CS60092: Information Retrieval

Relation Extraction

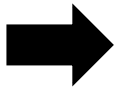
Debaditya Roy

Relation Extraction

IE requires structured predictions across an entire document

- multiple entities
- relations between entities
- events with arguments
- cross-sentence reasoning

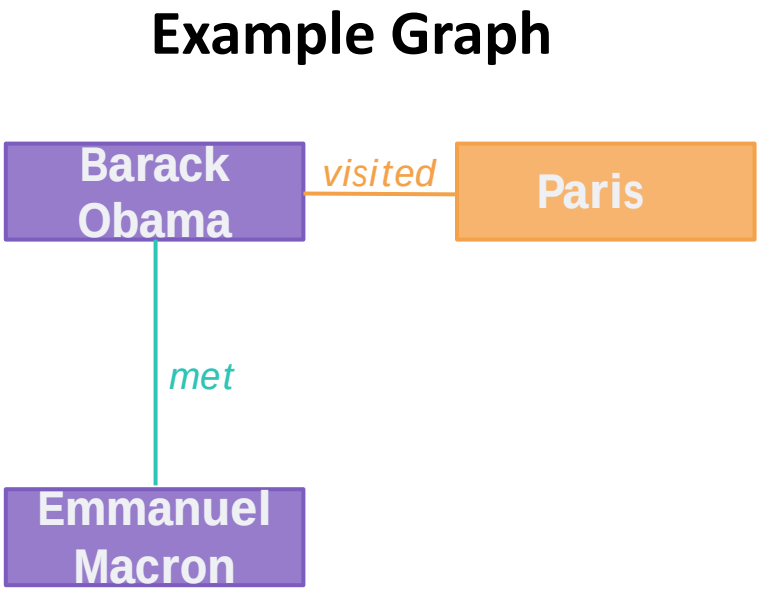
"Google acquired DeepMind in 2014 for $500 million."



```
Entity:
Google (Organization)
DeepMind (Organization)

Relation:
acquired(Google, DeepMind)

Event:
Acquisition
   Acquirer: Google
   Target: DeepMind
   Time: 2014
   Price: $500M
```

Relation Extraction

- BiLSTM-CRF works well for token-level labeling
  - Cannot naturally model the global structure



- DyGIE++ (Dynamic Graph IE) is a graph-based neural architecture for document-level IE

Nodes = entity spans
Edges = relations & event args

DyGIE++ : Span Graph Information Extraction

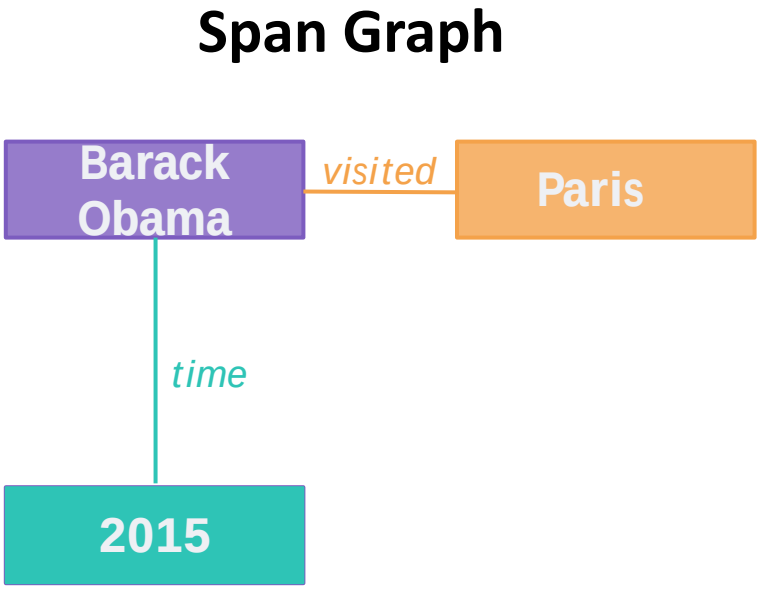
- DyGIE++ treats **text spans as nodes in a graph**
- **Span** is a contiguous sequence of tokens

Example sentence:

Barack Obama visited Paris in 2015

Candidate spans:

- Barack Obama
- Paris
- 2015
- visited



Span Representation

- DyGIE++ first encodes the document with a contextual encoder BiLSTM / BERT / Transformer
- Token embeddings

$$h_1, h_2, \ldots, h_n$$

- Span i: j representation

$$g_{\{i,j\}} = [h_i ; h_j ; \phi(i,j)]$$

  - h_i = start token representation
  - h_j = end token representation
  - $\phi(i,j)$ = span width embedding

Captures

- boundary tokens
- span length

Span Classification (Entity Detection)

