Hierarchical Context Retrieval on Knowledge Graphs Using the Tree-RAG Framework

Brij Kishor

Roll: 2411MC09

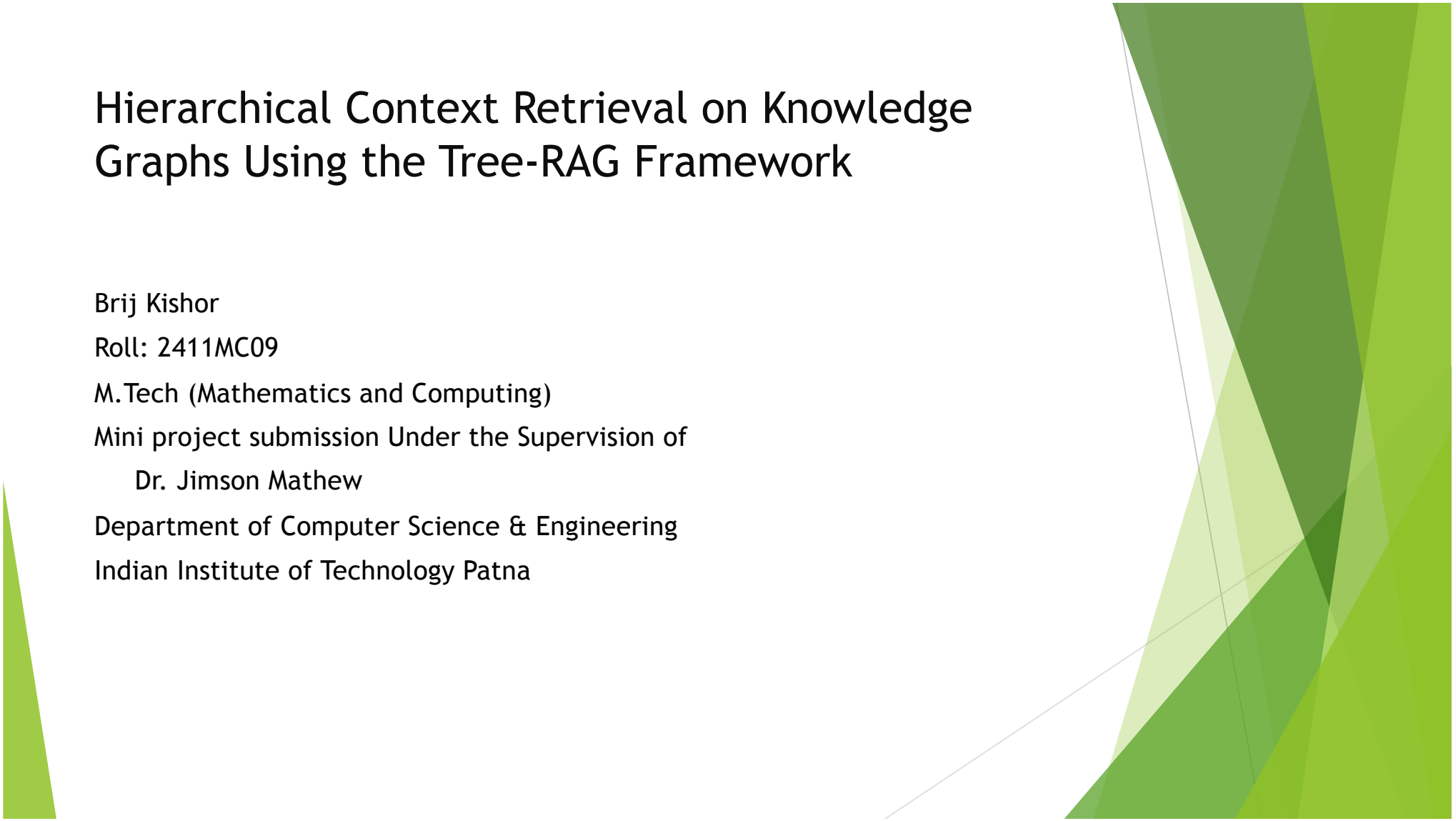
M.Tech (Mathematics and Computing)

