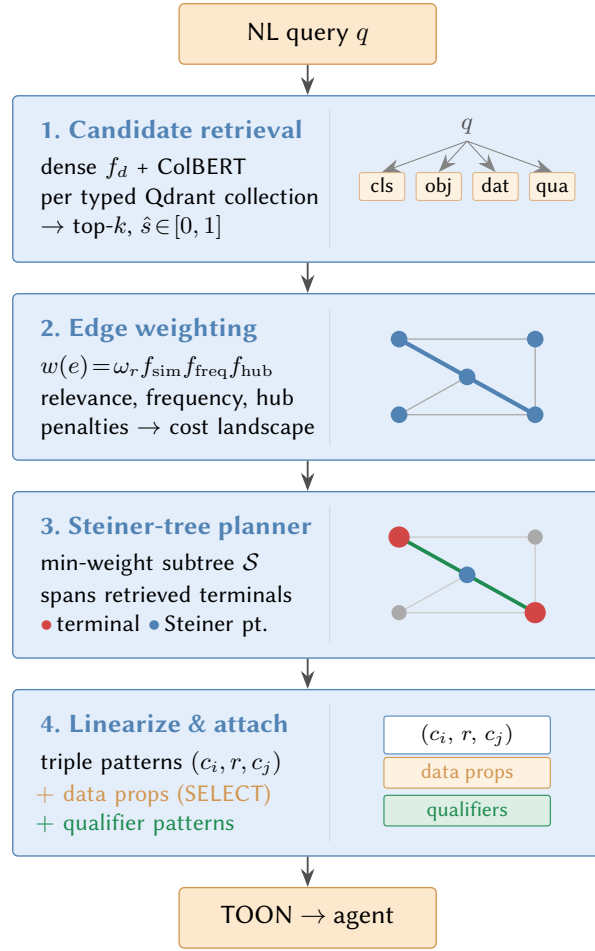


Figure 2: Query-planning pipeline executed by Explore_Ontology. A natural-language query q is retrieved against the four typed Qdrant collections via a dense encoder f_d followed by ColBERT re-ranking (Section 4.3). The normalized scores condition the schema graph through relevance, frequency, and hub penalties. Steiner tree (Section 4.4) selects the minimum-weight subtree $\mathcal{S}$ spanning all retrieved terminals, and the linearized triple patterns, together with attached data properties and qualifier patterns, are returned to the agent as a TOON payload.

4. Methods

4.1. Overview

Query planning proceeds in three stages. A natural-language query is first mapped to ranked candidate schema entities via dual-encoder retrieval over the typed Qdrant³ index, with retrieval scores assigned as edge weights on a directed schema graph. A Steiner tree [36] on the undirected projection then yields the minimum-weight join skeleton connecting all retrieved terminals. Data properties and qualifier constraints are attached to the Steiner nodes, producing a structured query-planning package for translation into executable SPARQL by an LLM agent. The workflow is detailed in the following sections and depicted in Figure 2. All hyperparameters – including the number of candidates retrieved per collection ($k=6$), the relation-type base weights, and the property stop list – were fixed prior to and independently of the evaluation.

²<https://developers.openai.com/api/docs/models/text-embedding-3-large>

³<https://qdrant.tech>

4.2. Schema Graph

Although Wikibase organizes its content around items, statements, and qualifiers rather than RDF triples, the endpoint exposes the data through a Wikibase-specific RDF projection. We adopt this RDF view throughout: Schema elements are treated as classes and properties, and the relations between them as directed triples. This allows reasoning about the schema with standard graph machinery while remaining faithful to how queries are actually evaluated against the endpoint.

Let $\mathcal{C}$, $\mathcal{D}$, and $\mathcal{O}$ denote the sets of ontology classes, data properties, and object properties extracted from the MathModDB Wikibase instance. We represent the schema as a directed, weighted graph

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}), \quad \mathcal{V} = \mathcal{C} \cup \mathcal{D}, \quad (1)$$

in which nodes are classes and data properties, and each directed edge $e = (c_i, r, c_j) \in \mathcal{E}$ with $c_i, c_j \in \mathcal{C}$ is labelled by an object property $r \in \mathcal{O}$ and carries a base weight $\omega_r > 0$. The base weight encodes how directly the edge expresses a triple pattern in the underlying RDF projection: we distinguish three relation types with increasing weights, namely (i) direct object-property links, (ii) qualifier-mediated object-property links, and (iii) qualifier-to-data-property usage paths. Higher weights make less direct relations more costly to traverse. To avoid degenerate shortcut paths through generic, high-frequency properties (e.g., *instance of*), we exclude a curated stop list of such properties from $\mathcal{E}$ before any further computation.

