Knowledge Graphs with Large Language Models

Panos Alexopoulos

Oct 23rd, 2024
6 - 9 p.m. Eastern European Summer Time

NLP & Robustness

NLP & Fragility

Connoisseurs of Chinese film will be pleased to discover that Tian's meticulous talent has not withered during his enforced hiatus.	Prediction: **Positive (77%)**
Aonnoisseurs of Chinese film will be pleased to discover that Tian's meticulous talent has not withered during his enforced hiatus.	Prediction: **Negative (52%)**
Connoisseurs of Chinese **footage** will be pleased to discover that Tian's meticulous talent has not withered during his enforced hiatus.	Prediction: **Negative (54%)**

An NLP system is fragile when small changes in its input can produce wrong output

Fragile NLP systems do not generalize well across different data and can perform quite badly in real-world applications.

A system's ability to be resilient to variation in data is called robustness

But how can we assess robustness?

Adversarial Evaluation

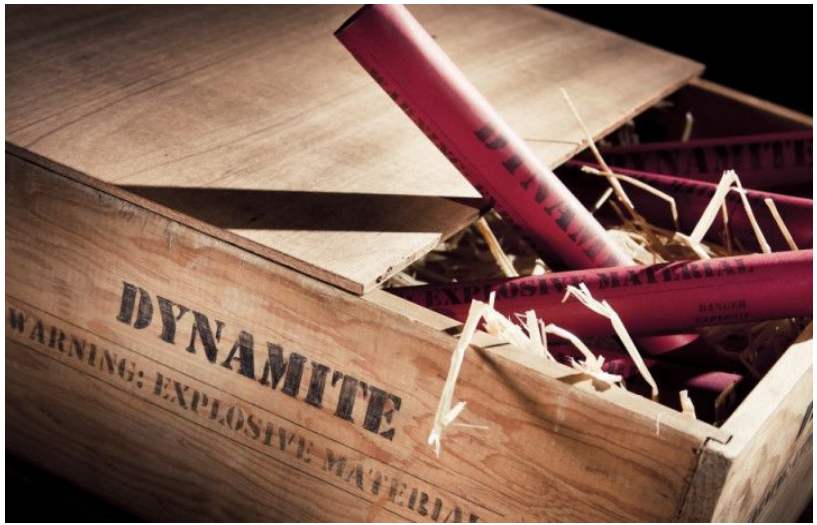
What is it and how it is performed

Typical NLP Evaluation

- NLP systems are typically evaluated on in-distribution data, i.e., data with the same distribution as the training data.
- This assumes that the data reflects the same distribution as the real-world scenarios the system will encounter.
- As a result, NLP systems may struggle when faced with out-of-distribution examples or adversarial inputs that differ significantly from the training data

Adversarial Evaluation

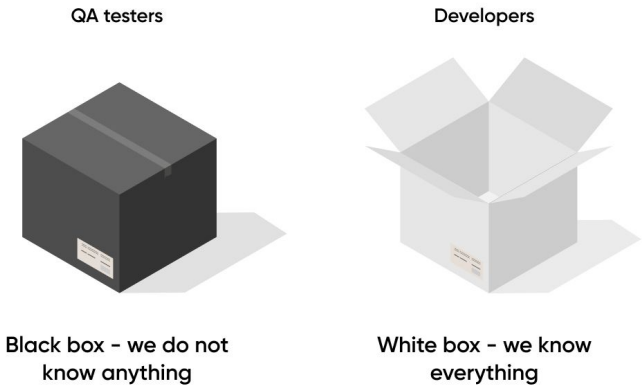
- In an adversarial evaluation, we “attack” the system with data that:
    - Are not necessarily derived from the same distribution as the data the system has been trained on.
    - Contain particular patterns and phenomena that we care to test about and on which we suspect it doesn’t perform well (e.g., negation, synonymy, typos, etc)
- Our goal is to make the system reveal its weaknesses and get a better understanding of its generalizability and robustness



Black-Box vs White-Box Attacks

- **White-box attacks:**
  - The adversary has access to all the information of the target system, including its architecture, parameters, gradients, etc.
  - The adversary can make full use of the model information to carefully craft adversarial examples

- **Black-box attacks:**
  - Adversaries can only feed the input data and query the outputs of the models.
  - They usually attack the models by keeping feeding samples to the box and observing the output to exploit the model's input-output relationship, and identity its weakness.



How to perform an Adversarial Evaluation

Step 1: Define task and target system(s)

Determine the NLP task to evaluate and the relevant systems

Step 2: Specify the adversaries

Determine the adversarial patterns and phenomena you want to test the system(s) on

Step 3: Generate adversarial data

Generate, manually or automatically, data that reflect the adversarial phenomena

Step 4: Apply the adversarial data

Feed the adversarial data to the system(s) under evaluation and get the output

Step 5: Measure adversarial impact

Measure the systems' performance on the adversarial data, along with other metrics that might be useful

Constructing Adversarial Examples

Perturbations

Word-Level Perturbations

Add, remove or replace words to alter the semantics and manipulate the interpretation

Character-Level Perturbations

Introduce small modifications at the character level, such as inserting, deleting, or substituting characters.

Sentence Insertion or Deletion

Insert additional sentences or remove critical or informative sections from the input text to mislead the model

Punctuation Manipulation

Feed the adversarial data to the system(s) under evaluation and get the output

Grammar and Syntax Modifications

Modify the sentence structure or word order, introduce grammatical errors or non-standard language constructions

Contextual Perturbations

Modify the surrounding context, introduce additional context, provide misleading or contradictory information

Examples

Original Text Prediction: **Contradiction** (Confidence = 91%)
Premise: A man and a woman stand in front of a Christmas tree contemplating a single thought. **Hypothesis:** Two **people talk** loudly in front of a cactus.
Adversarial Text Prediction: **Entailment** (Confidence = 51%)
Premise: A man and a woman stand in front of a Christmas tree contemplating a single thought. **Hypothesis:** Two **humans chitchat** loudly in front of a cactus.

Original Text Prediction: **Contradiction** (Confidence = 94%)
Premise: A young girl wearing yellow shorts and a white tank top using a cane pole to fish at a small pond. **Hypothesis:** A girl wearing a **dress** looks off a **cliff**.
Adversarial Text Prediction: **Entailment** (Confidence = 40%)
Premise: A young girl wearing yellow shorts and a white tank top using a cane pole to fish at a small pond. **Hypothesis:** A girl wearing a **skirt** looks off a **ravine**.

Original Text Prediction: **Entailment** (Confidence = 86%)
Premise: A large group of protesters are walking down the street with signs. **Hypothesis:** Some people are holding up **signs** of protest in the street.
Adversarial Text Prediction: **Contradiction** (Confidence = 43%)
Premise: A large group of protesters are walking down the street with signs. **Hypothesis:** Some people are holding up **signals** of protest in the street.