- Each span is classified as an entity type.

$$P(type \mid span) = softmax\left(W_{g_{i,j}}\right)$$

Span	Label
Barack Obama	PERSON
Paris	LOCATION
2015	DATE

- Replaces NER tagging from BiLSTM-CRF
- No tagging tokens **B-PER I-PER**, directly predicts entity spans

Relation Extraction Between Spans

- For two spans s_i and s_j
- Relation score $score(r_{ij}) = g_i^\top W_r g_j$
- Probability

$$P(r \mid s_i, s_j) = softmax(score)$$

- **Example:** visited(Obama, Paris)

Event Extraction in DyGIE++

- DyGIE++ also extracts events.
- Example sentence:

  `Amazon acquired Whole Foods in 2017`

- Event structure

```
Trigger: acquired
Arguments:
  Buyer → Amazon
  Target → Whole Foods
  Time → 2017
```

Event Extraction: Mathematical Model

- t = trigger, a_i = argument

Event extraction

$$P(event, a_1, a_2, \ldots, a_k \mid span)$$

Subtasks

1. Trigger detection

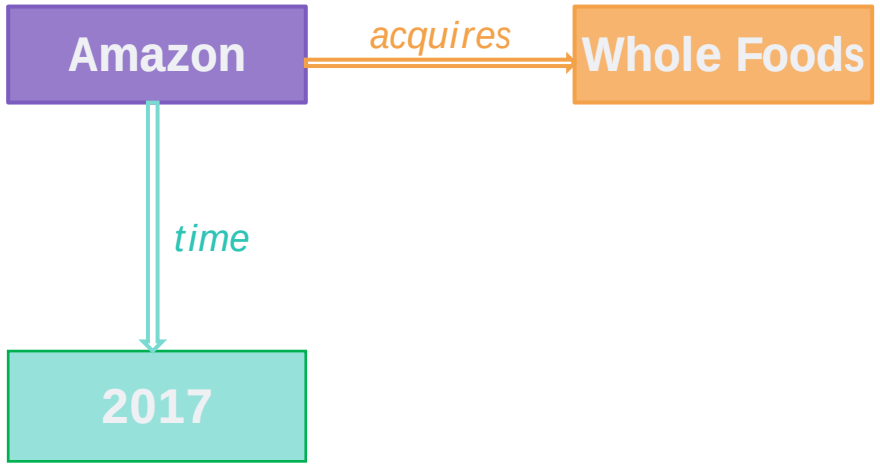
$$P(trigger_i \mid span)$$

1. Argument classification

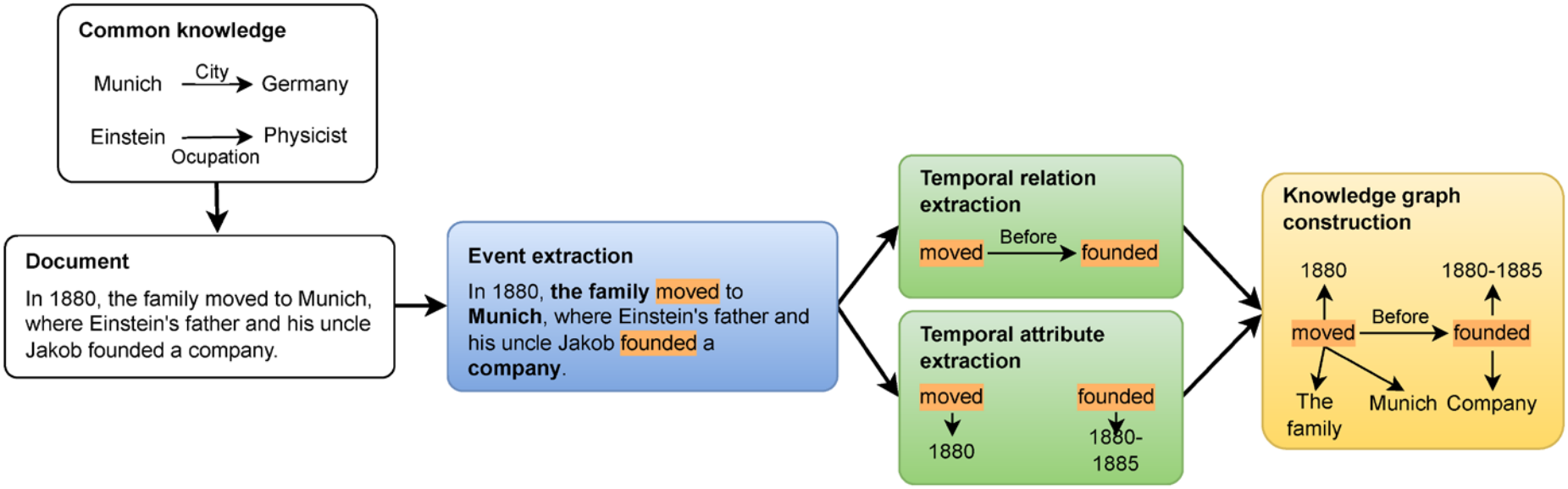
$$P(role(a_j) \mid trigger)$$

Dynamic Graph Construction

- DyGIE++ constructs a graph dynamically during inference
- Nodes
  - entity spans
  - event triggers
- Edges
  - relations
  - event arguments



