ADVANCED KNOWLEDGE ENGINEERING

Instructor:
KHUAT Thanh Tung
University of Technology Sydney

Subject's Outline

- **Topic 1:** An Overview of Knowledge Engineering
- **Topic 2:** An Overview of Knowledge-based Systems
- **Topic 3:** Knowledge Acquisition
- **Topic 4:** Knowledge Representation and Reasoning

Mid-term assessment

- **Topic 5:** Ontology
- **Topic 6:** Knowledge Graphs
- **Topic 7:** Expert Systems
- **Topic 8:** Uncertain Reasoning
- **Topic 9:** Hybrid Knowledge-based Systems
- **Topic 10:** Automated AI Planning

Group projects for the advanced topics

KNOWLEDGE GRAPHS

Objectives of this topic

By the end of this topic, you will be able to:

- ✓ explain how knowledge graphs represent data
- ✓ discuss the advantages and disadvantages of knowledge graphs
- ✓ understand how to query data from knowledge graphs
- ✓ know how to apply the knowledge graphs to practical projects
- ✓ understand several methods of reasoning over knowledge graphs
- ✓ know how to build knowledge graphs
- ✓ know how to combine knowledge graphs and Large Language models.

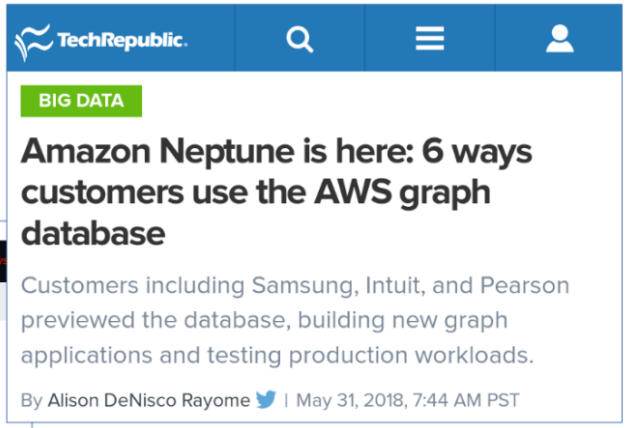
Agenda

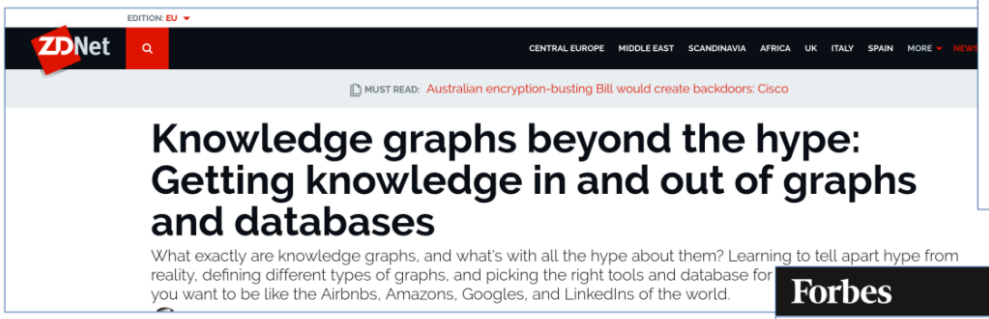
- Foundation of Knowledge Graph
  - What is Knowledge Graph
  - Applications
  - Knowledge Graph construction
  - Knowledge Graph Queries
- Advanced research topics
  - Knowledge Graph Embeddings
  - Knowledge Graph Reasoning
  - Knowledge Graph and Large Language Models (LLMs)

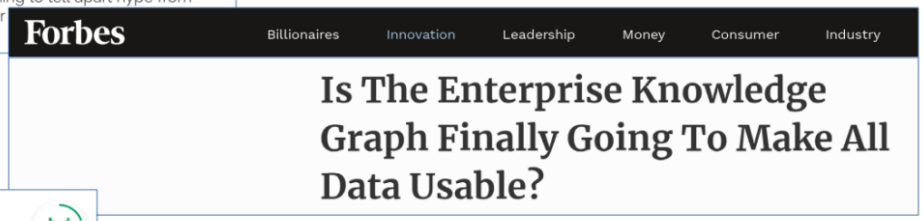
Knowledge Graphs Everywhere

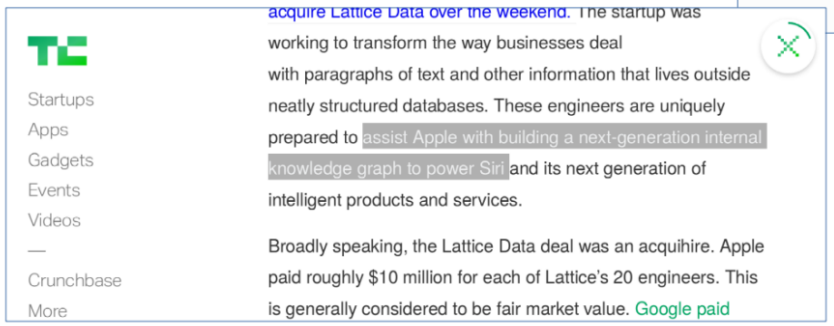
Google

Bing







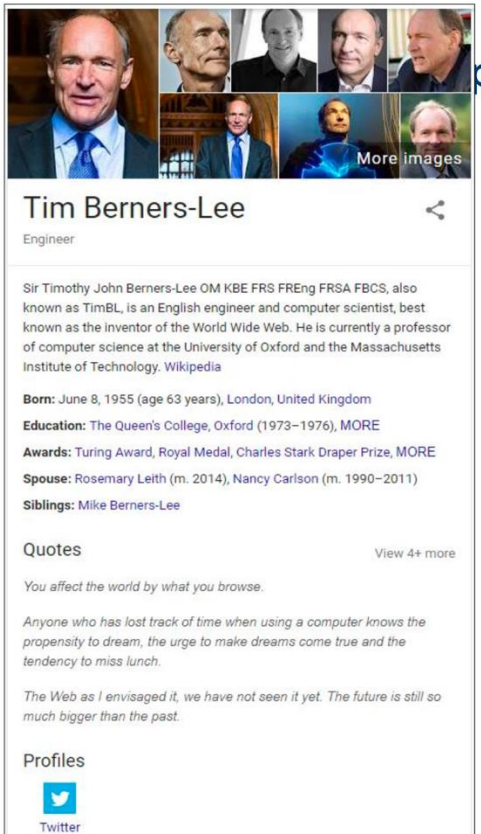


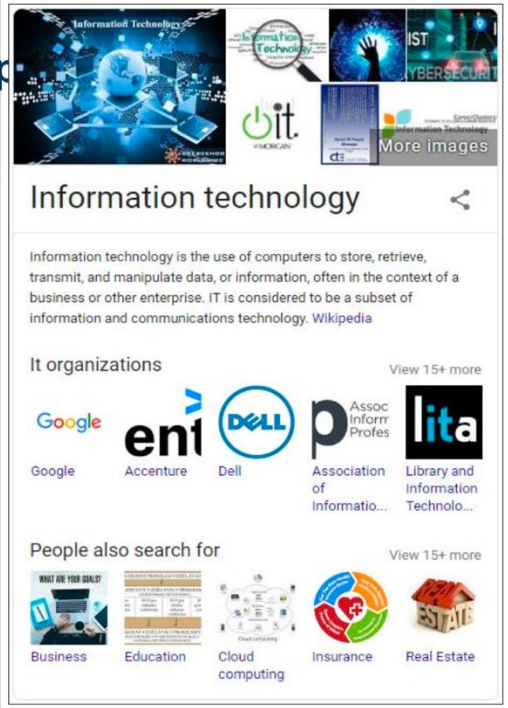
ebay

IBM

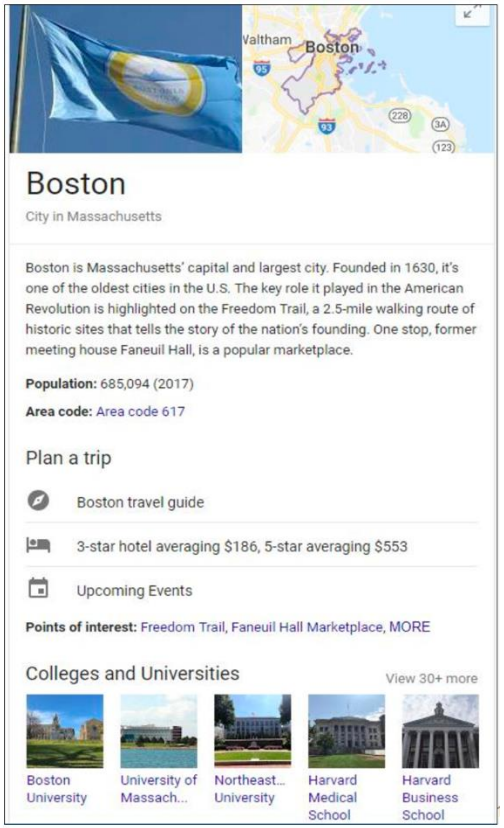
Knowledge Graph @ Google

- Key benefits
  - Find the right things
  - Get the best summary
  - Go deeper and broader



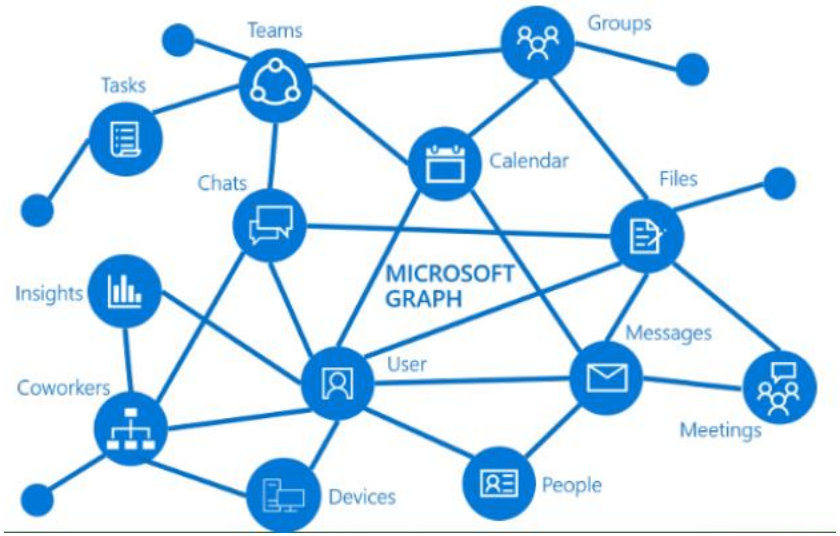


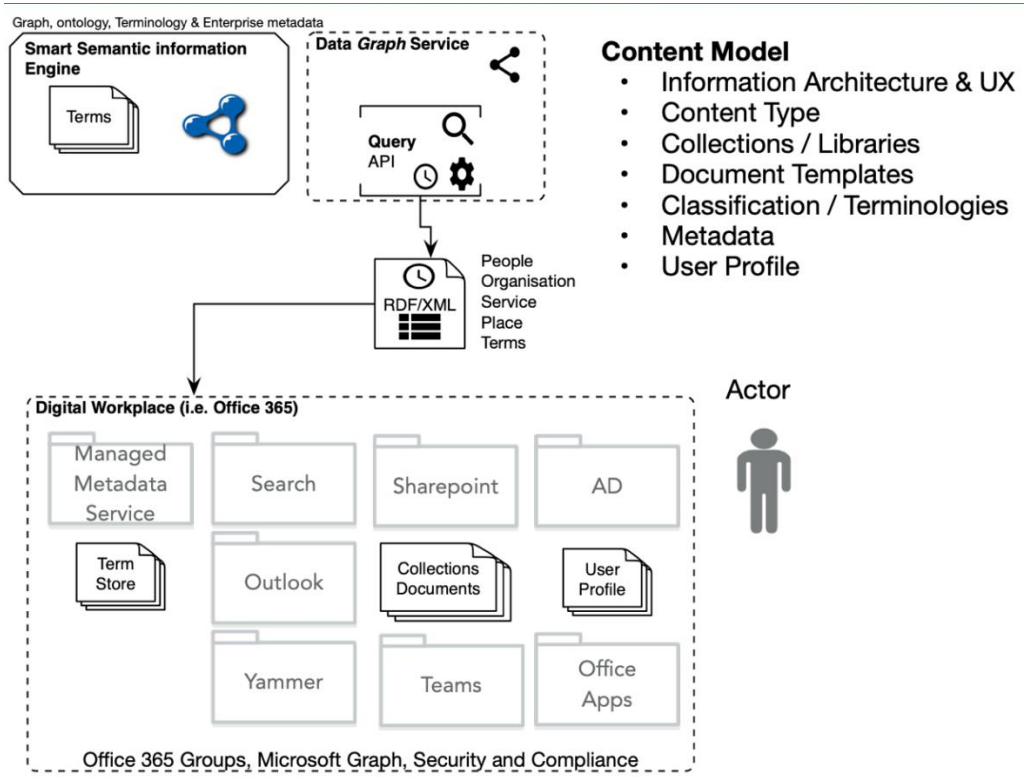
Knowledge Graphs from Google searches



Knowledge Graph @ MS Office 365

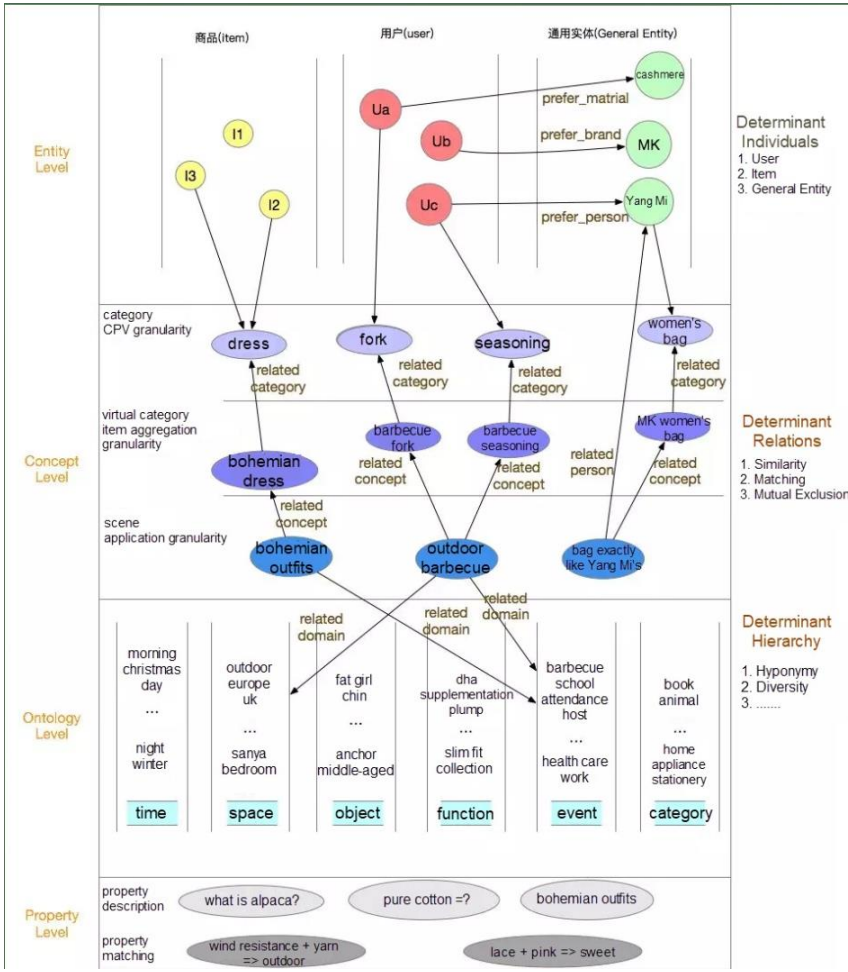
- Key benefits
  - Enterprise KG provides connected data supporting people at work
  - Personalized search