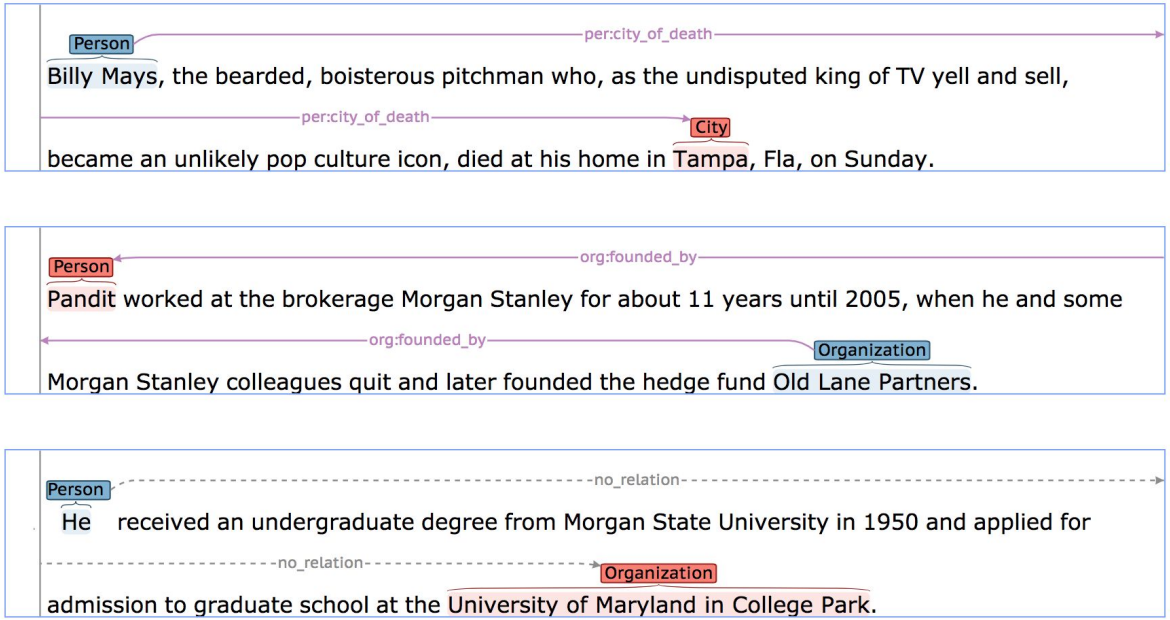
Frameworks and Tools

Case Study

Evaluating two SoA Relation Extraction Models

Relation Extraction and Classification

- **Task definition**: Locate and classify relations between entities mentioned in unstructured text into pre-defined relation categories



Target Systems

LUKE

- Pre-trained on Wikipedia for the relation extraction task
- **72.7 F1 score** in the TACRED benchmark

Ask2Transformers - NLI_DeBERTa

- Relies on a pre-trained textual entailment engine and reformulates relation extraction as an entailment task
- **73.9 F1 score** in the TACRED benchmark

Adversaries

NEGATION

What happens if the input sentence expresses the negation of a relation?

E.g. “Jane does not work for Google”

TENSE CHANGE

What happens if the input sentence expresses future tense version of the relation?

E.g. “Jane will work for Google”

MODALITY CHANGE

What happens if the input sentence expresses the relation in a different modality, such as uncertainty, intention, advice, etc.

E.g. “Jane might work for Google”

Adversaries

DEDUCTIVE PREMISE

What happens if the input sentence asserts a fact that results into the target relation being true

E.g. “Jane accepted a job offer by Google”

INDICATION

What happens if the input sentence sentence asserts a fact that indicates that the target relation might be true ?

E.g. “Jane got a job offer by Google”

PRESUPPOSITION

What happens if the input sentence asserts something that presupposes that the relation holds.

E.g. “Jane resigned from Google”

Adversarial Data

- We start with a "simple" set of sentences for each TACRED relation, with the pattern <subject entity, relation, object entity>, using a verb phrase lexicalization of the relation.
  - E.g. “Jane works for Google”
  - E.g. “John is married to Jane”

- We define and apply linguistic transformation templates to produce sentence variations for each adversary type.

Adversarial Data

- We generated a "simple" set of sentences for each TACRED relation, with the pattern <subject entity, relation, object entity>, using a verb phrase lexicalization of the relation.
- Then we applied linguistic transformation templates to produce sentence variations for each adversary type.

sentence	subject	object	reference_relation	adversary_type	adversary_subtype	is_relation_in_sentence
John works at Google	John	Google	employee_of	None	None	1
John is employed by Google	John	Google	employee_of	None	None	1
John does not work at Google	John	Google	employee_of	negation	None	0
John is not employed by Google	John	Google	employee_of	negation	None	0
John worked at Google	John	Google	employee_of	tense	past_simple	0
John was employed by Google	John	Google	employee_of	tense	past_simple	0
John will work at Google	John	Google	employee_of	tense	simple_future	0
John will be employed by Google	John	Google	employee_of	tense	simple_future	0
John has been working at Google	John	Google	employee_of	tense	present_perfect_continuous	1
John has been employed by Google	John	Google	employee_of	tense	present_perfect_continuous	1
John might work at Google	John	Google	employee_of	modality	uncertainty	0
John might be employed by Google	John	Google	employee_of	modality	uncertainty	0
John probably works at Google	John	Google	employee_of	modality	uncertainty	0
John is probably employed at Google	John	Google	employee_of	modality	uncertainty	0

Adversarial Impact

LUKE

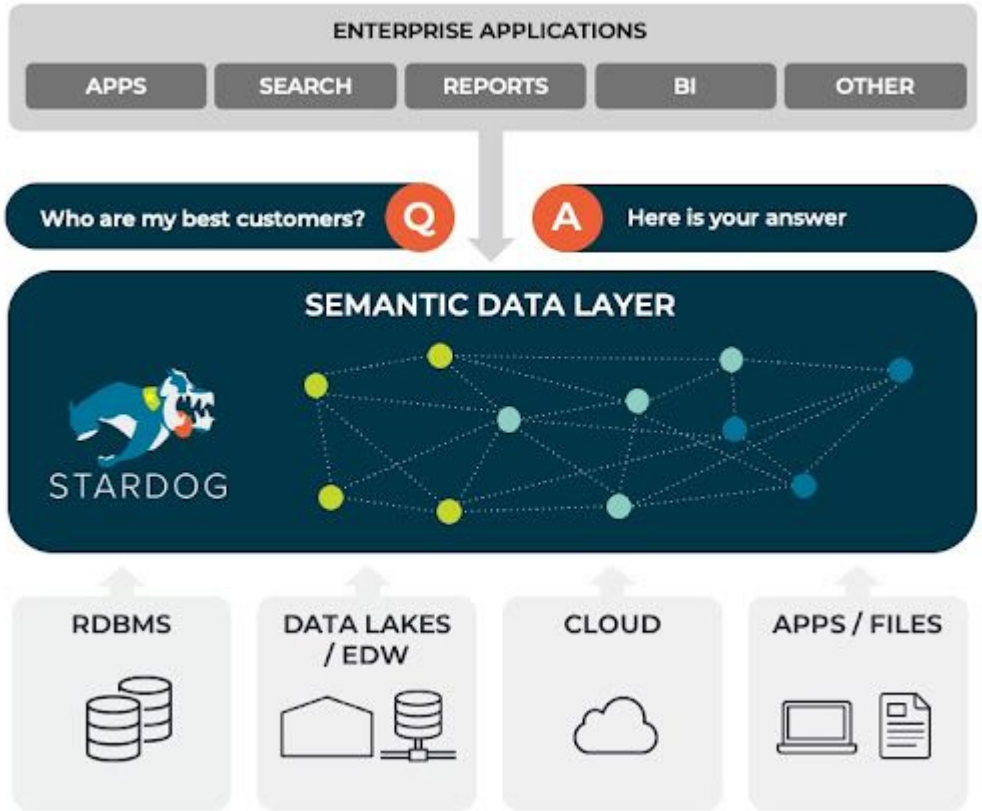
- Fooled by most negations (~80%)
- Successful in all positive tense variations but fooled by all negative ones.
- Fooled by all modality variations

Ask2Transformers - NLI_DeBERTa

- Successful in all negations
- Successful in all positive tense variations but fooled by 80% of the negative ones
- Fooled by 50% of modality variations

