Using Knowledge Graphs for Language Model and RAG Evaluation

Introduction to Knowledge Graphs

A **Knowledge Graph (KG)** is a structured representation of information, expressed as a set of **triplets**: `(subject, predicate, object)`.
Example:

- "Marie Curie discovered radium."
- Triplet: (Marie Curie, discovered, radium)

KGs capture semantic relationships, enabling machine-readable understanding and reasoning.

Knowledge Graphs, Documents Representation and LLM

Knowledge Graph and Document Representation:

- Capture **semantic structure**, not just surface text.
- Make implicit relationships **explicit**.
- Enable **efficient querying** and reasoning.
- Allow **paraphrase-robust** comparisons via embeddings.

Knowledge Graphs serve as structured semantic maps that help guide Large Language Models (LLMs) in understanding and generating relevant content.

Knowledge Graph as LLM Guidance Mechanisms

- **Semantic Scope Restriction:** KGs define which entities and relations are relevant, helping LLMs stay on topic and reduce hallucination.
- **Query Generation:** LLMs can use KG structure to generate focused questions based on known entity-relation paths.
- **Context Selection:** KGs help in selecting context that aligns with query intent by matching subgraphs or triplets.
- **Answer Structuring:** KGs can guide LLMs to generate answers that are type-consistent (e.g., answer a `who` question with a `Person` entity).
- **Inference and Reasoning:** Multi-hop reasoning can be simulated by traversing the KG to form intermediate facts or chains of logic.

KGs provide constraints and priors that make LLM behavior more accurate, interpretable, and aligned with knowledge.

Knowledge Graph Components

A knowledge graph is built from the following core components:

- **Entities:** The real-world objects or concepts (e.g., *Marie Curie*, *radium*)
- **Relationships:** Directed edges between entities indicating interactions or connections (e.g., *discovered*, *born_in*)
- **Attributes:** Properties or characteristics of entities (e.g., *name* = *Marie Curie*, *birth_year* = *1867*)

These components form a labeled, directed graph where nodes represent entities and edges encode their relationships.

What is an Entity in a Knowledge Graph?

- **Entity:** A real-world object, concept, or thing that is identifiable.
- In a KG, entities are represented as **nodes**.
- Entities are connected via **relationships (edges)**.
- Entities can have **attributes (properties)**.

Examples of Entities:

Domain	**Entities**
People	Barack Obama, Marie Curie
Places	Paris, Mount Everest
Organizations	UNESCO, OpenAI
Products	iPhone 13, Toyota Prius
Concepts	Quantum Mechanics, GDP

How to Define Entities in a Knowledge Graph

Step-by-step:

1. **Identify the Domain and Scope**
   Choose the focus of your KG (e.g., healthcare, finance).
2. **Determine Entity Types (Classes)**
   Define types like `Person`, `Organization`, `Place`.
3. **Assign Unique Identifiers (URIs)**
   Use global or local identifiers to ensure uniqueness.
4. **Describe with Attributes and Relationships**
   Add properties (e.g., birth date) and connections to other entities.

What is an Attribute?

- An **attribute** (also called a property or literal) is a **key-value pair** describing an **intrinsic feature** of an entity.
- Attributes do **not** link to other entities—they attach **literal values** such as strings, numbers, or dates.
- They enrich an entity with factual details.

Example:

- Entity: `:BarackObama`
- Attributes:
  - `:birthDate "1961-08-04"`
  - `:fullName "Barack Hussein Obama"`
  - `:height "1.85"`

What is a Relationship?

- A **relationship** (also called an edge or object property) **connects two entities**.
- It represents a **semantic link** between them.
- Formally described as a triple: `(subject, predicate, object)`, where both subject and object are entities.

Example:

- `:BarackObama :spouse :MichelleObama`
- `:BarackObama :presidentOf :USA`

Attribute vs Relationship

Feature	**Attribute**	**Relationship**
Value Type	Literal (string, number)	Entity (URI)
Connects	Entity to a value	Two entities
Example	`:birthDate "1961-08-04"`	`:spouse :MichelleObama`
Used For	Describing characteristics	Representing links

Creating Knowledge Graphs and Ontologies

Step-by-step to build a Knowledge Graph:

1. **Extract entities and relationships** from text using NER and relation extraction models.
2. **Normalize entities** to resolve duplicates and aliases (e.g., USA" = United States").
3. **Define schema or ontology:** list allowed types of entities and relationships.
4. **Link to external knowledge bases** like Wikidata or DBpedia for enrichment.
5. **Store in a graph database** (e.g., Neo4j, RDF store).

Ontology:

- A formal specification of types (classes), relationships (properties), and constraints.
- Example: *Person* can have *name*, *birth_place*, *works_at*.

Guiding LLM Query Generation with KGs

1. Use entity/relation types in KG to identify query opportunities.
2. Generate natural language queries reflecting KG structure.