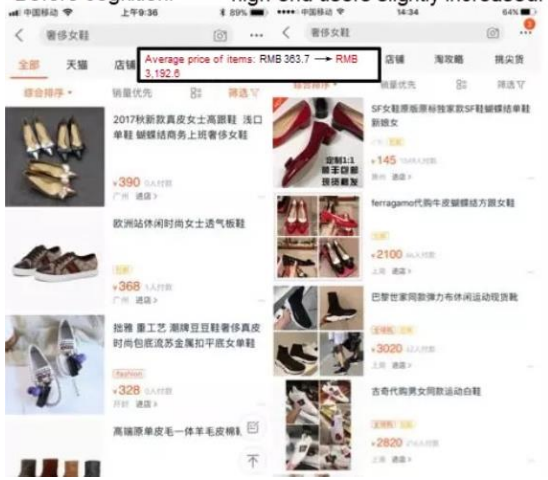
Knowledge Graph @ Alibaba Search & Recommendation Engines

- Key benefits
  - Structured data
  - Detecting noise in data
  - Connected data
  - Enable deep data cognition



Before cognition:

After cognition: The number of high-end users slightly increased.



Comparative data for 10 consecutive days from December 7, 2017 to December 18, 2017

Query types	all	High freq.	Medium freq.	Low freq.
GMV	+1.07%	+0.96%	+2.85%	+3.08%
PV coverage (per bucket)	61w	33w	24w	4w
UV coverage (per bucket)	4.2w	2.5w	1.7w	1w

Comparative data for 13 consecutive days from January 27, 2018 to February 8, 2018

Query types	all	High freq.	Medium freq.	Low freq.
GMV	+1.13%	-5.46%	+3.24%	+9.62%
UV coverage (per bucket)	11371	5120	5411	3293

Trial

Follow-ups:

Expansion of brand awareness categories: To expand categories, such as men's shoes, women's shoes, household items, which are being gradually expanded.

Expansion of the scope of cognition: To increase the cognition of different groups suc as online celebrities, stars, and designers, and to tease out extendable data.

What Knowledge Graphs are

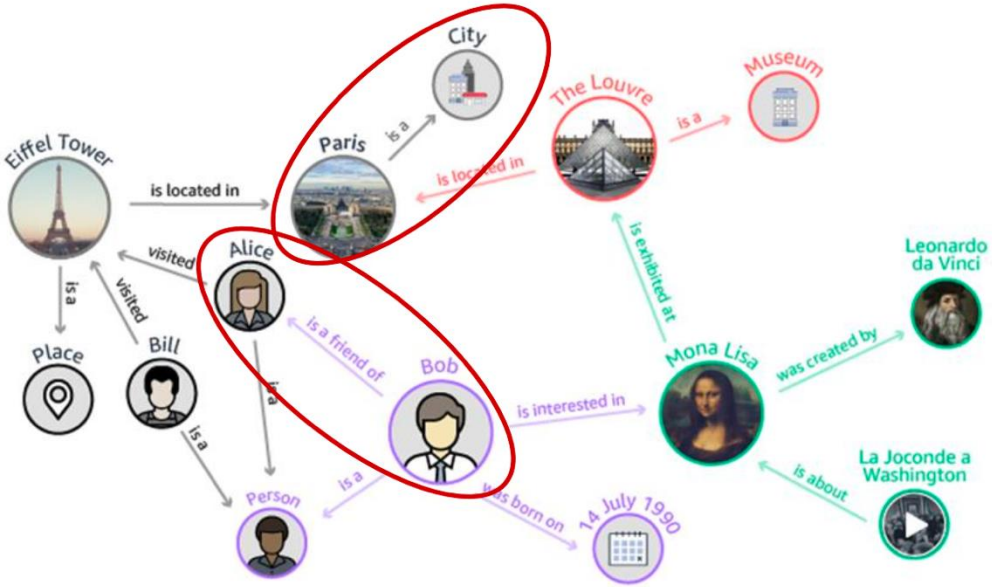
- The organization and representation of a knowledge base as a graph: as a network of nodes and links, not a table of rows and columns
- Usually based on data in graph databases, rather than relational databases
- Is both human-readable and machine-readable
- Usually includes, but not limited to, visualizations, such as of…
  - A display of interconnected nodes and links
  - A display of related information in a "fact box"
  - An output of graph analytics

What Knowledge Graphs are

- Represents a collection of interlinked descriptions of entities
  - Objects, events or concepts
  - Multiple types of entities and relations exist
- Facts are represented as triplets (h, r, t)
  - ('Paris', 'is_a', 'city')
  - ('Alice', 'is_friend_of', 'Bob')

Notation & Symbols

- h: the head entity
- r: the relation
- t: the tail entity
- (h, r, t): the triplet



Origin of Knowledge Graph

- The term "Knowledge Graph" was popularized by Google in 2012
  - when it announced the launch of its Knowledge Graph, aiming to improve search results by understanding the relationships between entities (people, places, things) rather than just matching keywords
- **Question:** Is Knowledge Graph invented Google
  - **Answer:** No, there has been a long tradition on research graph form of knowledge graph. Main characteristics of Knowledge Graph is inspired by Semantic Networks

Compare Knowledge Graph and Semantic Network

- Similarities
  - Use graph-based structures to represent relationships between entities.
  - Able to capture and represent knowledge and semantics.
- Differences
  - Knowledge graphs are generally more complex and can handle larger and more intricate datasets
  - Knowledge graphs may include additional features like context and metadata.
  - Knowledge graphs are typically associated with modern data integration and analytics tasks, while semantic networks have more historical significance in AI and cognitive modeling (to model knowledge and reasoning)

What is special about Knowledge Graphs?

- A Knowledge Graph is a data set that is:
  - structured (in the form of a specific data structure)
  - normalized (consisting of small units, such as vertices and edges)
  - connected (defined by the – possibly distant – connections between objects)
- Moreover, knowledge graphs are typically:
  - explicit (created purposefully with an intended meaning)
  - declarative (meaningful in itself, independent of a particular implementation or algorithm)
  - annotated (enriched with contextual information to record additional details and meta-data)
  - non-hierarchical (more than just a tree-structure)
  - large (millions rather than hundreds of elements)

Different Definitions of Knowledge Graphs?

"A knowledge graph (i) mainly describes real world entities and their interrelations, organized in a graph, (ii) defines possible classes and relations of entities in a schema, (iii) allows for potentially interrelating arbitrary entities with each other and (iv) covers various topical domains."
"Knowledge graphs are large networks of entities, their semantic types, properties, and relationships between entities."
"Knowledge graphs could be envisaged as a network of all kind things which are relevant to a specific domain or to an organization. They are not limited to abstract concepts and relations but can also contain instances of things like documents and datasets."
"We define a Knowledge Graph as an RDF graph. An RDF graph consists of a set of RDF triples where each RDF triple (s, p, o) is an ordered set of the following RDF terms: a subject $s \in U \cup B$, a predicate $p \in U$, and an object $U \cup B \cup L$. An RDF term is either a URI $u \in U$, a blank node $b \in B$, or a literal $l \in L$."
"[...] systems exist, [...], which use a variety of techniques to extract new knowledge, in the form of facts, from the web. These facts are interrelated, and hence, recently this extracted knowledge has been referred to as a knowledge graph."

Source: Towards a Definition of Knowledge Graphs

Knowledge Graphs vs Ontologies

- "A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge." (Eherlinger and Wöß - Towards a Definition of Knowledge Graphs.)
- Whereas an ontology can be a generic model template of how things are related to each other, a knowledge graph is the actual instance of that model
- A knowledge graph is an **ontology + instance data** (instance terms and links to data and content)

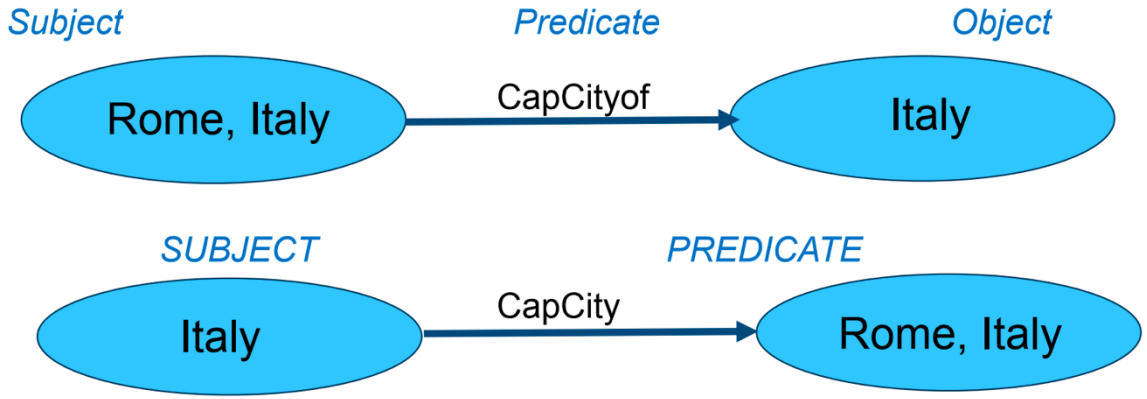
- Knowledge graphs are ontologies and more.
- A knowledge graph may also comprise multiple ontologies, or an ontology and other vocabularies.

Knowledge Graphs vs Ontologies

- Both Knowledge Graph and Ontology:
  - Represent nodes (things) and relationships between them
  - Can be visually represented in the same way of nodes and defined relationships and then may look the same in the visualization
  - *Are based on Semantic Web standards*, such as RDF triples
  - Tend to have been more the expertise domain of computer scientists and data scientists of than information professionals/taxonomists, but that's changing! Also a growing area of interest in knowledge management (business)

Knowledge Graphs vs Ontologies

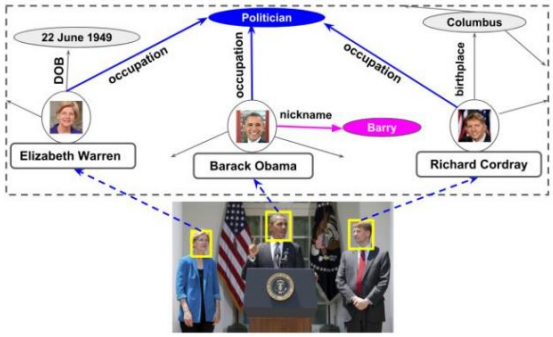
- Knowledge graphs and ontologies are based on RDF:
  - RDF, a standard model for data interchange on the Web, uses URIs to name things and the relationship between things, which are referred to as triples:
    (1) Subject – (2) Predicate – (3) Object.



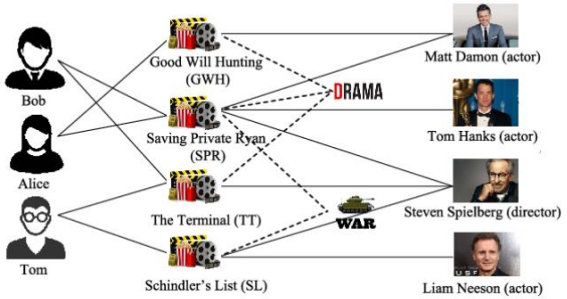
Applications of Knowledge Graphs

- Examples

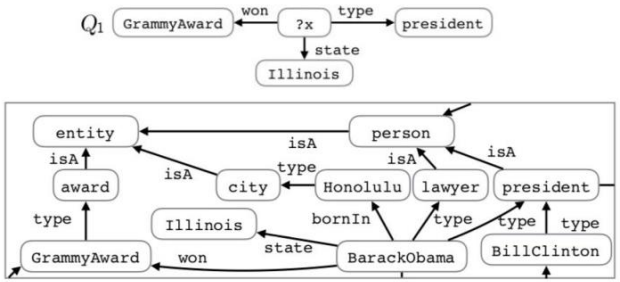
Computer Vision [1]



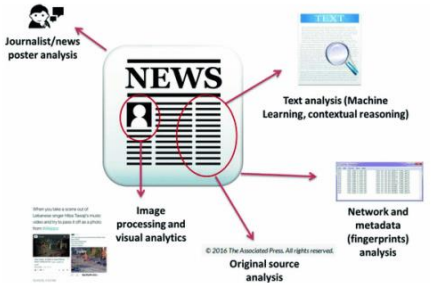
Recommendation [2]



Question Answering [3]



Fact Checking [4]



Applications of Knowledge Graphs

- What knowledge graphs can do
  - Integrate knowledge
  - Serve data governance
  - Provide semantic enrichment
  - Bring structured and unstructured data together
  - Provide unified view of different kinds of unconnected data sources
  - Provide a semantic layers on top of the metadata lay
  - Improve search results beyond machine learning and algorithms
  - Answer complex user questions instead of merely returning documents on a topic
  - Combine with deep text analytics, semantic AI, and machine learning

Applications of Knowledge Graphs

- Implementations of knowledge graphs
  - Recommendation engine (such as in ecommerce)
  - Expert finder
  - Question-answering based on data
  - Enterprise knowledge management
  - Search and discovery
  - Customer 360 – view of everything known about customers
  - Compliance

Knowledge Graphs in Companies

- Typical knowledge graphs:
  - Wikidata, Yago, Freebase, DBpedia
  - OpenStreetMap
  - Google Knowledge Graph, Microsoft Bing Satori
  - Amazon Knowledge Graph: Started as product categorization ontology
- Debatable cases
  - Facebook's social graph: structured, normalized, connected, but not explicit (emerging from user interactions, without intended meaning beyond local relations).
  - WordNet: structured dictionary and thesaurus, but with important unstructured content (descriptions); explicit, declarative model
  - Global data from schema.org: maybe not very connected
  - Document stores (Lucene, MongoDB, etc.): structured, but not normalized; connections sub-ordinary

Knowledge Graphs in Companies

- Primarily not knowledge graphs:
  - Wikipedia: mostly unstructured text; not normalized; connections (links) important but sub-ordinary
  - Relational database of company X: structured and possibly normalized, but no focus on connections (traditional RDBMS support connectivity queries only poorly)

Knowledge Graph Construction

Knowledge Graph Management Systems

AllegroGraph Franz Inc.	Graph database supporting SPARQL and Prolog reasoning
GRAKN.AI	Apache Cassandra-based KGMS providing schema support based on the Entity Relationship model
GP	Knowledge Graph-as-a-Service
VADALOG	Data source-agnostic KGMS supporting ontological and recursive reasoning based on Datalog
neo4j	Leading graph database system
Stardog	RDF-based unifying data-integration platform
metaphacts	SPARQL 1.1-graph database-based end-user-oriented platform
MAANA	Azure-based computation-focused platform
GNOSS	RDF and OWL-based metadata management solution.