Make it useful

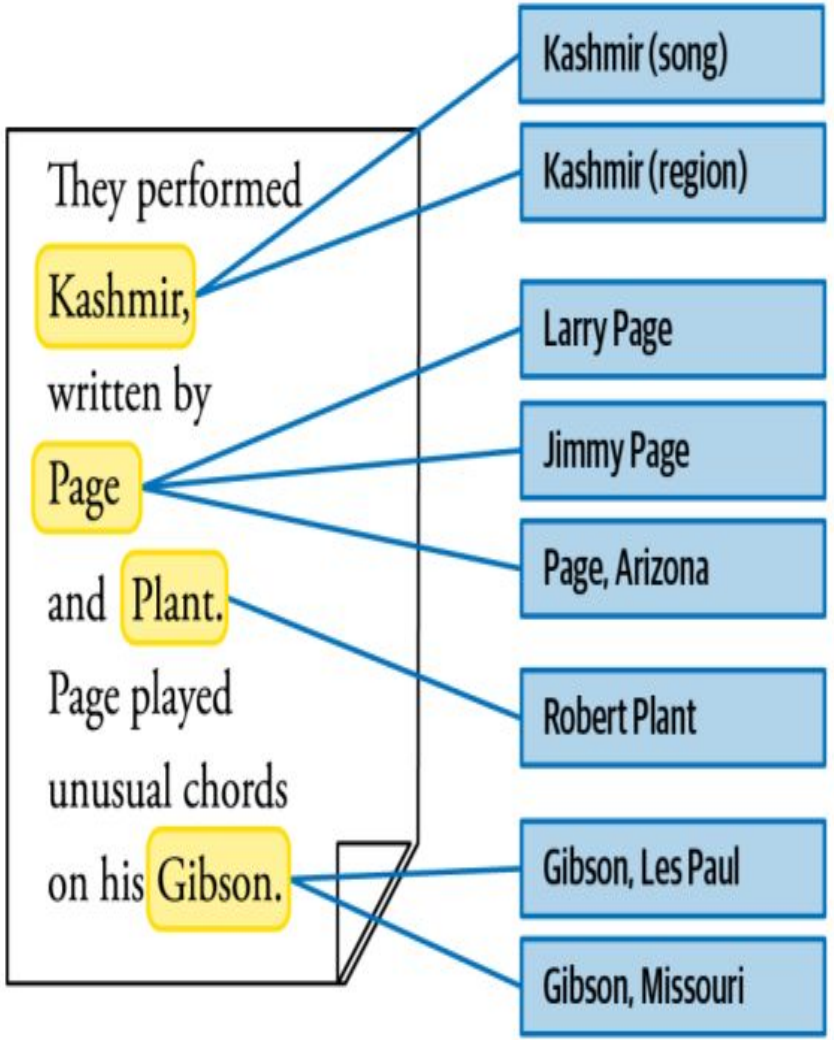
Semantic Data Layer



Graph Analytics Algorithms

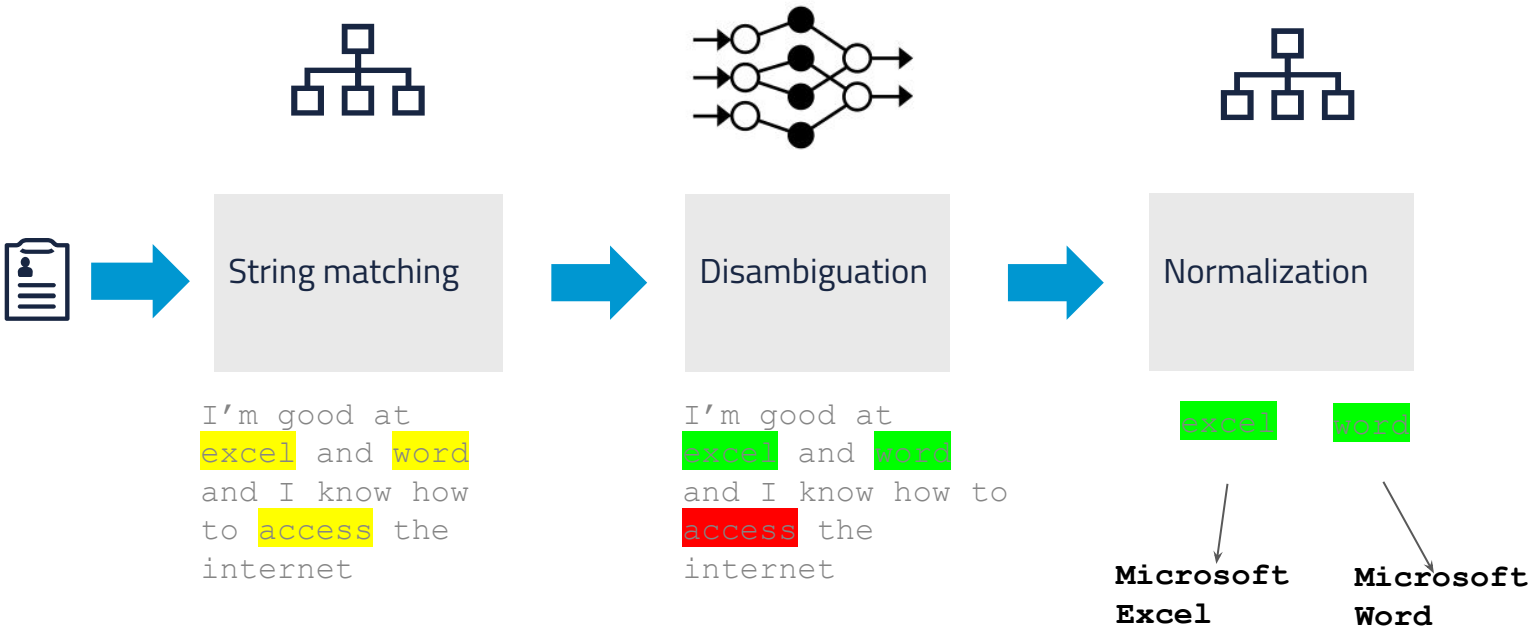
Algorithm Type	Graph Problem	Examples
 **Pathfinding & Search**	Find the optimal path or evaluate route availability and quality	• Find the quickest route to travel from A to B • Telephone call routing
 **Centrality**	Determine the importance of distinct nodes in the networks	• Determine social media influencers • Find likely attack targets in communication and transportation networks
 **Community Detection**	Evaluate how a group is clustered or partitioned	• Segment customers • Find potential members of a fraud ring

Entity Linking



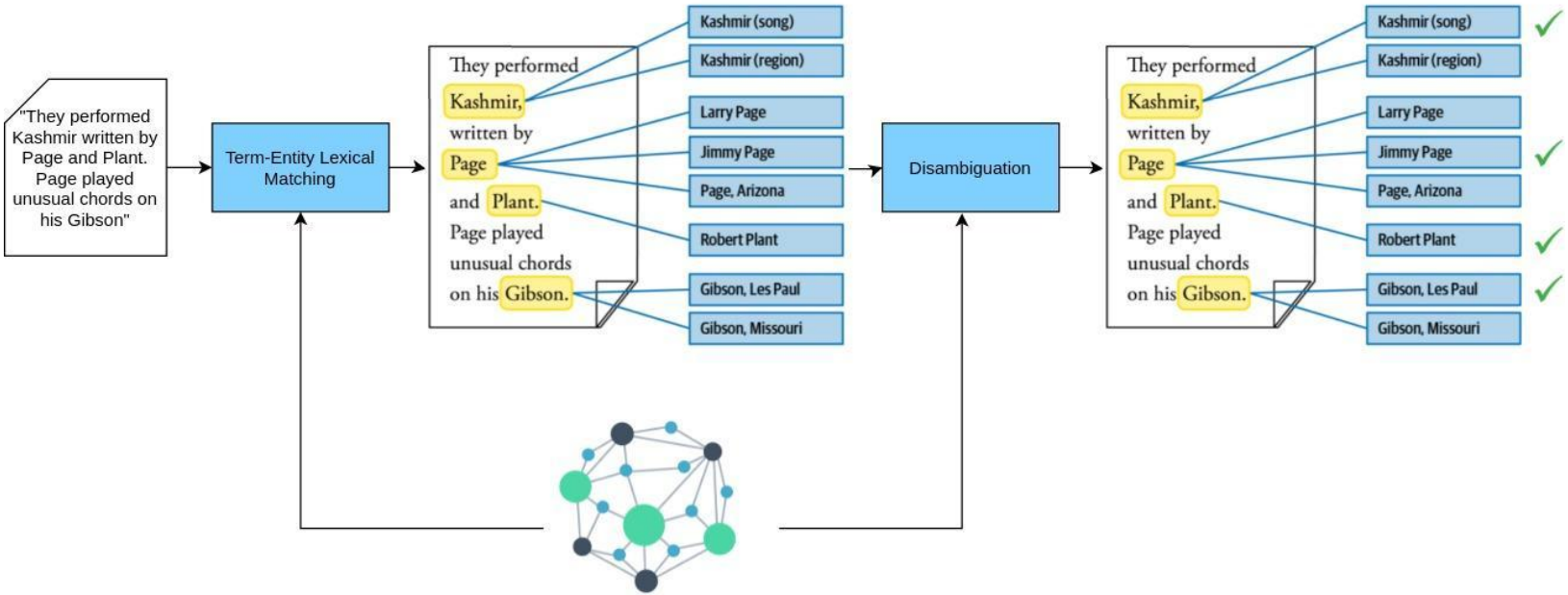
Entity Linking in Bloomberg

Entity Linking at Textkernel



In what ways does a KG enable/help entity linking?