Mini project submission Under the Supervision of

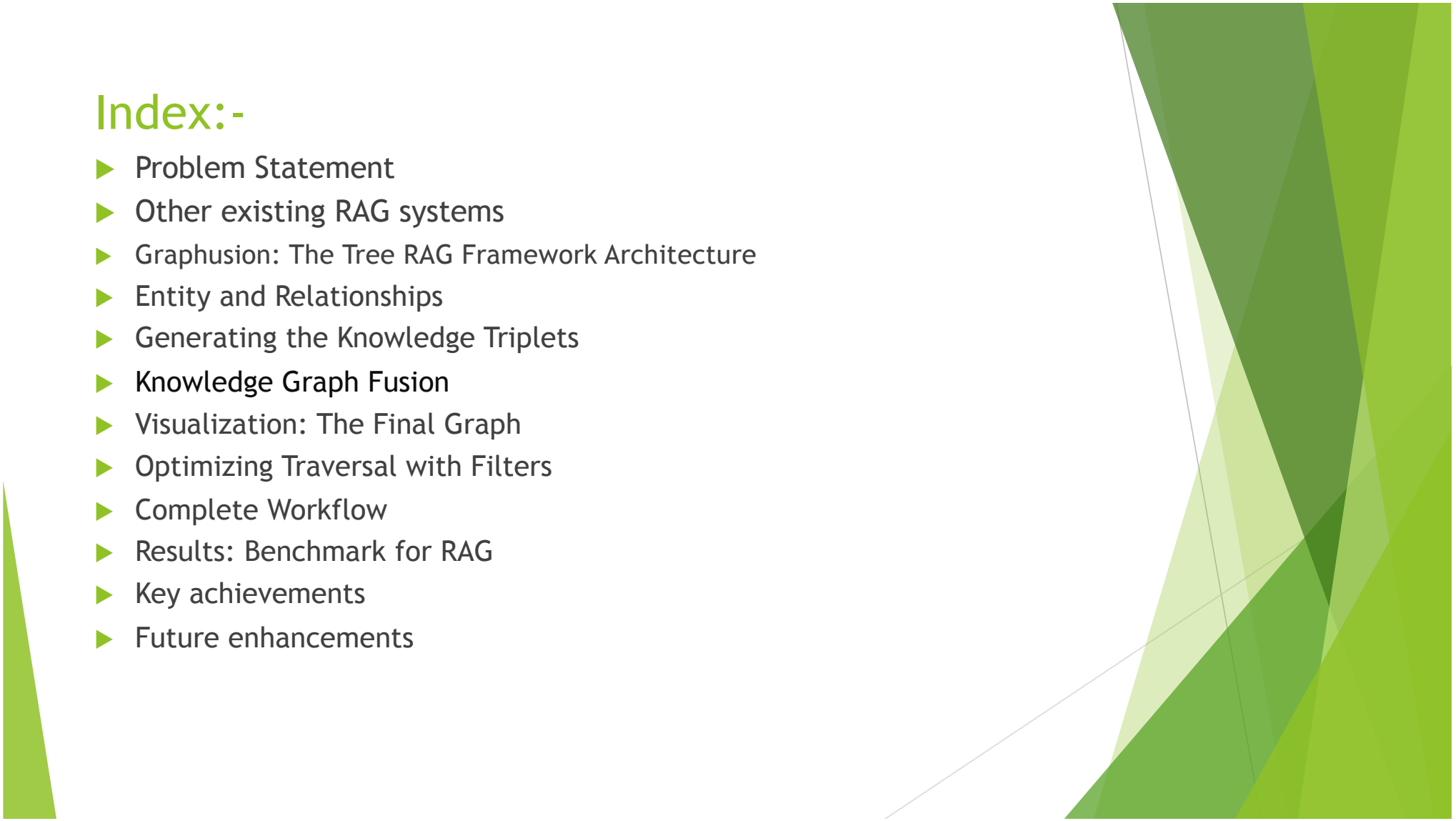
Dr. Jimson Mathew

Department of Computer Science & Engineering

Indian Institute of Technology Patna



Index:-



Problem Statement:-

Standard Retrieval-Augmented Generation (RAG):

It works well for simple questions on **unstructured text** (documents, articles, PDFs) but suffers for complex queries.