Data Format/Back-end in Knowledge Graph Management Systems

AllegroGraph Franz Inc.	Graph database sup	Graph
	d Prolog reasoning	
GRAKN.AI	Apache Cassandra-based KGMS providing sc	Cassandra
	based on the Entity Relationship model	
GP	Knowledge Graph-as-a-Service	
VADALOG	Data source-agnostic KGMS supporting onto	Multiple
	recursive reasoning based on Datalog	
neo4j	Leading graph datab	Graph
Stardog	R	RDF
	data-integration platform	
metaphacts	S	RDF
	database-based end-user-oriented	
	R	
MAANA	Azure-based computation-focused platform	Azure
GNOSS	I	RDF
	ed metadata management solution.	

Knowledge Graph Construction

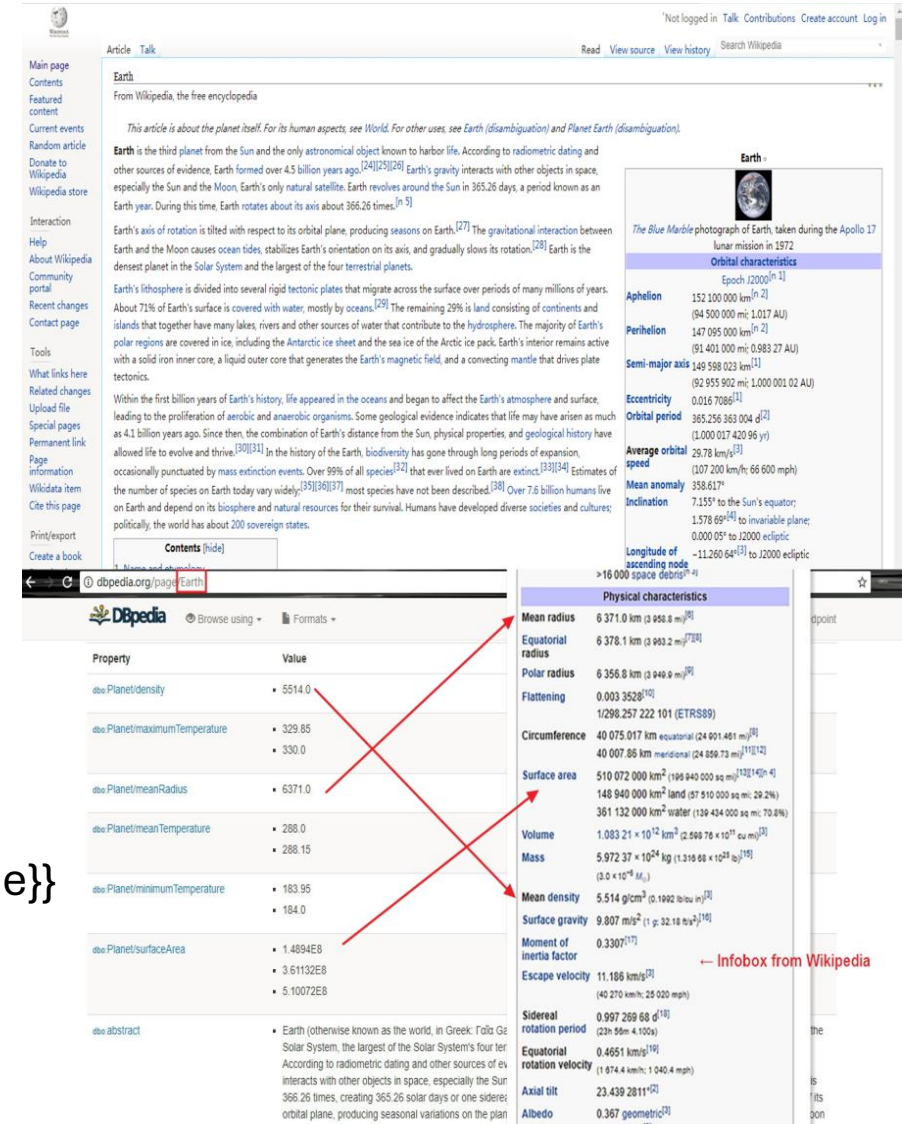
- Create or utilize taxonomies, apply ontologies, and link to data/content.
  - Follow SKOS, OWL, and RDF standards of the W3C
  - For example, all nodes must have URIs (Uniform Resource Identifies)
- Graph-database software tools can help.
  - For example, Neo4j Graph Database (Tutorial)
- Using Machine Learning algorithms
  - NELL - Never-Ending Language Learning (See the separated slides provided in Teams)
  - Large Language Models, such as Microsoft GraphRAG

Next slides will present several examples for building Knowledge Graphs

DBpedia Knowledge Graph Construction

- Mapping Wikipedia template elements to elements in DBpedia elements.
- Example: We consider some Wikipedia pages about actors

```
{{TemplateMapping
| mapToClass = Actor
| mappings =
    {{PropertyMapping | templateProperty =
        name | ontologyProperty = foaf:name}}
    {{PropertyMapping | templateProperty =
        birth_place | ontologyProperty = birthPlace}}
}}
```



DBpedia Knowledge Graph Construction

- Example (continued)
  - dbpedia:Vince_Vaughn rdf:type dbpedia-owl Actor .
  - dbpedia:Vince_Vaughn foaf:name “Vince Vaughn”@en .
  - dbpedia:Vince_Vaughn dbpedia-owl:birthplace dbpedia:Minneapolis .

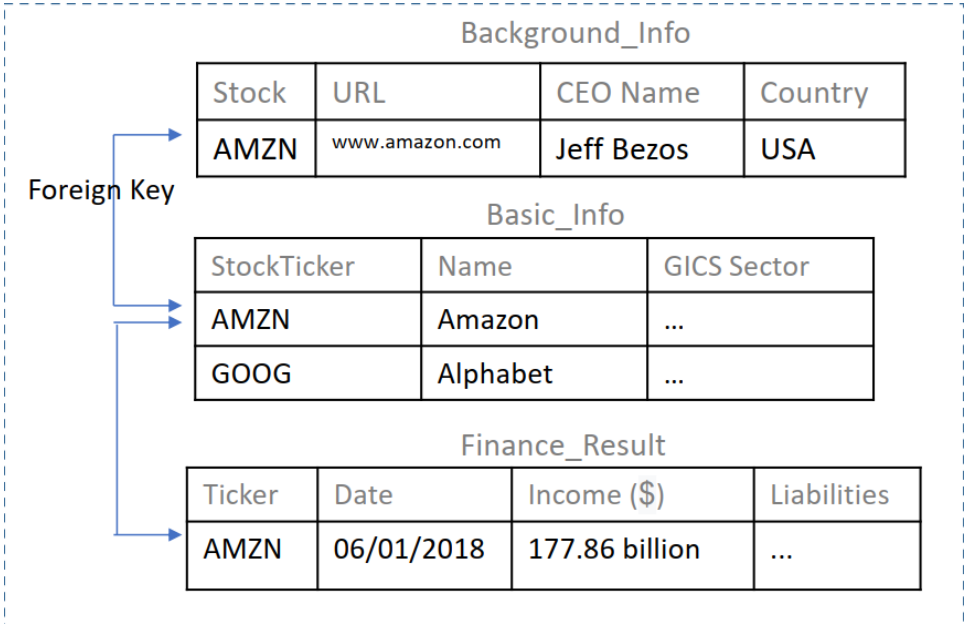
Vince Vaughn

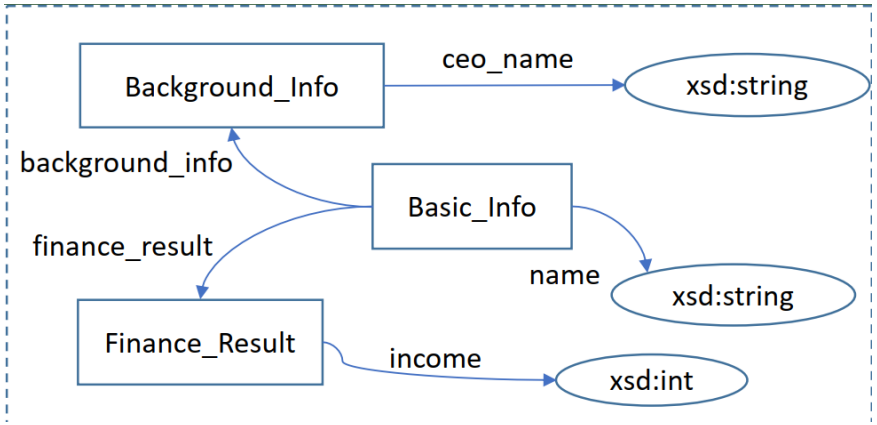
Vaughn in 2015

Born	Vincent Anthony Vaughn March 28, 1970 (age 51) Minneapolis, Minnesota, U.S.
Occupation	Actor, producer, screenwriter, comedian
Years active	1988–present
Height	6 ft 5 in (1.96 m)[1]
Spouse(s)	Kyla Weber (m. 2010)
Children	2

Knowledge Graph Construction from (Web) Tables

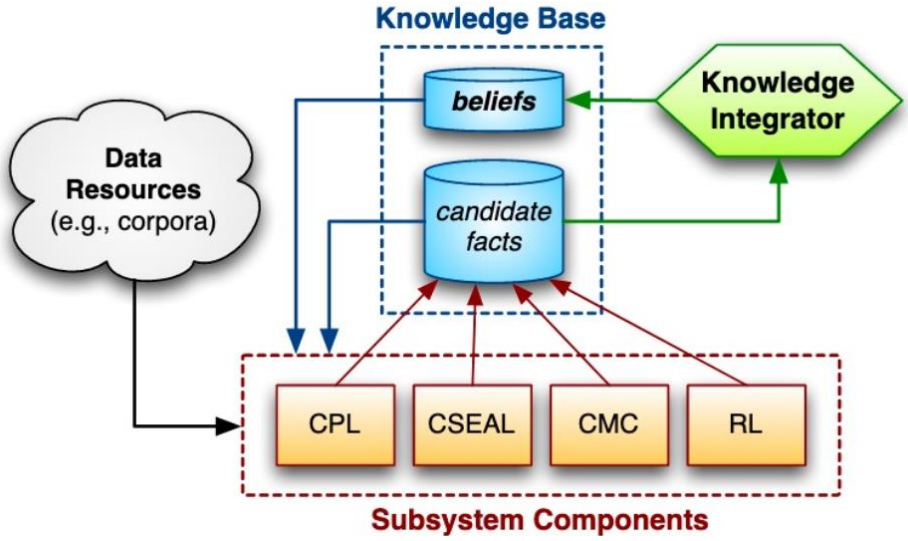
- (Web) Table understanding.
  - Extraction of web tables: HTML Tables, attribute-value pairs (such as info box), entity lists
  - Alignment with Knowledge Graph Schema
  - Column type prediction
  - Table enhancement