KG-Based Entity Linking



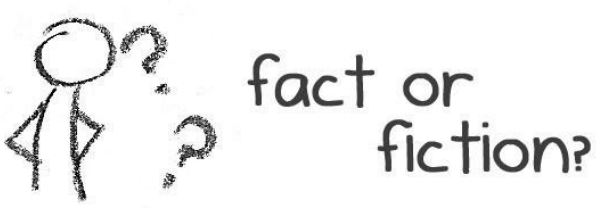
How would you build an KG-based entity linker with an LLM?

Grounding LLMs with Knowledge Graphs

The hallucination problem

- LLMs are designed to generate human-like text based on the patterns they’ve identified in vast amounts of data.

- Because of that they sometimes hallucinate or produce outputs that manifest as generating untrue facts, asserting details with unwarranted confidence, or crafting plausible ye nonsensical explanations.



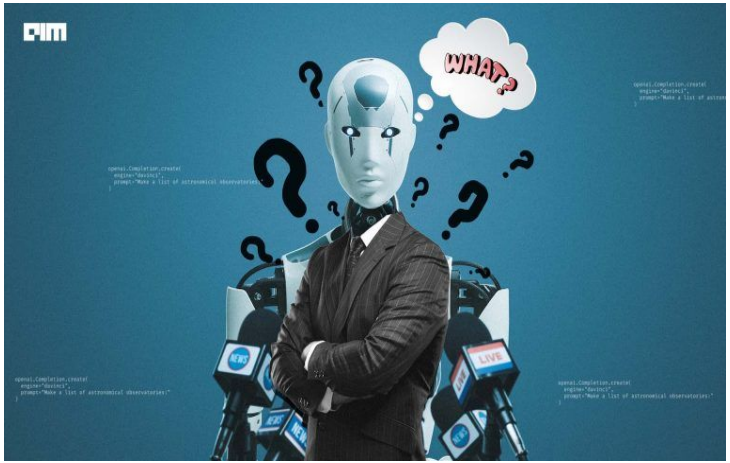
- These manifestations arise from a mix of *overfitting*, biases in the training data, and the model’s attempt to generalize from vast amounts of information

The knowledge cutoff problem

- The training process for LLMs is intricate and time-intensive, often requiring vast datasets compiled over extended periods.

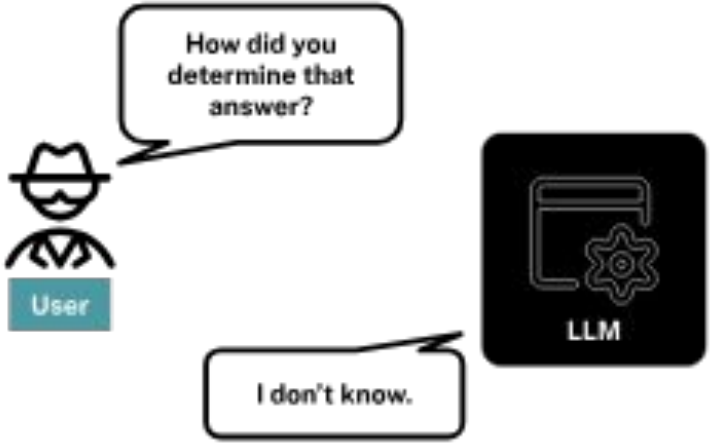
- LLMs are unaware of any events that happened after their training (knowledge cutoff) or are not present in their training dataset.

- LLMs don’t have any knowledge about private or confidential information that might be available even before the knowledge cutoff date.

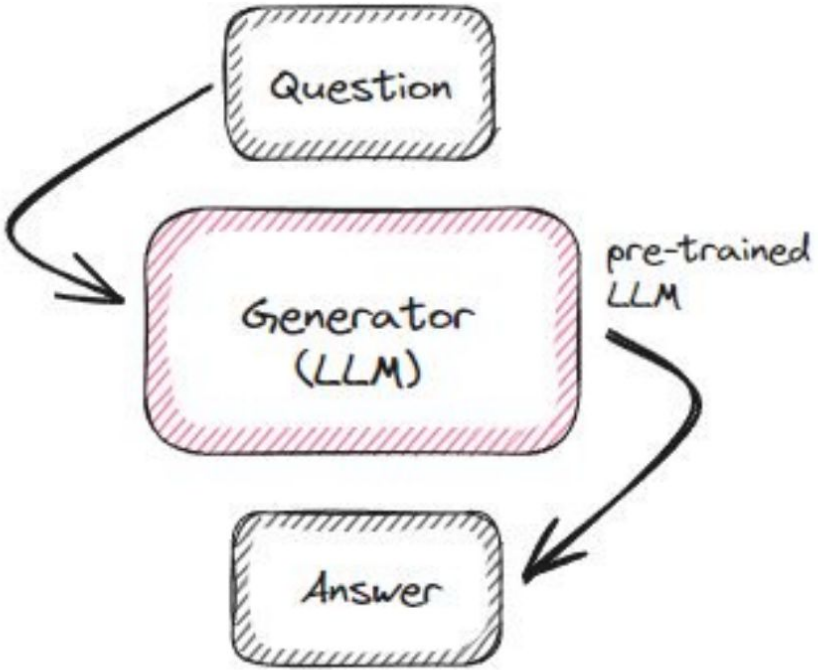


The explanation problem

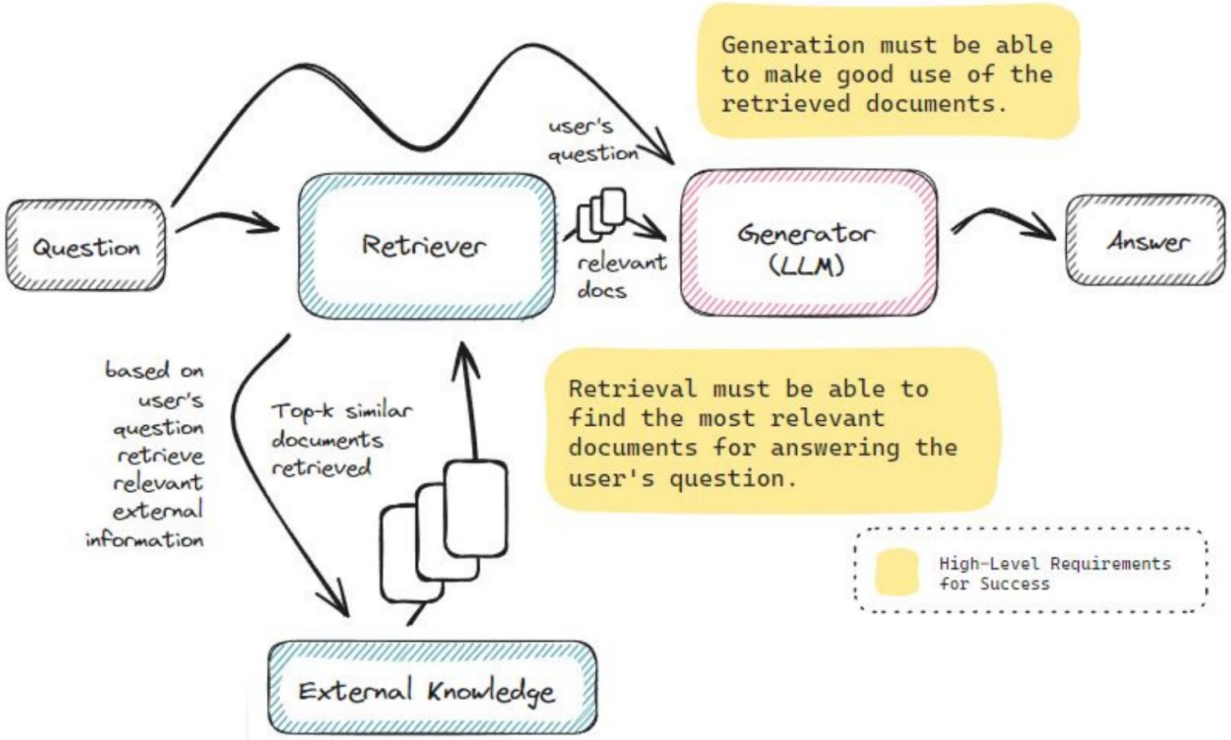
- LLMs are often considered "black boxes" due to the difficulty deciphering their decision-making processes.
- Their complexity means that their outputs can sometimes be unpredictable or inaccurate.
- It can be very hard to trace back how the model arrived at a given response or have the LLM provide the sources for its output or explain its reasoning.