Fragmented Context: Standard RAG retrieves isolated text chunks , missing the full hierarchical context and parent-child relationships

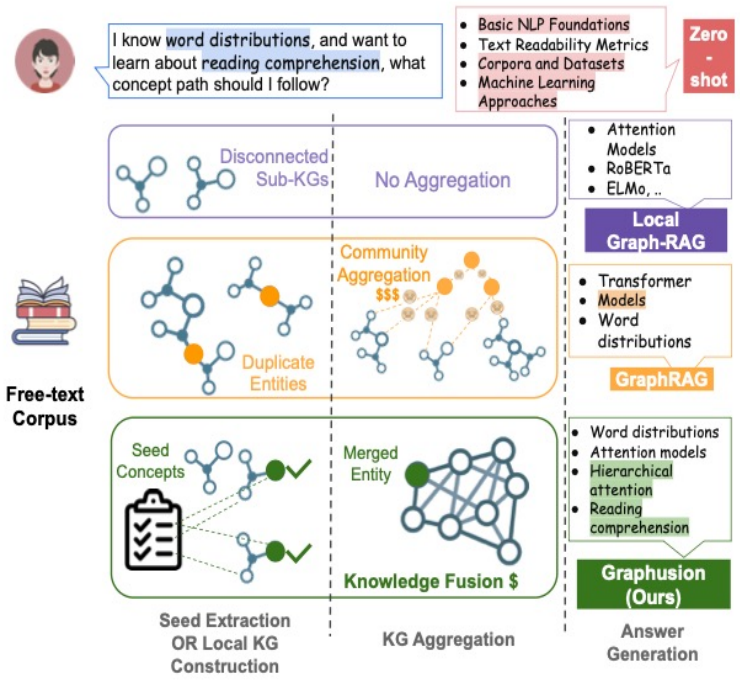
Ex-"When should we use BFS?”

Low Semantic Resilience: Standard RAG fails to retrieve correct information from ”noisy” or indirect queries

Ex-"Which algorithm quickly finds an element by repeatedly cutting the search region in half?"

High Latency: Slow for multi-hop questions due to sequential lookups.

Ex-"Which data structure should I use if I need the fastest lookup and I don’t care about order?"



Other RAG systems:-

- Local Graph RAG-Knowledge is extracted into many small seperate subgraphs hard to find relationships between concepts so it can't answer multi-step reasoning questions that leads to weak performance
- Graph-RAG-These graphs are not connected with duplicate entities, Still expensive and imperfect aggregation

Graphusion: The Tree RAG Framework

It achieves high performance by unifying three core components:

- **LLM-Driven Triplet Ingestion** :A powerful LLM automatically extracts key facts from documents.

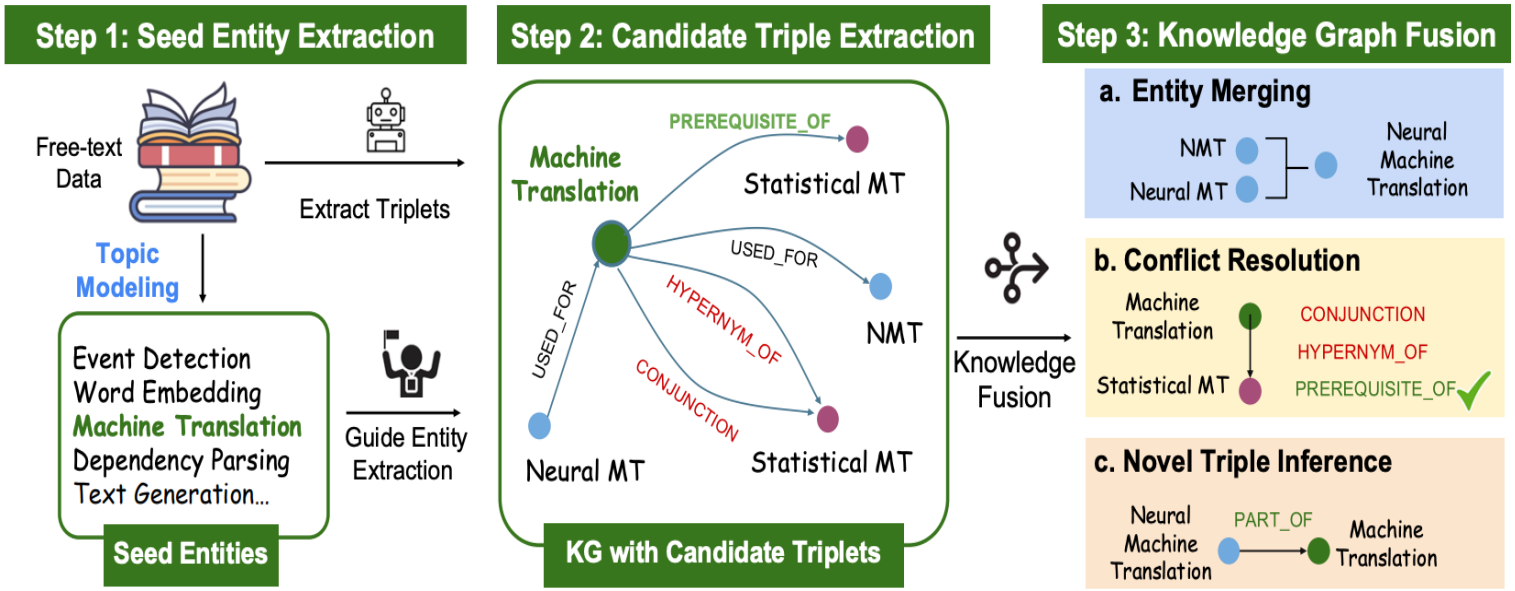
Converts them into structured sentences called triplets, e.g., (Array , Has-property, constant-time access).