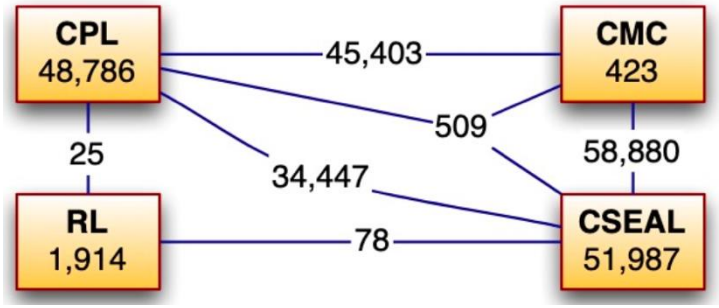




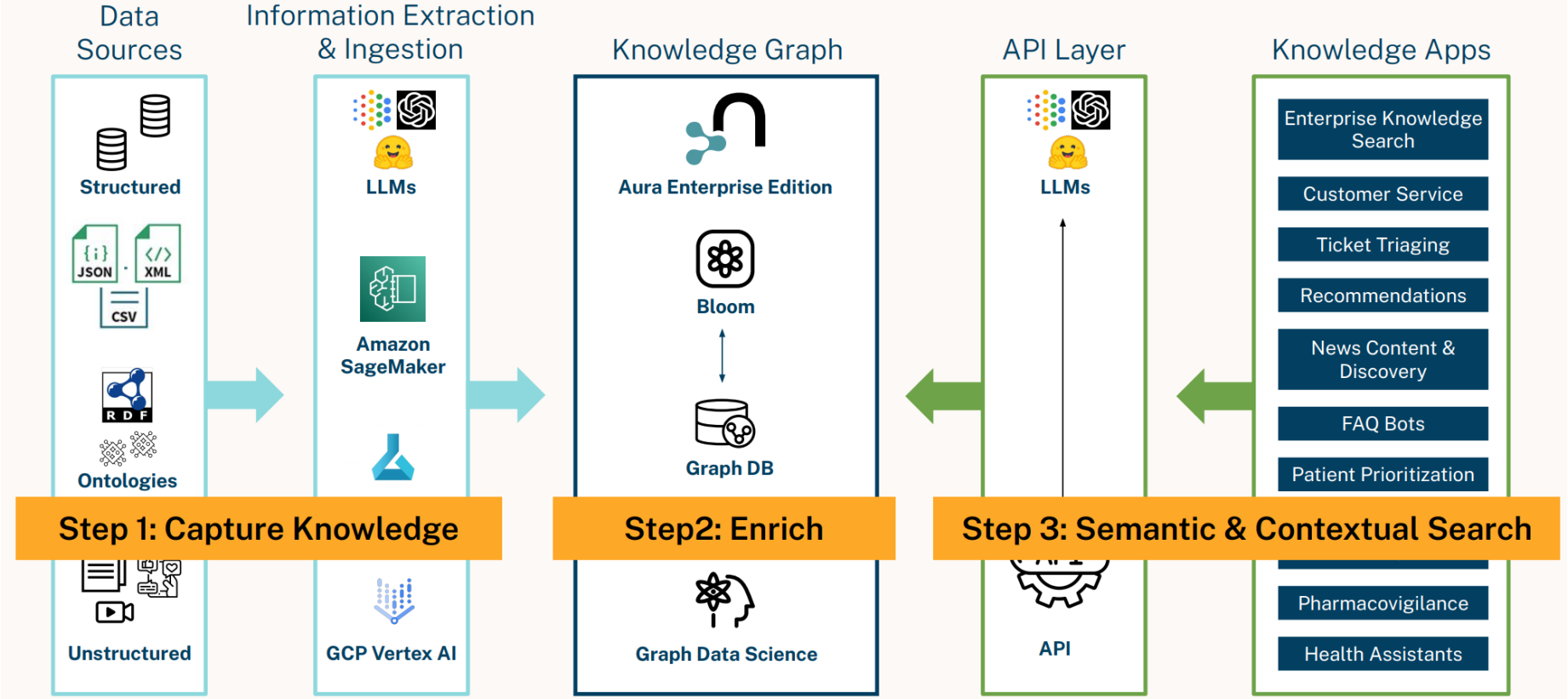
NELL: Never Ending Language Learner

- Goals
  - Extract info from **web texts** to construct Knowledge base (KB)
- Inputs
  - **schema** with 800 types and relations
  - 10-20 seed examples for each
- Output: continuously growing KB
- Key methods
  - Coupled Pattern Learner (CPL)
  - Coupled SEAL (CSEAL)
  - Coupled Morphological Classifier (CMC)
  - Rule Learner (RL)





Knowledge Graph Construction in Neo4j



Tutorial

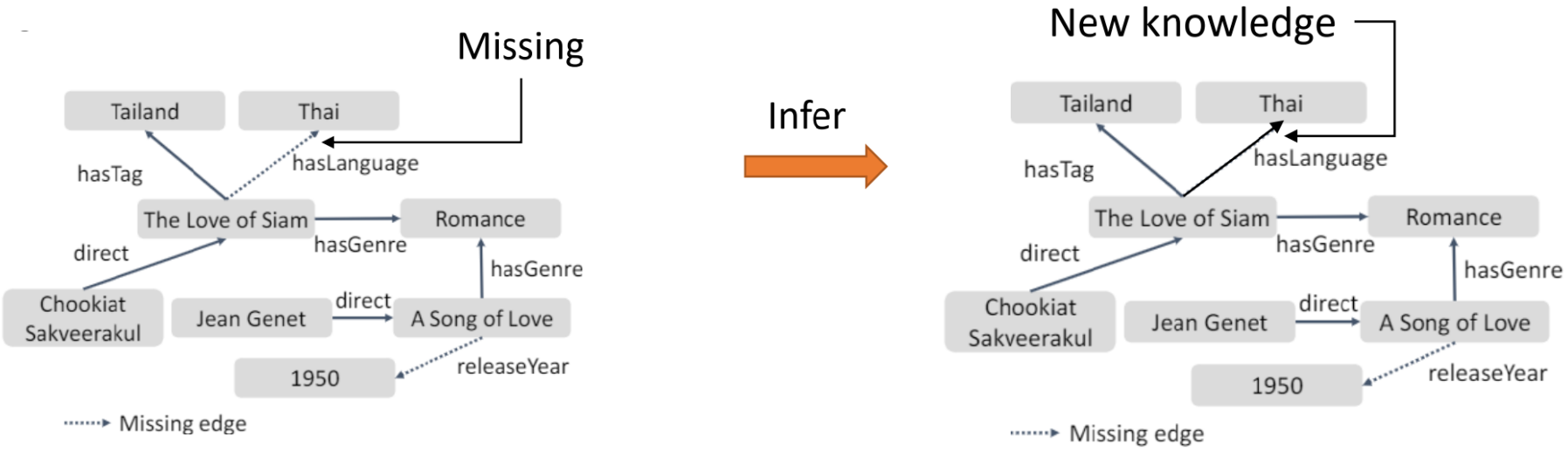
Knowledge Graph Querying

- If Knowledge Graphs are stored in the RDF formats.
  - Using SPARQL
- If Knowledge Graphs are stored in the Knowledge Graph Management systems
  - Using the query languages provided by the management systems
  - For examples, Neo4j provides Cypher
- If Knowledge Graphs are stored by third-party organizations
  - Using the provided APIs
  - For example, Google Knowledge Search API, Facebook's Graph API

Knowledge Graph Reasoning

What is Knowledge Graph Reasoning?

- **Goal**
  - Infer or discover new knowledge according to existing information in knowledge graphs in response to a query.
- **Benefit:** Enable models to learn from and reason with structured knowledge graph data



Knowledge Graph Reasoning Categorization

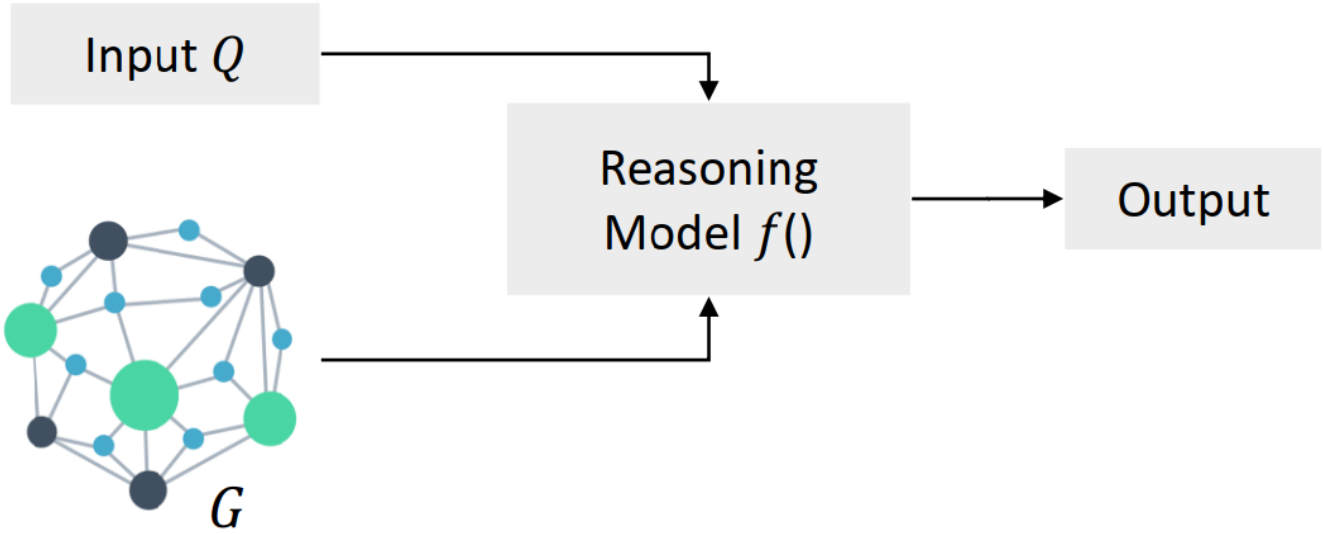
- **Class: Deductive reasoning**
- Apply **known rules** to derive knowledge
- (A, hasSon, B)&(B, hasSon, C)
- Infer: (A, isGrandFather, C)

- **Class: Abductive reasoning**
- Choose the best explanation that **explains an observation**
- (A, liveWith, B)
- Explanation: (A, hasSpouse, B)

- **Class: Inductive reasoning**
- Use patterns in observations to derive knowledge
- Give many examples of
  - (A, hasSon, B)&(B, hasSon, C)
  - (A, isGrandFather, C)
- Derive:
  - hasSon & hasSon means isGrandFather

Knowledge Graph Reasoning Formulation

- Knowledge graph reasoning can be formulated as $f(Q, G)$
  - f: the reasoning function/model
  - Q: the reasoning input/goal
    - A partial triplet (h, r, ?)
    - Natural language question
    - Query graph
  - G: the background knowledge graph



Knowledge Graph Reasoning Challenges

- **Size**: Knowledge Graphs are large
  - For example, DBpedia contains 4.6 million entities
  - Reasoning on large knowledge graph is time consuming
- **Quality**: KGs are noisy and incomplete
  - For example, half of entities in DBpedia contain less than 5 relationships
  - Reasoning on incomplete knowledge graphs can be difficult
- **Dynamics**: Almost every knowledge graph evolves over time
  - Present an additional challenge

Knowledge Graph Reasoning Challenges

- **Comprehensiveness:** Relations in KG have different properties
  - Symmetry, antisymmetry, composition
  - How to support these properties?
- **Efficiency:** Knowledge graphs are large
  - How to reason efficiently
- **Generalizability:** Generalization ability of the reasoning model
  - Handle new, unseen data effectively

Knowledge Graph Reasoning Challenges: Reasoning Inputs

- **Ambiguity:** Q is ambiguous
  - Users are not familiar with the background KG, likely to ask ambiguous questions
    - "John Litel role in *Declaration of Independence*" → role in : "profession" or "occupation"
    - "Thomas Jefferson role in *Declaration of Independence*" → role in: "film actor"
- **Interaction:** Q is changing iteratively
  - Users can gradually refine the queries
    - "John Litel role in declaration independence" → "Actor John Litel role in movie *Declaration of Independence*"
  - User's query intention may change correspondingly
    - "Who directed Interstellar?"
    - "What other movies did he also direct?"

Symmetric/Antisymmetric Relations

- **Symmetric/Antisymmetric Relations**
  - **Symmetric:** e.g., Marriage
  - **Antisymmetric:** e.g., hasChild
- Formally

r is **Symmetric**: $r(x, y) \Rightarrow r(y, x)$ if $\forall\, x, y$

r is **Antisymmetric**: $r(x, y) \Rightarrow \neg r(y, x)$ if $\forall\, x, y$

Inverse Relations

- **Inverse Relations**
  - Hypernym and hyponym
    - Color is the hypernym (r_2) of blue, and blue is the hyponym (r_1) of color
  - Husband (r_2) and wife (r_1)
- Formally

r_1 is inverse to relation r_2: $r_2(x, y) \Rightarrow r_1(y, x)$ if $\forall\, x, y$

Composition Relations

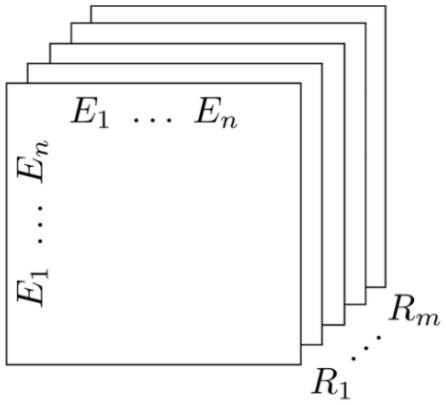
- **Composition Relations**
  - My mother's husband is my father
  - r_1: hasMother, r_2: hasHusband
  - r_3: hasFather
- Formally

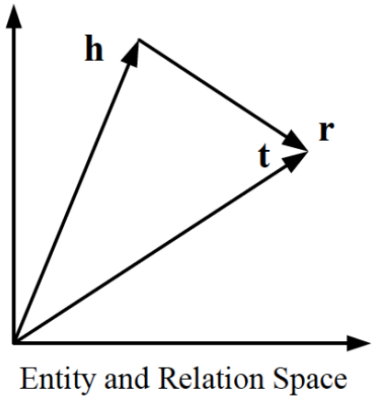
r_3 is a **composition** of relation r_1 and relation r_2:

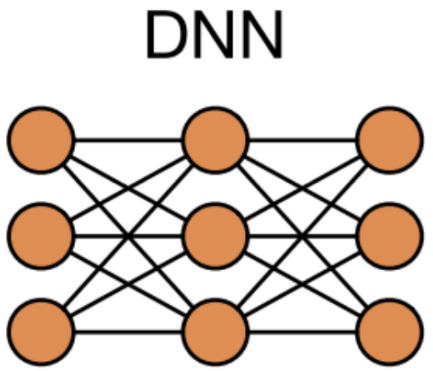
$$r_1(x, y) \wedge r_2(y, z) \Rightarrow r_3(x, z) \text{ if } \forall\, x, y, z$$

Reasoning in Continuous Space

- **Reasoning based on low-dimensional vector representations**
  - Knowledge graph embedding methods
- Project each entity and relation into a continuous vector space
- Tensor decomposition model, geometric model, deep learning model







Knowledge Graph Embedding

- **Goal:** Encode (1) entities as low-dimensional vectors and (2) relations as parametric algebraic operations in the continuous space
- **How-to:** Design a score function $f_{\mathbf{r}}(\mathbf{h}, \mathbf{t})$ w.r.t. such embedding vectors so that a true triplet receives higher score than a false one
- **KGE design rationale:** Capture KG patterns
  - Symmetry, antisymmetry, inversion and composition
- Applications of knowledge graph embedding
  - Knowledge graph completion
  - Question answering
  - Recommender system

