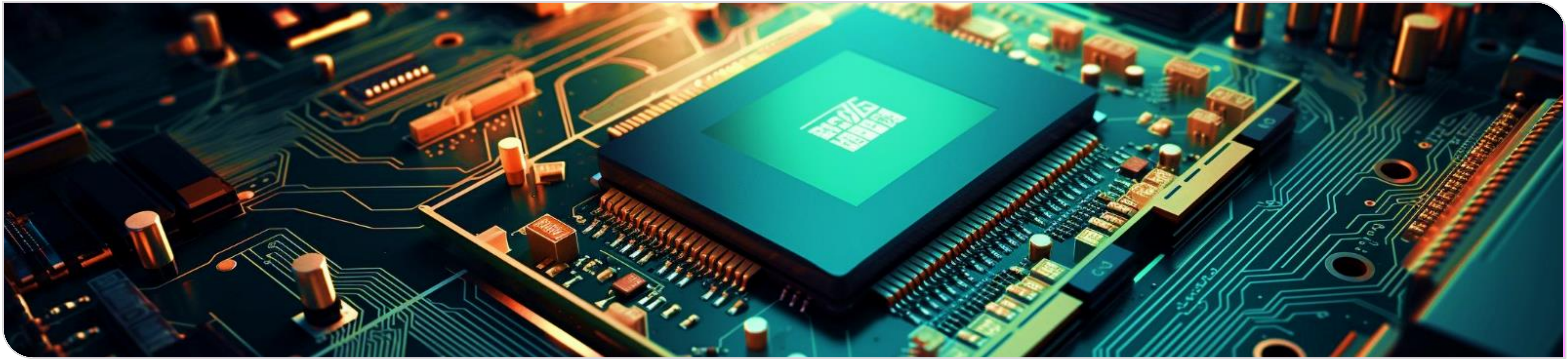
Karlsruhe Institute of Technology

Master Thesis – Overview:

Leveraging Embeddings and Knowledge Graphs for Enhanced Scholarly Information Retrieval: A Comparative Analysis of Retrieval Approaches Using Large Language Models

by Marco Schneider supervised by Angelika Kaplan



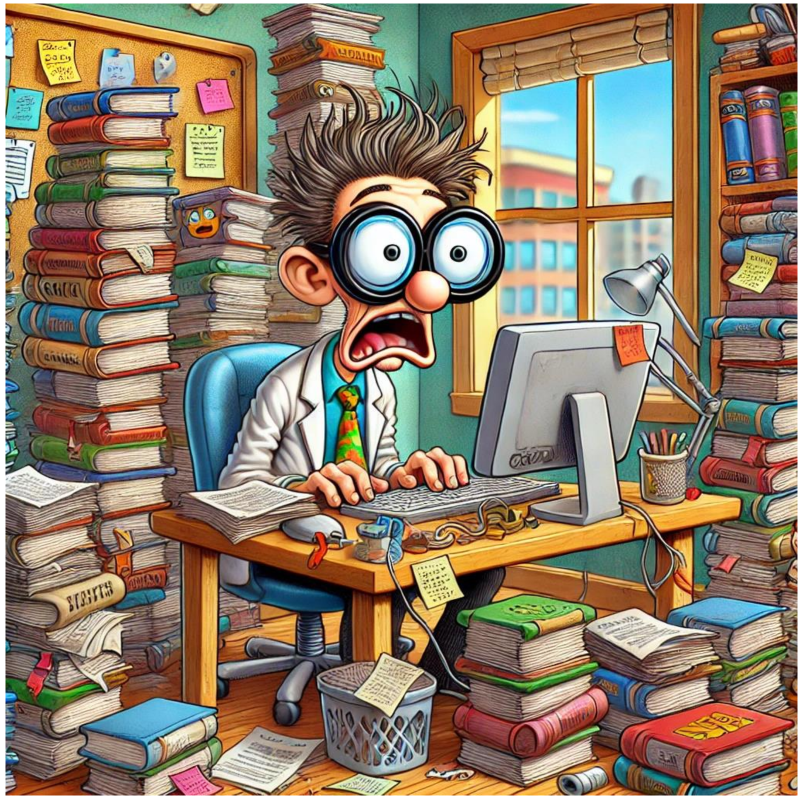
Overview for the Architecture Review

- Thank you for participating in the Code Review
- The following pages introduce you to the topic of the master thesis
- What you are supposed to review is explained in the **Review Manual** that you have been provided.
- These slides are for introduction purposes

Content

Literature Research is hard

Who doesn't know this situation?


OpenAI. (2024). Cartoon-style image of a frustrated academic researcher. Created with DALL-E. Retrieved on July 4, 2024

In modern academic settings, most research results are published in scholarly **digital articles**, presenting difficulties for human and automated processing [Jaradeh19].

Academic search engines return a **set of ranked documents** and users have the tedious task of finding relevant information from it [Thambiand22].

Future generation academic search engines should **focus on understanding meaning** rather than simply matching keywords [Hippel23].

Literature Research could be easy

Large Language Models (LLMs) demonstrate **remarkable abilities** in natural language tasks [Yangetal24]

The application of LLMs to academic search has the potential to **reduce barriers** to accessing information and **speed up** research tasks

Just research by chatting!

OpenAI. (2024). Cartoon-style image of a happy academic researcher. Created with DALL-E. Retrieved on July 4, 2024

LLMs alone are not sufficient

OpenAI. (2024). Cartoon-style image of a hallucinating ChatGPT. Created with DALL-E. Retrieved on July 4, 2024

LLM is trained and its knowledge is **frozen** in time

Especially in knowledge specific tasks LLMs tend to **hallucinate** where they contradict existing sources or lack supporting evidence [Yangetal24]

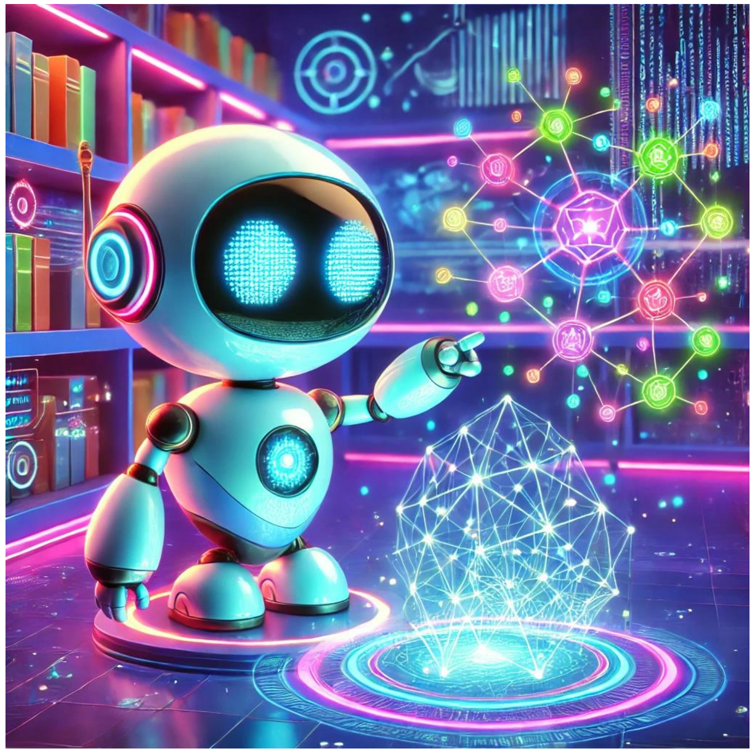
Retrieval Augmented Generation (RAG) allows to **integrate external knowledge** to retrieve current knowledge and reduce hallucinations [Lewis20]

Knowledge Graphs (KG) for Structured Information

The ORKG already offers a way to relate scientific papers in a structured manner in a graph format. [Jaradeh19]

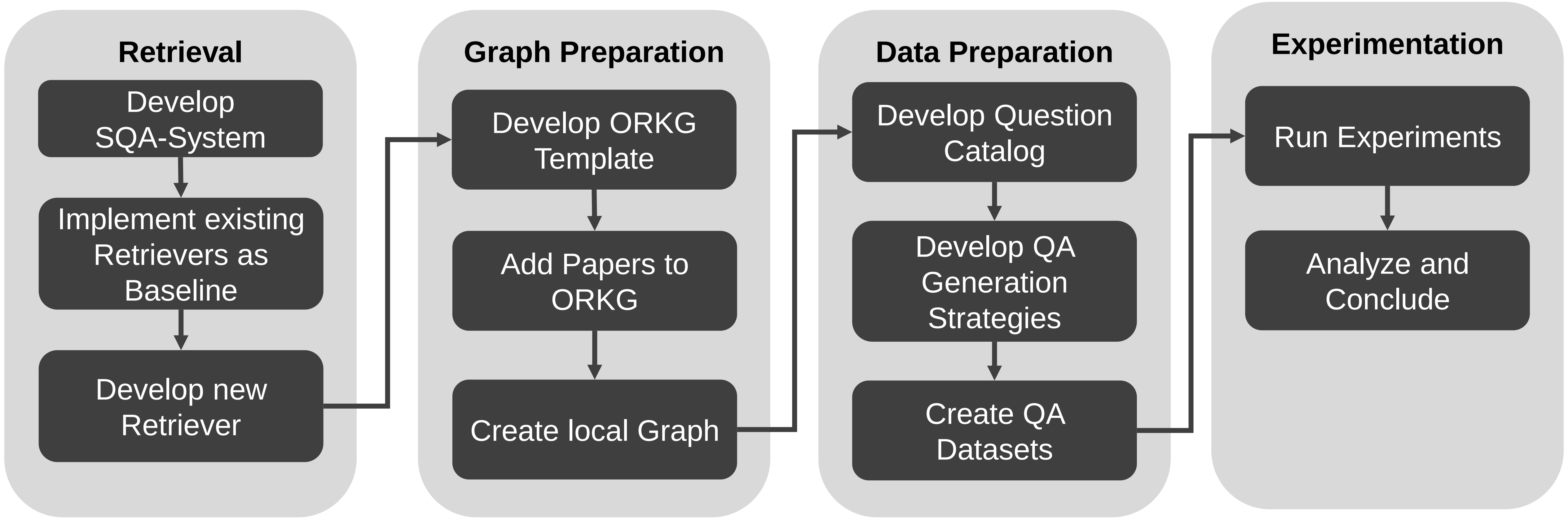
Applying LLM-based Retrieval on such KGs recently sparked major interest in the research community.

It is currently unknown how well a LLM-based Retrieval on the ORKG performs and what drawbacks these might have.



OpenAI. (2024). Cartoon-style image of a bot reading from a graph. Created with DALL-E. Retrieved on November 20, 2024

Overview


Retrieval
Develop SQA-System
Implement existing Retrievers as Baseline
Develop new Retriever
Graph Preparation
Develop ORKG Template
Add Papers to ORKG
Create local Graph
Data Preparation
Develop Question Catalog
Develop QA Generation Strategies
Create QA Datasets
Experimentation
Run Experiments
Analyze and Conclude

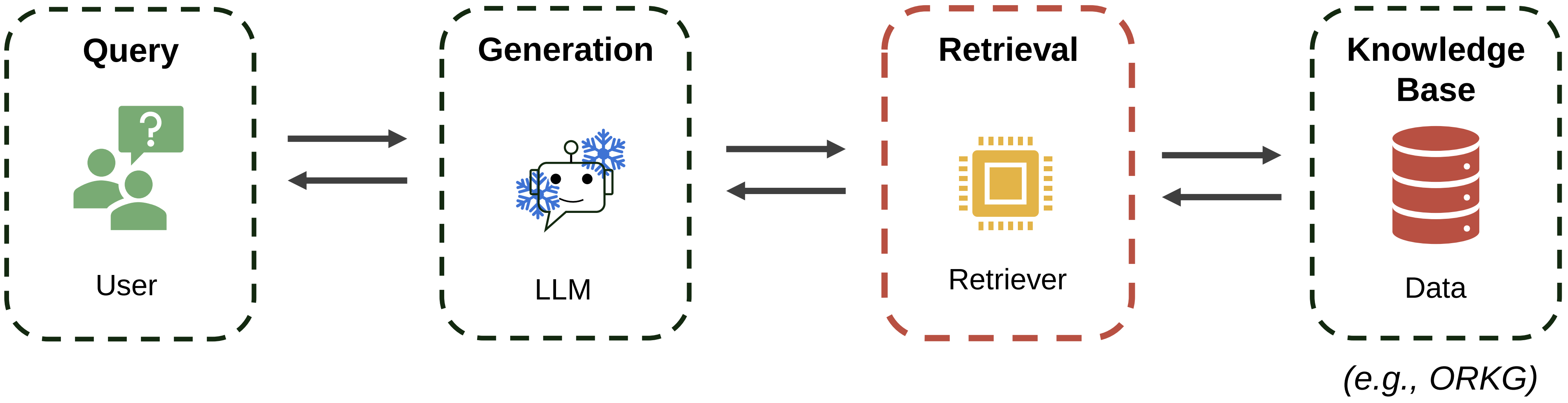
Content

Fundamentals: Retrieval Augmented Generation (RAG)

What is RAG?