- **Embedding-Based Fuzzy Matching using HNSW** :

Queries and graph entities are converted into numerical codes called embeddings .This enables semantic matching with help of HNSW, so the system finds relevant information even without exact keyword overlap.

- **In-Memory Forest Traversal** (”Instant Lookup” Phase):Instantly retrieves full context, including ancestors ,descendants and siblings , ensuring a complete, low-latency response in such a way that No need for multiple searches.

Architecture:-



Entity and Relationships :-

- An entity is a fundamental unit of information in a knowledge graph that represents a real-world object, concept, or event, these are nodes of graph
- Relations = connections between entities that give meaning to the graph
- **Leads-to**: One concept triggers another. Example: Recursion Leads-to function calls.
- **Is-similar-to**: Entities share similarities. Example: Stack Is-similar-to Queue.
- **Has-property**: Concept has certain attributes. Example: Binary Tree Has-property two children.
- **Has-function**: Defines functional role of a concept. Example: Heap Has-function priority queues.
- **Is-implemented-by**: Concept is realized by another. Example: QuickSort Is-implemented-by recursive partitioning.
- **Depends-on**: One concept relies on another. Example: Doubly Linked List Depends-on two pointers.

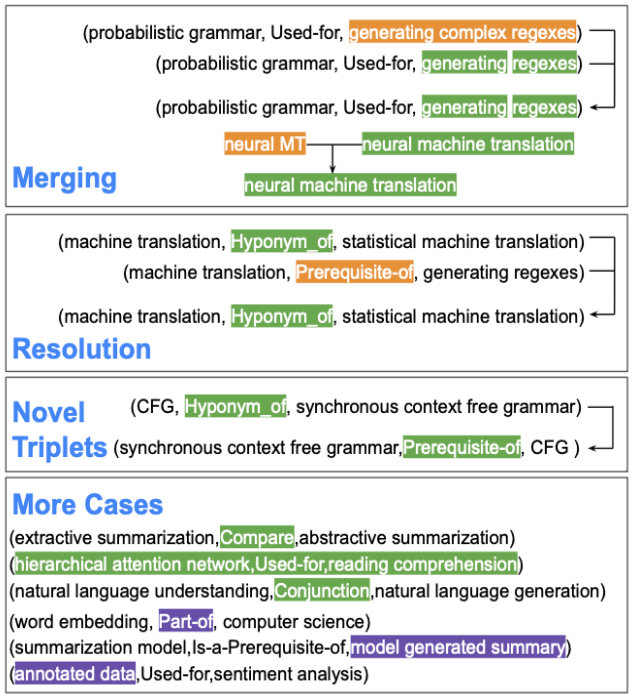
Generating the Knowledge Triplets

- **Step1 -Seed Entity Extraction**
- The system uses a specialized prompt to instruct a language model to act as a ”knowledge graph builder.”
- This LLM is tasked with identifying key nouns or noun phrases (concepts) directly from the raw text of the document.
- This process results in a clean list of educational concepts, such as ‘Array‘, ‘Quicksort‘, and ‘Time Complexity‘.
- **Step2- Candidate Triplet Extraction :-**
- The system takes the list of extracted entities and the original text.
- It prompts the LLM again, this time to identify relationships between these entities using a strict set of predefined relations.
- The output is a collection of ‘(subject, predicate, object)‘ triplets.
- Example:(Array, Has-property, Constant-time access)