LLM Answer Generation



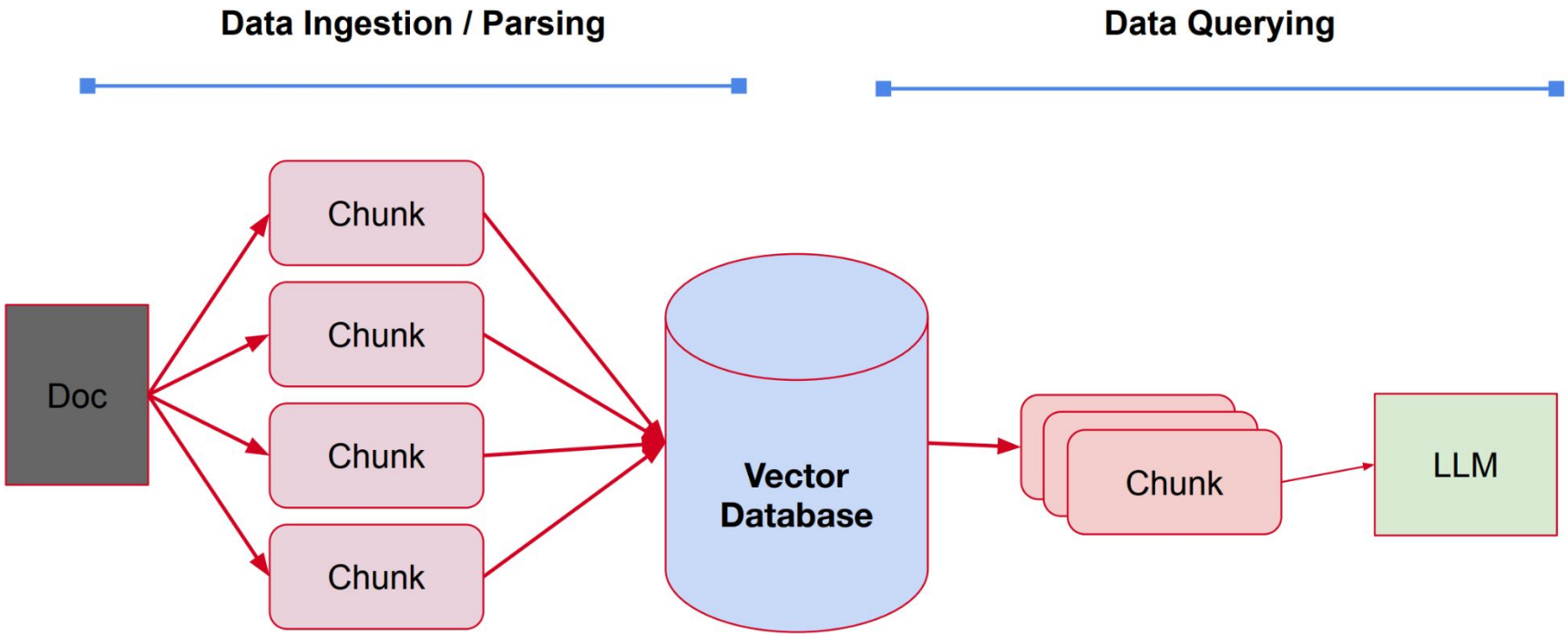
Retrieval Augmented Generation (RAG)



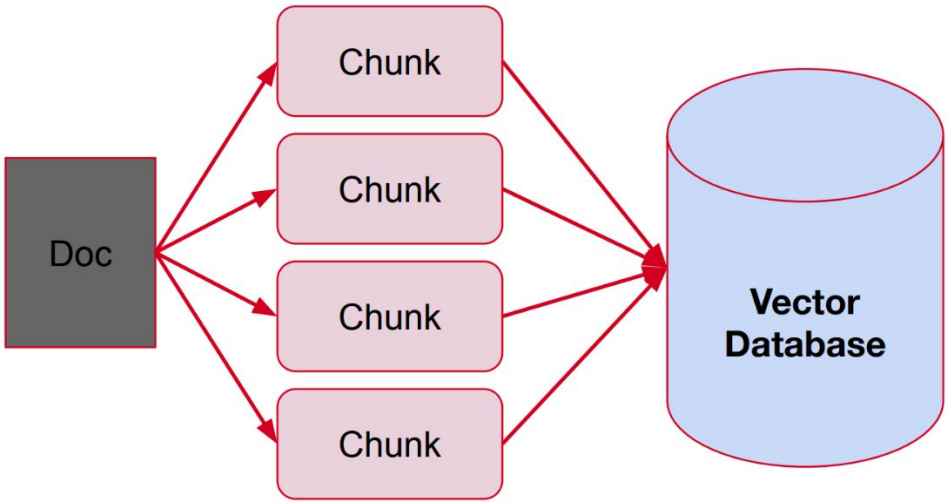
RAG System Components

- **An indexer:** A mechanism to compress data into vectors and stored in a database
- **A retriever**: Closely tied to the indexer, something to retrieve data from that database given a query.
- **A generator:** An LLM to reason through the user's query and the retrieved knowledge to provide a response.

Retrieval Augmented Generation (RAG)



Retrieval Augmented Generation (RAG)



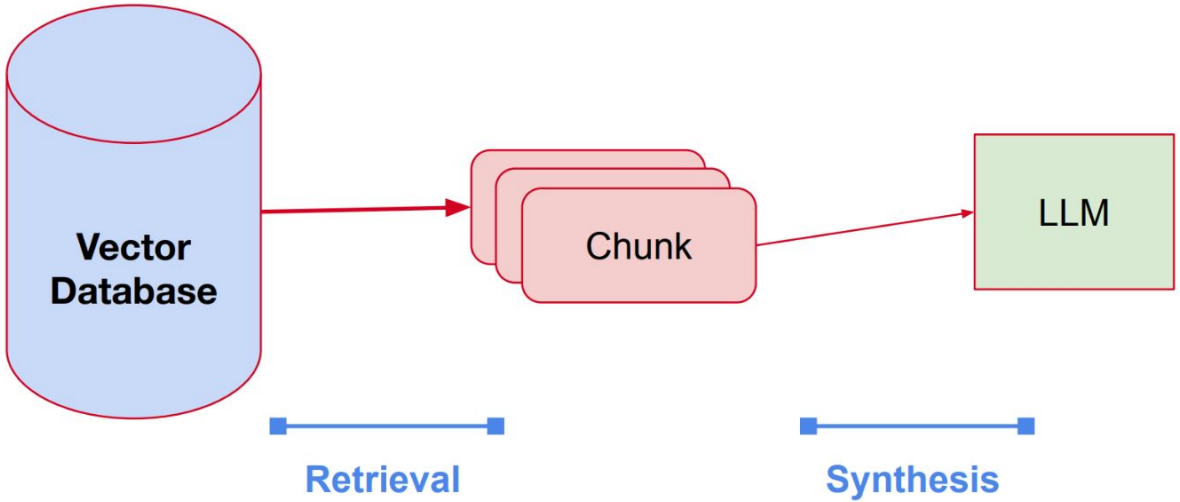
Process:

- Split up document(s) into even chunks.
- Each chunk is a piece of raw text.
- Generate embedding for each chunk (e.g. OpenAI embeddings)
- Store each chunk into a vector database

