

universität freiburg

Enhancing Retrieval and LLM Understanding of Structured Data for Financial Table Question Answering

Master Thesis

Presented by

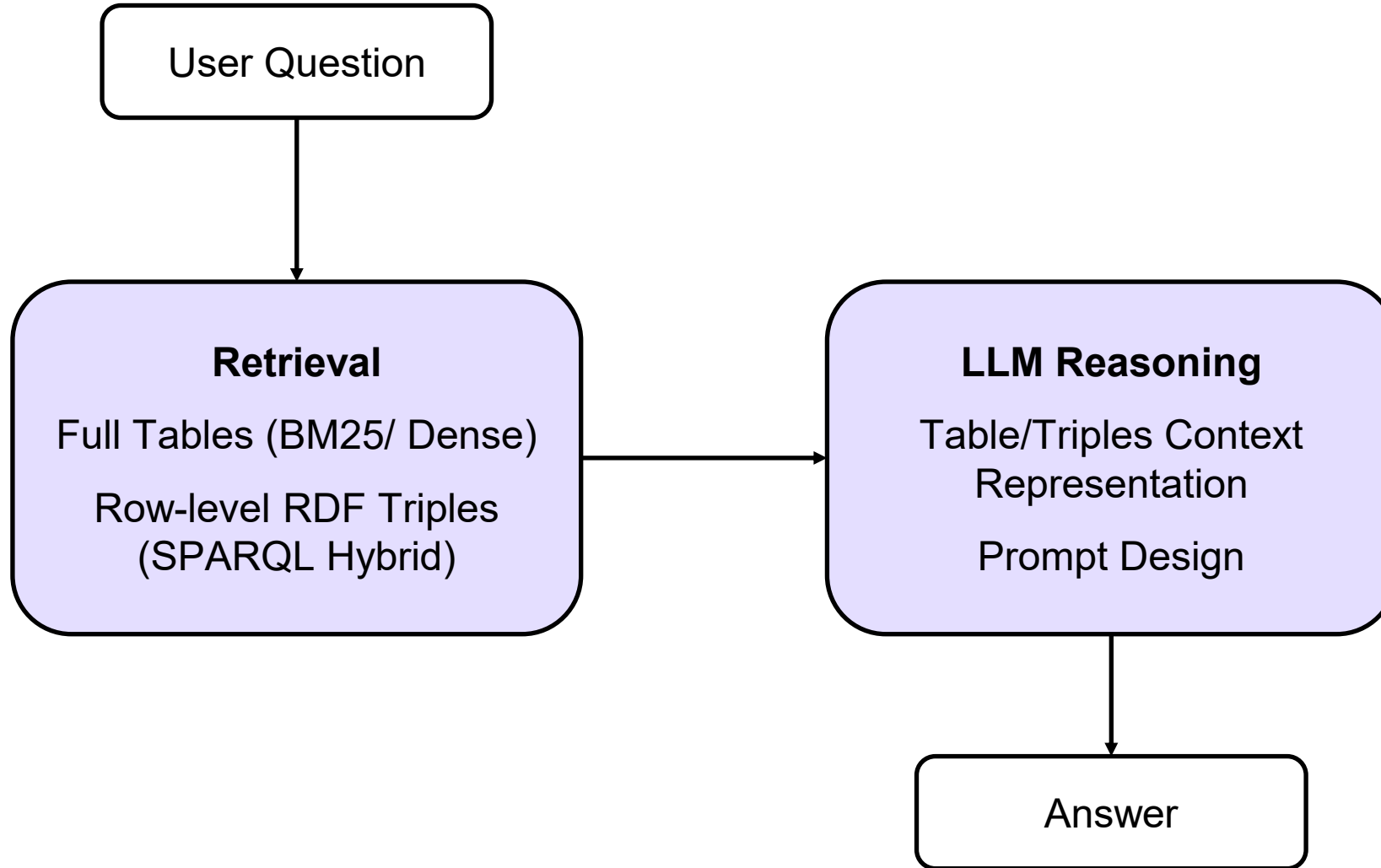
Shanthini Malarvizhi

M.Sc. Computer Science (AI Specialisation)