Temporal Information Extraction



Some temporal relations:

- BEFORE
- AFTER
- OVERLAPS
- INCLUDES

Knez, T., & Žitnik, S. (2023). Event-centric temporal knowledge graph construction: A survey. *Mathematics*, *11*(23), 4852.

Graph Propagation (Message Passing)

- Key innovation of DyGIE++ is propagation over the graph
- Each node updates its representation using neighbors

$$h_i^{(k+1)} = \sigma\left(W h_i^{(k)} + \sum_{j \in N} W_r h_j^{(k)}\right)$$

- Update rule
  - $N(i)$ = neighbors
  - k = propagation step
- entity information → relation prediction
- relation info → event prediction

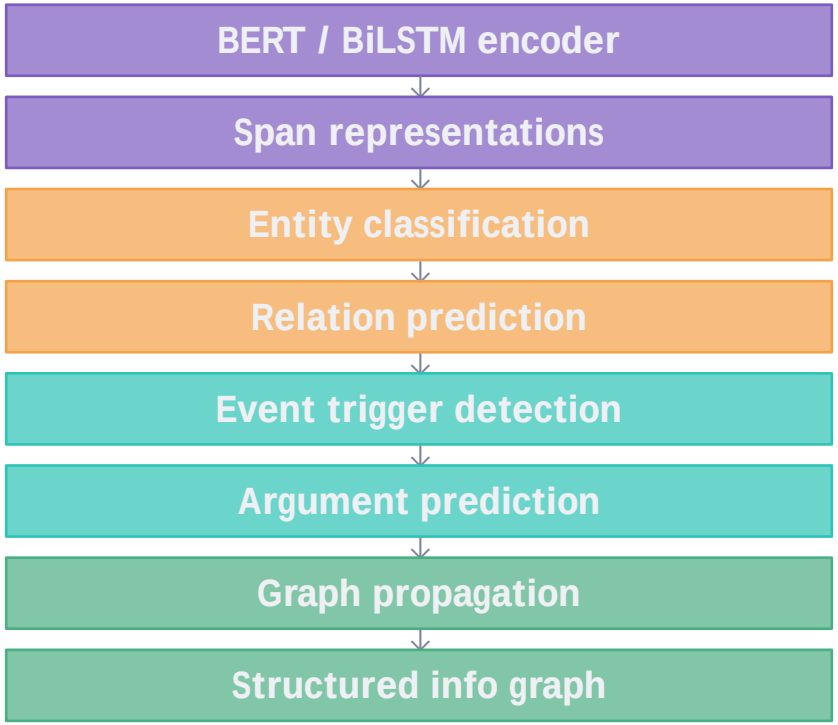
Multi-task Learning

- DyGIE++ jointly predicts
  - entity spans
  - Relations
  - event triggers
  - event arguments
- Loss

$$L = L_{entity} + L_{relation} + L_{trigger} + L_{argument}$$

- Shared learning improves accuracy.

DyGIE++ pipeline



Open Information Extraction (OpenIE)

No predefined schema — extracts relation phrases from text

Example: "Tesla acquired SolarCity in 2016"

```
(Tesla, acquired, SolarCity)
(Tesla, acquired SolarCity in, 2016)
```

Pipeline

1. Dependency parsing
2. Pattern detection
3. Relation phrase extraction
4. Triple generation

Strengths

- ✓ No schema required
- ✓ Works on web-scale text
- ✓ Useful for large knowledge graphs

Weaknesses

- ✕ Inconsistent relation phrases
- ✕ Duplicates and noise
- ✕ Hard to normalize — all of these mean the same thing:

```
acquired / bought /
purchased / took over
```