Retrieval Augmented Generation (RAG)

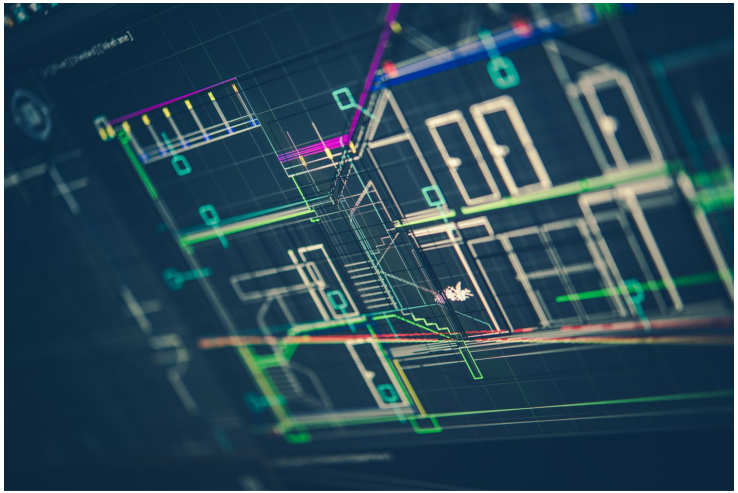
Process:

- Find top-k most similar chunks from vector database collection
- Plug into LLM **response synthesis module**

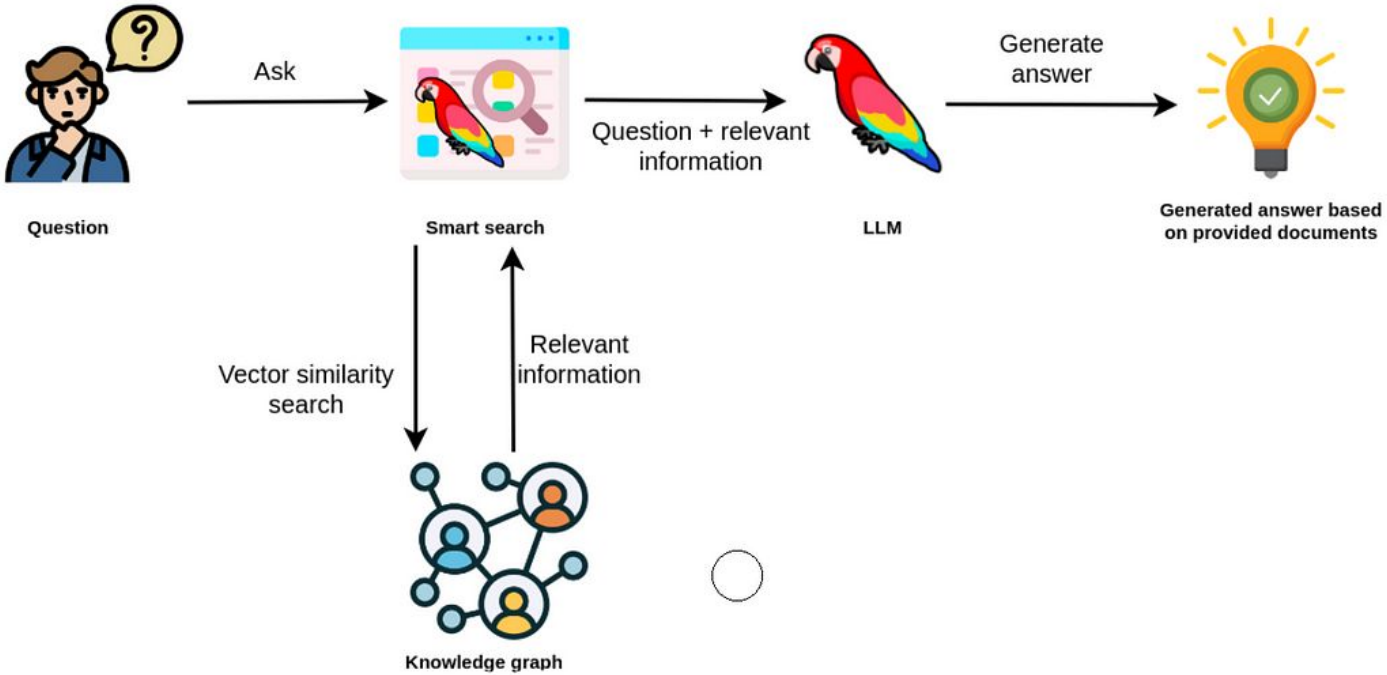


Challenges with Basic RAG

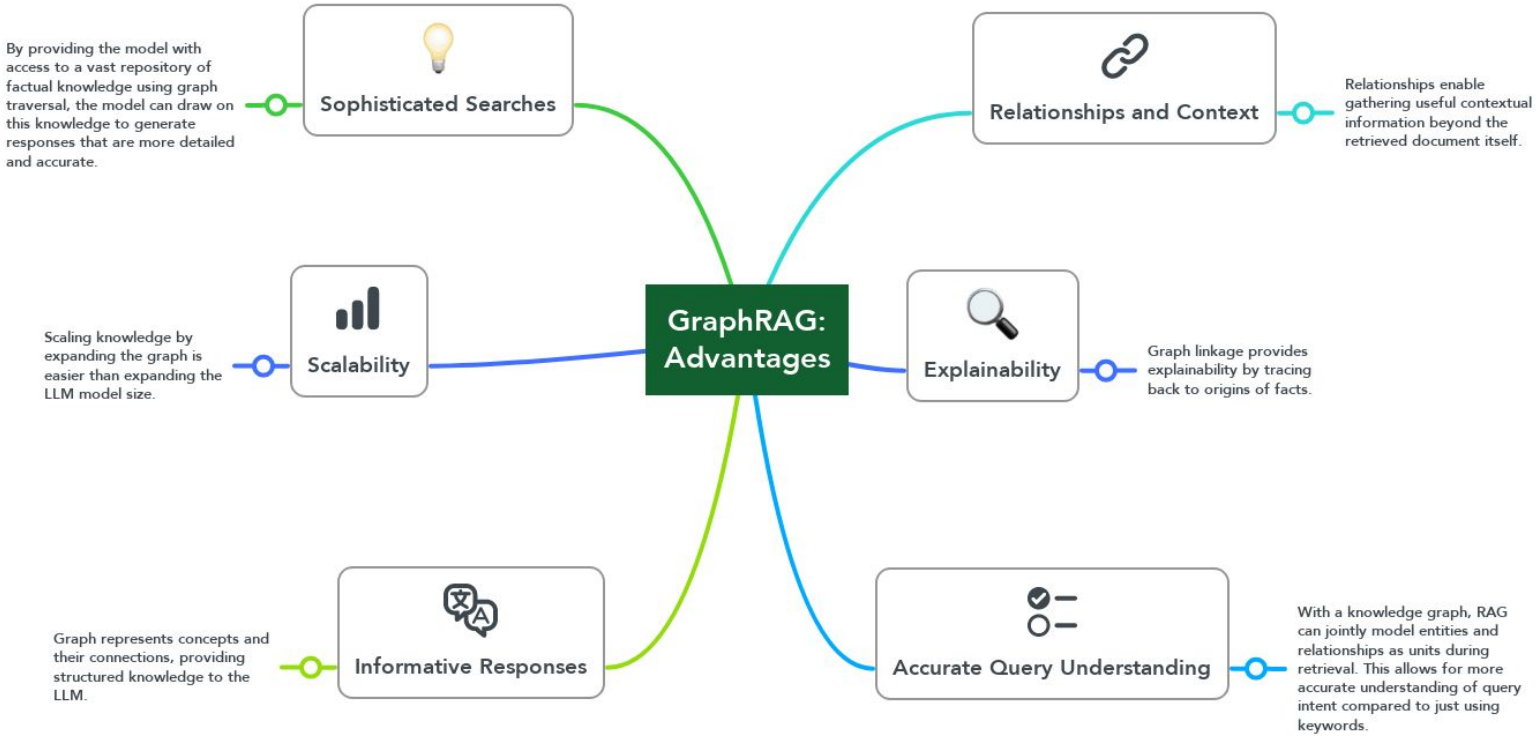
- **Poor Retrieval**
  - Low Precision: Not all chunks in retrieved set are relevant
    - Hallucination + Lost in the Middle Problems
  - Low Recall: Not all relevant chunks are retrieved.
    - Lacks enough context for LLM to synthesize an answer
- **Poor Response Generation**
  - Hallucination: Model makes up an answer that isn't in the context.
  - Toxicity/Bias: Model makes up an answer that's harmful/offensive.