4.3. Candidate Retrieval and Schema Graph Conditioning

Given a natural-language query q , the server first identifies which schema entities are most likely relevant. We use a two-stage retriever that operates separately on each typed Qdrant collection, which are classes, object properties, data properties or qualifier properties.

Stage 1: Dense Retrieval A dense encoder f_d maps the query and each candidate entity e into a single vector. Candidates are scored by cosine similarity,

$$s_d(q, e) = \frac{f_d(q)^\top f_d(e)}{\|f_d(q)\| \|f_d(e)\|}, \quad (2)$$

and the top- k candidates per collection are kept. This stage is tuned for recall and broadly collects candidates.

Stage 2: Late-interaction Re-ranking The candidates are re-ranked with a ColBERT scorer [35] that compares the query and entity at the token level rather than as single vectors. Let $\mathbf{Q} = [\mathbf{q}_1, \dots, \mathbf{q}_m]$ and $\mathbf{E} = [\mathbf{e}_1, \dots, \mathbf{e}_n]$ be the token embeddings of the query and the entity text. The MaxSim score

$$s_{ci}(q, e) = \sum_{i=1}^m \max_{j=1}^n \mathbf{q}_i^\top \mathbf{e}_j \quad (3)$$

lets each query token pick its best-matching entity token before the scores are summed, which improves precision on short, terminology-heavy queries that are typical of schema lookup. To make scores comparable across the four collections, we normalize them within each type to obtain $\hat{s} \in [0, 1]$.

Conditioning the schema graph The normalized scores are then injected into the schema graph as a query-dependent cost. For each directed edge $e_{ij} = (c_i, r, c_j) \in \mathcal{E}$, we replace the base weight ω_r by

$$w(e_{ij}) = \omega_r \cdot f_{sim}(e_{ij}) \cdot f_{freq}(r) \cdot f_{hub}(c_i, c_j), \quad (4)$$

where the three multiplicative factors penalize, respectively, semantically irrelevant edges, over-represented properties, and edges that pass through high-degree hub nodes. Each factor has a clear intuition. The cost f_{sim} penalizes semantically irrelevant edges by combining the retrieval scores of the subject class, the predicate, and the object class into a single edge-level relevance and turning it into a cost,

$$f_{\text{sim}}(e_{ij}) = \epsilon + (1 - \hat{s}_{\text{edge}}(e_{ij}))^\gamma, \quad \hat{s}_{\text{edge}} = \alpha_s \hat{s}(c_i) + \alpha_p \hat{s}(r) + \alpha_o \hat{s}(c_j), \quad (5)$$

so that edges close to the query become cheap and edges far from it become expensive. The constant $\epsilon > 0$ keeps weights strictly positive, $\gamma > 1$ sharpens the contrast between relevant and irrelevant edges, and the weights satisfy $\alpha_s + \alpha_p + \alpha_o = 1$ with α_p dominant, since the predicate is the most informative element of a triple pattern. Next, the factor f_{freq} discourages over-represented properties, since properties reused across many triples (e.g. generic linking properties) tend to produce ambiguous joins. We penalize them linearly through

$$f_{\text{freq}}(r) = 1 + \alpha_f \cdot \frac{\text{freq}(r)}{\max_{r'} \text{freq}(r')}, \quad (6)$$

where r' is the most frequent property, so that the frequent properties pay larger penalties. Finally, the factor

$$f_{\text{hub}}(c_i, c_j) = 1 + \alpha_h \cdot \delta, \quad \delta \in [0, 1] \quad (7)$$

is the normalized maximum degree of c_i and c_j , discourages paths that funnel through high-connectivity nodes and would otherwise act as cheap but uninformative shortcuts.

Together, these three factors turn the static schema graph into a query-conditioned cost landscape, without requiring any further model inference at query time.