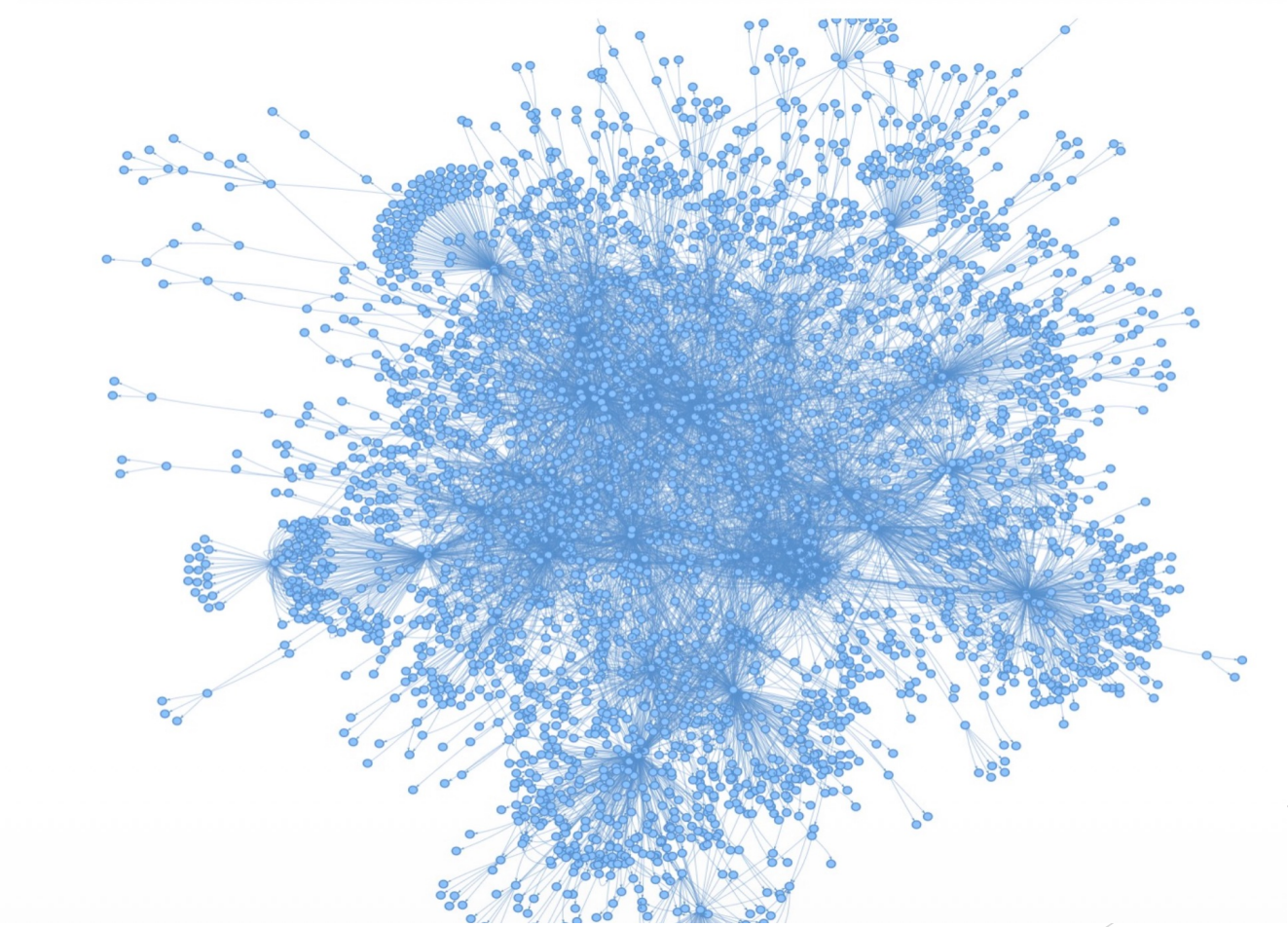
Step3-Knowledge Graph Fusion

- Goal: To transform the raw extracted triplets into a high-quality, accurate, and semantically rich knowledge graph.
- **Entity Fusion** :-Identifies and merges duplicate or semantically similar entities into a single, canonical representation.
- Example: ‘Dynamic Programming‘ and ‘DP‘ are fused into a single canonical entity, ensuring all related triplets point to one node.
- **Conflict Resolution** :-Resolves conflicting relationships between the same two entities. Using the original source text as a ground truth.
- Example: If the LLM generates both ‘(Array, Used-for, Sorting)‘ and ‘(Array, Has-property, Sorting)‘, the conflict resolution logic would analyze the text to determine the correct relationship and discard the incorrect triplet.
- **Novel Triplet Inference :-**Novel Triplet Inference enriches the knowledge graph by logically deducing new facts that weren’t explicitly written but are implied by existing relationships..
- Example: (Quicksort, Has-paradigm, Divide-and-Conquer), (Quicksort, Uses, Recursion), (Quicksort, Has-time-complexity, O(n log n))