RAG with a Knowledge Graph



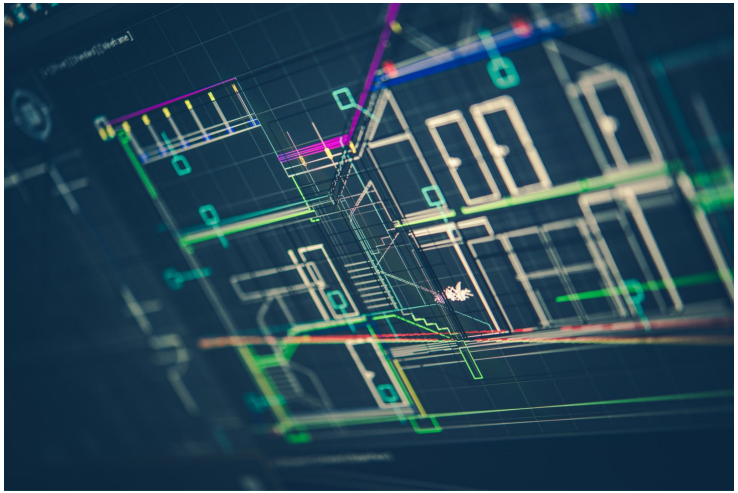
RAG with a Knowledge Graph



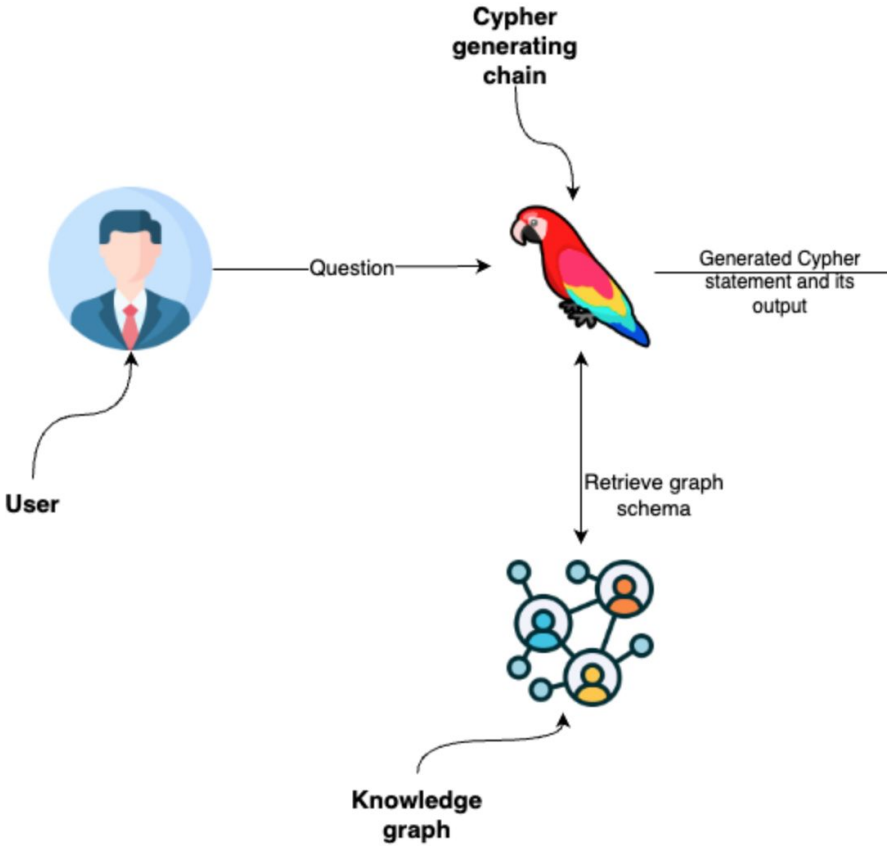
Implementing GraphRAG

- There is no standardized approach for integrating knowledge graphs into the RAG pipeline

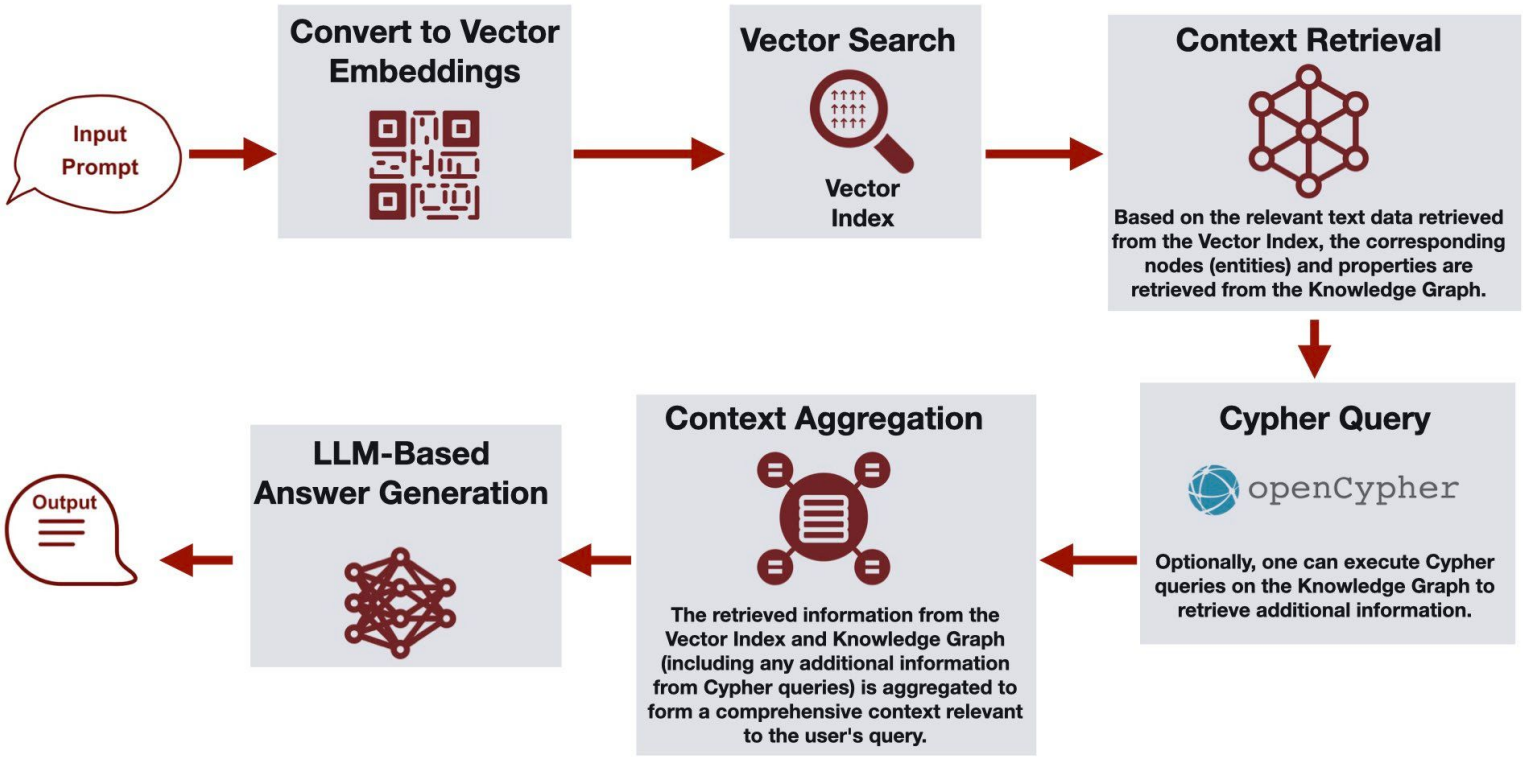
- There are, though, some few common GraphRAG architectures



Question answering over Knowledge Graphs



Knowledge Graph-Enhanced Question Answering Pipeline



Knowledge Graph-Based Query Augmentation and Generation

Input Prompt

Interpretation and Entity Recognition

The LLM interprets the prompt and identifies relevant entities and relationships within the query.