4.4. Steiner-Based Join Planning and Qualifier Conditioning

The retrieved classes and data properties form the terminals that the join plan must connect:

$$T = T_{\mathcal{C}} \cup T_{\mathcal{D}} \subseteq \mathcal{V}, \quad T_{\mathcal{C}} \subseteq \mathcal{C}, \quad T_{\mathcal{D}} \subseteq \mathcal{D}. \quad (8)$$

On the undirected projection $\tilde{\mathcal{G}}$ of the conditioned schema graph, we look for the minimum-weight connected subtree $\mathcal{S} \subseteq \tilde{\mathcal{G}}$ that spans all terminals in T . This is the classical Steiner tree problem in graphs [36] and produces subgraph $\mathcal{S}$. Intuitively, the Steiner tree is the cheapest backbone in the schema that touches every retrieved entity, and additional non-terminal classes (Steiner points) are added only when they shorten the overall structure. Because the optimization runs on the undirected projection, we restore the original triple direction in a post-processing step by mapping each undirected Steiner edge to its best-matching directed counterpart in $\mathcal{G}$, using property identity and edge weight as tiebreakers.

The resulting subgraph $\mathcal{S}$ is then linearized into candidate triple patterns (c_i, r, c_j) , each annotated with the empirical occurrence count taken from the schema index. $\mathcal{S}$ is not a query but a join blueprint and it constrains subsequent SPARQL formulation to schema-valid paths, which reduces the combinatorial space in which an LLM agent must search to produce a correct WHERE clause.

Two further pieces of information are then attached to this blueprint. First, for each Steiner class node $c \in \mathcal{S}$, the data properties that the schema index records as valid for c become projection candidates for the SPARQL SELECT clause. Second, qualifier properties are matched to Steiner subject classes and main properties via the precomputed usage paths, yielding conditional statement patterns for qualified triples. The final output is a structured package containing the Steiner-derived join candidates, the class-conditioned projection fields, and the qualifier-conditioned statement patterns, serialized in TOON format⁴ and handed to the LLM agent. All retrieved classes and data properties are passed as Steiner

⁴<https://github.com/toon-format/toon>

terminals to the join planner without applying a similarity threshold; precision is delegated to the Steiner optimization, which selects the minimum-weight subgraph over the full terminal set.

5. Results

For the evaluation we first analyse the schema-retrieval quality on a set of typical researcher queries, then study two datasets from MathModDB representing different mathematical models from continuum mechanics and enzyme kinetics. The queries for the analysis of the schema-retrieval quality were chosen to cover the main topical areas of the MathModDB schema, targeting different parts of the ontology. The two case studies were selected to represent complementary interoperability scenarios: Case 1 exercises cross-repository linking between a data repository and MathModDB, while Case 2 demonstrates query simplification and federated retrieval within MathModDB and across external knowledge bases including Wikidata. Together, the quantitative analysis and the case studies demonstrate that coupling an LLM with a curated knowledge base yields both epistemically reliable responses and a level of accessibility that neither component achieves alone. All queries, prompts, LLM chats and other material supporting the results are available in the Supplemental Material section (c.f. Sec. 7).

Table 1

Schema-context provision evaluation: recall at 10 (R@10), precision at 10 (P@10), F1 and mean input-token usage per query, aggregated by topical group.

Topical group	R@10	P@10	F1	#Tokens
Formulations & equations	0.79	0.28	0.42	41 851
Model transformations	0.91	0.43	0.58	27 229
Tasks & problem types	0.83	0.44	0.58	47 388
Domain & provenance	0.95	0.28	0.42	32 218
Mean	0.88	0.39	0.53	37 172

Schema retrieval quality To assess how reliably `Explore_Ontology` exposes the relevant schema scaffold for typical researcher questions, we evaluated the system on 11 natural-language queries spanning four broad areas of the MathModDB schema: model formulations and equations, model transformations, computational tasks and problem types, and domain or provenance information.

For each query we measure recall at 10 (R@10), precision at 10 (P@10), F1 and token usage per topical group and display the results in Table 1. R@10 is defined as the fraction of hand-curated expected schema elements appearing in the agent’s output and P@10 as the fraction of returned elements that are in the reference set. The system retrieves $k=6$ candidates per collection per `Explore_Ontology` call by default, however R@10 is reported to provide a more informative assessment of retrieval quality, as $k=6$ would yield a sample size too small to meaningfully evaluate the vector index. Our approach achieves a mean R@10 of 0.88 across all topical groups. Recall is strongest for domain and provenance queries (0.95) and weakest for formulations and equations (0.79). The mean precision of 0.39 reflects a design choice: since all retrieved schema elements are passed as Steiner terminals to the join planner, the retrieval stage is optimized for recall rather than precision. False-positive retrievals increase the terminal set but do not invalidate the join blueprint, as the Steiner optimization acts as a structural filter downstream, making over-prediction an acceptable trade-off for high recall. The associated token cost is a known consequence of this design; as this work focuses on demonstrating the feasibility of the approach, prompt and retrieval optimization remain future work. Results are stable across repeated runs, with mean P, R, and F1 varying by less than 0.03 across six independent runs (P: 0.39-0.40, R: 0.85-0.88, F1: 0.52-0.53).