LLM-based Information Extraction

Prompt the model instead of training a classifier

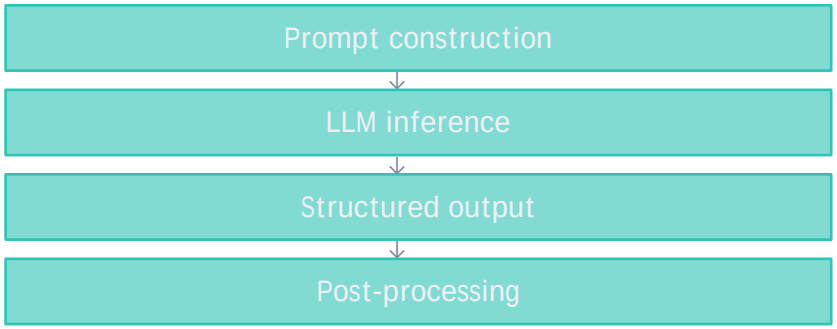
Example Prompt

```
Extract entities & relations.
Schema: Person, Org → works_for

Text: "Sam Altman is CEO of OpenAI."

Return JSON.
```

Model Output

```
{ "entities": ["Sam Altman","OpenAI"],
  "relations": [{
    "type":"works_for",
    "head":"Sam Altman",
    "tail":"OpenAI" }] }
```

IE Pipeline



Strengths

✓ Extremely flexible

✓ Zero-shot extraction

✓ Adapts to new schemas instantly

Weaknesses

✕ Hallucinations

✕ Inconsistent outputs

Comparison through an example

"Google acquired DeepMind in 2014."

DyGIE++

```
Entity: Google
Entity: DeepMind
Entity: 2014

Event: Acquisition
  buyer  → Google
  target → DeepMind
  time   → 2014
```

Structured & normalised

OpenIE

```
(Google, acquired,
DeepMind)

(Google, acquired
 DeepMind in, 2014)
```

Less structured

LLM-based IE

```
{
"event":"acquisition",
  "buyer":"Google",
  "target":"DeepMind",
  "year":2014 }
```

Schema depends on prompt

Modern Hybrid Pipeline

State-of-the-art systems combine all three approaches

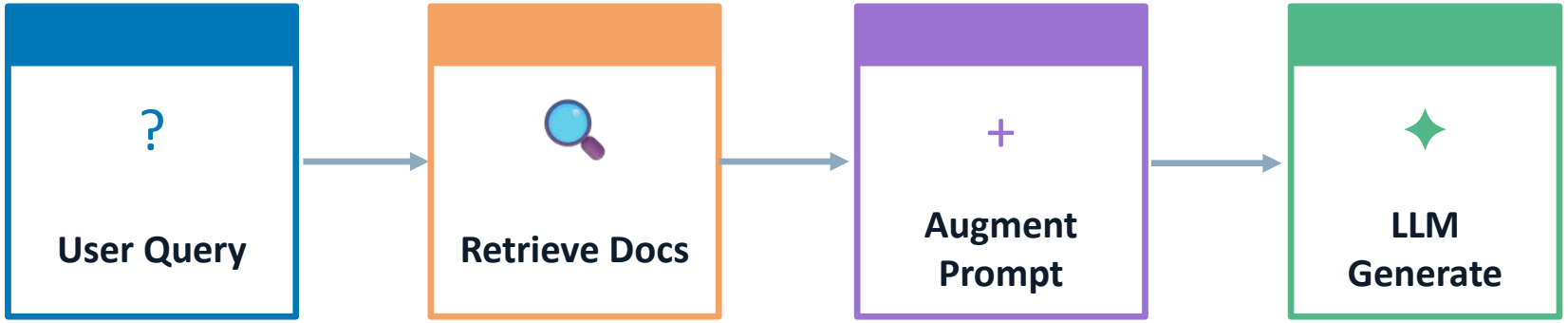
Graph RAG

Retrieval-Augmented Generation

Powered by Knowledge Graphs

What is RAG?