February 18, 2026



Introduction



Preprocessing of Tables – Example 1 (Normal Table)

Table linearized formats

Year	Pension	Retiree medical and other
2011	574	173
2012	602	170

(a) Normal Table

```
| year | pension | retiree medical and other |  
|---|---|---|  
| 2011 | 574 | 173 |  
| 2012 | 602 | 170 |
```

(b) Markdown Format

```
year: 2011 , pension: 574 , retiree medical and other: 173  
year: 2012 , pension: 602 , retiree medical and other: 170
```

(c) Plaintext Format

```
pension retiree medical and other  
2011 2012
```

(d) Header + First Column Format

```
year,pension,retiree medical and other  
2011,574,173  
2012,602,170
```

(e) CSV Format

Preprocessing of Tables – Example 1 (Normal Table)

Table linearized formats

The table represents annual financial contributions (in millions of dollars) towards pension and retiree medical and other benefits over a span of years from 2011 to 2012. Each column in the table specifies a different type of benefit contribution: the 'pension' column shows the amount contributed towards pension funds, while the 'retiree medical and other' column details the contributions towards retiree medical benefits and other similar expenses. The key values in the first column, which represents the years, include 2011, and 2012. Each row in the table presents data for a specific year or range of years; for example, in 2011, \$574 million was contributed to pensions and \$173 million to retiree medical and other benefits.

(f) Table Schema Format

```
<table>
<thead><tr><th>year</th><th>pension</th><th>retiree medical and other</th></tr></thead>
<tbody>
<tr><td>2011</td><td>574</td><td>173</td></tr>
<tr><td>2012</td><td>602</td><td>170</td></tr>
</tbody>
</table>
```

(g) HTML Format

```
Ungrouped:
2011:
- has pension: 574
- has retiree medical and other: 173
2012:
- has pension: 602
- has retiree medical and other: 170
```

(h) Text Format

Preprocessing of Tables – Example 2 (Hierarchical Table)

Account	2016	2017	2018
Net Income	615	585	595
Profit	671	654	695
Costs	562	678	706

(a) Hierarchical Table

```
| account | 2016 | 2017 | 2018 |  
|---|---|---|---|  
| net Income | 615 | 585 | 595 |  
| profit | 671 | 654 | 695 |  
| costs | 562 | 678 | 706 |
```

(b) Markdown Format

```
account: net Income , 2016: 615 , 2017: 585 , 2018: 595  
account: profit , 2016: 671 , 2017: 654 , 2018: 695  
account: costs , 2016: 562 , 2017: 678 , 2018: 706
```

(c) Plaintext Format

```
account 2016 2017 2018  
net income profit costs
```

(d) Header + First Column Format

```
account,2016,2017,2018  
net Income,615,585,595  
profit,671,654,695  
costs,562,678,706
```

(e) CSV Format

Preprocessing of Tables – Example 2 (Hierarchical Table)

The table represents annual financial figures for a company over a span of years from 2016 to 2018. Each column in the table specifies a different financial account: the net income column shows the company's earnings after expenses and taxes, the profit column details total profit, and the costs column shows total expenses. The key values in the first column are net income, profit and costs. Each row in the table presents data for a specific year; for example, in 2016, the company reported \$615 million in net income, and \$671 million in profit, \$562 million in costs.