Case 1: Bridging Dataverse and MathModDB To evaluate the interoperability between data repositories and MathModDB through the combination of multiple MCP servers, we consider an experimental dataset⁵ deposited in a Dataverse data repository, which provides concentration trajectories from the enzymatic synthesis of the β -lactam antibiotic cephalixin, catalyzed by an α -amino ester hydrolase (AEH from *X. campestris*) acting on the substrates PGME and 7-ADCA [37].

The agent first queried the Dataverse installation via its MCP interface to retrieve the dataset's metadata and files, identifying the four reactant species (PGME, 7-ADCA, CEX, PG) and the catalyzing enzyme. It then accessed MathModDB through its MCP interface, using the ontology-exploration endpoint to obtain the relevant schema scaffolds, specifically the classes for dynamic models, ODEs, and chemical constants, along with their associated direct and qualifier properties. Guided by these scaffolds, the agent issued targeted SPARQL queries against the knowledge graph, retrieving instance-level entries that correspond to the dataset's kinetic structure and resolving their formal definitions and source references.

The combined output from both servers enabled identification of the bi-bi ping-pong mechanism and adjacent variants, which matches the underlying system kinetics. This process demonstrates end-to-end interoperability between the research-data layer and the mathematical-model knowledge layer.

Case 2: From complex queries to simple prompts The Stokes Darcy model in MathModDB is a coupled mathematical model describing fluid flow across two distinct physical domains [38]. Typical applications include groundwater-surface water interaction, biomedical flow problems, and industrial filtration.

Retrieving information about the composite Stokes Darcy model from MathModDB via SPARQL requires detailed knowledge of its schema: the relations linking models to their equations, the properties that identify equations as coupling conditions, and the specific identifiers used throughout the knowledge base. A sample SPARQL query for the coupling conditions of the Stokes Darcy model is available in the Supplementary Materials section. When biographical or historical information is requested – such as the creators of a model – federated queries across knowledge bases like FactGrid or Wikidata become necessary, since this information is not stored in MathModDB. Queries over multiple knowledge bases demand familiarity with SPARQL federation syntax, endpoint URLs, and the key structures of external databases. Our approach of coupling an LLM with a curated knowledge base reduces the user-facing complexity to a natural language question while preserving epistemic rigor. For the coupling condition case, the prompt *What are the coupling conditions for the Stokes Darcy model in MathModDB?* suffices. For the biographical case, *Based on MathModDB, who are the creators of the Stokes Darcy model, and where were they born and where did they die?* yields the same result as the federated query. Internally, the agent first accesses MathModDB via the MCP interface and uses the ontology-exploration endpoint to identify the relevant schema scaffolds, namely the classes for mathematical models, equations, and coupling conditions together with their associated properties. Using these scaffolds as a guide, the agent issues targeted SPARQL queries against the knowledge graph, and for biographical attributes follows external links to Wikidata already present in MathModDB to retrieve additional information without explicit federation.

6. Discussion

Our work presents an MCP server for MathModDB that combines a vector-indexed schema retriever, a Steiner-tree-based join planner, and guarded SPARQL execution. The schema-first interface provides LLM agents with ontology-grounded query paths.

Steiner-tree join planning has been explored in the broader KGQA literature, but our approach integrates it with Wikibase-specific schema extraction – including qualifier properties and typed entity sets – dual-encoder schema retrieval with ColBERT re-ranking, and exposure through a stateless MCP interface to LLM agents. Existing SPARQL-MCP approaches such as Dobriy et al. [24] operate on generic

⁵<https://doi.org/10.18419/DARUS-5539>

SPARQL endpoints with VoID-based schema exploration and do not address the Wikibase-specific data model or qualifier-mediated triple patterns. The Wikidata MCP server provides broad general-purpose access to a collaboratively edited open knowledge base but lacks a query-conditioned schema planner. Our system adds precisely this layer, targeting a domain-specific, expert-curated resource where schema-valid query paths are essential for scientific reproducibility.