Retrieval-Augmented Generation = Search + Generate



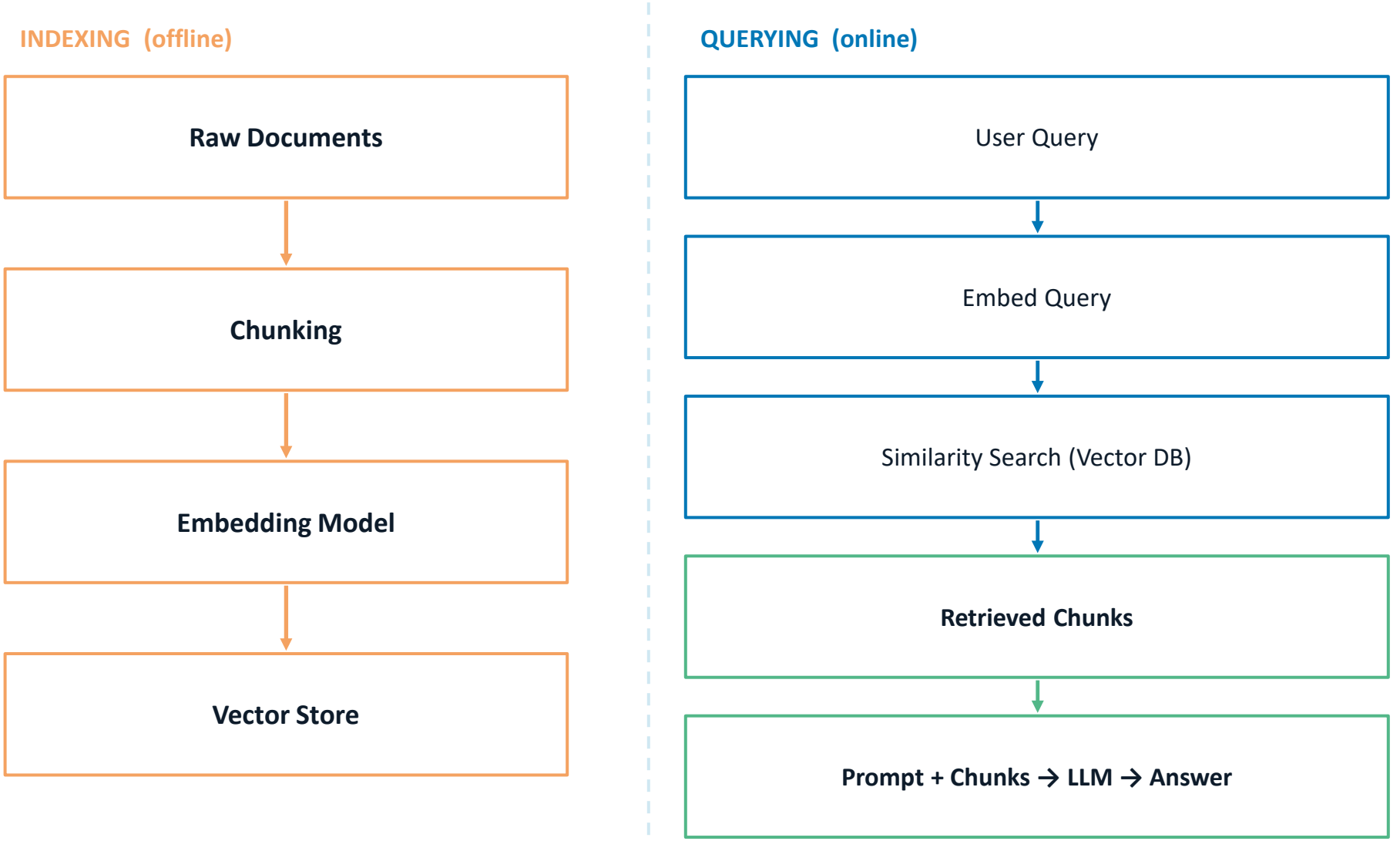
Why RAG? LLMs have a knowledge cutoff and limited context. RAG grounds responses in your documents — **reducing hallucinations** and enabling up-to-date, source-backed answers.

Standard RAG — Formal Definition

$$P(answer \mid query, D) = \boldsymbol{P}_{LLM}(answer \mid query, \boldsymbol{TopK}(Embed(query), D))$$

where D is the document corpus and Top-K retrieves the K most similar chunks by vector similarity.

Standard RAG — Indexing & Retrieval Pipeline



Limitation 1 of 4 — Flat, Isolated Chunks

Documents are split into fixed-size chunks with no awareness of how entities or concepts relate across the corpus. Each chunk is treated as an island.

What RAG sees:

Chunk A

Marie Curie won the Nobel Prize in Physics in 1903.

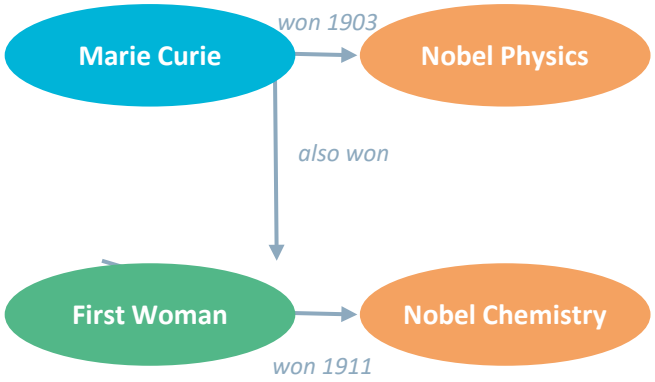
Chunk B

The Nobel Prize in Chemistry was awarded in 1911.

Chunk C

She was the first woman to win a Nobel Prize.