(f) Table Schema Format

```
<table>
<thead> <tr><th>account</th><th>2016</th><th>2017</th><th>2018</th></tr></thead>
<tbody>
<tr><td>net Income</td><td>615</td><td>585</td><td>595</td></tr>
<tr><td>profit</td><td>671</td><td>654</td><td>695</td></tr>
<tr><td>costs</td><td>562</td><td>678</td><td>706</td> </tr>
</tbody>
</table>
```

(g) HTML Format

```
Ungrouped:
Net Income:
- in 2016: 615
- in 2017: 585
- in 2018: 595
```

```
Group: Net Income
Profit:
- in 2016: 671
- in 2017: 654
- in 2018: 695
Costs:
- in 2016: 562
- in 2017: 678
- in 2018: 706
```

(h) Text Format

Preprocessing of Tables - Tables to RDF Triples

- RDF – (Subject, Predicate, Object)
- First column values → Subjects
- Second and subsequent columns → Predicates

- Column names → hasColumnName
- Symbolic notations (e.g., Δ) → hasDelta
- Dates / Years / Quarters → in{ColumnName}
- No table headers → hasValue{columnNumber}

Example Predicates

```
,mainline operations,wholly-owned regional carriers,total  
pilots and flight crew training instructors,13400,3400,16800  
flight attendants,24700,2200,26900  
maintenance personnel,14900,2000,16900
```

hasTotal
hasMainlineOperations
hasWhollyOwnedRegionalCarriers

```
€ millions,2024,2023, $\Delta$   
"Cash and cash equivalents","9,609","8,124","1,485"  
"Current time deposits and debt securities","1,471","3,151","-1,680"  
"Group liquidity","11,080","11,275","-195"
```

2024 - in2024

2023 - in2023

Δ - hasDelta

Preprocessing of Tables - Tables to RDF Triples

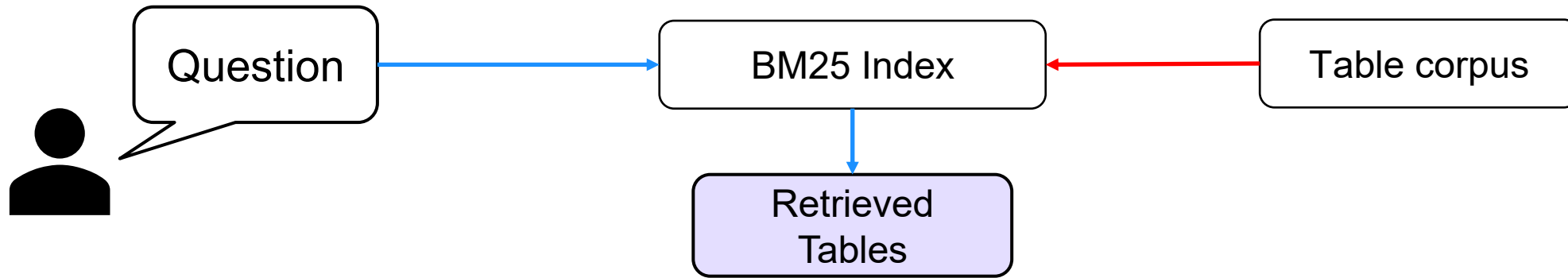
- Numerical cell values:
 - Represented with explicit units and scale
 - Incorporated QUDT ontology (qudt: and unit:)
 - Standardized scales:
 - Millions → M
 - Billions → B
 - Thousands → T
- Hierarchical tables: Relationships preserved using ex:group predicate

```
€ millions,2024,2023
Assets,,
Intangible assets,659,"1,111"
"Property, plant, and equipment","1,562","1,451"
Financial assets,"36,114","34,323"
Fixed assets,"38,334","36,885"
Inventories,0,1
Accounts receivable and other assets,"5,914","5,712"
```