- RAG aims to **retrieve relevant data from an external knowledge base** to give the LLM recent and factual data for answer generation

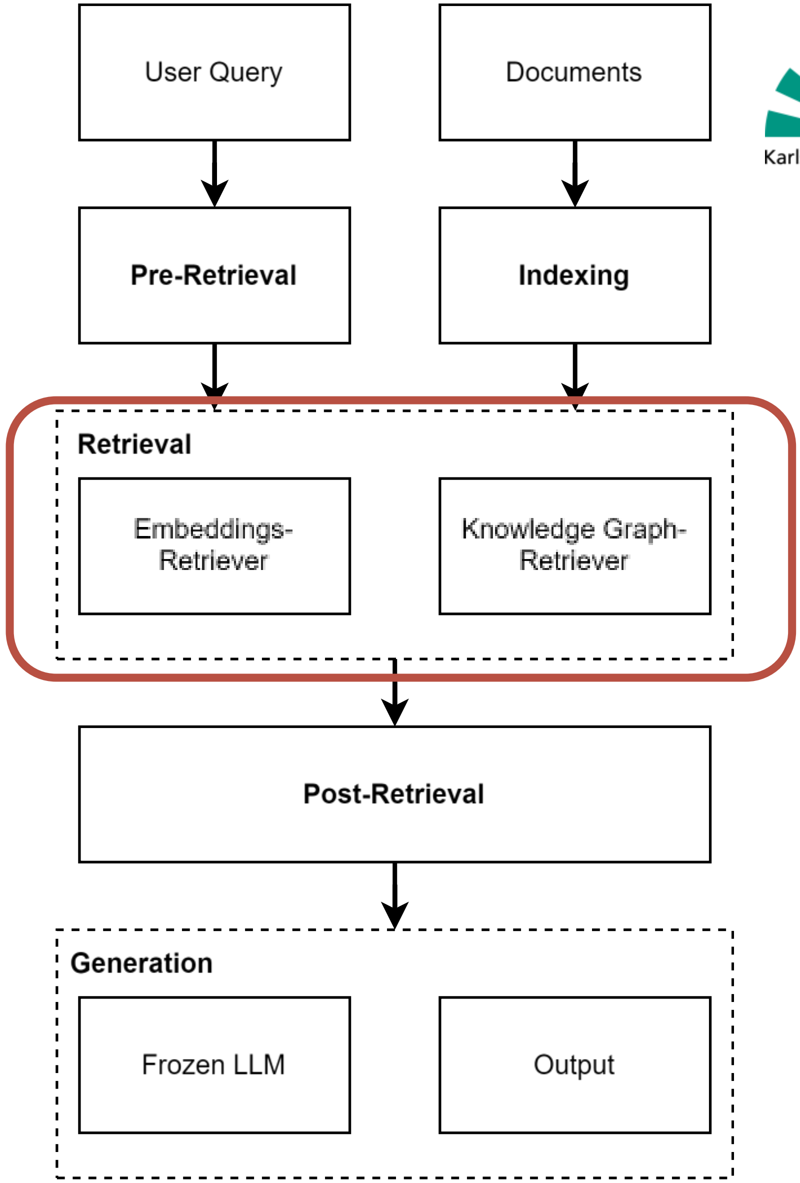
Overview RAG Process



Fundamentals: Deeper look

- The RAG approach basically has 4 main phases

1. It starts with the **indexing** where data is saved in an appropriate format in a knowledge base (KB)
2. **Pre-Retrieval** are processing steps applied to a user query (question) to prepare the question for the retriever
3. **Retrieval** here a retriever object receives the query and retrieves the most relevant documents from the KB. This can be either embedding-based or knowledge graph-based depending on the KB used
4. **Generation** here a language model (like GPT) is tasked to generate an answer based on the documents retrieved

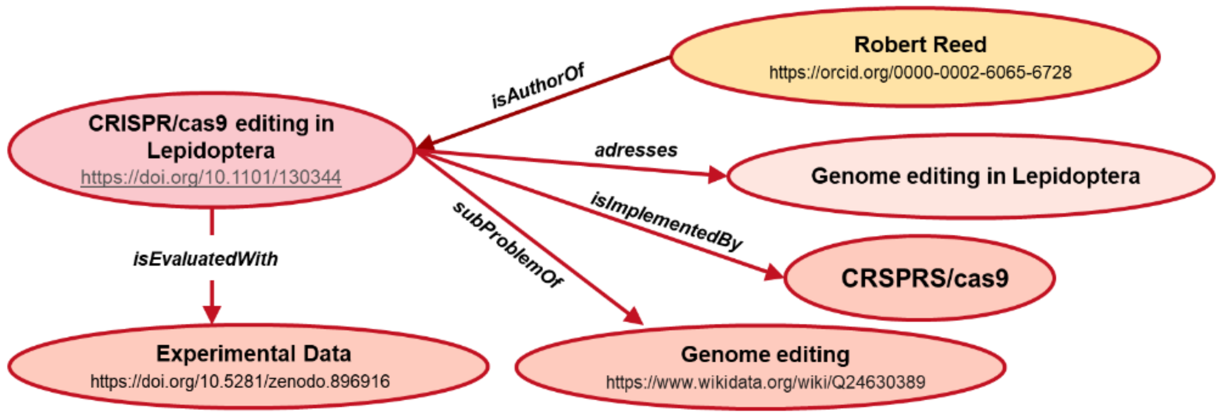


Fundamentals: Open Research Knowledge Graph (ORKG)

- Leverages **knowledge graphs** to represent scholarly literature semantically.
- Papers are added in a community effort by manually adding them based on **templates**

Templates

- Are **predefined structures** that guide users in adding a new paper to the ORKG graph
- Help **standardize** the representation of various types of research contributions
- **Domain specific templates** are needed to ensure that the unique requirements and characteristics of different fields of study are captured.


Robert Reed
https://orcid.org/0000-0002-6065-6728
isAuthorOf
CRISPR/cas9 editing in Lepidoptera
https://doi.org/10.1101/130344
adresses
Genome editing in Lepidoptera
isImplementedBy
CRSPRS/cas9
subProblemOf
isEvaluatedWith
Experimental Data
https://doi.org/10.5281/zenodo.896916
Genome editing
https://www.wikidata.org/wiki/Q24630389

Example graph representation [Jaradeh19]

Content

Related Research Questions

RQ 1

How to improve the QA performance of LLMs on scholarly literature research, by utilizing the structure of scholarly research graphs like the ORKG?

RQ 2

How effective is the proposed approach in the QA setting on a scholarly knowledge graph for addressing desirable question types for software architecture literature research?

- The research questions above are what we will be looking at in this codereview
- We answer **RQ1** with our new retrieval approach called HubLink
- To answer **RQ2** we test this approach in a ORKG use case using competency questions.
  - To enable this evaluation we implemented the SQA-System that allows to run experiments with retrievers on a Knowledge Graph

First of.. Why is the Retrieval so difficult?

- Before we start with the Algorithm, we would like to point out some of the difficulties that make the retrieval hard. You can skip this, but we recommend to read this to understand the issue.

- **What we want:** The user should be able to input a question in natural language. The system then looks through the Knowledge Graph and returns an answer considering all relevant facts from the graph.
- **How the retrieval works without LLMs:**
  - Classically, the search would be keyword based. Here a exact keyword matching is done on the graph to find relevant facts = (Subject, Predicate, Object) = (Node, Edge, Node).
  - However, doing so has limited success with a natural language question. While we could apply Entity Recognition to find and match words from the question sentence to the graph, this proved to be difficult because of ambiguity issues between the word in the question and in the graph. It's also hard to consider context in the question using this approach.
- **Using LLMs** allows to consider the context of the question which has the possibility to increase the meaningfulness and relevance of the retrieved information towards the question

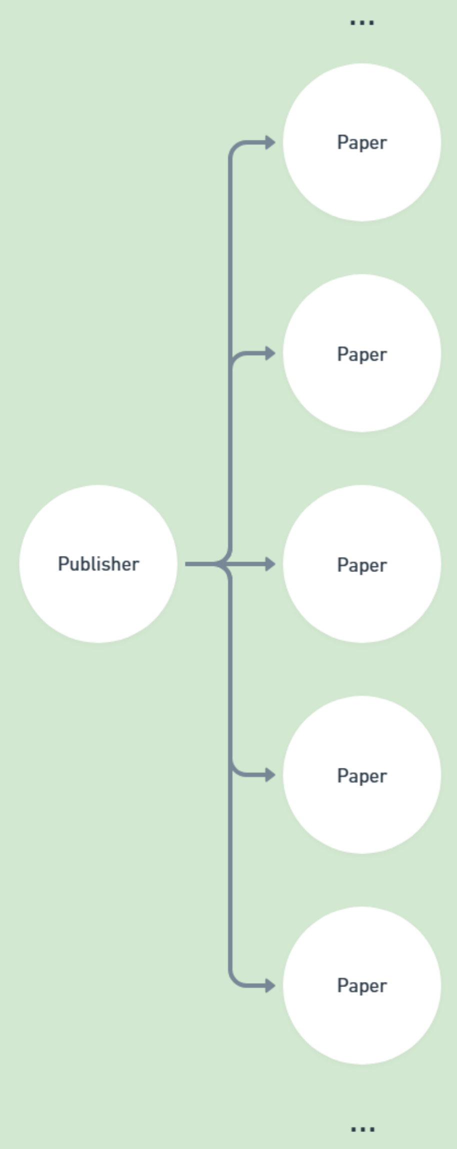
First of.. Why is the Retrieval so difficult?

- **But using the LLM for Retrieval is not trivial:** Obviously it is not feasible to just give the whole graph as context to the LLM as some graph contain millions of entries.. This means that we can only provide a selected amount of information from the graph to the LLM. Therefore, given a Question:
  1. How to select what we give the LLM as context?
  2. Where in the Graph should we even start looking for the context of interest?
  3. How do we determine which triples in the graph are relevant?
  4. LLM calls are costly, so how do we traverse through the graph as efficient as possible?
  5. When should we stop searching to avoid looking though the whole graph?
  6. Most graphs are directed, when do we need to traverse “against the direction”?
  7. What if we need to traverse multiple paths to find the answer?
  8. How do we merge information if multiple triples from the graph are needed?
  9. Information is simplified when stored in a graph because whole sentences or paragraphs are now stored as a (Subject, Predicate, Object) format inheritably losing context. Therefore, how do we process the triple information?

The Width is Difficult

- We want to highlight the following issue: **What if we need to traverse multiple paths to find the answer?** Which is classified as a **Multi-Fact Retrieval**, meaning that it requires multiple triples from the graph to give an answer to a question.
- Current Retrievers struggle with these types of questions. **But why?** Consider the following question: “What papers published by IEEE talk about the concept of prompt engineering on LLMs?”
- To answer this question, every paper with an edge to the publisher needs to be processed to find out if it contains relevant content for the question. As illustrated on the right graph this question quickly becomes very difficult as the **WIDTH** of the publisher can be very large (possible more than thousands of edges)
- We highlight this issue, because during development of our own retriever we concentrated on solving this problem

Sample Graph

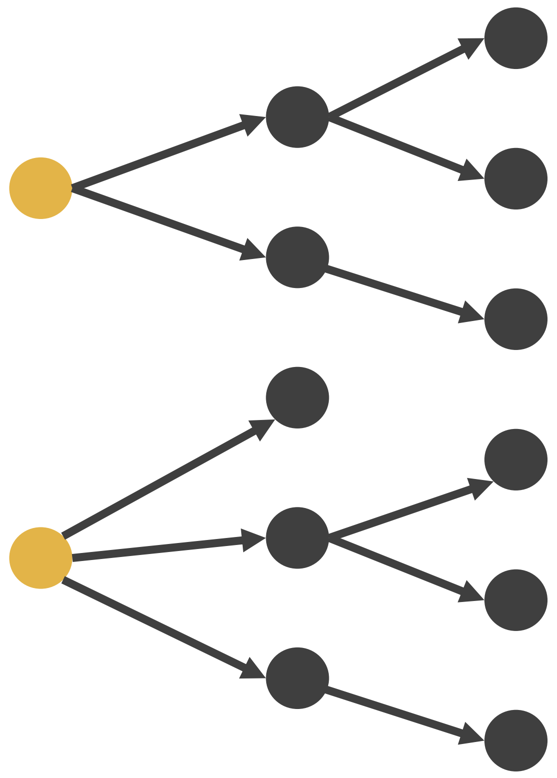


HubLink Retrieval Approach

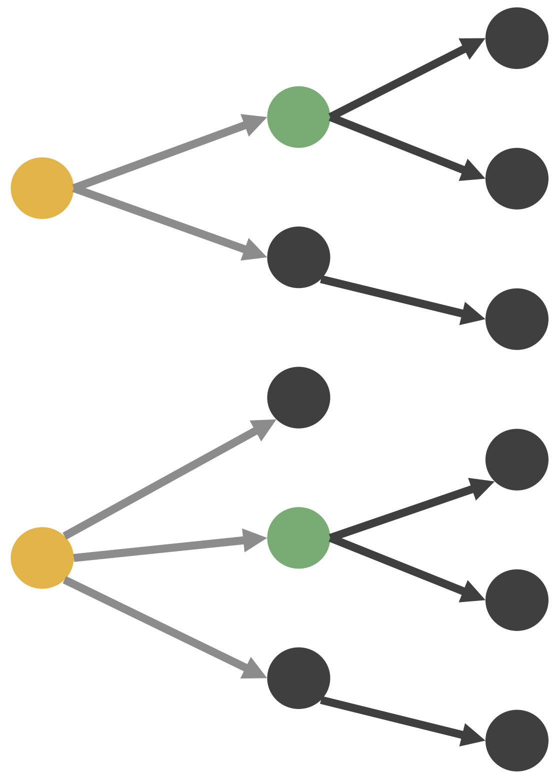
“Reasoning on subgraphs without additional training costs.”