As shown in *Case 1*, the MCP interface enables cross-repository interoperability, linking research data from Dataverse to mathematical models in MathModDB via natural language. As demonstrated in *Case 2*, the same principle reduces the effort required to access mathematical knowledge within MathModDB. The LLM does not replace the structured knowledge base but uses its schema and data as context for generating targeted SPARQL queries. This allows users to access information through natural-language prompts without having to formulate the underlying queries themselves.

7. Conclusion and Future Work

The presented system combines the MathModDB knowledge graph with an LLM through a schema-first MCP interface. The quantitative evaluation shows schema-context retrieval with a mean R@10 of 0.88 across four topical groups. The two case studies demonstrate end-to-end interoperability between heterogeneous data sources – linking an enzyme-kinetics dataset from Dataverse to its mathematical model in MathModDB (*Case 1*), and simplifying structurally complex federated queries over the Stokes-Darcy model to plain natural-language prompts (*Case 2*). By grounding LLM outputs in epistemically curated knowledge, the system facilitates steps toward XAIR-compliant research outputs: the association of datasets with formal model descriptions, governing equations, and coupling conditions.

The architecture generalizes to any Wikibase-based system exposing a SPARQL endpoint and an indexable schema, making it a reusable integration pattern for the broader Linked Open Data ecosystem, including Wikidata itself. What our solution currently lacks is the reverse process: the extraction of mathematical knowledge from scientific articles and the ingest of extracted model data into MathModDB, which we have identified as the natural next step. Current limitations include the small evaluation set of 11 queries, the absence of a baseline and ablation study, and the lack of a systematic end-to-end reliability analysis. Prompt and retrieval optimization and a controlled usability comparison with the MaRDI portal remain future work.

Acknowledgments

This work was funded by the MaRDI project [6] under Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) grant number 460135501, and funded by DFG under Germany’s Excellence Strategy – EXC 2075 – 390740016. We acknowledge the support by the Stuttgart Center for Simulation Science (SimTech). We also thank the reviewers for their thorough and constructive feedback, which has substantially improved the work.

Supplemental Material Statement

Source Code

- MathModDB MCP Server
 - github repository: <https://github.com/MaRDI4NFDI/MathModDB-MCP>
 - Version release used in this paper: <https://doi.org/10.5281/zenodo.22207787>
- Schema-retrieval evaluation pipeline: <https://github.com/JR-1991/MathModDB-ISWC-Evaluation>

LLM Chats/Prompts

- Prompt: What are the coupling conditions for the Stokes Darcy model in MathModDB?
 - Claude Chat share: <https://claude.ai/share/71419ffd-fed2-4b51-b36e-b3851cd588bd>
 - Stable deposited version: <https://doi.org/10.5281/zenodo.22208236><https://claude.ai/share/71419ffd-fed2-4b51-b36e-b3851cd588bd>
- Prompt: Based on MathModDB, who are the creators of the Stokes Darcy model, and where were they born and where did they die?
 - Claude Chat share: <https://claude.ai/share/dcdc556e-c97b-41dd-a58f-5d44c0643fd9>
 - Stable deposited version: <https://doi.org/10.5281/zenodo.22129551>
- Prompt: Please inspect the dataset 10.18419/DARUS-5539 and gather information on the modelled reaction. Then find a fitting mathematical model for the present reaction system.
 - Claude chat share: <https://claude.ai/share/fe79b9d6-12b4-45d0-a55d-4e56ec1ea704>
 - Stable deposited version: <https://doi.org/10.5281/zenodo.22670774>

SPARQL Queries

- SPARQL query to determine the coupling conditions of the Stokes Darcy model: <https://gist.github.com/bjschembera/b555e3c1af5aece36c297e01423725dd>
- Federated SPARQL query to find biographical information about the creators of the Stokes Darcy model: <https://gist.github.com/bjschembera/1700f34ed6e59d89e519f06ab5384c5b>

Declaration of use of Generative AI

In the preparation of this work, the following generative AI tools were used:

- **Claude Opus 4.7 & Sonnet 4.6** has been used for language polishing, stylistic refinement, and feedback on the structure and organization of paragraphs. It has also been used for the creating figures 1 and 2.
- **Grammarly** has been used for proofreading, grammar and spelling checks, and to obtain feedback on reader reactions and clarity.

All AI-assisted outputs were reviewed and edited by the authors, who take full responsibility for the content of the work.