  -> (Quicksort, Is-implemented-by, recursive partitioning)

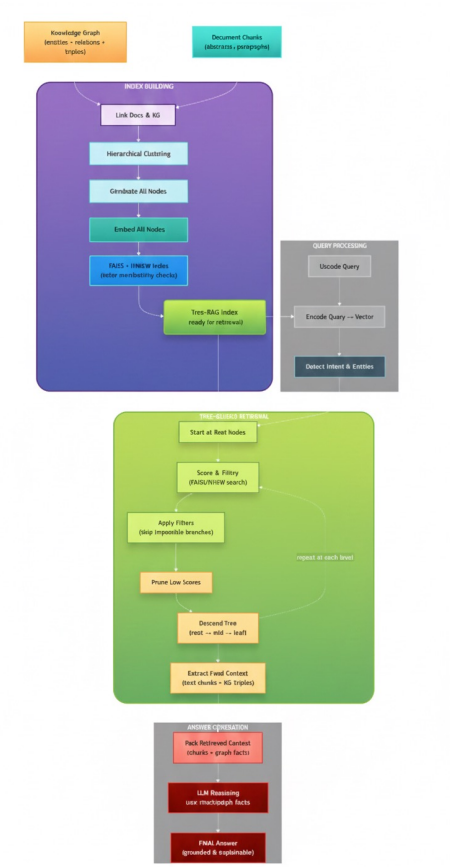


Visualization: The Final Graph



Optimizing Traversal with Filters

- Goal: When searching through a large knowledge graph, we don't want to waste time exploring parts of the graph that definitely do **not** contain the information we need.
- **Bloom Filters :-**
- Functionality:- A highly space-efficient filter that provides a fast, one-way check. It can definitively tell us if an entity is not present.
- Benefit:- If it returns 'false', we can skip that entire branch of the forest, saving significant time. If it return 'true' then it *might* be there, so we check further.
- **Cuckoo Filters :-**
- Functionality:- An advanced filter(works like a Bloom filter but with extra abilities) have faster lookups and support for deletions. It is designed for dynamic graphs where entities and relations are updated.
- Benefit:- It maintains search efficiency while allowing for a mutable dataset, which is essential for a real-world system.



Complete Workflow

- Tree RAG enables single-shot query with full hierarchical context.
- Query embedded into vector.
- FAISS IV+HNSW (Hierarchical Navigable Small World graph) finds semantic match (with filters).
- In-memory traversal retrieves ancestors + descendants .so instead of returning only **one chunk**, it returns the **full structured context**.
- Result: Complete, low-latency, context-rich response.

Results: Benchmark for Tree-RAG

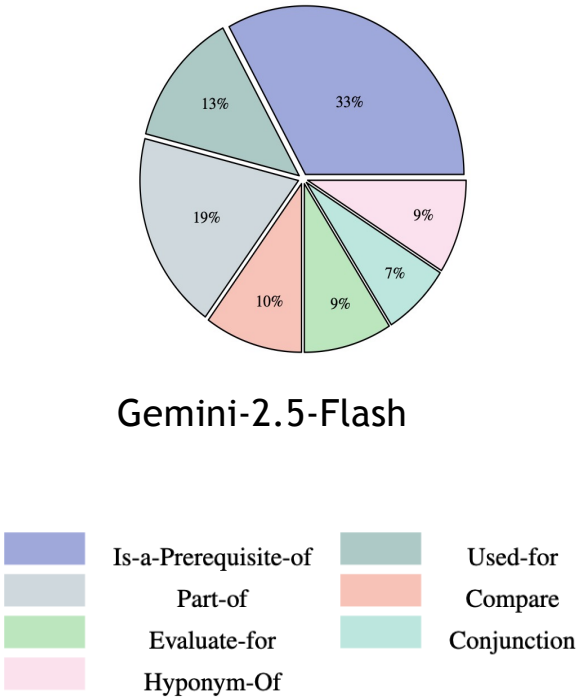
Setting	T1	T2	T3	T4	T5
GPT4o zs	69.20	64.42	66.61	44.00	11.45
GPT4o RAG	64.40	65.06	69.31	40.80	10.02
GraphRAG	60.40	64.19	67.45	42.00	8.96
Ours	**92.00**	**80.29**	**77.85**	**50.00**	**15.65**

Evaluation on Tasks 1-5. T1, T4: accuracy; T2, T3: similarity score; T5: hit rate.