HubLink: Indexing

1. **Given:** A list of Start Points ● from which the indexing is started

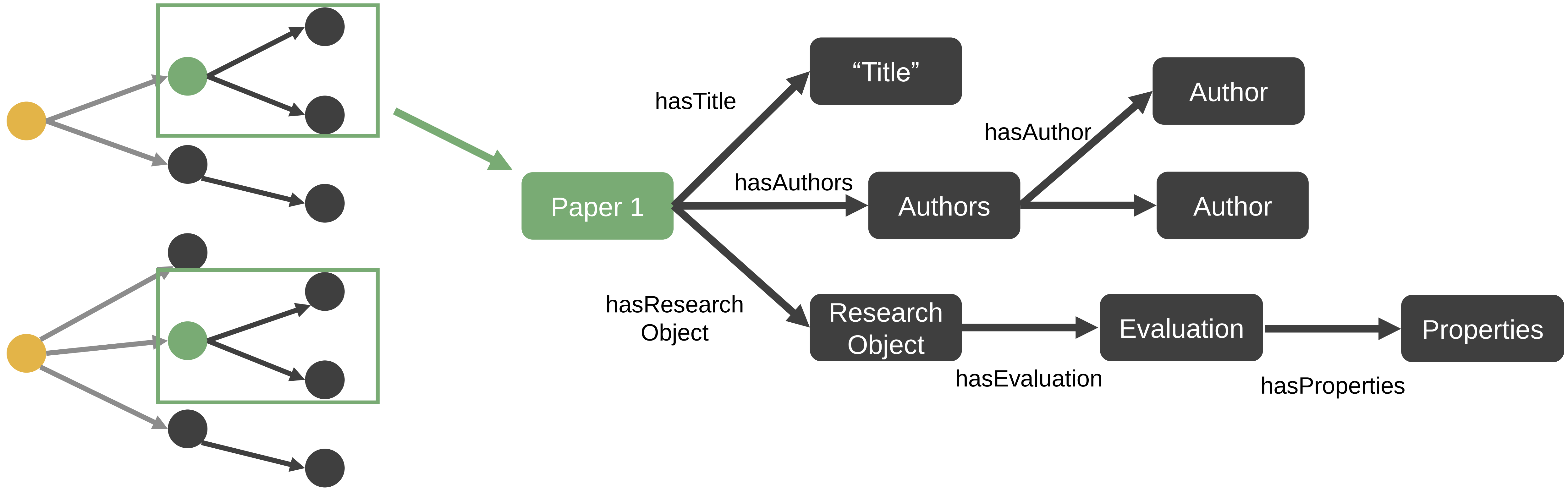


2. **Search:** The graph is traversed to find entities ● that are Hub types.



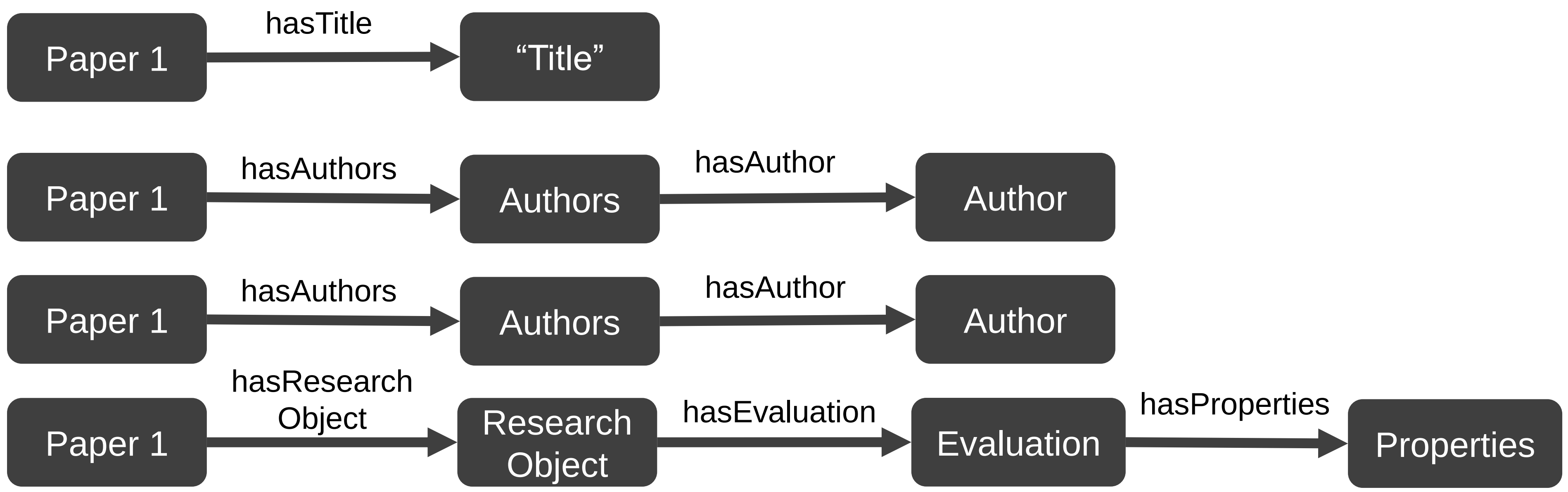
HubLink: Indexing

3. **Building Hubs:** Each root entity of a Hub is further processed. The following slides will explain this processing.



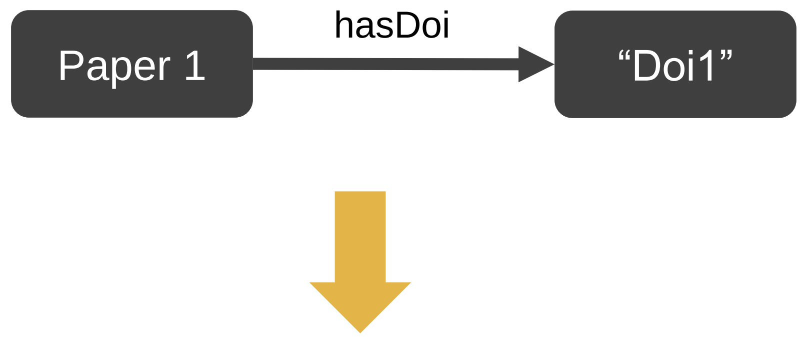
HubLink: Indexing

3. **Building Hubs – Find HubPaths:** The algorithm searches for the individual paths of a hub in the graph

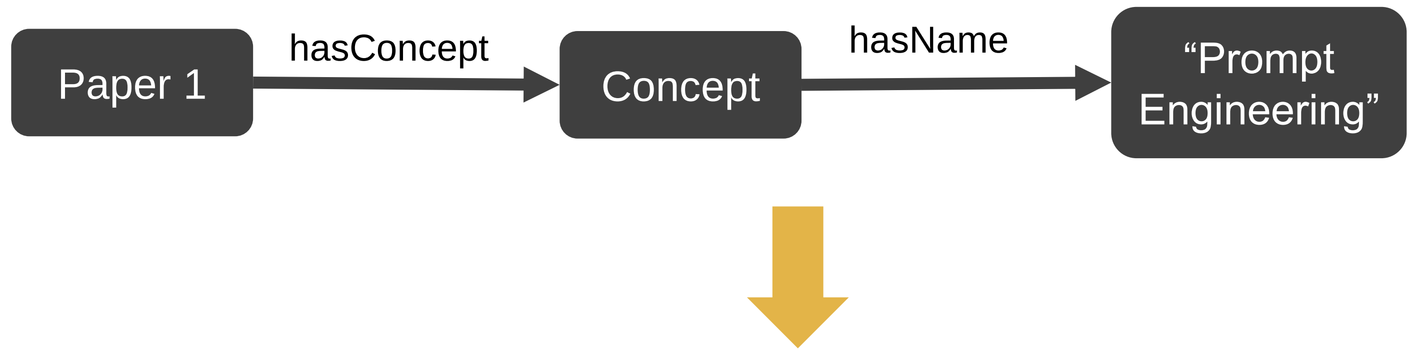


HubLink: Indexing

3. **Building Hubs - Build HubPaths:** Each path is converted to a textual representation.



The paper has the doi "Doi1"



The paper with the title "Paper1" has a concept with the name Prompt Engineering

HubLink: Indexing

3. **Building Hubs - Build HubPaths :** Each text is converted to an embedding vector which encodes the text into a vector space. This allows to apply distance calculations to find out how close one text is to another.

The paper has the doi “Doi1”

↓

(0, 1, 5, 2, 1, 5, -5, 1, -2, …)

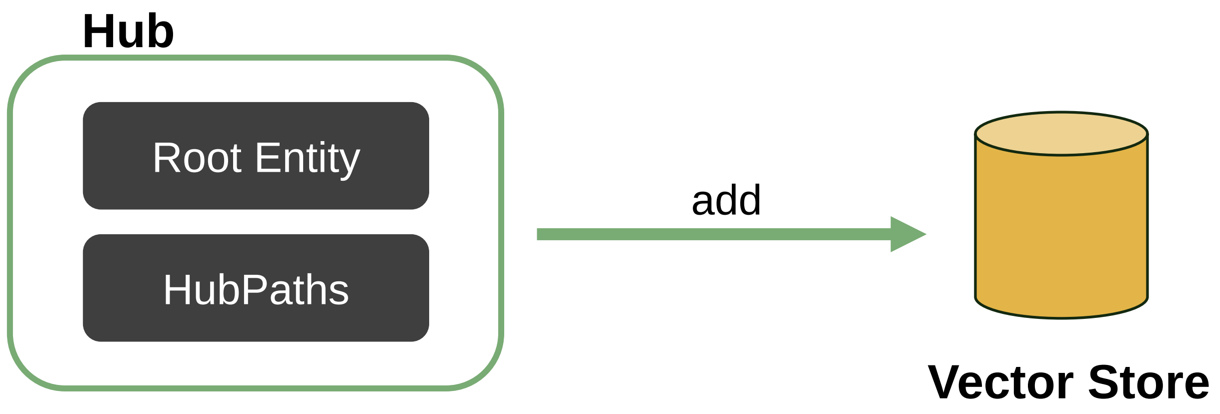
The paper with the title “Paper1” has a concept with the name Prompt Engineering

↓

(6, 2, 5, -5, 2, 9, 5, -1, -7, …)

HubLink: Indexing

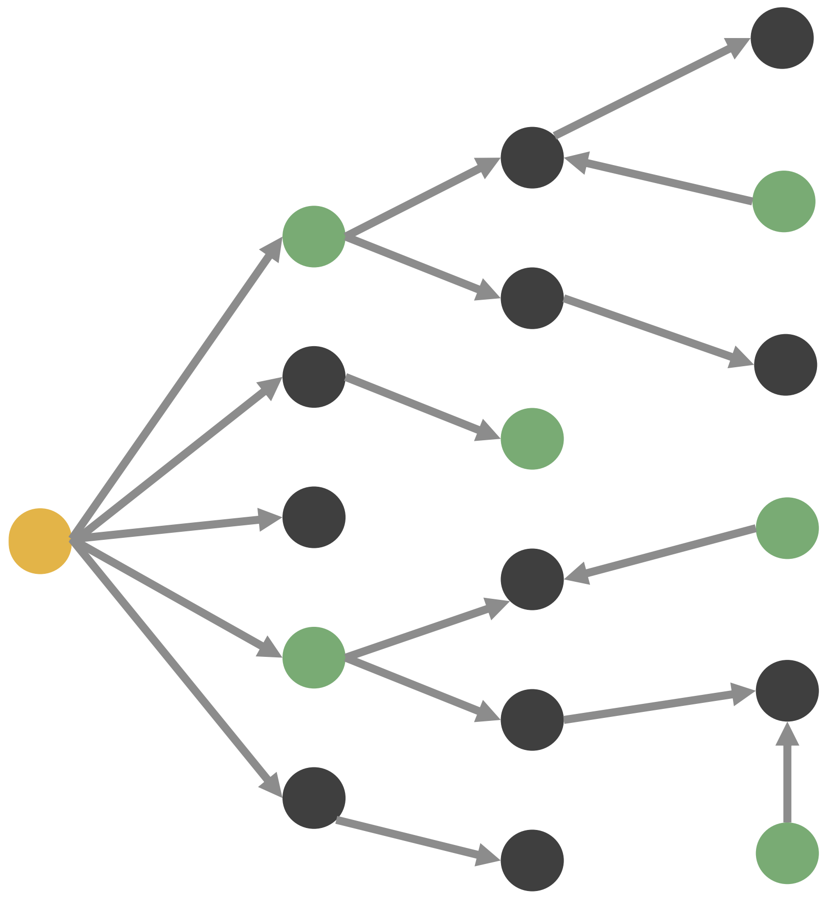
3. **Building Hubs:** After the Hub has been build, it contains information about the root entity in the graph and the HubPaths with their textual description and embedding. This Hub is than added to the vector store.