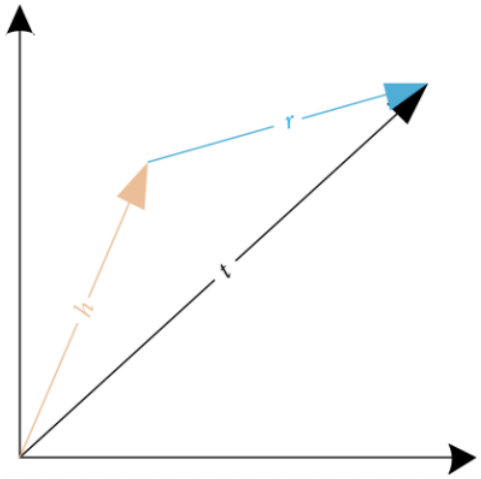
Notation & Symbols

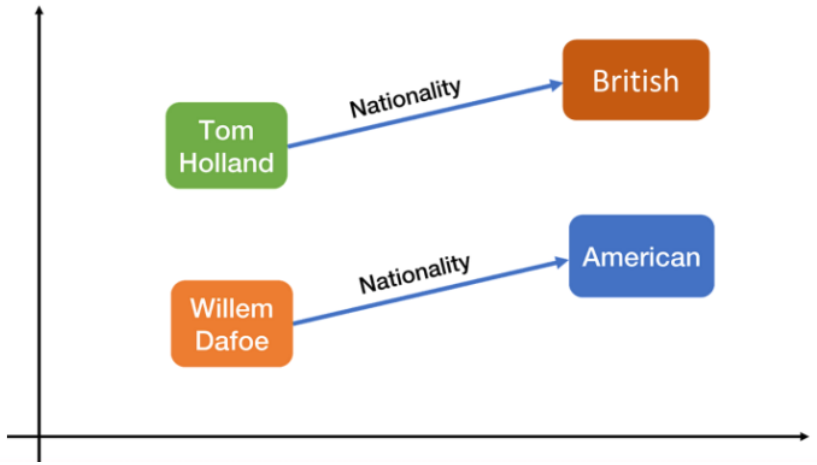
- h: head entity
- r: *relation*
- t: tail entity
- $\mathbf{h}$: head entity embedding
- $\mathbf{r}$: *relation embedding*
- $\mathbf{t}$: tail entity embedding
- $f_{\mathbf{r}}(\mathbf{h}, \mathbf{t})$: the score function
- $d_{\mathbf{r}}(\mathbf{h}, \mathbf{t})$: the distance function
- True/positive triplet: (h, r, t)
- False/negative triplet: (h', r, t), (h, r, t'), (h', r, t')

Knowledge Graph Embedding Methods: TransE

- **Embedding space:** Each entity and relation as a low-dimensional vector in R^k
- **Key idea:** Relation r as a translation from the head entity h to the tail entity t
  - An ideal/predicted tail entity: $\mathbf{t_{pred}} = \mathbf{h} + \mathbf{r}$
- **Score function:** $f_{\mathbf{r}}(\mathbf{h}, \mathbf{t}) = -d_{\mathbf{r}}(\mathbf{h}, \mathbf{t}) = -\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|$
- **Distance function:** $d_{\mathbf{r}}(\mathbf{h}, \mathbf{t}) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|$

- Triplet: (h, r, t)
- Embedding vectors: $\mathbf{h}, \mathbf{r}, \mathbf{t}$





TransE: Training Procedure

- For each positive triplet $(h, r, t) \in S$,
  - Sample a set of corrupted triplets $(h, r, t') \in S'_{(h,r,t)}$ or $(h', r, t) \in S'_{(h,r,t)}$

- True triplet: (h, r, t)
- Corrupted triplets:
  - (h', r, t), (h, r, t'), (h', r, t')
- γ: the margin

- Learning: a margin-based ranking loss

- Minimize $L = \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'_{(h,r,t)}} max(\gamma + d_r(\mathbf{h}, \mathbf{t}) - d_r(\mathbf{h}', \mathbf{t}'), 0)$

Margin: gap should be at least γ

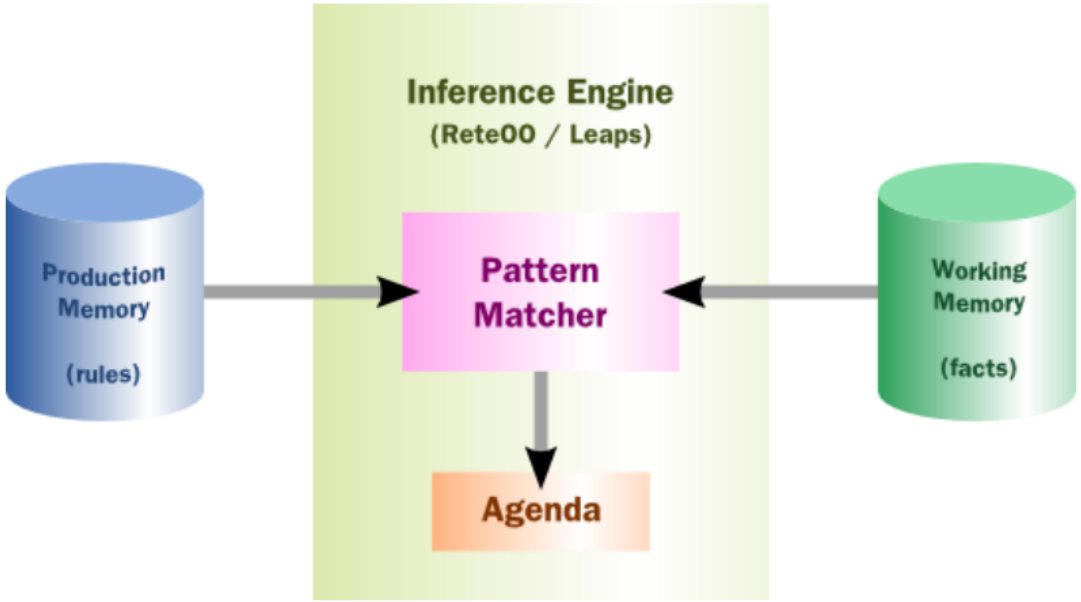
Paper: A. Bordes, N. Usunier, and A. Garcia-Duran. 2013. Translating Embeddings for Modeling Multi-relational Data (NeurIPS 13).

TransE: Key Properties

- **Pros:**
  - Can model antisymmetric relations: $\mathbf{h} + \mathbf{r} = \mathbf{t}$, but $\mathbf{t} + \mathbf{r} \neq \mathbf{h}$ if $\mathbf{r} \neq 0$
  - Can model inverse relations: $\mathbf{h} + \mathbf{r}_1 = \mathbf{t}$, $\mathbf{t} + \mathbf{r}_2 = \mathbf{h}$, $\mathbf{r}_1 = -\mathbf{r}_2$
  - Can model composition relations: $\mathbf{r}_3 = \mathbf{r}_1 + \mathbf{r}_2$
- **Cons**
  - Cannot model symmetric relations: $\mathbf{h} + \mathbf{r} = \mathbf{t}$, $\mathbf{t} + \mathbf{r} = \mathbf{h}$, then $\mathbf{r} = 0$

Rule-based Knowledge Graph Reasoning

- **Key idea:** Apply rules iteratively to generate new facts
  - New facts represent conclusions about the state of the domain given the observations
- **Major components**
  - The inference engine
  - The knowledge bases and rules

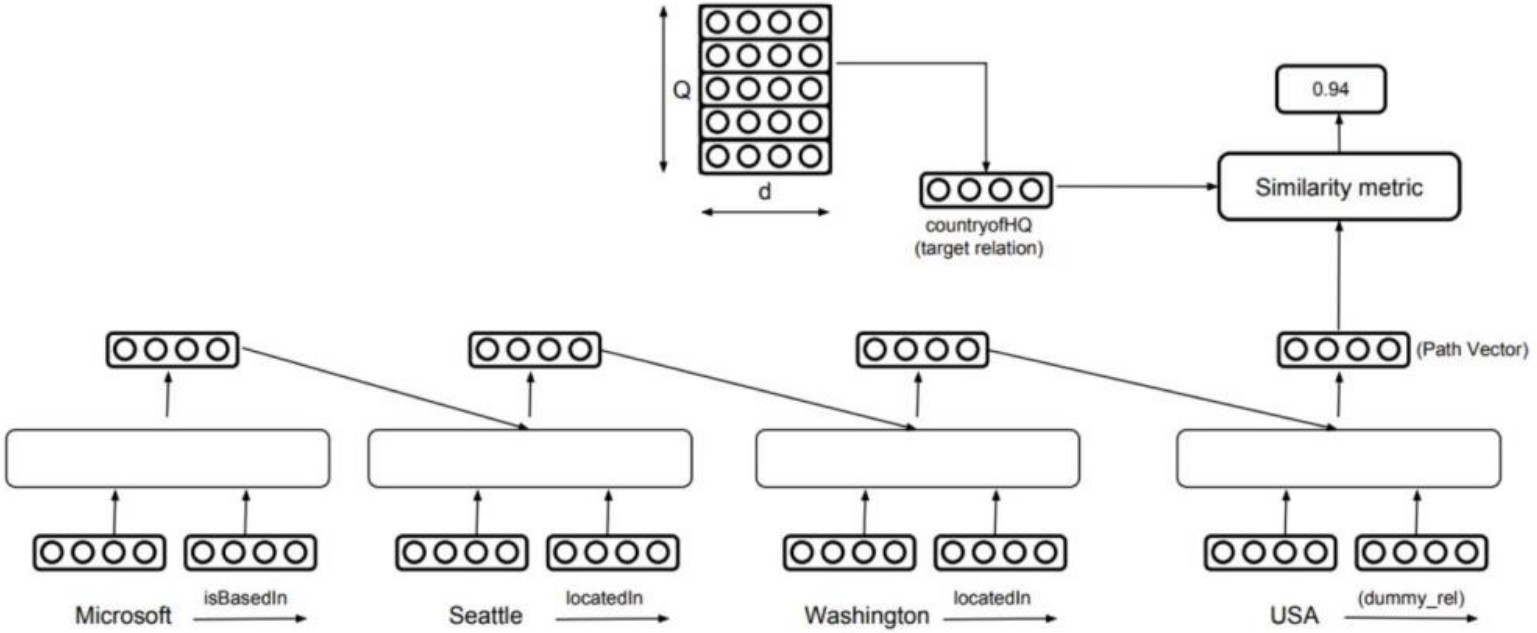


Rule-based Knowledge Graph Reasoning: Inference Engine

- Brain of the reasoning system
- Inferring knowledge by applying forward or backward chaining
  - Forward chaining (data driven)
    - Start with facts, determine applicable rules, and apply one
    - Repeat until no more rules can be applied
  - Backward chaining (goal oriented)
    - Give a goal, rules are applied by matching the goal to infer the answer
    - E.g. (Alan Turing, wasBorn, ?)

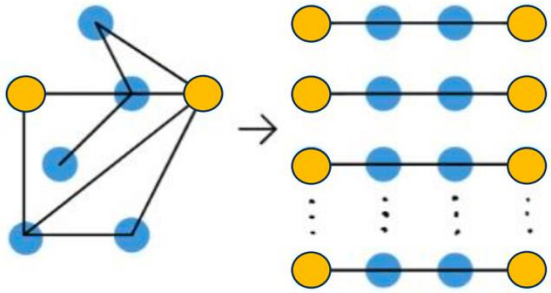
Path-based Knowledge Graph Reasoning

- Logical rule reasoning
  - Require given logic rules as input
- Path-based reasoning
  - Involves traversing paths in a KG to infer new information
  - Paths can be directed or undirected



Path-based Knowledge Graph Reasoning: Random Walks Method

- **Assumption:** random walks → relational features
- **Key ideas**
  - A relation path $p = (R_1, \ldots, R_n)$ is a sequence of relations
  - Run random walk to derive many paths
  - Use supervised training to predict the score of triplet



	Path 1	Path 2	…	Path n	Label
Query 1	Score 1.1	Score 1.2	…	Score 1.n	y1
Query 2	…	…	…	…	y2
…	…	…	…	…	…
Query k	Score k.1	Score k.2	…	Score k.n	yk

Paper: N. Lao, T. Mitchell, and W. Cohen. 2011. Random walk inference and learning in a large-scale knowledge base (ACL 11).

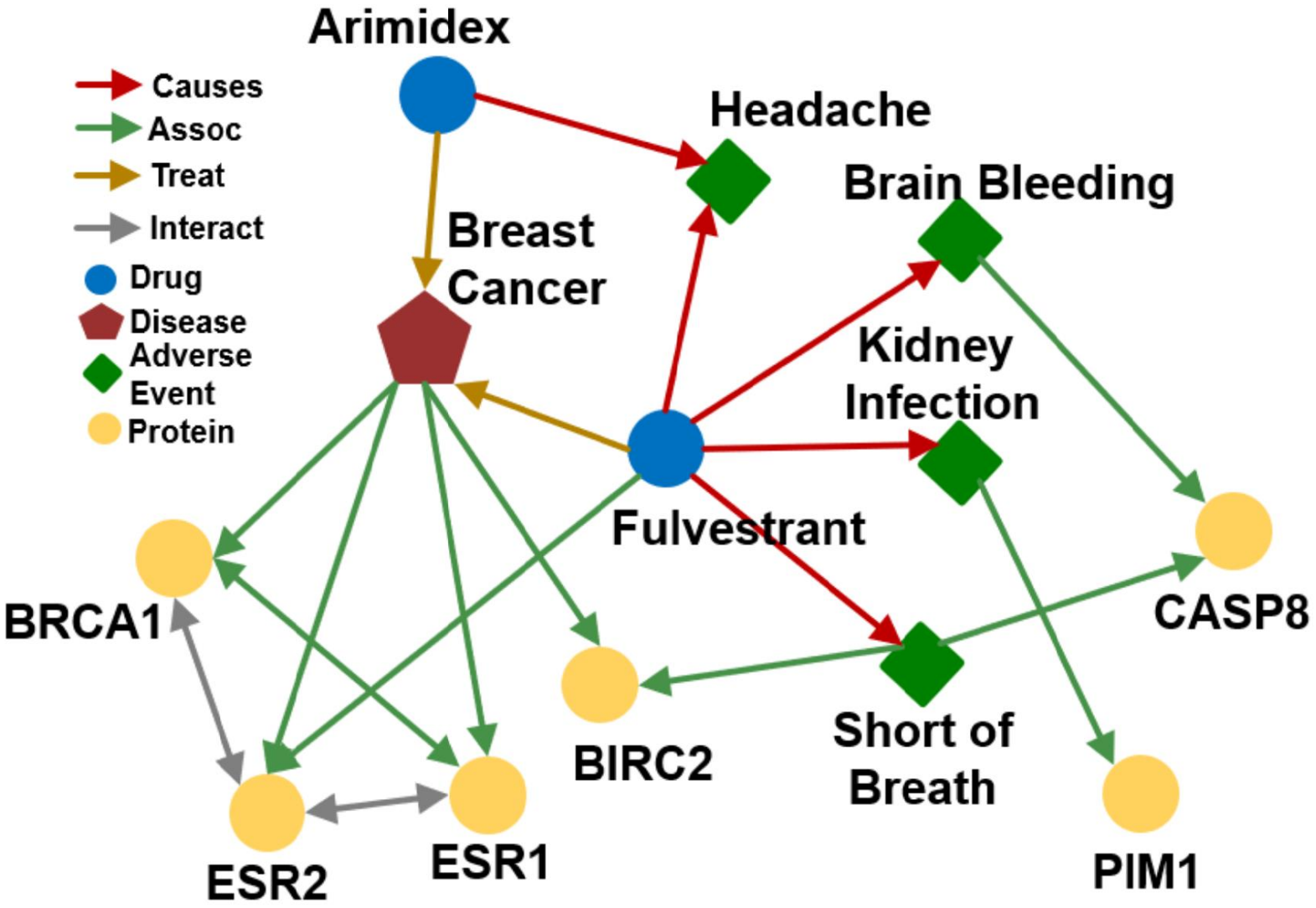
Neural Reasoning for Natural Language Query:

Background

Knowledge Graph Question-Answering

- **Definition**
  - Given: (1) a natural language question (2) the topic entity, and (3) a knowledge graph
  - **Find**: a set of nodes of the knowledge graph to answer the question
- Multi-hop question (the focus of this part)
  - Can be transformed to a path on KG
  - Example: "what is the language of the film directed by Steven Spielberg?"
- Other types of questions
  - One-hop question: "What is the genre of Interstellar?"
  - Logic query: "Where did Canadian citizens with Turing Award graduate?"