Method	NLP		CV		BIO		Overall	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1
Supervised Baselines								
P2V[3]	0.6369	0.5961	0.7642	0.7570	0.7200	0.7367	0.7070	0.6966
BERT[7]	0.7088	0.6963	0.7572	0.7495	0.7067	0.7189	0.7242	0.7216
DeepWalk[31]	0.6292	0.5860	0.7988	0.7910	0.7911	0.8079	0.7397	0.7283
Node2vec[12]	0.6209	0.6181	0.8197	0.8172	0.7956	0.8060	0.7454	0.7471
Zero-shot (zs)								
LLaMA	0.6058	0.6937	0.6092	0.6989	0.6261	0.6957	0.6137	0.6961
GPT-3.5	0.6123	0.7139	0.6667	0.7271	0.6696	0.6801	0.6495	0.7070
GPT-4	0.7639	0.7946	**0.7391**	**0.7629**	**0.7348**	**0.7737**	**0.7459**	**0.7771**
Zero-shot + RAG								
GPT-3.5	0.7587	0.7793	0.6828	0.7123	0.6870	0.7006	0.7095	0.7307
GPT-4	**0.7755**	**0.7958**	0.7230	0.7441	0.7174	0.7200	0.7386	0.7533

Results: Benchmark for RAG

- Overall Performance: Our Tree RAG framework achieves a 2-5x reduction in latency compared to traditional methods, with sub-100 ms lookups on large-scale datasets.
- Core Innovation: By unifying embedding-based fuzzy matching with in-memory forest traversal, we enable a single, low-latency lookup that retrieves complete hierarchical context with high accuracy, even from noisy queries.
- Benefits of the Approach: This system delivers 10-20% higher recall under noisy query conditions and scales efficiently to millions of relations, making it ideal for information-rich domains.
- High-Level Summary: The Tree RAG framework successfully bridges the gap between semantic resilience and hierarchical awareness , providing a robust, high-performance solution for a new generation of knowledge-intensive RAG applications.



Key achievements-

- **Better Long Context Understanding**

Advantage: Gemini 2.5 Flash handles longer educational abstracts (10,000+ characters) more effectively

Evidence: The system processes entire academic documents in chunks without losing context

Comparison: LLaMA has shorter context windows, GPT models may lose coherence in longer texts

- **Superior Structured Output Generation**

Advantage: Gemini consistently produces well-formatted triplets in the exact (subject, predicate, object) format required

Evidence: The prompts show strict formatting requirements that Gemini handles reliably

Comparison: LLaMA often struggles with consistent formatting, GPT-3.5/4 may hallucinate extra formatting

- **Faster Processing Speed**

Advantage: Gemini 2.5 Flash optimized for rapid inference

Evidence: Pipeline processes large datasets efficiently

Comparison: LLaMA requires more computational resources, GPT-4 has slower response times

- **Improved Factual Accuracy in Knowledge Extraction**

Advantage: Higher precision in identifying genuine relationships vs. spurious connections

Evidence: Generated triples show meaningful educational relationships

Comparison: LLaMA may hallucinate relationships, GPT can be overly creative

Future enhancements :-

- **Advanced Graph Operations:**
  - Implement graph validation and consistency checking
  - Support for graph analytics (centrality, communities, paths)
  - Enable graph versioning and diff tracking
- **Automated Quality Assurance:**
  - Add confidence scores for extracted triples
  - Implement conflict resolution mechanisms (beyond fusion step)
  - Create validation benchmarks and metrics
- **Language Extension:**
  - Complete Japanese medical domain implementation (currently incomplete)
  - Add support for other languages beyond English
  - Cross-lingual knowledge graph fusion
- **Multi-modal Integration:**
  - Extract knowledge from figures, tables, equations
  - Support PDF, HTML, and structured documents (currently only .txt)
  - Integrate with citation networks

Thank You for your time!