HubLink: Indexing

4. **Finding more Hubs:** After the hubs of the current level have been built and stored in the vector store, the algorithm proceeds to search for more hubs in the graph.

5. After the whole graph (or a subgraph depending on the configuration) have been searched and built, the indexing process finishes



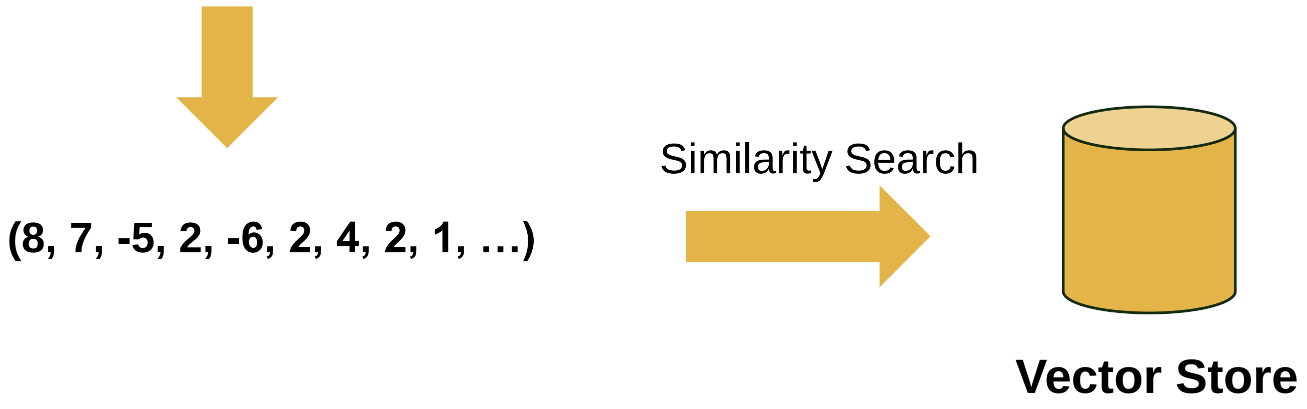
HubLink: Direct Retrieval Strategy

“Answering natural language questions on the graph without providing a entry point into the graph.”

HubLink: Direct Retrieval Strategy

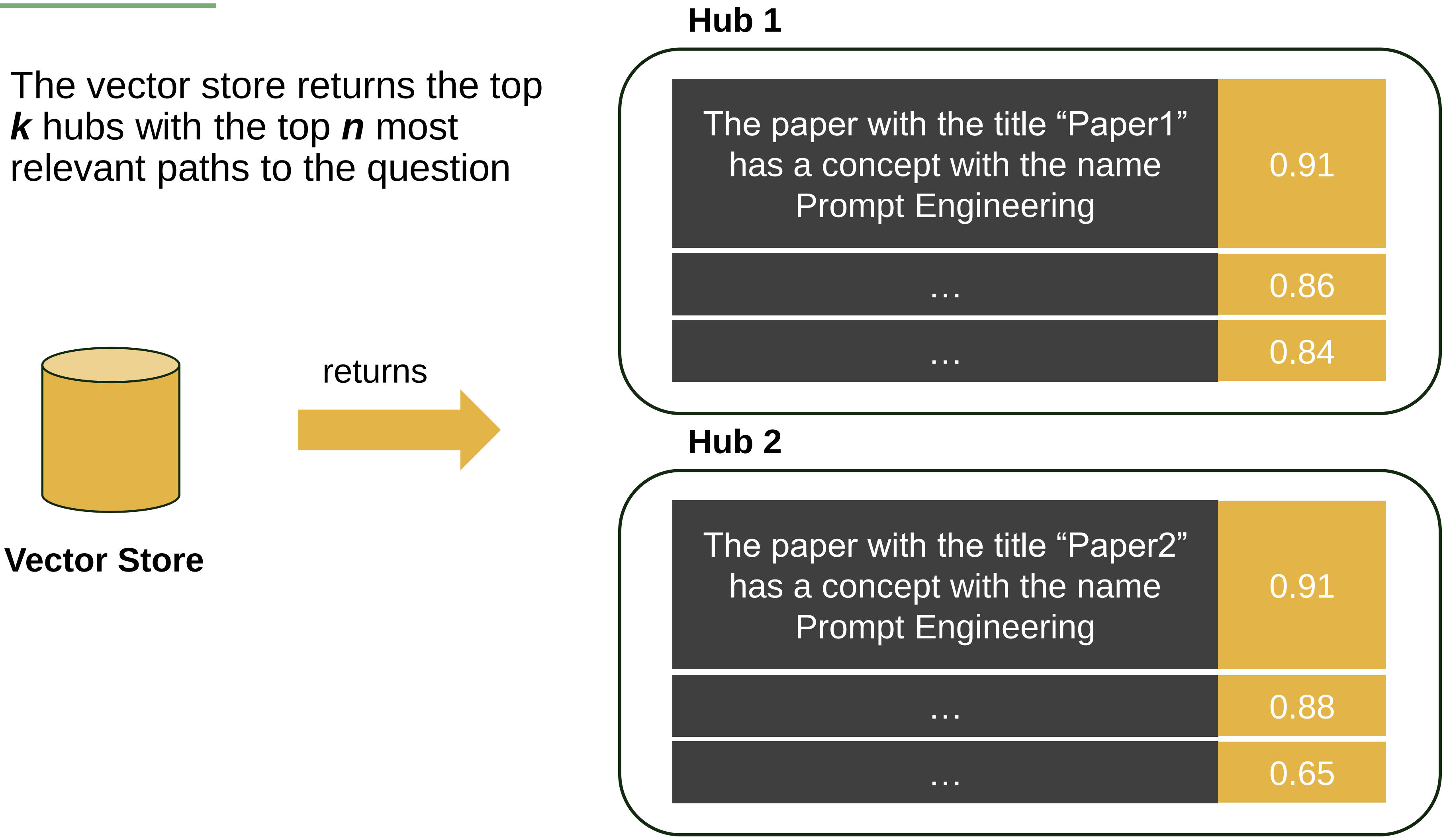
Question: “What papers published by IEEE talk about the concept of prompt engineering on LLMs?”

1. A question in natural language is asked
2. The retriever converts the question to an embedding
3. This is queried on the vector store



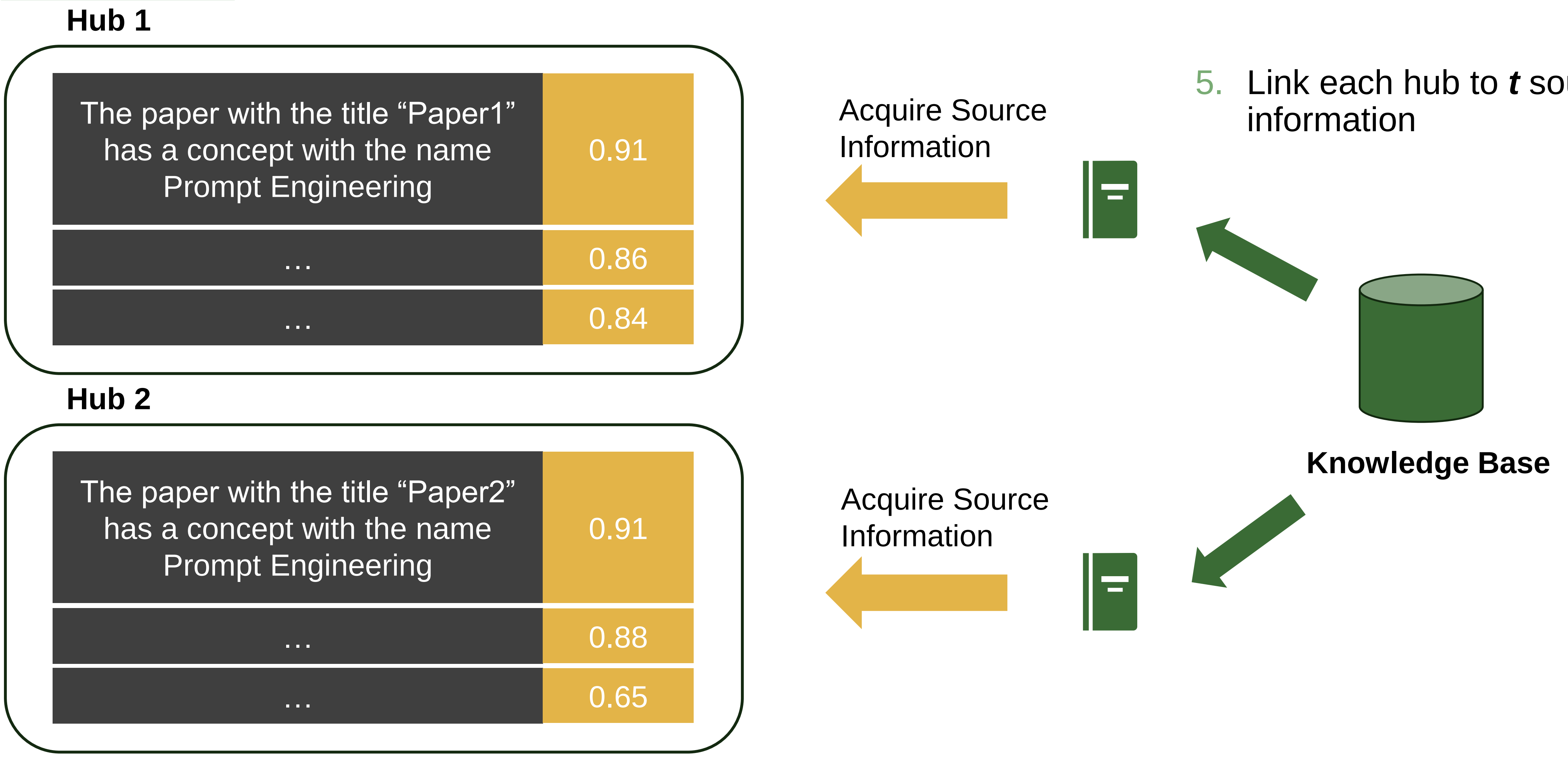
HubLink: Direct Retrieval Strategy

4. The vector store returns the top ***k*** hubs with the top ***n*** most relevant paths to the question



Vector Store
returns
Hub 1
The paper with the title "Paper1" has a concept with the name Prompt Engineering
0.91
…
0.86
…
0.84
Hub 2
The paper with the title "Paper2" has a concept with the name Prompt Engineering
0.91
…
0.88
…
0.65

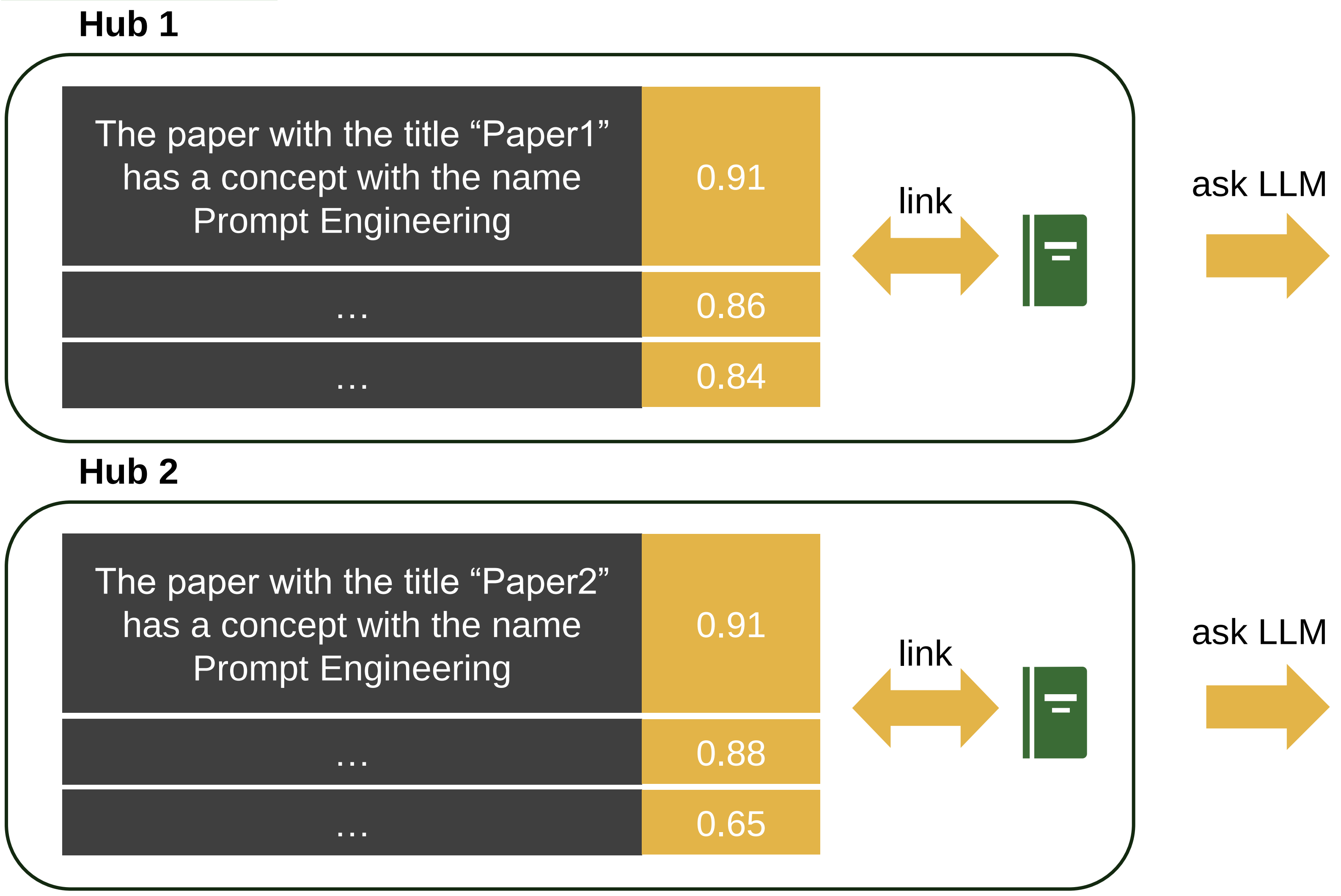
HubLink: Direct Retrieval Strategy



Hub 1
The paper with the title "Paper1" has a concept with the name Prompt Engineering
0.91
…
0.86
…
0.84
Hub 2
The paper with the title "Paper2" has a concept with the name Prompt Engineering
0.91
…
0.88
…
0.65
Acquire Source Information
Acquire Source Information
Knowledge Base