References

- [1] T. O. Conrad, E. Ferrer, D. Mietchen, L. Pusch, J. Stegmüller, M. Schubotz, Making mathematical research data FAIR: pathways to improved data sharing, *Scientific Data* 11 (2024) 676. URL: <https://doi.org/10.1038/s41597-024-03480-0>. doi:10.1038/s41597-024-03480-0.
- [2] K. Hulek, F. Müller, M. Schubotz, O. Teschke, Mathematical research data—an analysis through zbMATH references, *European Mathematical Society Magazine* (2019) 54–57. URL: <https://doi.org/10.4171/news/113/14>. doi:10.4171/news/113/14.
- [3] M. Schubotz, E. Ferrer, J. Stegmüller, D. Mietchen, O. Teschke, L. Pusch, T. O. Conrad, Bravo MaRDI: a wikibase powered knowledge graph on mathematics, in: *Proc. ISWC 2023*, 2023. URL: https://wikidataworkshop.github.io/2023/papers/15__novel_bravo_mardi_a_wikibase_%5B1%5D.pdf.
- [4] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, et al., The FAIR Guiding Principles for scientific data management and stewardship, *Scientific data* 3 (2016) 1–9. URL: <http://doi.org/10.1038/sdata.2016.18>.
- [5] B. Schembera, F. Wübbeling, H. Kleikamp, B. Schmidt, A. Shehu, M. Reidelbach, C. Biedinger, J. Fiedler, T. Koprucki, D. Iglezakis, D. Göddeke, Towards a Knowledge Graph for Models and Algorithms in Applied Mathematics, in: M. Sfakakis, E. Garoufallou, M. Damigos, A. Salaba, C. Papatheodorou (Eds.), *Metadata and Semantic Research*, Springer Nature Switzerland, Cham, 2025, pp. 95–109. doi:10.1007/978-3-031-81974-2_8.
- [6] The MaRDI consortium, MaRDI: Mathematical Research Data Initiative Proposal, 2022. doi:10.5281/zenodo.6552436.
- [7] D. Vrandečić, M. Krötzsch, Wikidata: a free collaborative knowledgebase, *Communications of the ACM* 57 (2014) 78–85. URL: <https://dx.doi.org/10.1145/2629489>. doi:10.1145/2629489.
- [8] T. Berners-Lee, J. Hendler, Publishing on the Semantic Web, *Nature* 410 (2001) 1023–1024. doi:10.1038/35074206.
- [9] M. T. Horsch, S. Chiacchiera, G. Guevara Carrión, M. Kohns, E. A. Müller, D. Šarić, S. Stephan, I. T. Todorov, J. Vrabec, B. Schembera, Epistemic metadata for computational engineering information systems, in: *Formal Ontology in Information Systems: Proceedings of the 13th International Conference (FOIS 2023)*, SAGE Publications 1 Oliver’s Yard, 55 City Road, London, EC1Y 1SP, 2023, pp. 302–317. URL: <https://dx.doi.org/10.3233/FAIA231136>. doi:10.3233/FAIA231136.
- [10] B. Schembera, Data, Information, Knowledge, Wisdom, and Explainable Artificial Intelligence, in: F. Al Machot, M. T. Horsch, S. Scholze (Eds.), *Designing the Conceptual Landscape for a XAIR Validation Infrastructure*, Springer Nature Switzerland, Cham, 2025, pp. 122–132. URL: https://doi.org/10.1007/978-3-031-89274-5_9. doi:10.1007/978-3-031-89274-5_9.
- [11] W. M. Farmer, MKM: a new interdisciplinary field of research, *SIGSAM Bull.* 38 (2004) 47–52. URL: <https://doi.org/10.1145/1041791.1041795>. doi:10.1145/1041791.1041795.
- [12] V. Chelliah, C. Laibe, N. Le Novère, BioModels database: a repository of mathematical models of biological processes, *In Silico Systems Biology* (2013) 189–199. URL: http://doi.org/10.1007/978-1-62703-450-0_10.
- [13] O. Inizan, V. Fromion, A. Goelzer, F. Saïs, D. Symeonidou, An Ontology to structure Biological Data: The Contribution of Mathematical Models, in: *Research Conference on Metadata and Semantics Research*, Springer, 2021, pp. 57–64. URL: http://doi.org/10.1007/978-3-030-98876-0_5.
- [14] P. Suresh, G. Joglekar, S. Hsu, P. Akkisetty, L. Hailemariam, A. Jain, G. Reklaitis, V. Venkatasubramanian, Onto MODEL: Ontological mathematical modeling knowledge management, in: *Computer Aided Chemical Engineering*, volume 25, Elsevier, 2008, pp. 985–990. URL: [http://doi.org/10.1016/S1570-7946\(08\)80170-8](http://doi.org/10.1016/S1570-7946(08)80170-8).
- [15] P. Suresh, S.-H. Hsu, P. Akkisetty, G. V. Reklaitis, V. Venkatasubramanian, OntoMODEL: ontological mathematical modeling knowledge management in pharmaceutical product development, 1: conceptual framework, *Industrial & Engineering Chemistry Research* 49 (2010) 7758–7767. URL: <http://doi.org/10.1021/ie100246w>.
- [16] A. Snytnikov, B. Glinskiy, G. Zagorulko, Y. Zagorulko, Ontological Approach to Formalization of