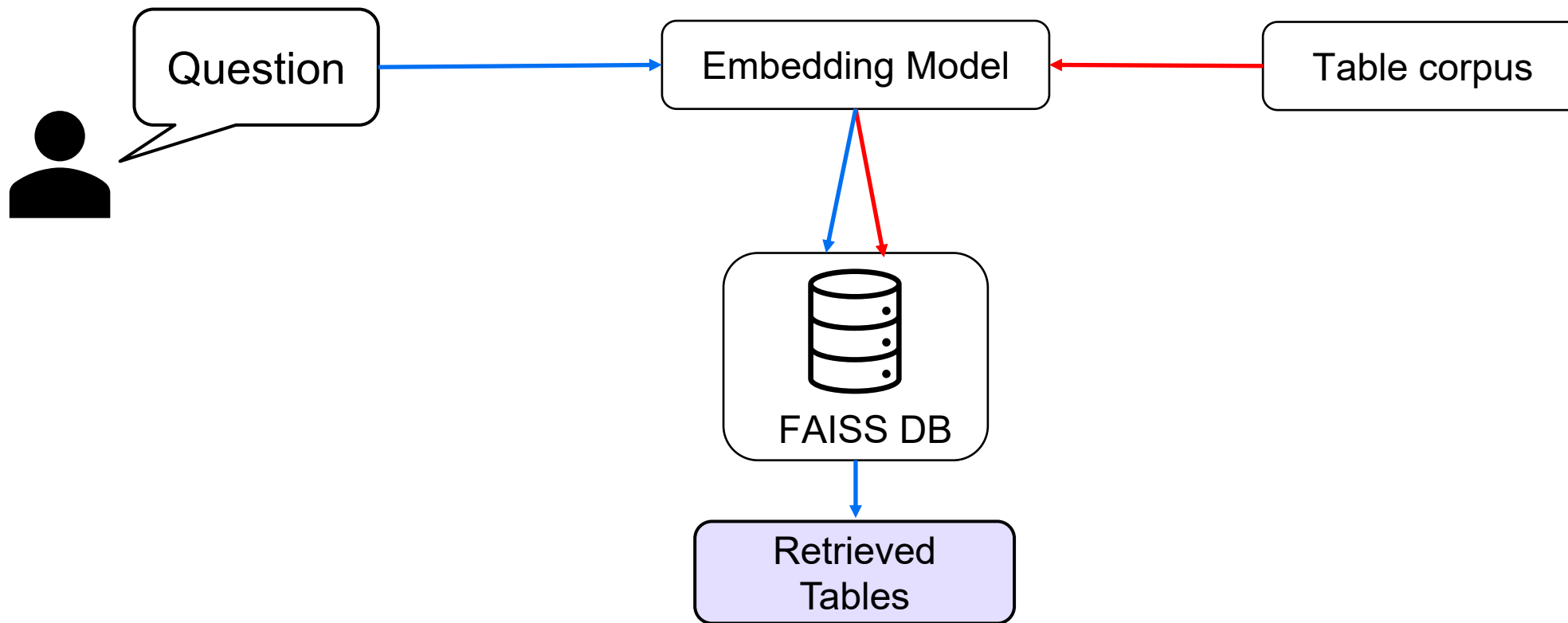
```
@prefix data: <http://example.org/data#> .
@prefix ex: <http://example.org/ontology#> .
@prefix qudt: <http://qudt.org/schema/qudt/> .
@prefix unit: <http://qudt.org/vocab/unit/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

data:accounts_receivable_and_other_assets ex:group data:assets ;
  ex:in2023 [ a qudt:QuantityValue ;
    ex:note "POSITIVE" ;
    ex:signed_str "+5712.00" ;
    ex:unit_scale "M" ;
    qudt:numericValue 5712.0 ;
    qudt:unit unit:EUR ] ;
  ex:in2024 [ a qudt:QuantityValue ;
    ex:note "POSITIVE" ;
    ex:signed_str "+5914.00" ;
    ex:unit_scale "M" ;
    qudt:numericValue 5914.0 ;
    qudt:unit unit:EUR ] ;
  ex:label "Accounts receivable and other assets" .
```