5. Link each hub to ***t*** source information

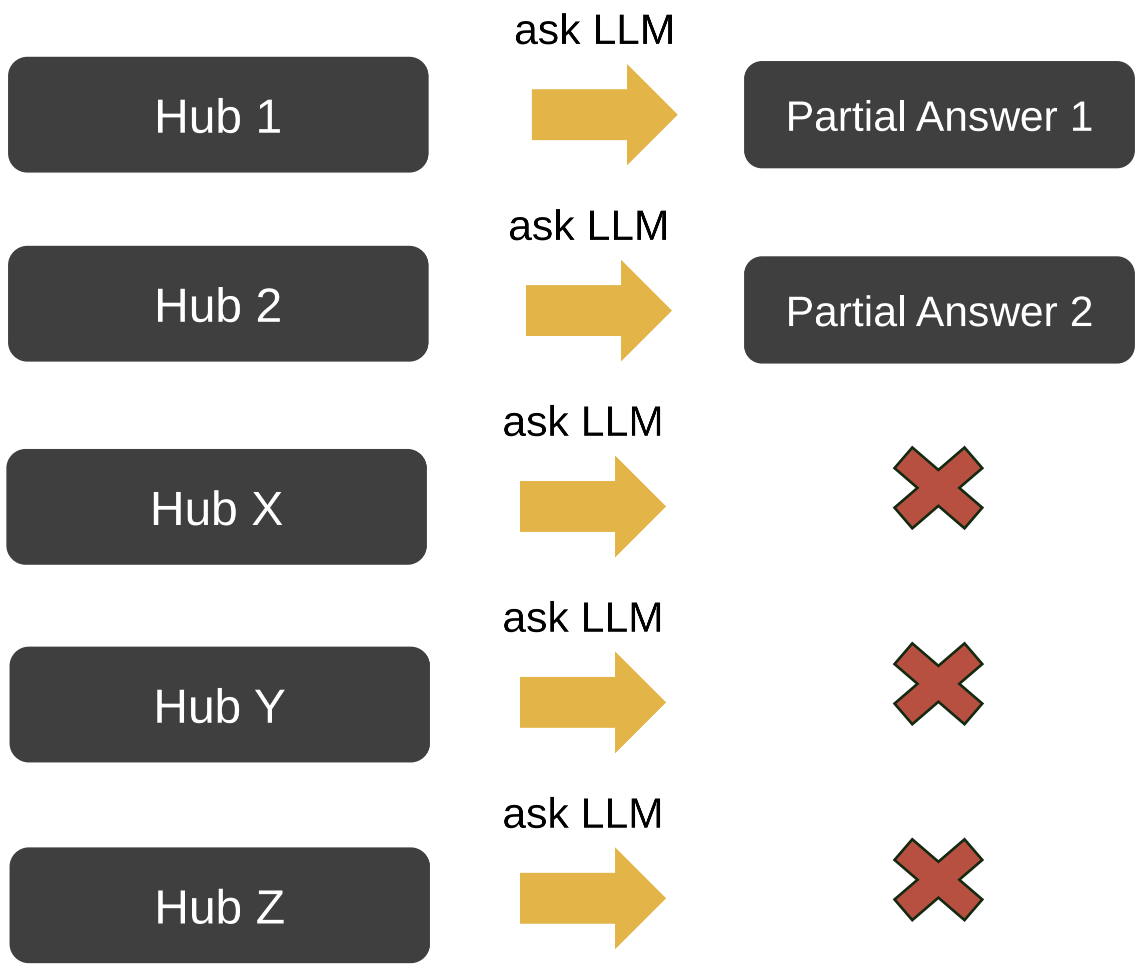
HubLink: Direct Retrieval Strategy



Hub 1
The paper with the title “Paper1” has a concept with the name Prompt Engineering
0.91
…
0.86
…
0.84
link
ask LLM
Partial Answer 1
Hub 2
The paper with the title “Paper2” has a concept with the name Prompt Engineering
0.91
…
0.88
…
0.65
link
ask LLM
Partial Answer 2

6. Generate partial answers for each HubSummary

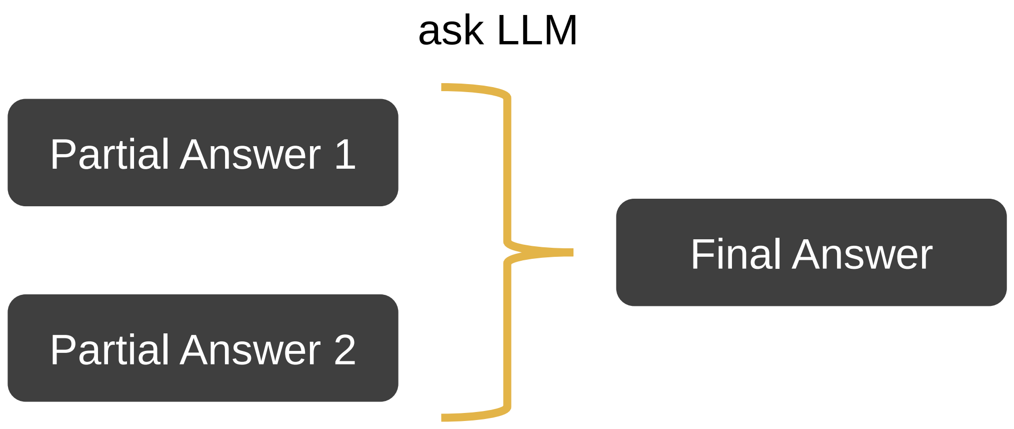
HubLink: Direct Retrieval Strategy



7. Hubs that where not relevant enough for the answer generation do not provide a partial answer

HubLink: Direct Retrieval Strategy

8. If partial answers are generated, they are used for the final answer generation. Else the retrieval stops without an answer.



HubLink: Graph Traversal Retrieval Strategy

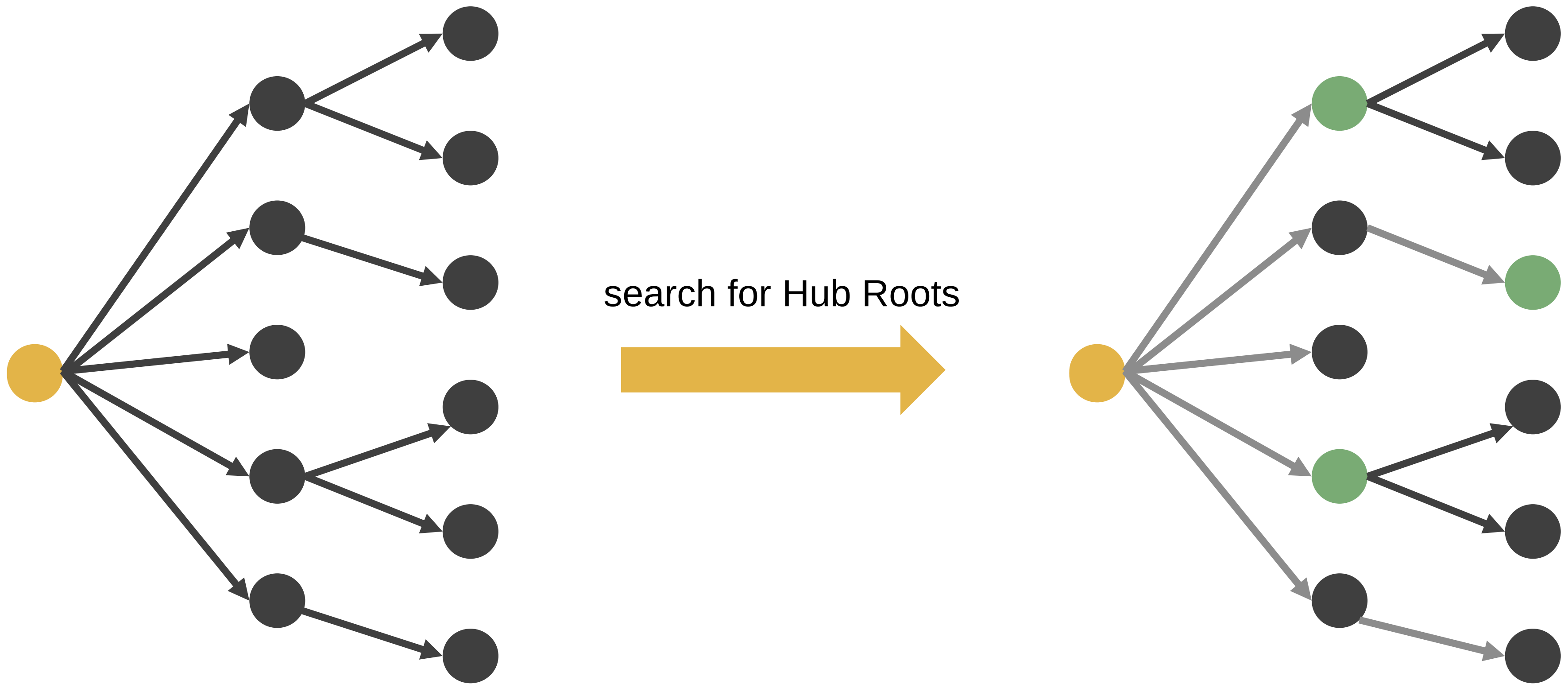
“Answering natural language questions on the graph while providing an entry point into the graph.”

Why does it make sense to provide an entry point?

- Without an entry point, the retrieval process must consider the whole indexed graph
- Providing an entry point helps the retriever to focus on a specific part of the graph
- Think of it as a kind of filter you would use in a search query giving the search more information about what type the answer should have. E.g. the answer should be from a paper with a specific Research Field, Publisher, Author, …