- Knowledge in Computational PlasmaPhysics, *Journal of Physics: Conference Series* 1640 (2020) 012013. URL: <http://doi.org/10.1088/1742-6596/1640/1/012013>.
- [17] A. Kirillovich, O. Nevzorova, M. Falileeva, E. Lipachev, L. Shakirova, Ontomathedu: A linguistically grounded educational mathematical ontology, in: C. Benz Müller, B. Miller (Eds.), *Intelligent Computer Mathematics*, Springer International Publishing, Cham, 2020, pp. 157–172. URL: https://doi.org/10.1007/978-3-030-53518-6_10. doi:10.1007/978-3-030-53518-6_10.
 - [18] A. Kirillovich, M. Falileeva, O. Nevzorova, E. Lipachev, A. Dyupina, L. Shakirova, Prerequisite relationships of the ontomathedu educational mathematical ontology, in: J. C. Figueroa-García, Y. Díaz-Gutierrez, E. E. Gaona-García, A. D. Orjuela-Cañón (Eds.), *Applied Computer Sciences in Engineering*, Springer International Publishing, Cham, 2021, pp. 517–524. URL: https://doi.org/10.1007/978-3-030-86702-7_44. doi:10.1007/978-3-030-86702-7_44.
 - [19] B. Schembera, F. Wübbeling, H. Kleikamp, C. Biedinger, J. Fiedler, M. Reidelbach, A. Shehu, B. Schmidt, T. Koprucki, D. Iglezakis, D. Göddeke, Ontologies for Models and Algorithms in Applied Mathematics and Related Disciplines, in: E. Garoufallou, F. Sartori (Eds.), *Communications in Computer and Information Science*, Springer Nature Switzerland, Cham, 2024, pp. 161–168. URL: http://doi.org/10.1007/978-3-031-65990-4_14.
 - [20] B. Schembera, F. Wübbeling, A. Shehu, C. Biedinger, J. Fiedler, M. Reidelbach, B. Schmidt, E. Ferrer, T. Koprucki, FAIR Representation of Mathematical Research Data: MathModDB and MathAlgoDB as Knowledge Graphs for Mathematical Models and Numerical Algorithms, 2025. URL: <https://doi.org/10.5281/zenodo.16735911>. doi:10.5281/zenodo.16735911.
 - [21] P. Scharpf, M. Schubotz, B. Gipp, Mathematics in Wikidata, in: *Proc. ISWC 2021*, 2021.
 - [22] X. Hou, Y. Zhao, S. Wang, H. Wang, Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions, *ACM Trans. Softw. Eng. Methodol.* (2026). URL: <https://doi.org/10.1145/3796519>. doi:10.1145/3796519, just Accepted.
 - [23] P. P. Ray, A survey on model context protocol: Architecture, state-of-the-art, challenges and future directions, *Authorea Preprints* (2025). URL: <https://doi.org/10.36227/techrxiv.174495492.22752319/v1>. doi:10.36227/techrxiv.174495492.22752319/v1.
 - [24] D. Dobriy, F. Bauer, A. Azzam, D. Banerjee, A. Polleres, Agentic SPARQL: Evaluating SPARQL-MCP-powered Intelligent Agents on the Federated KGQA Benchmark, *arXiv preprint arXiv:2603.06582* (2026). doi:10.48550/arXiv.2603.06582.
 - [25] A. C. Sima, T. M. de Farias, E. Zbinden, M. Anisimova, M. Gil, H. Stockinger, K. Stockinger, M. Robinson-Rechavi, C. Dessimoz, Bio-SODA: Enabling natural language question answering over knowledge graphs without training data, in: *Proceedings of the 33rd International Conference on Scientific and Statistical Database Management (SSDBM)*, 2021. doi:10.1145/3468791.3469119.
 - [26] V. Emonet, J. Bolleman, S. Duvaud, T. M. de Farias, A. C. Sima, LLM-based SPARQL Query Generation from Natural Language over Federated Knowledge Graphs, 2025. URL: <https://arxiv.org/abs/2410.06062>. arXiv:2410.06062.
 - [27] Y. Zhang, K. Tsuda, SAGA: Schema-aware grounding for agentic text-to-SPARQL generation, *arXiv preprint arXiv:2607.14494* (2026). URL: <https://arxiv.org/abs/2607.14494>.
 - [28] H. Pan, R. Chard, R. Mello, C. Grams, T. He, A. Brace, O. P. Skelly, W. Engler, H. Holbrook, S. Y. Oh, et al., Experiences with Model Context Protocol Servers for Science and High Performance Computing, *arXiv preprint arXiv:2508.18489* (2025). doi:10.48550/arXiv.2508.18489.
 - [29] T. Dasgupta, M. Neogi, P. Mukhopadhyay, Enhancing Dspace with Large Language Models: Designing an Integration Framework Using the Model Context Protocol, in: *Proceedings of the 4th International Symposium on Knowledge Engineering in Digital Library Design (KEDLD-2025)*, 235–245., 2025. URL: <https://doi.org/10.5281/zenodo.17598560>. doi:10.5281/zenodo.17598560.
 - [30] M. Kuehl, D. P. Schaub, F. Carli, L. Heumos, C. Fernández-Zapata, N. Kaiser, J. Schaul, U. Panzer, S. Bonn, S. Lobentanzer, J. Saez-Rodriguez, V. G. Puelles, Community-based biomedical context to unlock agentic systems, *bioRxiv* (2025). URL: <https://www.biorxiv.org/content/early/2025/07/25/2025.07.21.665729>. doi:10.1101/2025.07.21.665729.
 - [31] M. T. Horsch, B. Schembera, H. A. Preisig, European standardization efforts from FAIR toward explainable-AI-ready data documentation in materials modelling, in: *2023 3rd International*