What actually exists:

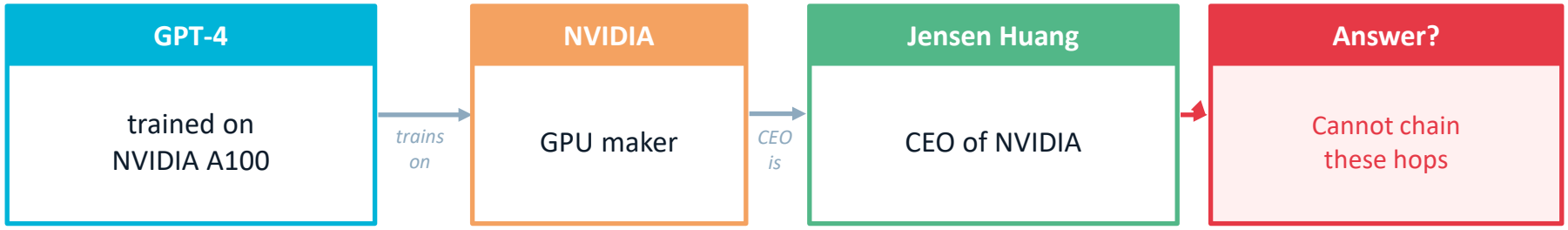


Example query: *"How many Nobel Prizes did Marie Curie win?"* -> RAG retrieves Chunk A or C separately, never seeing the full picture.

Limitation 2 of 4 — No Relational Reasoning

Standard RAG cannot answer multi-hop questions. Each retrieved chunk is scored independently with no mechanism to chain facts across documents.

Query: *"Who is the CEO of the company that makes the GPU used to train GPT-4?"*



What standard RAG does instead:

Chunk 1 (sim: 0.81)

"...GPT-4 was trained using NVIDIA H100 GPUs..."

Chunk 2 (sim: 0.76)

"...NVIDIA produces leading AI accelerators..."

Chunk 3 (sim: 0.71)

"...Jensen Huang is a tech entrepreneur..."

Unconnected chunks passed to LLM. The link GPT-4 → NVIDIA → Jensen Huang is never assembled.

Limitation 3 of 4 — Context & Metadata Loss

Embedding flattens rich structured documents into a single dense vector. Provenance, authorship, date, document hierarchy, and cross-document links are all discarded.

Original Document — rich structure

SEC Filing — Apple Inc. — Q3 2024

Author:	Apple Legal Dept.
Date:	July 30, 2024
Section:	Risk Factors > Competition
References:	SEC Rule 10-K, Exhibit 32
Sentiment:	Cautionary / Forward-looking
Topic chain:	Supply chain -> Revenue -> China ops
Hierarchy:	Part II > Item 1A > Sub-section 3

embed

After Embedding — all context gone

1536-dim Vector

```
[ 0.0823, -0.1142, 0.3401, 0.0092,
  -0.2280, 0.1567, -0.0443, 0.3892,
   0.1120, -0.0671, 0.4102, ... ]
```

Author: X

Date: X

Section: X

References: X

Sentiment: X

Topic chain: X

Hierarchy: X

Limitation 4 of 4 — Retrieval Redundancy & Noise

- Top-K vector search returns the K most similar chunks, but similar does not equal useful.
- Results are often overlapping, repeated, or off-topic, wasting the LLM's context window.

Query: *"What are the side effects of Ibuprofen?"*

K=1 sim:0.91	USEFUL	Ibuprofen may cause stomach pain, heartburn, nausea, and dizziness. Rare effects include kidney issues.
K=2 sim:0.89	REDUNDANT	Common side effects of Ibuprofen: nausea, upset stomach, heartburn. Seek help if serious effects occur.
K=3 sim:0.87	REDUNDANT	Patients taking Ibuprofen reported gastrointestinal issues including nausea and stomach discomfort.
K=4 sim:0.83	OFF-TOPIC	Ibuprofen dosage for adults is 200-400mg. It was first synthesized by Stewart Adams in 1961.
K=5 sim:0.80	NOISE	NSAIDs such as aspirin, naproxen, and ibuprofen are widely used. Drug interactions vary.

3 of 5 retrieved chunks are redundant or off-topic, consuming 60% of the context window with low-value content.

Graph-Based Information Extraction

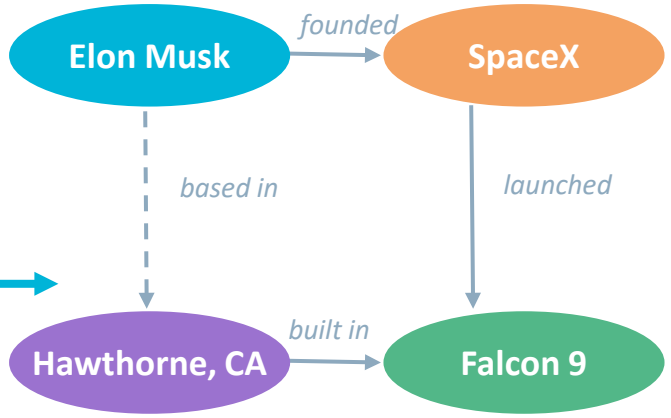
Raw Document

"Elon Musk founded SpaceX in 2002. SpaceX launched Falcon 9. Falcon 9 is a reusable rocket built in Hawthorne, California."