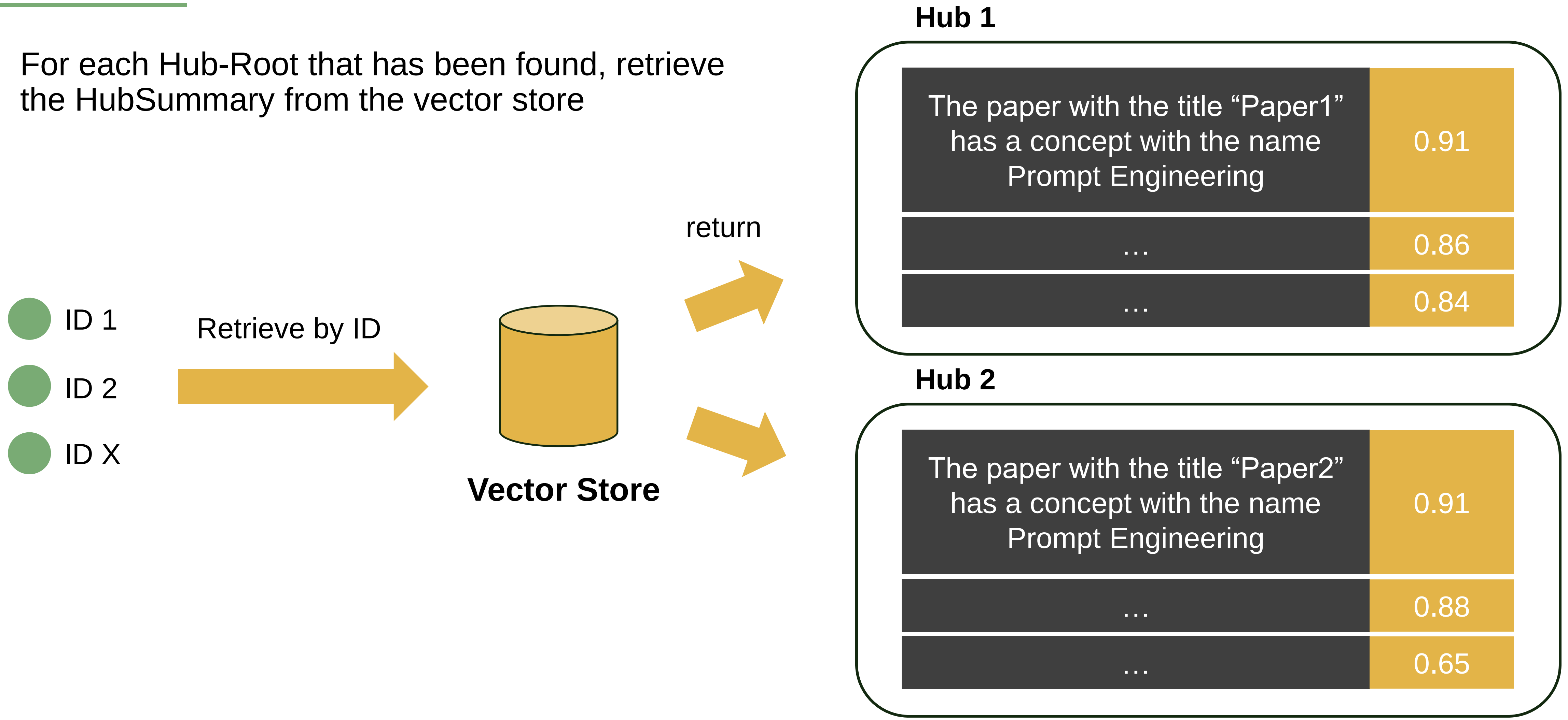
HubLink: Graph Traversal Retrieval Strategy

1. A question is asked, and a topic entity ● is provided
2. HubLink starts the retrieval process by searching for Hub-Root-Entities ● starting from the topic entity

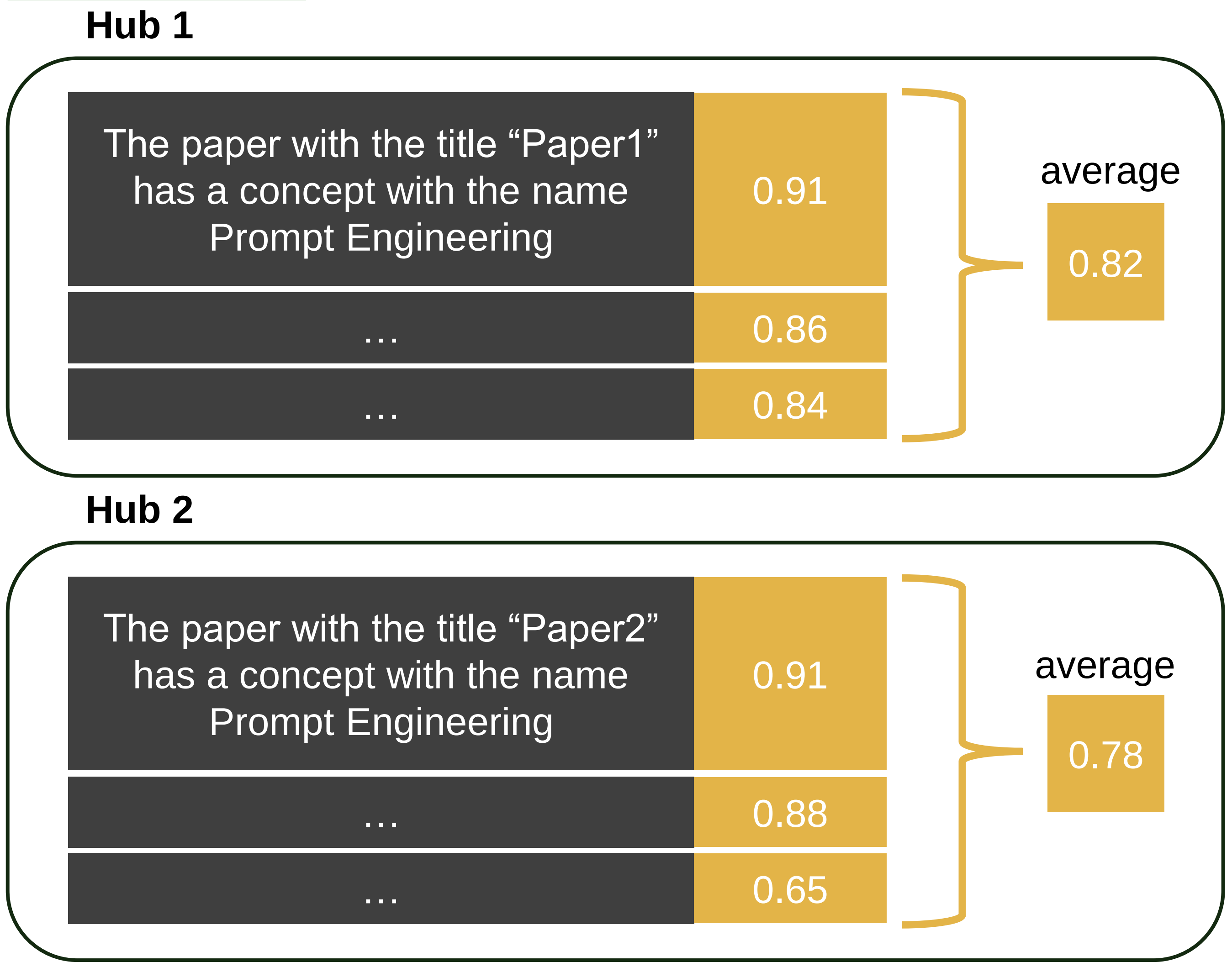


HubLink: Graph Traversal Retrieval Strategy

3. For each Hub-Root that has been found, retrieve the HubSummary from the vector store


ID 1
ID 2
ID X
Retrieve by ID
Vector Store
return
Hub 1
The paper with the title “Paper1” has a concept with the name Prompt Engineering
0.91
…
0.86
…
0.84
Hub 2
The paper with the title “Paper2” has a concept with the name Prompt Engineering
0.91
…
0.88
…
0.65

HubLink: Graph Traversal Retrieval Strategy

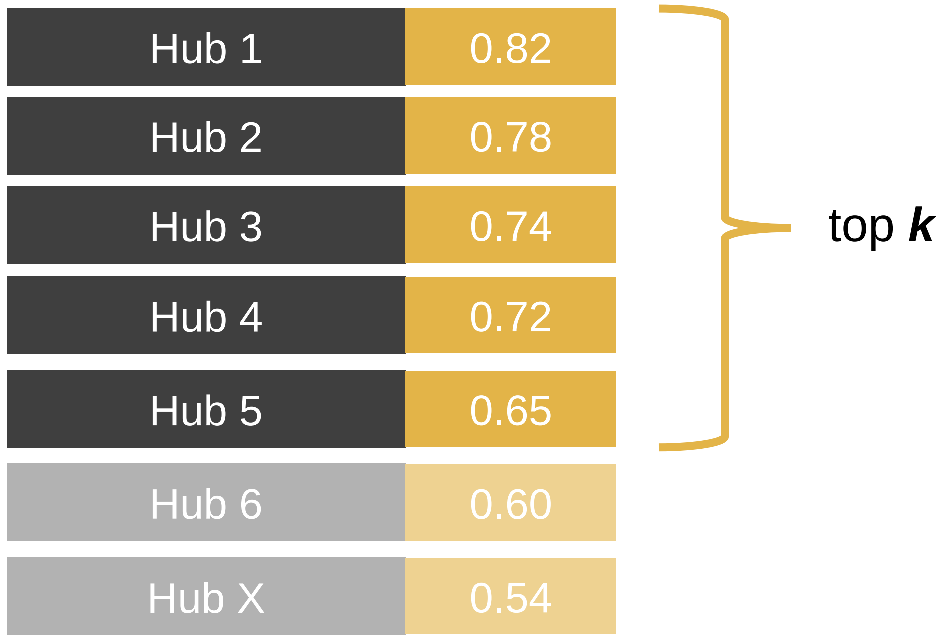


Hub 1
The paper with the title “Paper1” has a concept with the name Prompt Engineering
0.91
…
0.86
…
0.84
average
0.82
Hub 2
The paper with the title “Paper2” has a concept with the name Prompt Engineering
0.91
…
0.88
…
0.65
average
0.78

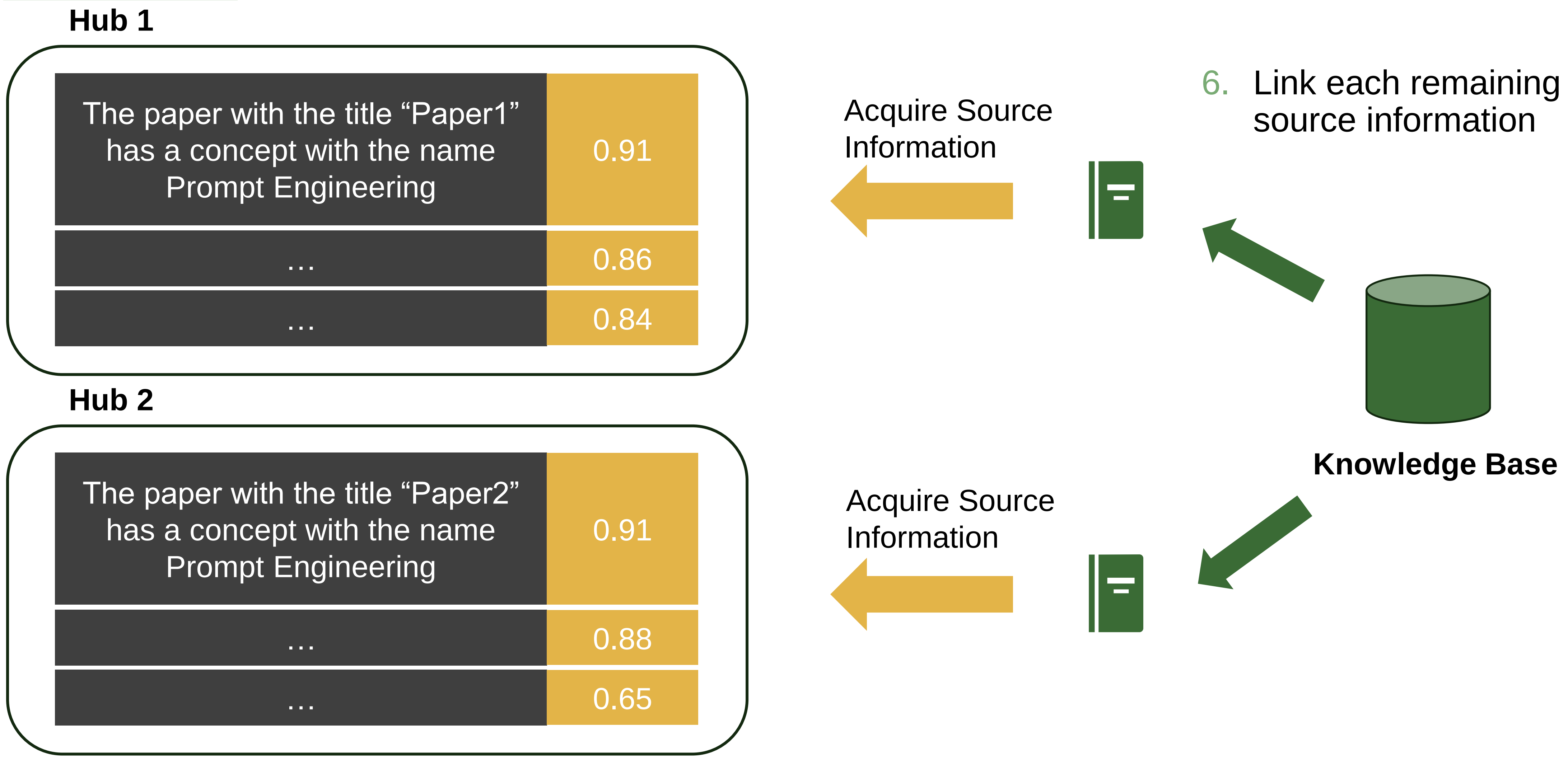
4. Calculate the average score of the retrieved HubPaths for each Hub