Example KG: Medicine



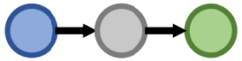
Predictive Queries on KG

- Can we do multi-hop reasoning, i.e., answer complex queries on an incomplete, massive KG?

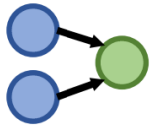
Query Types	Examples: Natural Language Question, Query
One-hop Queries	What adverse event is caused by Fulvestrant? (e:Fulvestrant, (r:Causes))
Path Queries	What protein is associated with the adverse event caused by Fulvestrant? (e:Fulvestrant, (r:Causes, r:Assoc))
Conjunctive Queries	What is the drug that treats breast cancer and caused headache? ((e:BreastCancer, (r:TreatedBy)), (e:Migraine, (r:CausedBy))

In this lecture, **we only focus on answering queries on a KG!** The notation will be detailed next

One-hop Queries



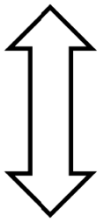
Path Queries



Conjunctive Queries

Predictive One-hop Queries

- We can formulate knowledge graph completion problems as answering one-hop queries
- KG completion: Is link (h, r, t) in the KG?



- One-hop query: Is t an answer to query (h, r)?
  - For example: What side effects are caused by drug Fulvestrant?

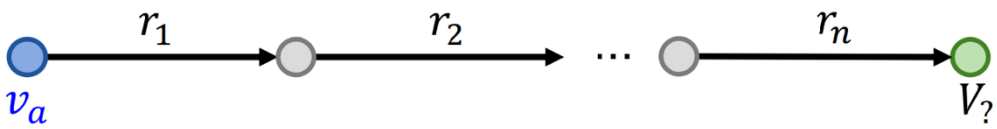
Path Queries

- Generalize one-hop queries to path queries by adding more relations on the path.
- An n-hop path query q can be represented by

$$q = (v_a, (r_1, \ldots, r_n))$$

  - v_a is an "anchor" entity
  - Let answers to q in graph G be denoted by $[\![q]\!]_G$

Query Plan of q:

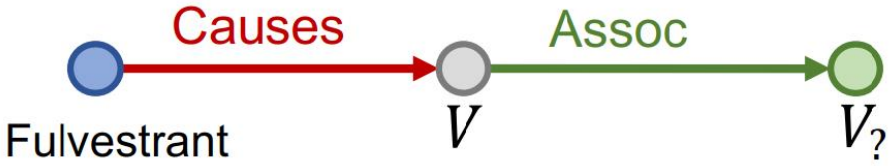


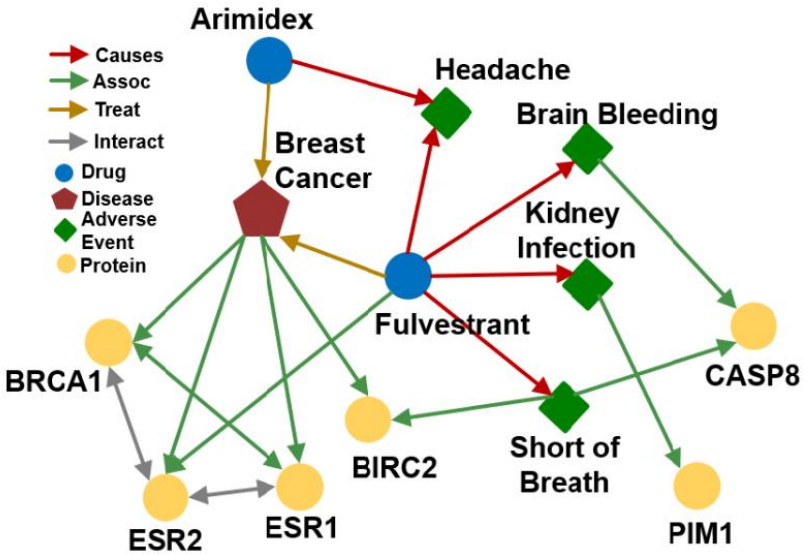
Query plan of path queries is a chain.

Path Queries

Question: "*What proteins are associated with adverse events caused by Fulvestrant ?*"

- v_a is e:Fulvestrant
- (r_1, r_2) is (r:Causes, r:Assoc)
- **Query:** (e:Fulvestrant, (r:Causes, r:Assoc))



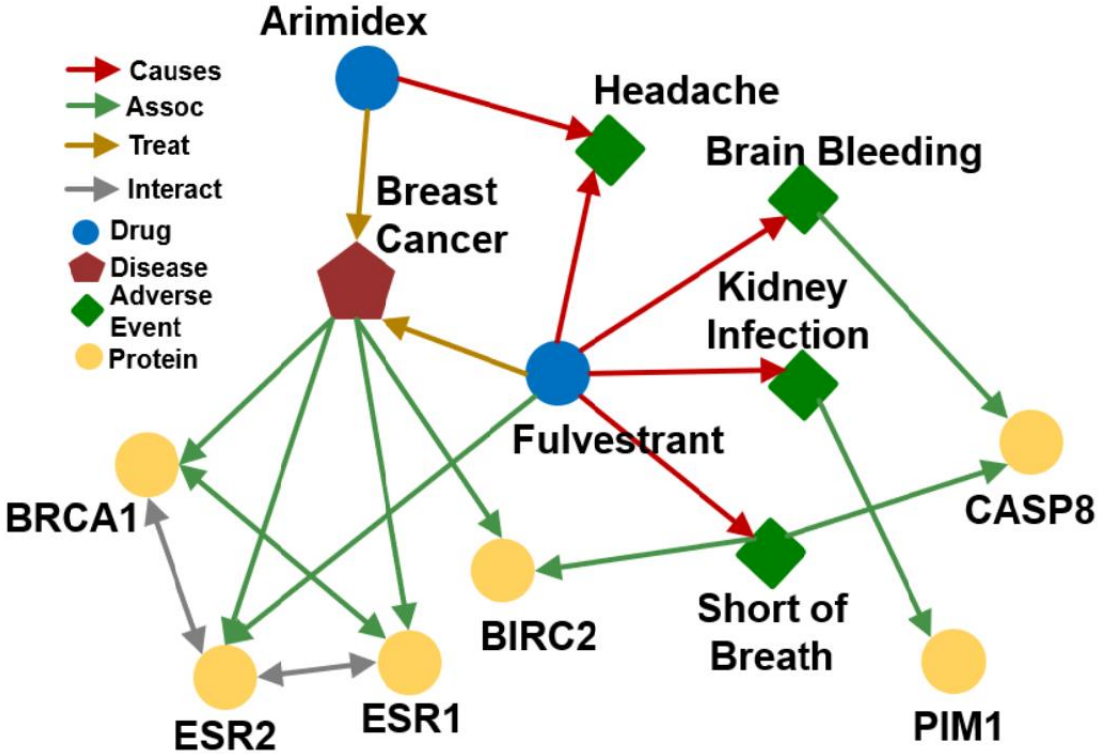


Path Queries

Question: "*What proteins are associated with adverse events caused by Fulvestrant ?*"

- **Query:** (e:Fulvestrant, (r:Causes, r:Assoc))

Given a KG, how to answer a path query?

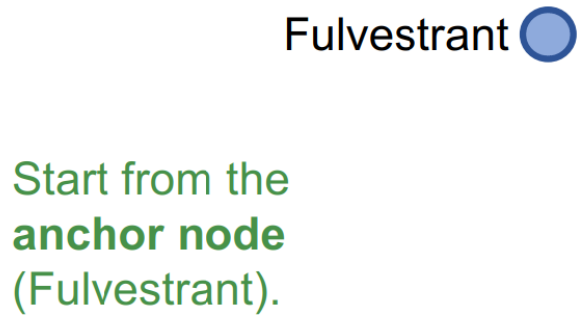


Traversing Knowledge Graphs

- **We answer path queries by traversing the KG:**

"*What proteins are associated with adverse events caused by Fulvestrant ?*"

- **Query:** (e:Fulvestrant, (r:Causes, r:Assoc))

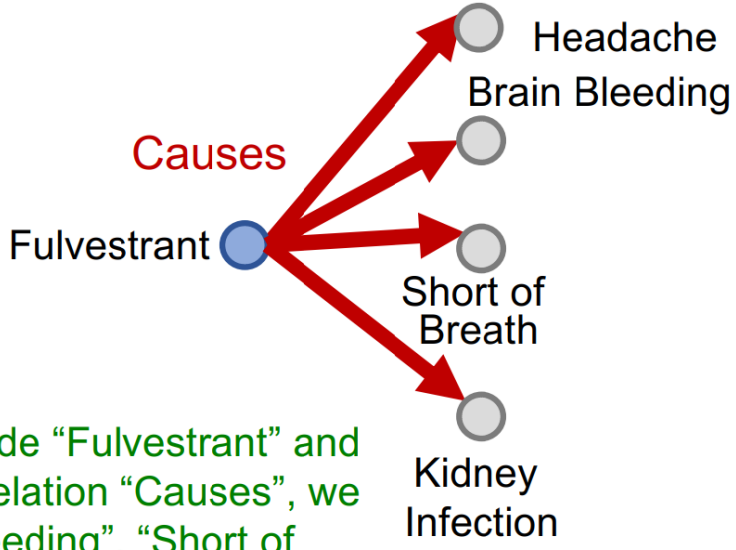


Traversing Knowledge Graphs

- **We answer path queries by traversing the KG:**

"*What proteins are associated with adverse events caused by Fulvestrant ?*"

- **Query:** (e:Fulvestrant, (r:Causes, r:Assoc))



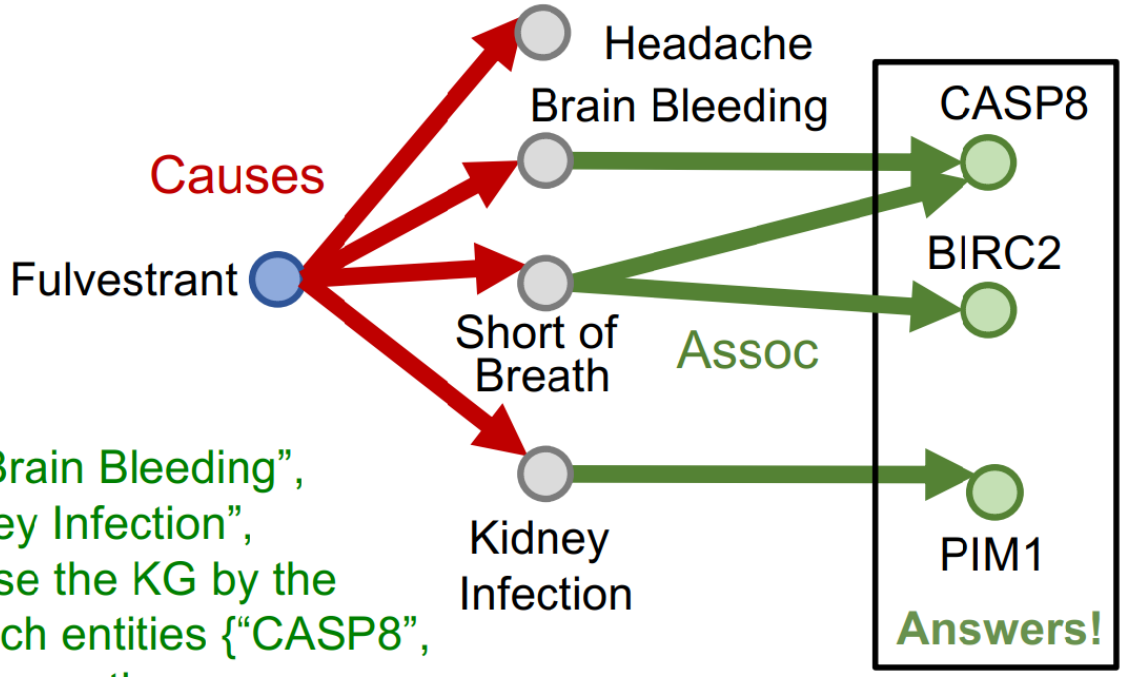
Start from the anchor node "Fulvestrant" and traverse the KG by the relation "Causes", we reach entities {"Brain Bleeding", "Short of Breath", "Kidney Infection", "Headache"}.

Traversing Knowledge Graphs

- **We answer path queries by traversing the KG:**

"*What proteins are associated with adverse events caused by Fulvestrant ?*"

- **Query:** (e:Fulvestrant, (r:Causes, r:Assoc))



Start from the nodes {"Brain Bleeding", "Short of Breath", "Kidney Infection", "Headache"} and traverse the KG by the relation "Assoc", we reach entities {"CASP8", "BIRC2", "PIM1"}. These are the answers.