PERSON ORG ROCKET PLACE

NLP Pipeline

Knowledge Graph



Information Extraction Steps

1. NER — Identify entity spans: persons, orgs, locations, products
2. Relation Extraction — Classify edges between entity pairs (founded, acquired, located_in ...)
3. Coreference Resolution — Merge "she", "the company" to their canonical entities
4. Event Extraction — Anchor facts to time, location, and participants

Graph RAG — Mathematical Formulation

1. Knowledge Graph

$$G = (V, E, \varphi, \psi)$$

V = entity nodes $E \subseteq V \times V$ = directed relation edges

$\varphi: V \to T_v$ = entity type fn. $\psi: E \to T_e$ = relation type fn

2. Seed Node Retrieval

$$S(q) = \text{Top}-\text{K}\ \{\, v \in V \mid sim(\, h_v, \text{Embed}(q)\,)\,\}$$

where h_v = f_enc(v) is the entity embedding and sim(·,·) is cosine similarity

3. Subgraph Extraction (k-hop neighbourhood)

$$G_q = N_k(S(q), G) = \{\, v \in V \mid d_G(s, v) \le k,\ s \in S(q)\,\}$$

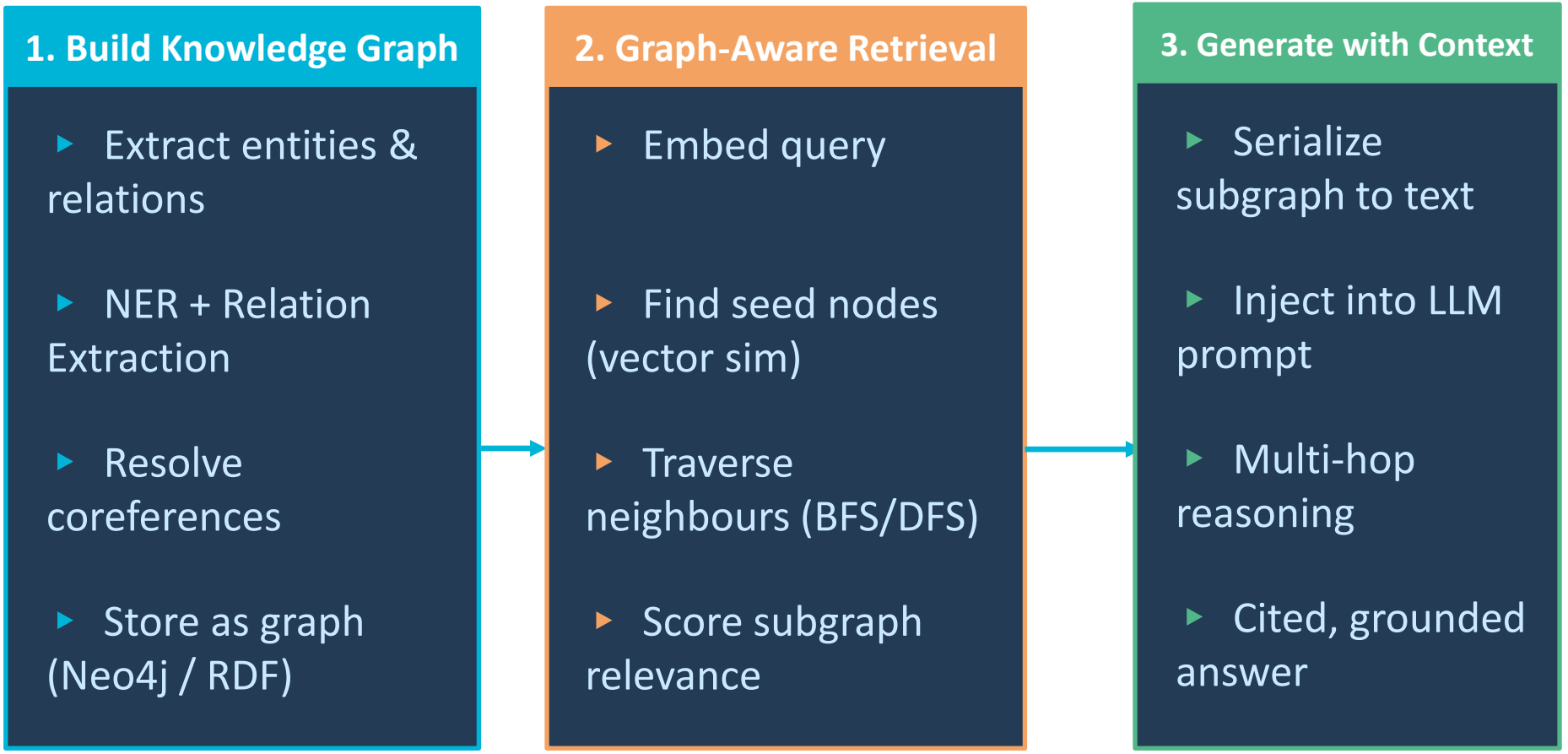
where d_G(s, v) is the shortest-path distance in G and k is the hop limit