HubLink: Graph Traversal Retrieval Strategy

5. Only keep the top ***k*** hubs with the highest score and prune the rest.



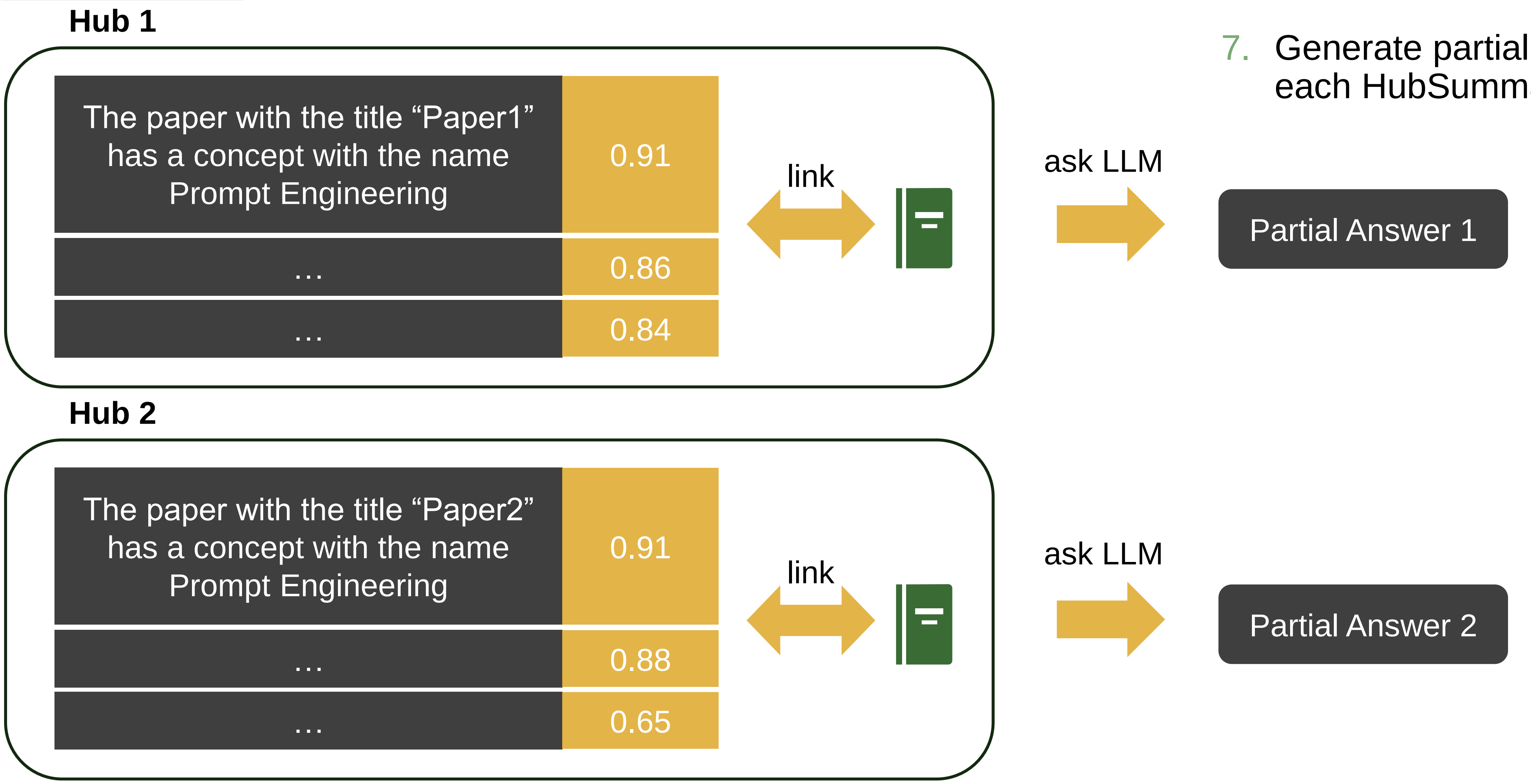
HubLink: Graph Traversal Retrieval Strategy


Hub 1
The paper with the title "Paper1" has a concept with the name Prompt Engineering
0.91
…
0.86
…
0.84
Hub 2
The paper with the title "Paper2" has a concept with the name Prompt Engineering
0.91
…
0.88
…
0.65
Acquire Source Information
Acquire Source Information
Knowledge Base

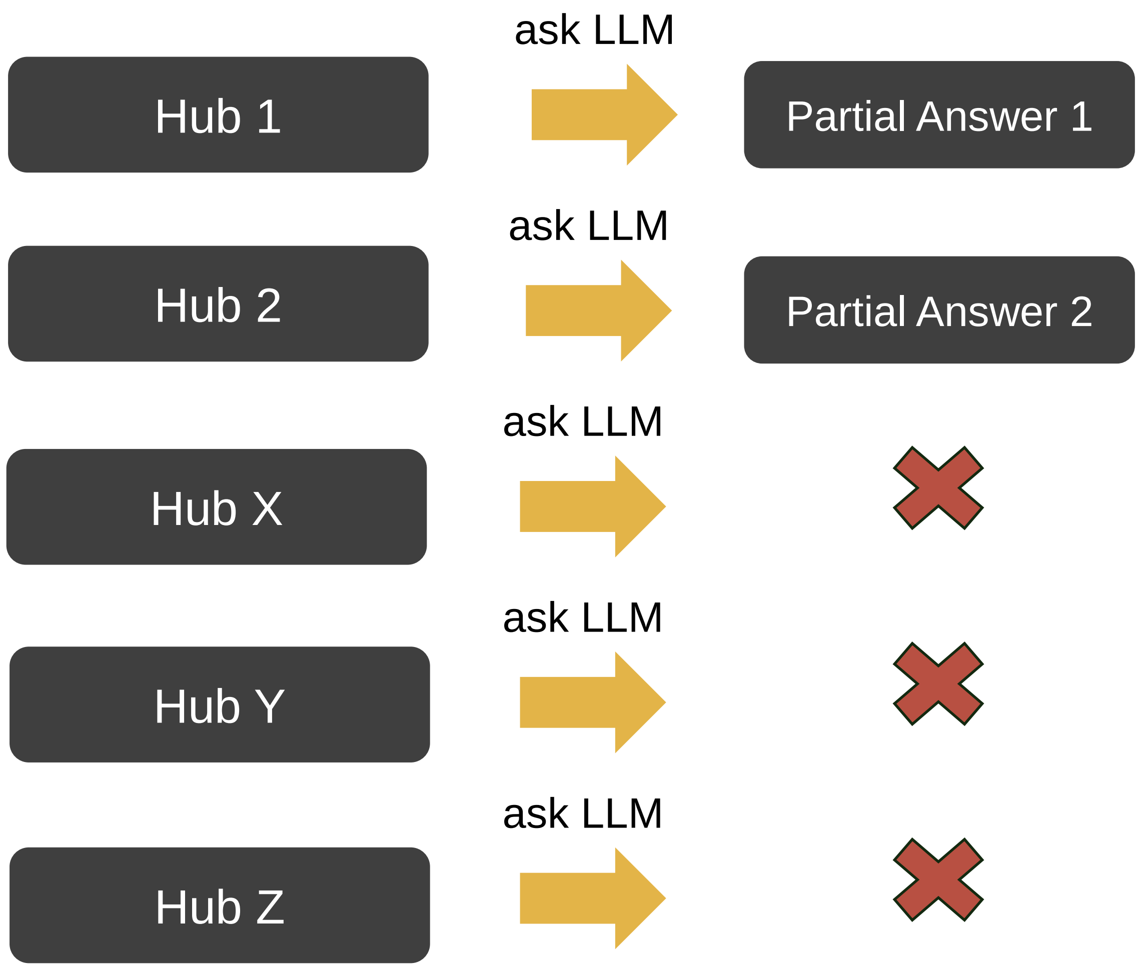
6. Link each remaining hub to ***t*** source information

HubLink: Graph Traversal Retrieval Strategy

7. Generate partial answers for each HubSummary


Hub 1
The paper with the title “Paper1” has a concept with the name Prompt Engineering
0.91
…
0.86
…
0.84
link
ask LLM
Partial Answer 1
Hub 2
The paper with the title “Paper2” has a concept with the name Prompt Engineering
0.91
…
0.88
…
0.65
link
ask LLM
Partial Answer 2

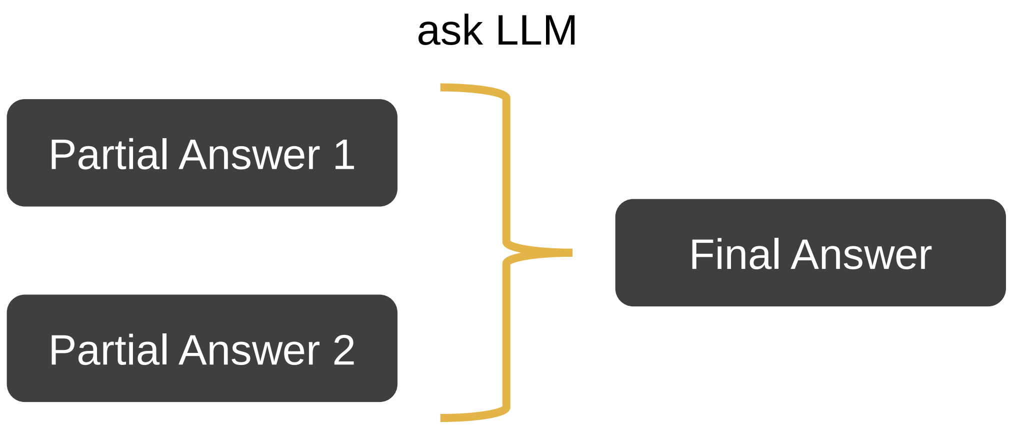
HubLink: Graph Traversal Retrieval Strategy



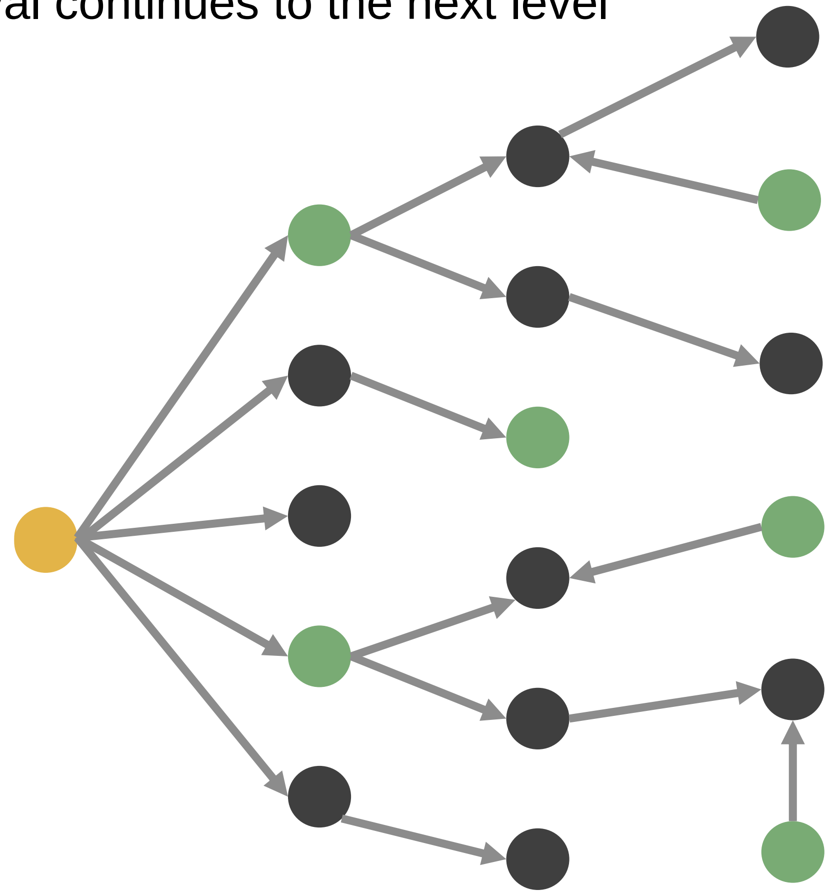
8. Hubs that where not relevant enough for the answer generation do not provide a partial answer

HubLink: Graph Traversal Retrieval Strategy

9. If partial answers are generated, they are used for the final answer generation.



10. If no partial answer are generated, the retrieval continues to the next level



Content

Related Research Questions

RQ 1

How to improve the QA performance of LLMs on scholarly literature research, by utilizing the structure of scholarly research graphs like the ORKG?