Example:

- KG: (Einstein, born_in, Germany) $\Rightarrow$ Where was Einstein born?
- KG: (Einstein, educated_in, Zurich) $\Rightarrow$ Where did Einstein go to school?

Evaluating Context Relevancy with KG

1. Extract triplets from query.
2. Extract triplets from retrieved context.
3. Compare triplets via **embedding similarity**.
    1. Embed each triplet using models (e.g., Sentence-BERT).
    2. Compute cosine similarity.

Example:

- Query: (Marie Curie, discovered, radium)
- Retrieved: (Curie, found, radium) $\Rightarrow$ High similarity

Relevant if one or more context triplets semantically match the query triplet.

Evaluating with Triplet Embedding Similarity

Given a query represented as a set of triplets $Q = \{q_1, \ldots, q_m\}$ and a context $C = \{c_1, \ldots, c_n\}$, we compute similarity:

$$\vec{q}_i = \text{Embed}(q_i) \qquad \text{and} \qquad \vec{c}_j = \text{Embed}(c_j)$$

Cosine similarity: $\text{Sim}(q_i, c_j) = \frac{\vec{q}_i \cdot \vec{c}_j}{\|\vec{q}_i\| \|\vec{c}_j\|}$
Max similarity per query triplet: $S_{max}(q_i) = \max_{1 \le j \le n} \text{Sim}(q_i, c_j)$
Aggregate scores:

- Average: $S_{avg} = \frac{1}{m} \sum_{i=1}^{m} S_{\max}(q_i)$
- Weighted: $S_{wavg} = \sum_{i=1}^{m} w_i \cdot S_{\max}(q_i)$, with $\sum_{i=1}^{m} w_i = 1$
- Minimax: $S_{min} = \min_{1 \le i \le m} S_{max}(q_i)$

Evaluating Groundedness and Completeness

Groundedness: Are LLM answer triplets supported by context?
Completeness: Does the answer cover all relevant context triplets?

- Compare triplets from context and answer.
- Use embedding similarity to find matches.

Given a context represented as a set of triplets $C = \{c_1, \ldots, c_m\}$ and an answer $A = \{a_1, \ldots, a_n\}$, we compute similarity:

$$\vec{c_i} = \text{Embed}(c_i) \qquad \text{and} \qquad \vec{a_j} = \text{Embed}(a_j)$$

Cosine similarity: $\text{Sim}(c_i, a_j) = \frac{\vec{c_i} \cdot \vec{a_j}}{\|\vec{c_i}\| \|\vec{a_j}\|}$

Aggregate Groundedness and Completeness Scores

Triplet Max similarity:

- $S_{max}(a_j) = \max_{1 \leq i \leq m} \mathrm{Sim}(c_i, a_j)$
- $S_{max}(c_i) = \max_{1 \leq j \leq n} \mathrm{Sim}(c_i, a_j)$

Aggregate Groundedness score:

- Average: $S_{avg\_groundedness} = \frac{1}{n} \sum_{j=1}^{n} S_{\max}(a_j)$
- Weighted: $S_{wavg\_groundedness} = \sum_{j=1}^{n} w_j \cdot S_{\max}(a_j)$, with $\sum_{j=1}^{n} w_j = 1$
- Minimax: $S_{min\_groundedness} = \min_{1 \leq j \leq n} S_{max}(a_j)$

Aggregate Completeness score:

- Average: $S_{avg\_completeness} = \frac{1}{n} \sum_{i=1}^{m} S_{\max}(c_i)$
- Weighted: $S_{wavg\_completeness} = \sum_{i=1}^{m} w_i \cdot S_{\max}(c_i)$, with $\sum_{i=1}^{m} w_i = 1$
- Minimax: $S_{min\_completeness} = \min_{1 \leq i \leq m} S_{max}(c_i)$

Evaluating Answer Relevancy with KGs

1. Extract triplets from query and answer.
2. Compare them using embedding similarity.
3. Determine if answer addresses the query's semantic structure.

Example:

- Query: Where was Obama born?" $\Rightarrow$ (Obama, born_in, ?place)
- Answer A: Obama was born in Hawaii." $\Rightarrow$ Relevant
- Answer B: "Obama was the 44th President." $\Rightarrow$ Irrelevant

Given a context represented as a set of triplets $Q = \{q_1, \ldots, q_m\}$ and an answer $A = \{a_1, \ldots, a_n\}$, we compute similarity:

$$\vec{q}_i = \text{Embed}(q_i) \qquad \text{and} \qquad \vec{a}_j = \text{Embed}(a_j)$$

Compute cosine similarity, maximum similarities and aggregate metrics

Conclusion

Knowledge Graphs offer a structured, interpretable approach to:

- Representing document semantics
- Guiding query generation
- Evaluating retrieval relevance
- Verifying groundedness and completeness
- Assessing answer relevancy

KGs + embeddings provide a powerful toolkit for robust evaluation of LLM and RAG systems.