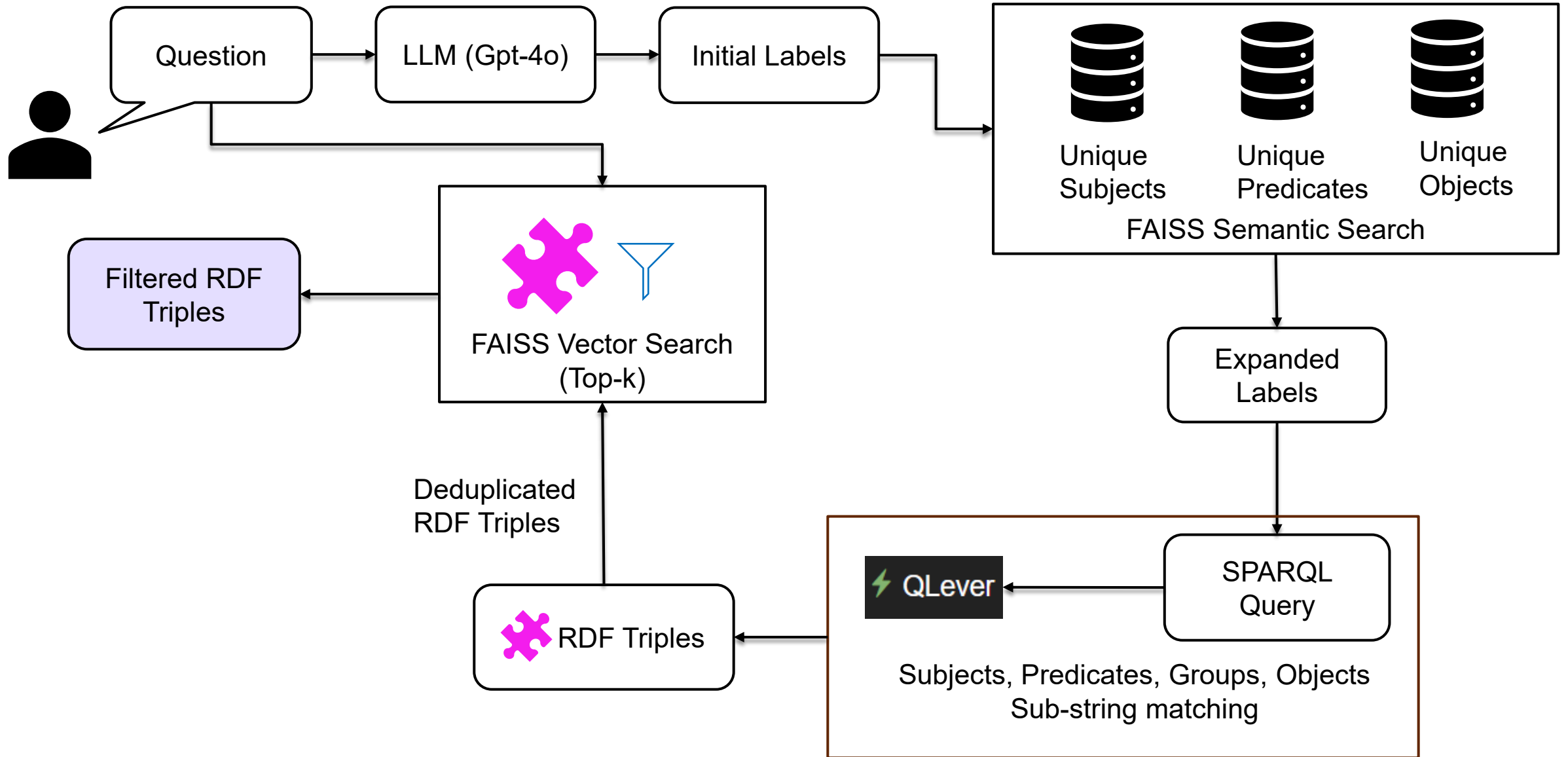
Approach – Table Retrieval (BM25 Retrieval)



Approach – Table Retrieval (Dense Retrieval)



Approach – SPARQL-based Hybrid Retrieval



Approach – LLM Answer Generation

Zero-shot prompting

Table Representation in Context

- Markdown
- HTML
- Plaintext

Table ID: 13

Product Name	Units Sold	Revenue (\$)
Product A	120	2,400
Product B	95	1,900
Product C	150	3,750

Table ID: 2

Service Name	Customer Rating	Revenue (\$)
Premium Support	4.8	5,200
Cloud Backup	4.5	3,750
Data Analytics	4.9	6,100

...