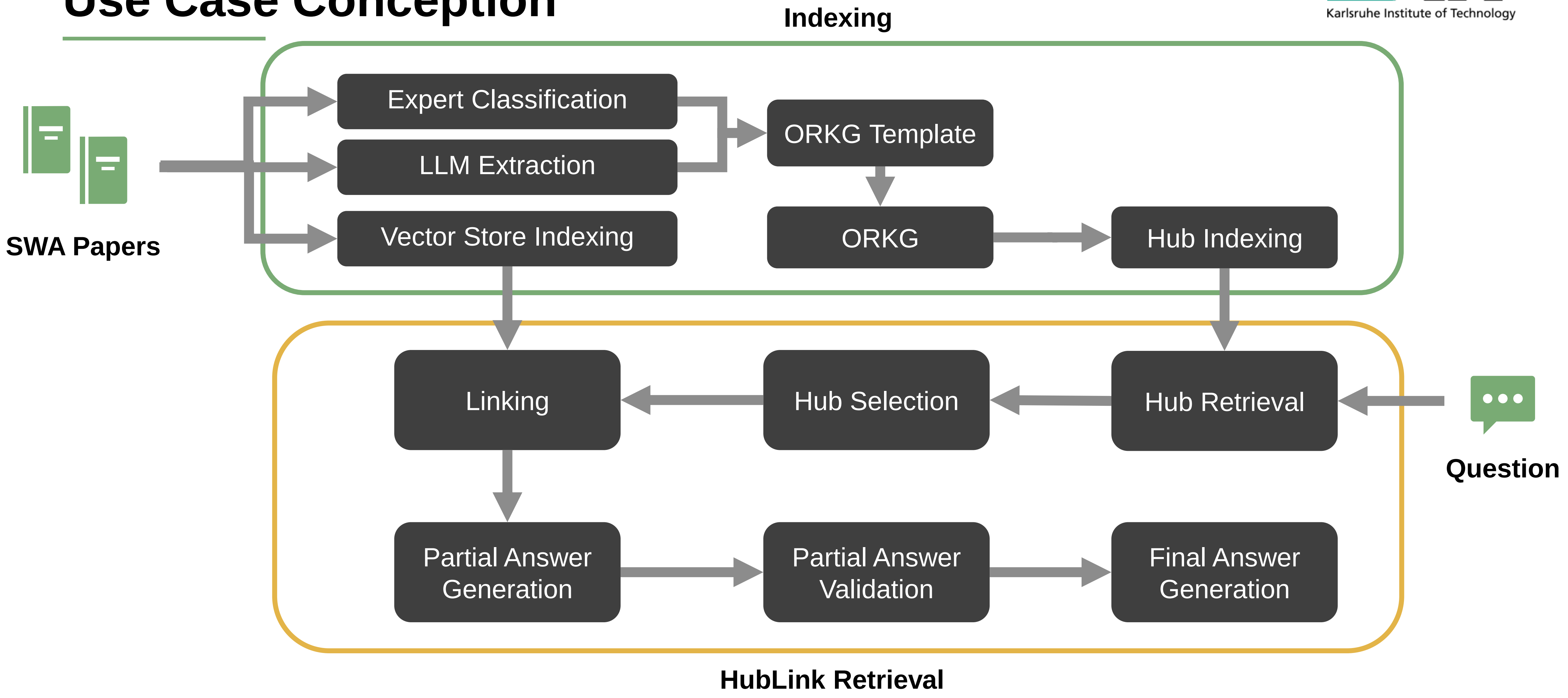
RQ 2

How effective is the proposed approach in the QA setting on a scholarly knowledge graph for addressing desirable question types for software architecture literature research?

- Ok so we have proposed an approach to answer **RQ1** now lets focus on **RQ2** which requires to determine the effectiveness of the approach
- For this we apply HubLink in a use case on the ORKG and evaluate its effectiveness against baselines
- For realizing both the use case and its evaluation, we implemented the SQA-System

- In the next slides we will first give a short overview on the concept of realizing the use case, then we introduce the SQA-System

Use Case Conception



Use Case Conception

- Given a set of SWA papers, we use an Expert Classification and a LLM Extraction to fill the information of a ORKG template.

- Using this template, we upload the paper data to the ORKG

- Using our HubLink algorithm we can now query question on the publications we have uploaded on the ORKG.

Content

SQA-System: Scholarly Question Answering System

“Providing a Framework for the integration and evaluation of KG-and Document-focused retrieval approaches.”

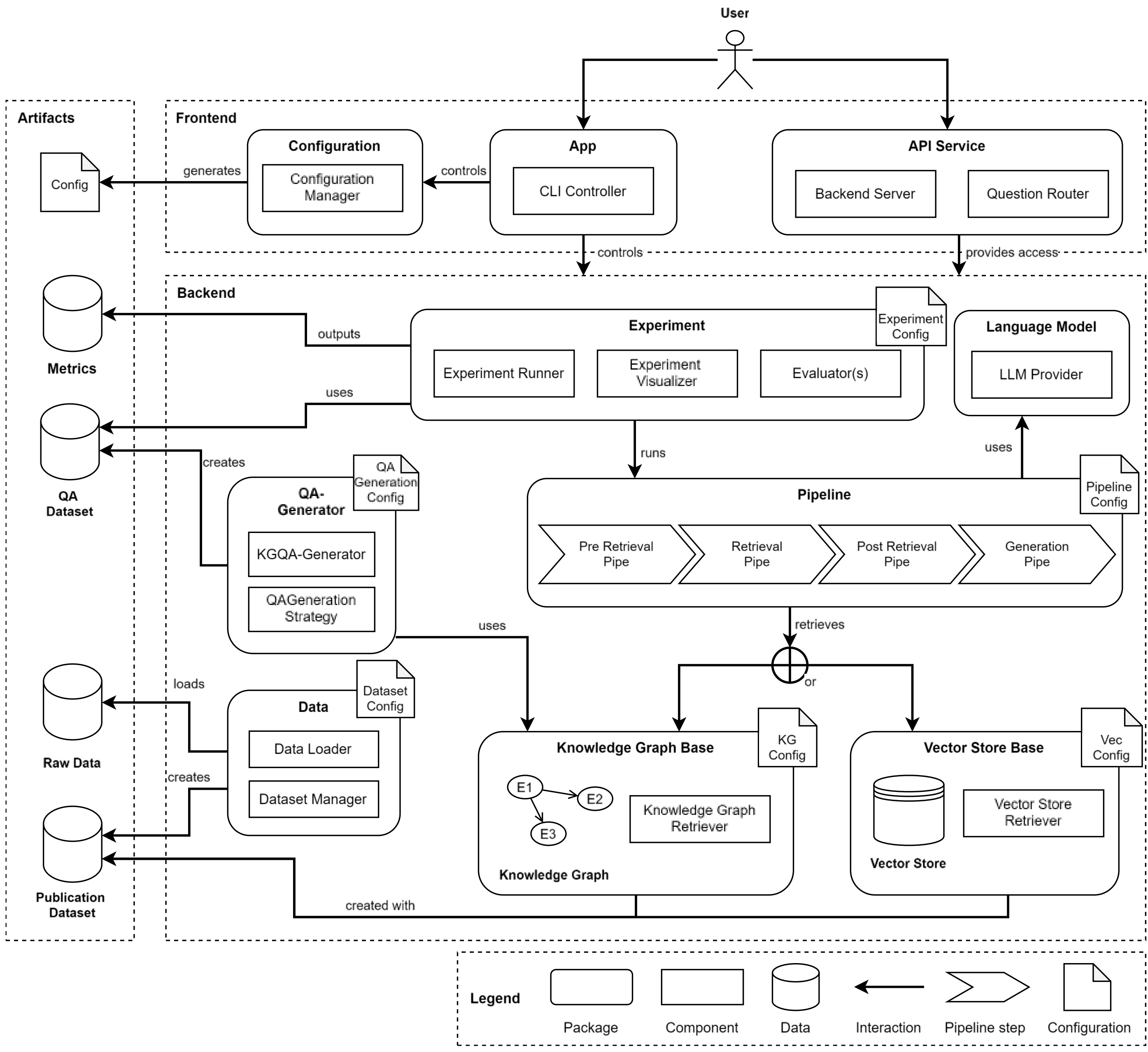
Preparation

- In the following slides we will be presenting an **overview** of the components implemented in our system

- We also provide several **links** to the repository which you can use to get comfortable with the code

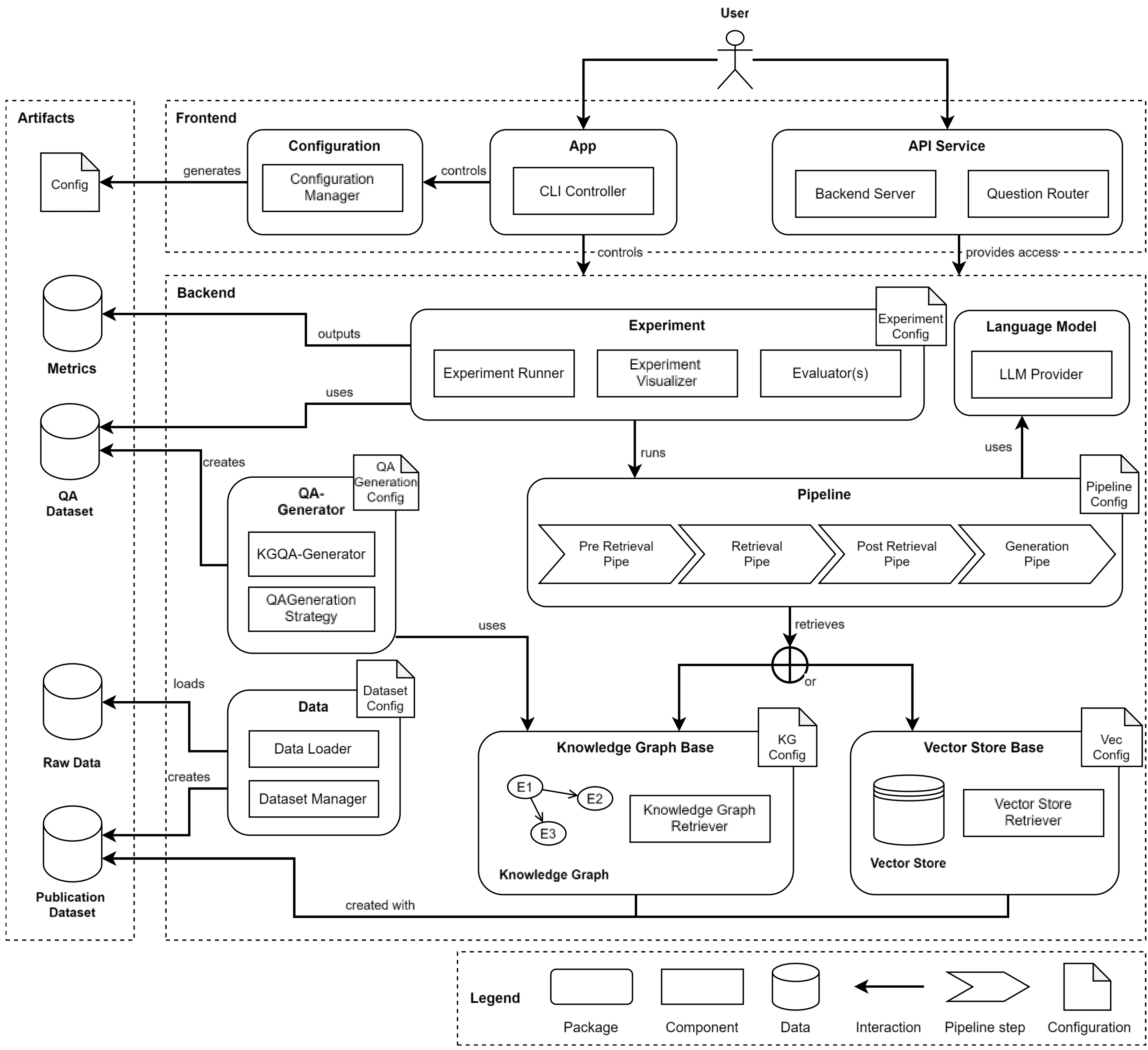
Overview of the SQA- System

- The illustration on the right shows an overview of each component of the implemented SQA-System.



Overview of the SQA- System

- The Pipeline Component runs the retrieval. It receives the question and outputs an answer
- The Pipeline retrieves information from Knowledge Bases which can be a Knowledge Graph or a Vector Store
- A Pipeline is run from the Experiment Component which queries question from the QA Dataset and evaluates the performance using Evaluators
- A QA-Dataset is created from the data loaded from the Data Component using the strategies from the QA-Generator to semi-automatically create questions using an LLM
- The whole system is controlled with configuration files managed by the Configuration Component which allow to reproduce experiments
- In addition, a CLI interface can be used to execute experiments, run pipelines, or manage configuration files



References

1. [Jaradeh19] Mohamad Yaser Jaradeh et al. “Open Research Knowledge Graph: Next Generation Infrastructure for Semantic Scholarly Knowledge”. In: Proceedings of the 10th Inter national Conference on Knowledge Capture. K-CAP ’19. Marina Del Rey, CA, USA: Association for Computing Machinery, 2019, pp. 243–246. isbn: 9781450370080. doi: 10.1145/3360901.3364435. url: https://doi.org/10.1145/3360901.3364435.
2. [Thambiand22] SincyV.ThambiandP.C.Reghuraj.“TowardsImprovingthePerformanceofQuestion Answering System using Knowledge Graph- A Survey”. In: 2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS) (2022), pp. 672–679. url: https://api.semanticscholar.org/CorpusID:247795830.
3. [Yangetal24] LinyaoYangetal. Give Us the Facts: Enhancing Large Language Models with Knowledge Graphs for Fact-aware Language Modeling. 2024. arXiv: 2306.11489.
4. [Lewis20] Patrick Lewis et al. “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks”. In: ArXiv abs/2005.11401 (2020). url: https://api.semanticscholar.org/ CorpusID:218869575.
5. [Gaoetal24] YunfanGaoetal.Retrieval-AugmentedGenerationforLargeLanguageModels:ASurvey. 2024. arXiv: 2312.10997.
6. [AbuSalih21] Bilal Abu-Salih. Domain-specific Knowledge Graphs: A survey. 2021. arXiv: 2011.00235.