- conference on applied artificial intelligence (ICAPAI), IEEE, 2023, pp. 1–6. doi:10.1109/ICAPAI.58366.2023.10193944.
- [32] S. Liu, S. Semnani, H. Triedman, J. Xu, I. D. Zhao, M. Lam, Spinach: Sparql-based information navigation for challenging real-world questions, in: Findings of the association for computational linguistics: EMNLP 2024, 2024, pp. 15977–16001. URL: <https://doi.org/10.18653/v1/2024.findings-emnlp.938>. doi:10.18653/v1/2024.findings-emnlp.938.
 - [33] H. Abu-Rasheed, C. Weber, M. Fathi, Knowledge graphs as context sources for llm-based explanations of learning recommendations, in: 2024 IEEE Global Engineering Education Conference (EDUCON), IEEE, 2024, pp. 1–5. doi:10.1109/EDUCON60312.2024.10578654.
 - [34] J. Sequeda, D. Allemang, B. Jacob, Knowledge Graphs as a source of trust for LLM-powered enterprise question answering, *Journal of Web Semantics* 85 (2025) 100858. URL: <https://www.sciencedirect.com/science/article/pii/S1570826824000441>. doi:<https://doi.org/10.1016/j.websem.2024.100858>.
 - [35] O. Khattab, M. Zaharia, ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT, *CoRR* abs/2004.12832 (2020). URL: <https://arxiv.org/abs/2004.12832>. arXiv:2004.12832.
 - [36] F. K. Hwang, D. S. Richards, Steiner tree problems, *Networks* 22 (1992) 55–89. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230220105>. doi:<https://doi.org/10.1002/net.3230220105>.
 - [37] C. E. Lagerman, M. A. Grover, R. W. Rousseau, A. S. Bommarius, Kinetic model development for α -amino ester hydrolase (aeh)-catalyzed synthesis of β -lactam antibiotics, *Chemical Engineering Journal* 426 (2021) 131816. doi:10.1016/j.cej.2021.131816.
 - [38] J. Schmalfuß, C. Riethmüller, M. Altenbernd, K. Weishaupt, D. Göddeke, Partitioned Coupling vs. Monolithic Block-Preconditioning Approaches for Solving Stokes-Darcy Systems, in: 9th International Conference on Computational Methods for Coupled Problems in Science and Engineering (COUPLED PROBLEMS 2021), 2021. doi:10.23967/coupled.2021.043.