Approach – LLM Answer Generation

RDF Triples Representation in Context

- Subject Predicate Object
- Sequential RDF Triples
- Triples grouped based on tables
- Triples grouped based on tables and subject

Triples grouped based on tables Example

Table ID: 13
Product A hasUnitsSold 120
ProductC hasUnitsSold 150
ProductA hasRevenuedollars 2400 USD

Table ID: 2
Premium Support hasCustomerRating 4.8
Premium Support hasRevenuedollars 5,200 USD

Sequential RDF Triples Example

ProductA hasUnitsSold 120
ProductC hasUnitsSold 150
Premium Support hasCustomerRating 4.8
Premium Support hasRevenuedollars 5200 USD
ProductA hasRevenuedollars 2400 USD

Triples grouped based on tables and Subjects Example

Table ID: 13
Product A hasUnitsSold 120
ProductA hasRevenuedollars 2400 USD
ProductC hasUnitsSold 150

Table ID: 2
Premium Support hasCustomerRating 4.8
Premium Support hasRevenuedollars 5,200 USD

Approach – LLM Answer Generation

Prompt Design Variations

- Context-before-instruction design
- Instruction-before-context design
- Explicit separator-based prompts
- Few-shot examples

Instruction
Question:
{question}
Tables:
{context}

Question:
{question}
Tables:
{context}
Instruction

Simple Prompt

Instruction
==Context Start==
{context}
==Context End==
==Question Start==
{question}
==Question End==

==Context Start==
{context}
==Context End==
==Question Start==
{question}
==Question End==
Instruction

Explicit separator-based prompts

Instruction
Examples
Question:
{question}
Tables:
{context}

Few-shot Examples prompt

Evaluation – Datasets and Metrics

Dataset	#Tables	Hierarchical Tables	Train	Dev	Test	Domain
Enterprise (Benchmark 1)	397	120	1,187	–	1,135	Finance
Enterprise (Benchmark 2)	397	120	96	–	60	Finance
FinQA	2,515	4	3,983	566	729	Finance
WikiTableQuestions	1,116	6	2,422	2,086	1,812	Open-domain

Metrics: Average Recall, Weighted Accuracy

Evaluation – BM25 Retrieval

Benchmark	Format	Average Recall
Enterprise Dataset Benchmark 1	Header first column	0.901
Enterprise Dataset Benchmark 2	Table schema	0.771
FinQA Benchmark	Header first column	0.685
WikiTableQuestions Benchmark	Table schema	0.568

Evaluation – Dense Retrieval

Benchmark	Format	Average Recall	Model
Enterprise Dataset Benchmark 1	plaintext	0.974	NovaSearch/stella_en_1.5B_v5
Enterprise Dataset Benchmark 2	plaintext	0.974	NovaSearch/stella_en_1.5B_v5
FinQA Benchmark	markdown	0.865	Qwen/Qwen3-Embedding-0.6B
WikiTableQuestions Benchmark	plaintext	0.909	text-embedding-3-large

Evaluation – SPARQL-based Hybrid Retrieval

Benchmark	Questions Sampled	Top-K Triples Retrieved	Average Recall
Enterprise Dataset Benchmark 1	200	3,000	0.944
Enterprise Dataset Benchmark 2	96	3,000	0.933
FinQA Benchmark	200	1,000	0.873
WikiTableQuestions Benchmark	200	11,000	0.818