However, KGs are incomplete

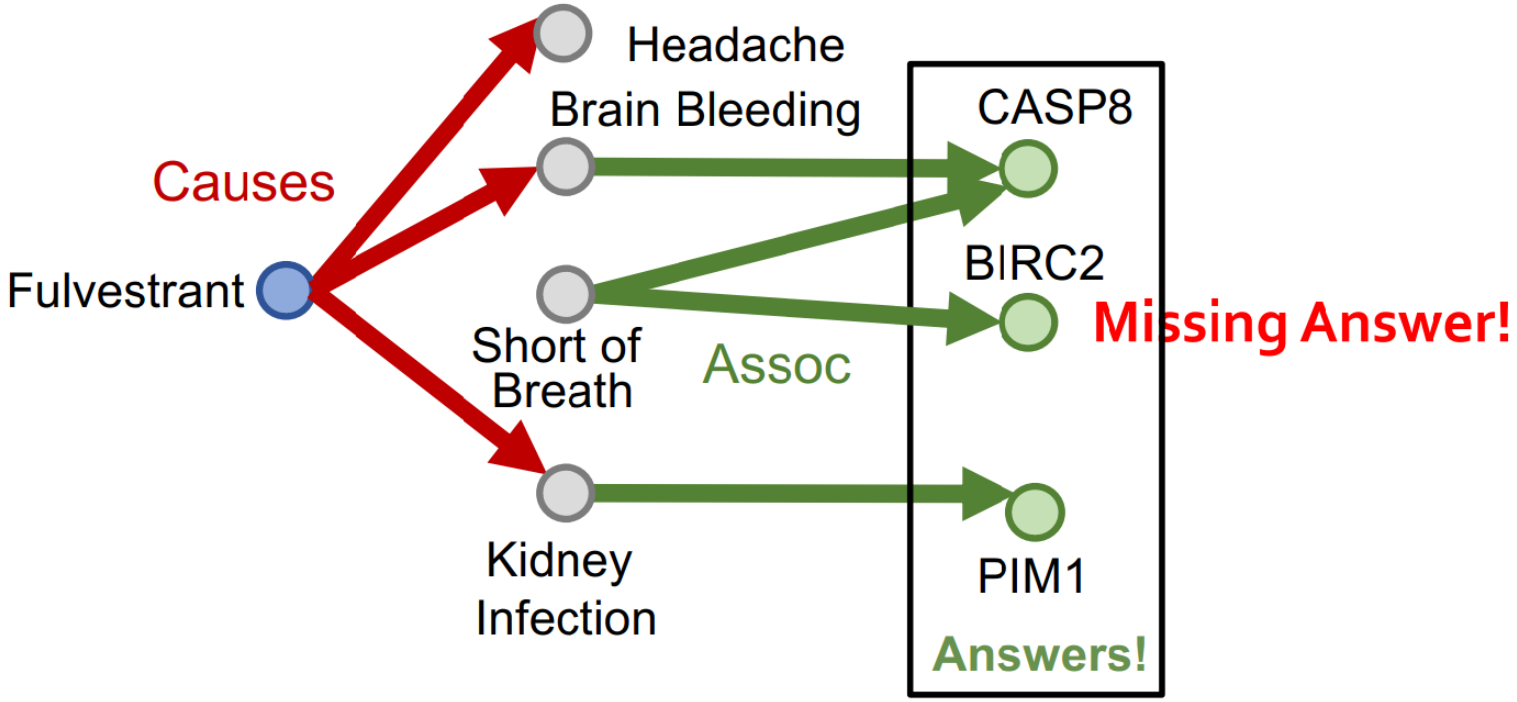
- **Answering queries seems easy: Just traverse the graph**
- **But KGs are incomplete and unknown:**
  - Many relations between entities are missing or are incomplete
    - For example, we lack all the biomedical knowledge
    - Enumerating all the facts takes non-trivial time and cost, we cannot hope that KGs will ever be fully complete
- **Due to KG incompleteness, one is not able to identify all the answer entities**

Example: Incomplete KG

- **We answer path queries by traversing the KG:**

"*What proteins are associated with adverse events caused by Fulvestrant ?*"

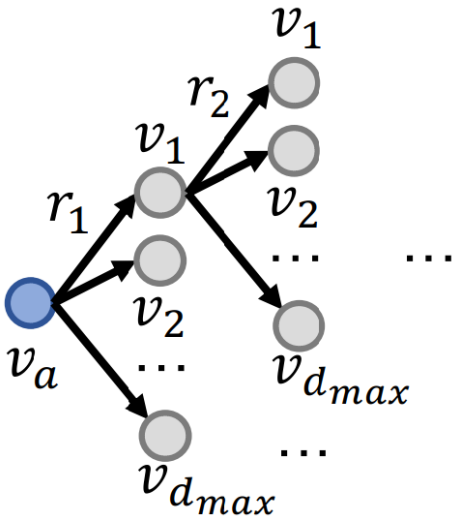
- **Query:** (e:Fulvestrant, (r:Causes, r:Assoc))



Can KG Completion help?

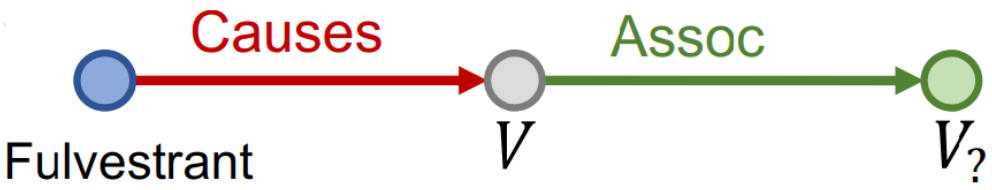
Can we first do KG completion and then traverse the completed (probabilistic) KG?

- **No!** The "completed" KG is a **dense graph**!
  - Most (h, r, t) triples (edge on KG) will have some non-zero probability
- Time complexity of traversing a dense KG is exponential as a function of the path length L: $\boldsymbol{O}(d_{max}^{L})$



Task: Predictive Queries

- **We need a way to answer path-based queries over an incomplete knowledge graph.**
- We want our approach to implicitly impute and account for the incomplete KG.
- **Task:** Predictive queries
  - Want to be able to answer arbitrary queries while implicitly imputing for the missing information.
  - **Generalization of the link prediction task**



Solutions for Predictive Queries

1) **Given entity embeddings, how do we answer an arbitrary query?**

- Path queries: Using a generalization of TransE.
- Conjunctive queries: Using Query2Box
- And-Or Queries: Using Query2Box and query rewriting

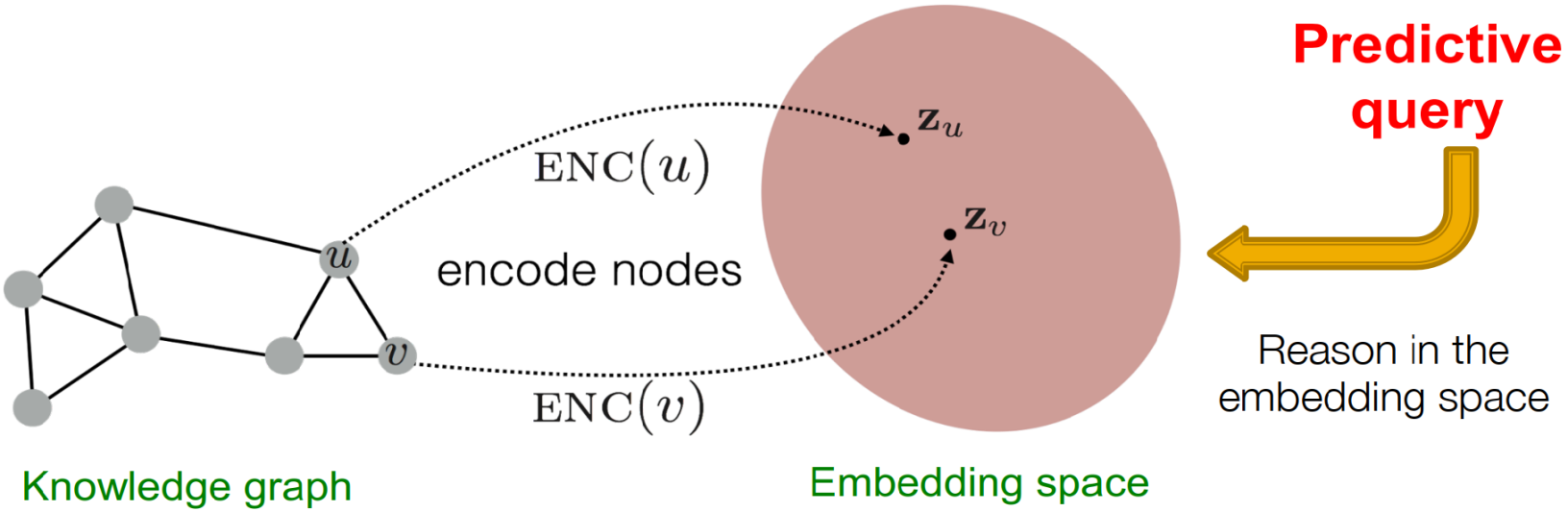
2) **How do we train the embeddings?**

- The process of determining entity and relation embeddings which allow us to embed a query

Neural Reasoning for Natural Language Query:

Answering Predictive Queries on Knowledge Graphs

General Ideas



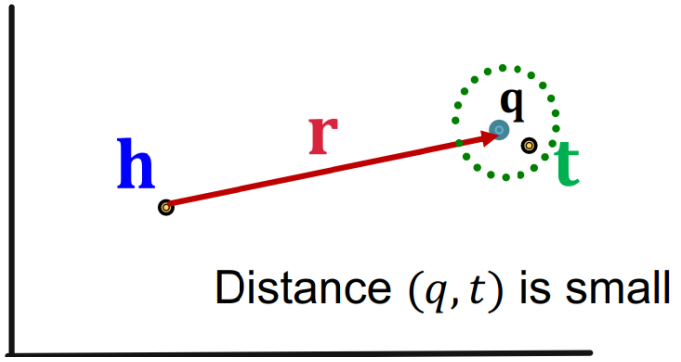
Map queries into embedding space. Learn to reason in that space.

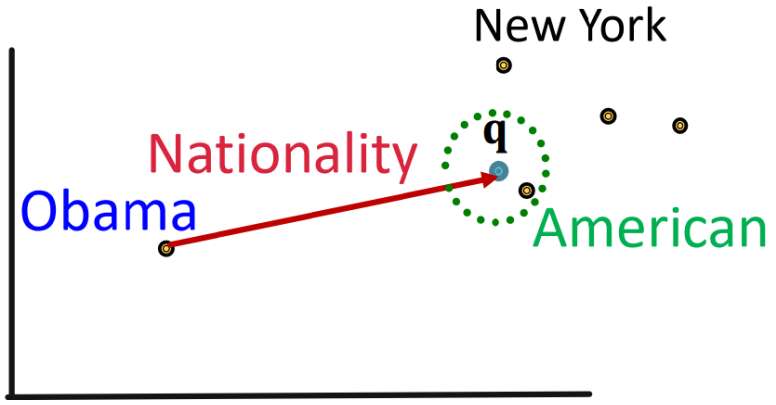
- Embed query into a single **point** in the Euclidean space: answer nodes are close to the query.
- **Query2Box**: Embed query into a hyper-rectangle (box) in the Euclidean space: answer nodes are enclosed in the box.

Embedding Logical Queries on Knowledge Graphs. Hamilton, et al., NeurIPS 2018
Query2box: Reasoning over Knowledge Graphs in Vector Space Using Box Embeddings. Ren, et al., ICLR 2020

Traversing KG in Vector Space

- **Key idea: Embed queries!**
  - Generalize TransE to multi-hop reasoning.
  - **Recap:** TransE: Translate $\mathbf{h}$ to $\mathbf{t}$ using $\mathbf{r}$ with score function:
    $f_{\mathbf{r}}(\mathbf{h}, \mathbf{t}) = -\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|$
  - Another way to interpret this is that:
    - Query embedding: $\mathbf{q} = \mathbf{h} + \mathbf{r}$
    - Goal: query embedding $\mathbf{q}$ is close to the answer embedding $\mathbf{t}$
      $f_{\mathbf{q}}(\mathbf{t}) = -\|\mathbf{q} - \mathbf{t}\|$

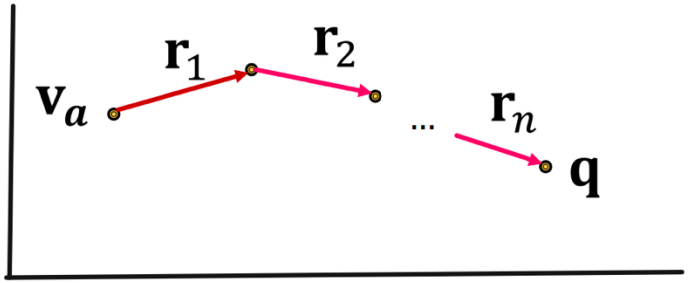




Traversing KG in Vector Space

- **Key idea: Embed queries!**
  - Generalize TransE to multi-hop reasoning.

Given a path query $q = (v_a, (r_1, \ldots, r_n))$,



$$\mathbf{q} = \mathbf{v}_a + \mathbf{r}_1 + \cdots + \mathbf{r}_n$$

  - The embedding process only involves vector addition, **independent of the number of entities** in the KG!

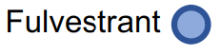
Traversing KG in Vector Space

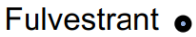
Embed path queries in vector space.