4. Graph-Conditioned Generation

$$P(a \mid q, G) = P_{LLM}(a \mid q, \text{serialize}(G_q))$$

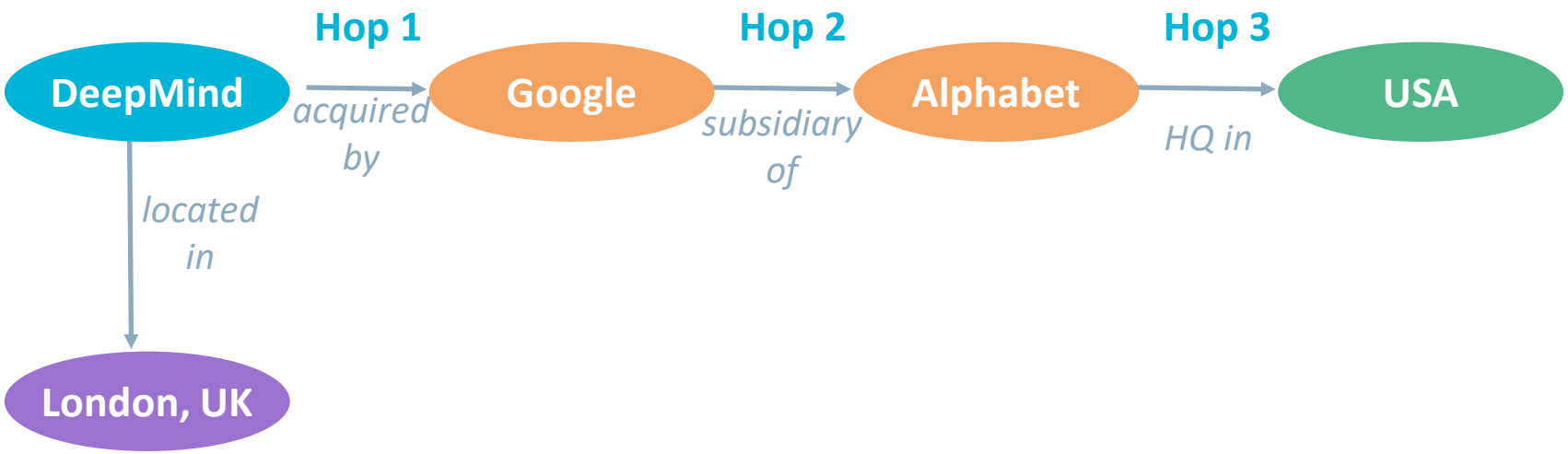
compare Standard RAG: P(a | q, D) = P_LLM(a | q, Top-K(Embed(q), D)) ← *flat chunks only*

Graph RAG Architecture



Multi-Hop Reasoning: Graph RAG in Action

Query: *"Which country is the headquarters of the company that acquired DeepMind?"*



Formal mapping to Graph RAG:

$S(q)$ = {DeepMind} $k = 3$ hops

G_q = {DeepMind, Google, Alphabet, USA}

answer = $P_{LLM}(a \mid q, \text{serialize}(G_q))$

Answer: Alphabet (Google's parent company) acquired DeepMind. Alphabet is headquartered in the **United States**. Requires 3 hops — impossible with flat chunk retrieval.

Where Graph RAG Shines

Healthcare

Drug-gene-disease interaction graphs enable multi-hop clinical reasoning across biomedical literature.

Legal

Entity graphs over case law let attorneys trace precedent chains and ownership structures.

Enterprise Q&A

Org charts + product-knowledge graphs answer questions like 'Who owns the risk for system X?'

Research

Citation & concept graphs help researchers discover indirect connections between papers.

Cybersecurity

Attack-path graphs allow analysts to trace lateral movement across CVEs, assets, and actors.

Finance

Corporate ownership graphs surface hidden counterparty risk and cross-sector dependencies.