Evaluation – Dense Vs SPARQL-based Hybrid Retrieval

	Enterprise Dataset Benchmark 1		Enterprise Dataset Benchmark 2		FinQA Benchmark		WikiTableQuestions Benchmark	
	Dense Retriever	SPARQL	Dense Retriever	SPARQL	Dense Retriever	SPARQL	Dense Retriever	SPARQL
Average Recall	0.965	0.944	0.968	0.933	0.535	0.873	0.965	0.818
Queries Failed	7	5	3	0	93	32	7	11

Evaluation - Answer Generation (BM25 & Dense Retrieval)

Benchmark	Zero-Shot	RAG (BM25)	RAG (Dense Retrieval)	Best Prompt Design
Enterprise Dataset Benchmark 1	5.91%	68.30%	74.03%	Context-before-instruction
Enterprise Dataset Benchmark 2	4.17%	39.47%	40.0%	Context-before-instruction
FinQA Benchmark	4.5%	40.52%	46.06%	Explicit separator-based
WikiTableQuestions Benchmark	7.33%	19.45%	38.34%	Explicit separator-based + Context-before-instruction

Evaluation - Answer Generation (SPARQL)

Benchmark	Best Approach	Weighted Accuracy	Best Prompt Design
Enterprise Dataset Benchmark 1	Sequential RDF Triples	18.75%	Few-shot examples + Context-before-instruction
Enterprise Dataset Benchmark 2	Triples grouped based on tables	48.50%	Explicit separator-based + Context-before-instruction
FinQA Benchmark	Triples grouped based on tables	23.50%	Explicit separator-based
WikiTableQuestions Benchmark	Triples grouped based on tables And subject	14.25%	Explicit separator-based + Context-before-instruction

Evaluation Against Existing Method - TableRAG

- Focus on table retrieval for QA
- Use dense embeddings + FAISS (markdown formatted-tables)
- Retrieve Top-30 tables
- Evaluate on WikiTableQuestions
- Handle both:
 - Single-table
 - Multi-table scenarios

Our System Vs TableRAG – Retrieval Results

Benchmark	TableRAG	Our Dense Retriever	Our Hybrid Approach
Enterprise Dataset Benchmark 1	86.0	96.0	89.2
Enterprise Dataset Benchmark 2	83.3	94.1	96.5
FinQA	77.0	84.0	80.7
WikiTableQuestions	88.0	99.0	86.1

Our System Vs TableRAG - Results

Benchmark	TableRAG	Our Dense Retriever + LLM	Our Hybrid Approach + LLM
Enterprise Dataset Benchmark 1	30.5	61.0	50.5
Enterprise Dataset Benchmark 2	8.33	52.5	44.92
FinQA	3.0	50.5	21.0
WikiTableQuestions	22.5	59.0	39.0

Conclusion

- **Dense retrieval** → Best with text-like formats.
- **SPARQL-based hybrid retrieval.**
 - Fine-grained row-level reasoning
 - Strong in multi-table scenarios
- **LLM Answer Generation**
 - Placing **context before instructions** improves reasoning quality.
 - Use **Markdown or HTML table formats.**
 - For RDF triples: Explicit separators or few-shot examples further enhance reasoning.

References

1. X. Yu, P. Jian, and C. Chen, “Tablerag: A retrieval augmented generation framework for heterogeneous document reasoning,” CoRR, vol. abs/2506.10380, 2025. [Online]. Available: <https://doi.org/10.48550/arXiv.2506.10380>
2. MTEB Leaderboard - K. C. Enevoldsen, I. Chung, I. Kerboua, M. Kardos, A. Mathur, D. Stap, J. Gala, W. Siblino, D. Krzeminski, G. I. Winata, S. Sturua, S. Utpala, M. Ciancone, M. Schaeffer, D. Misra, S. Dhakal, J. Rystrom, R. Solomatin, Ö. V. Çagatan, A. Kundu, and et al., “MMTEB: massive multilingual text embedding benchmark,” in ICLR. OpenReview.net, 2025. [Online]. Available: <https://openreview.net/forum?id=zl3pfz4VCV>

universität freiburg

Thank you!