Context Augmentation

The system augments the user's query with additional context to enhance the retrieval process.

Vector Search on Knowledge Graph

A vector search is performed on the knowledge graph to find nodes and edges that are semantically similar to the augmented query.

Cypher Query Generation

openCypher

Based on the vector search results, the system generates Cypher queries to retrieve structured information from the knowledge graph.

Graph Traversal & Search

The generated Cypher queries are executed to traverse the knowledge graph and retrieve relevant data

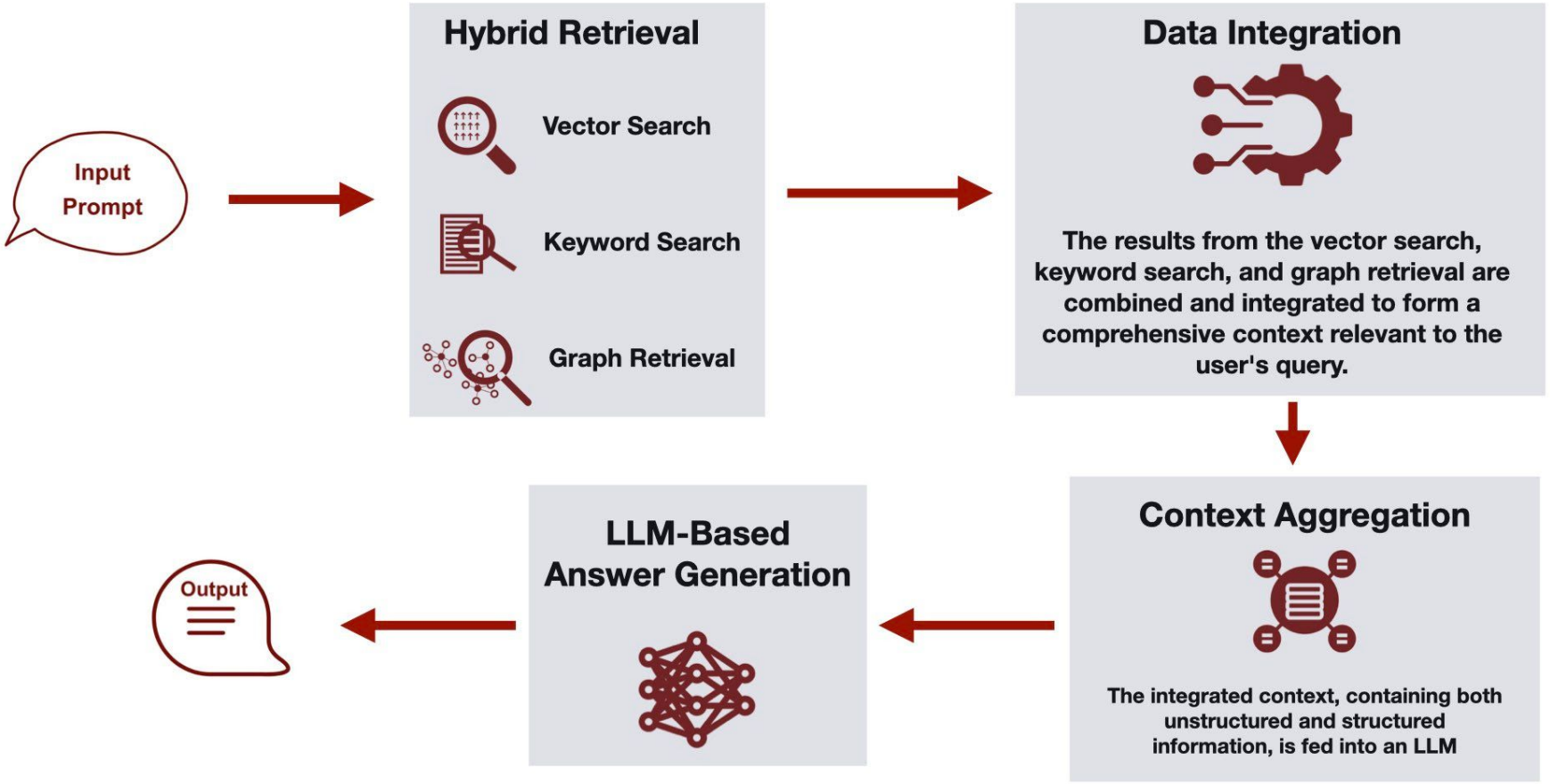
Context Enrichment

The retrieved information from the knowledge graph is used to enrich the context window of the LLM.

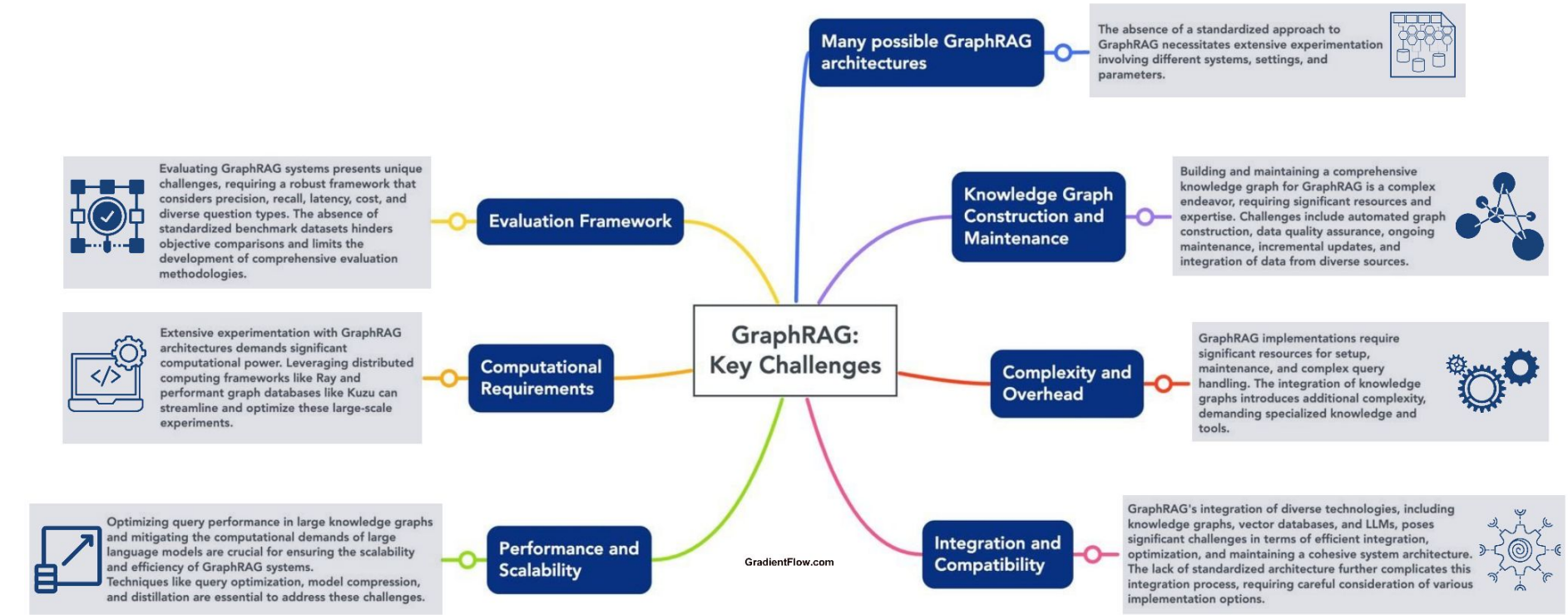
LLM-Based Answer Generation

Output

Graph-Enhanced Hybrid Retrieval



Graph RAG Challenges



Grounding LLMs with Knowledge Graphs

Hands-On