Additional Experiments



Preprocessing of Tables

Context Augmented Table Formats

- Financial data – Extracted from Reports
- Text surrounding the tables
 - Heading
 - Subheading
 - Pretext
 - Posttext
- Table – plaintext format

Monthly Sales Report → Heading

Overview → Sub-heading

This report provides a brief summary of product sales performance for the month of January. The table below shows the number of units sold and total revenue generated for each product. → Pretext

Product Name	Units Sold	Revenue (\$)
Product A	120	2,400
Product B	95	1,900
Product C	150	3,750

→ Table

Overall, Product C generated the highest revenue, while Product B had the lowest sales figures for the month. → Posttext

Preprocessing of Tables - Example

Context Augmented Table Formats

Monthly Sales Report | Overview

a) Combined Heading + Subheading

Monthly Sales Report | Overview | Product Name: Product A, Units Sold: 120, Revenue (\$): 2,400\nProduct Name: Product B, Units Sold: 95, Revenue (\$): 1,900\nProduct Name: Product C, Units Sold: 150, Revenue (\$): 3,750

b) Combined Heading + Subheading + Table

Preprocessing of Tables - Example

Context Augmented Table Formats

[Monthly Sales Report | Overview](#) | This report provides a brief summary of product sales performance for the month of January. The table below shows the number of units sold and total revenue generated for each product.

c) Combined Heading + Subheading + Pretext

[Monthly Sales Report | Overview](#) | This report provides a brief summary of product sales performance for the month of January. The table below shows the number of units sold and total revenue generated for each product. | **Product Name: Product A, Units Sold: 120, Revenue (\$): 2,400**
Product Name: Product B, Units Sold: 95, Revenue (\$): 1,900
Product Name: Product C, Units Sold: 150, Revenue (\$): 3,750

d) Combined Heading + Subheading + Pretext + Table

Preprocessing of Tables - Example

Context Augmented Table Formats

Monthly Sales Report | Overview | This report provides a brief summary of product sales performance for the month of January. The table below shows the number of units sold and total revenue generated for each product. | Overall, Product C generated the highest revenue, while Product B recorded the lowest sales for the month.

e) Combined Heading + Subheading + Pretext + Posttext – Combined Context

Monthly Sales Report | Overview | This report provides a brief summary of product sales performance for the month of January. The table below shows the number of units sold and total revenue generated for each product. | Product Name: Product A, Units Sold: 120, Revenue (\$): 2,400\nProduct Name: Product B, Units Sold: 95, Revenue (\$): 1,900\nProduct Name: Product C, Units Sold: 150, Revenue (\$): 3,750 | Overall, Product C generated the highest revenue, while Product B recorded the lowest sales for the month.

f) Combined Heading + Subheading + Pretext + Table + Posttext – Combined Context Table

Evaluation – BM25 and Dense Retrieval

Benchmark	Format	Average Recall (BM25)	Average Recall (Dense)	Embedding Model
Enterprise Dataset Benchmark 1	Combined heading subheading table	0.897	0.960	NovaSearch/stella_en_1.5B_v5
Enterprise Dataset Benchmark 2	Combined heading subheading table	0.677	0.939	NovaSearch/stella_en_1.5B_v5
FinQA Benchmark	Combined context table	0.737	0.682	Alibaba-NLP/gte-base-en-v1.5

Key Findings

- Relevant contextual enrichment improves retrieval.
- Excessive or loosely related context introduces noise.
- **Dense Retrieval** → Best with formats that have tables.
- In **FinQA**, BM25 outperforms dense retrieval on context-augmented tables.
 - Due to strong reliance on exact lexical overlap