- **Question:** "*What proteins are associated with adverse events caused by Fulvestrant?*"
- **Query:** (e:Fulvestrant, (r:Causes , r:Assoc))

Follow the query plan:

Query Plan **Embedding Process**



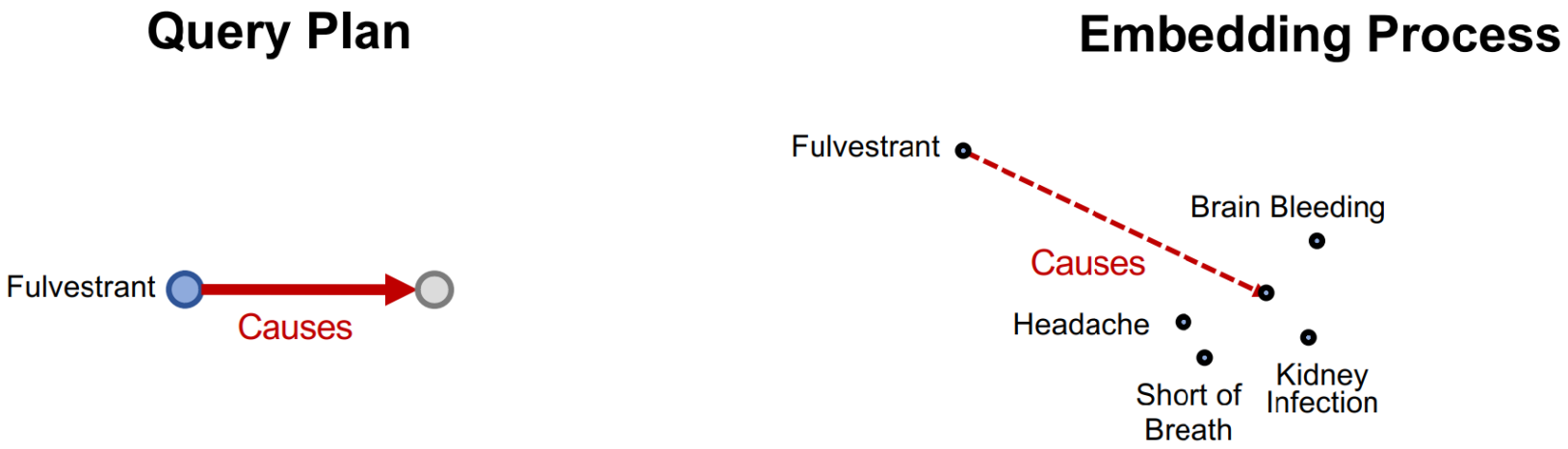


Traversing KG in Vector Space

Embed path queries in vector space.

- **Question:** "*What proteins are associated with adverse events caused by Fulvestrant?*"
- **Query:** (e:Fulvestrant, (r:Causes , r:Assoc))

Follow the query plan:

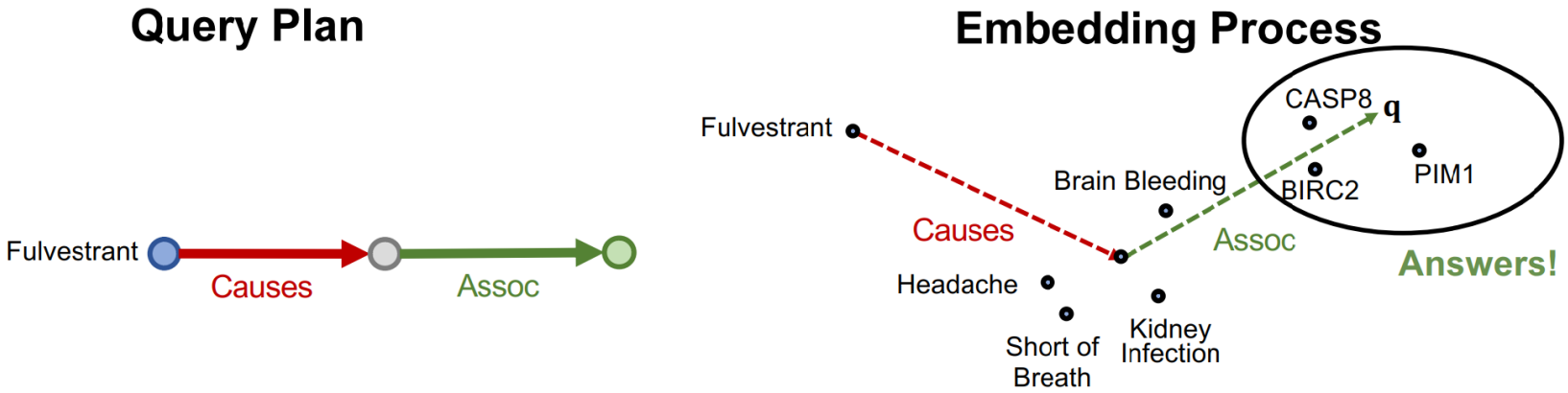


Traversing KG in Vector Space

Embed path queries in vector space.

- **Question:** "*What proteins are associated with adverse events caused by Fulvestrant?*"
- **Query:** (e:Fulvestrant, (r:Causes , r:Assoc))

Follow the query plan:



Traversing KG in Vector Space

Insight

- We can train **TransE** to optimize knowledge graph completion objective.
- Since **TransE** can naturally handle **compositional relations**, it can handle path queries by translating in the latent space **for multiple hops using addition of relation embeddings.**

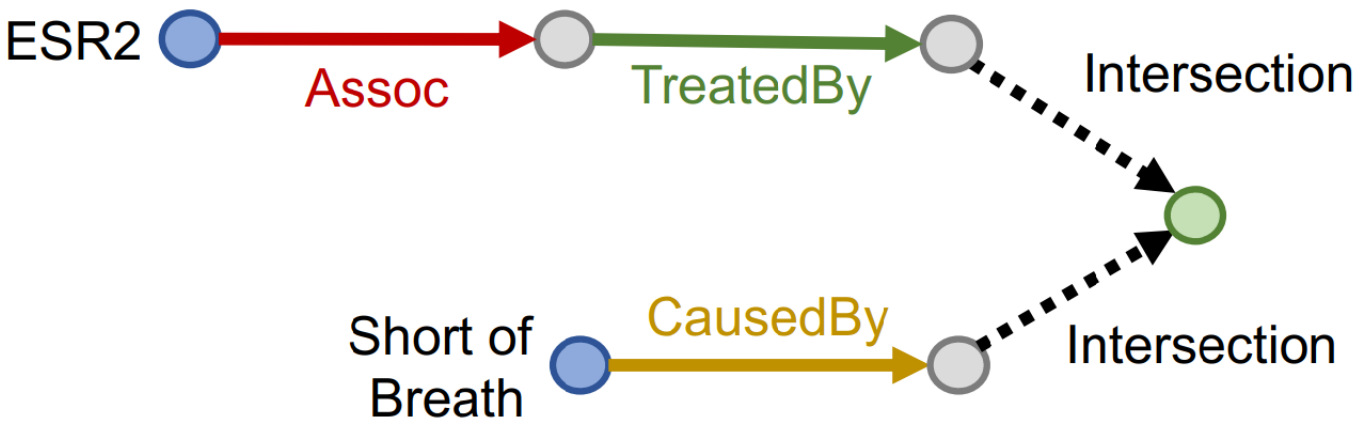
Conjunctive Queries

Can we answer **more complex queries** with logic conjunction operation?

- **Conjunctive Queries:** "*What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?*".

((**e:ESR2**, (r:Assoc, r:TreatedBy)), (**e:Short of Breath**, (r:CausedBy))

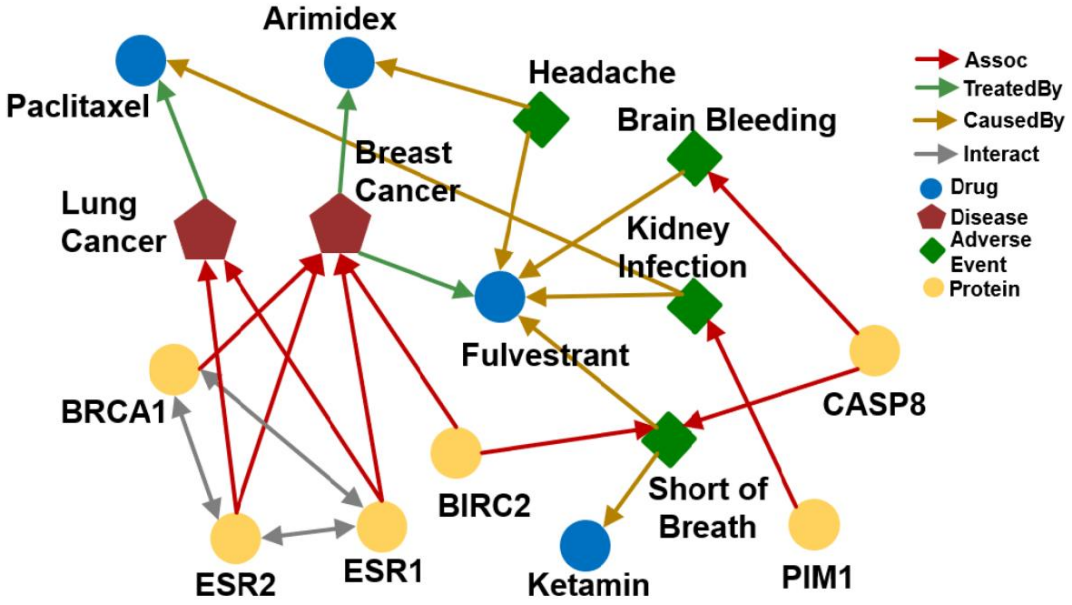
Query plan:



Conjunctive Queries

Conjunctive Queries: "*What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?*".
((**e:ESR2**, (r:Assoc, r:TreatedBy)), (**e:Short of Breath**, (r:CausedBy))

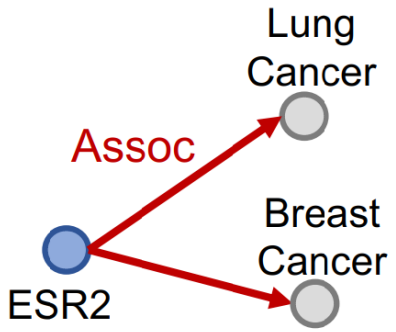
How do we answer the question by KG traversal?



Traversing KG for Conjunctive Queries

Conjunctive Queries: "*What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?*".
((**e:ESR2**, (r:Assoc, r:TreatedBy)), (**e:Short of Breath**, (r:CausedBy))

Traverse KG from **anchor nodes**: **ESR2** and **Short of Breath:**

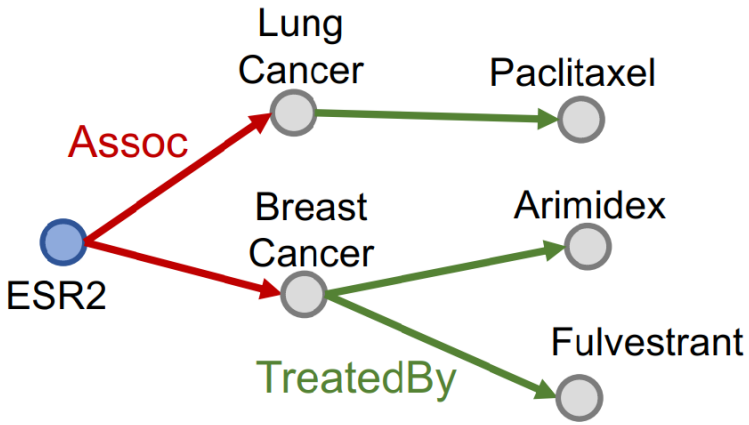


Traverse from the first anchor "ESR2" by relation "Assoc", we reach a set of entities {"Lung Cancer", "Breast Cancer"}

Traversing KG for Conjunctive Queries

Conjunctive Queries: "*What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?*".
((**e:ESR2**, (r:Assoc, r:TreatedBy)), (**e:Short of Breath**, (r:CausedBy))

Traverse KG from **anchor nodes**: **ESR2** and **Short of Breath:**

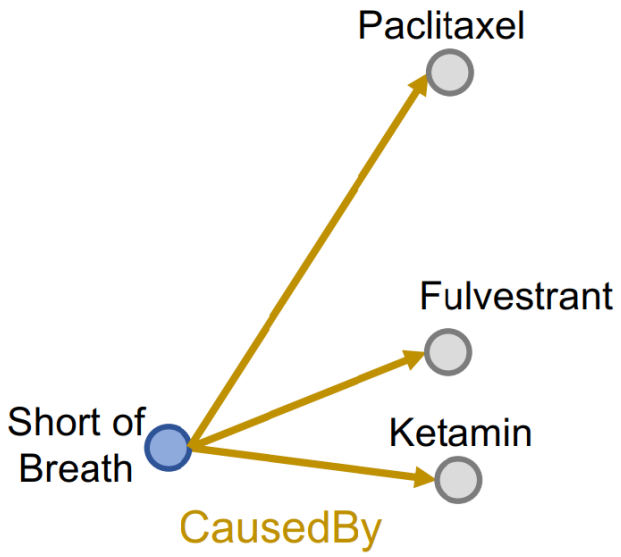


Traverse from the set of entities {"Lung Cancer", "Breast Cancer"} by relation TreatedBy, we reach a set of entities {"Paclitaxel", "Arimidex", "Fulvestrant"}

Traversing KG for Conjunctive Queries

Conjunctive Queries: "*What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?*".
((**e:ESR2**, (r:Assoc, r:TreatedBy)), (**e:Short of Breath**, (r:CausedBy))

Traverse KG from **anchor nodes**: **ESR2** and **Short of Breath:**

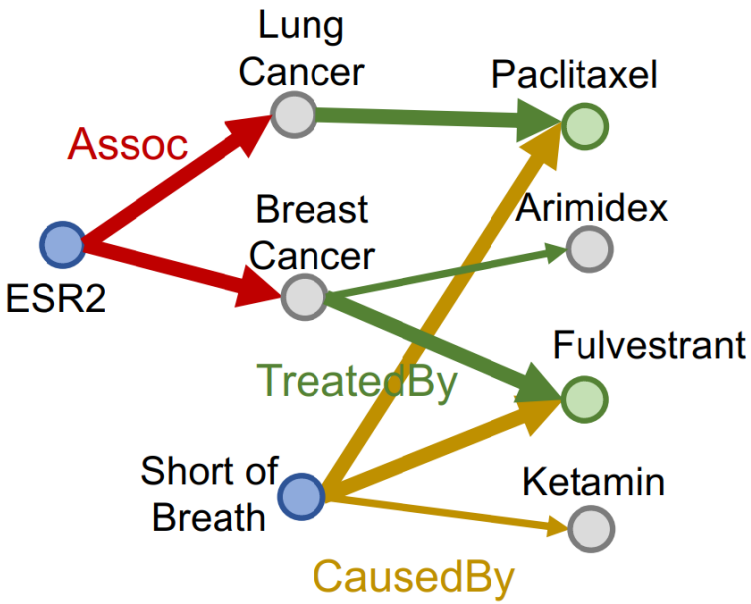


Traverse from the second anchor "Short of Breath" by relation "CausedBy", we reach a set of entities {"Fulvestrant", "Ketamin", "Paclitaxel"}

Traversing KG for Conjunctive Queries

Conjunctive Queries: "*What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?*".
((**e:ESR2**, (r:Assoc, r:TreatedBy)), (**e:Short of Breath**, (r:CausedBy))

Traverse KG from **anchor nodes**: **ESR2** and **Short of Breath:**

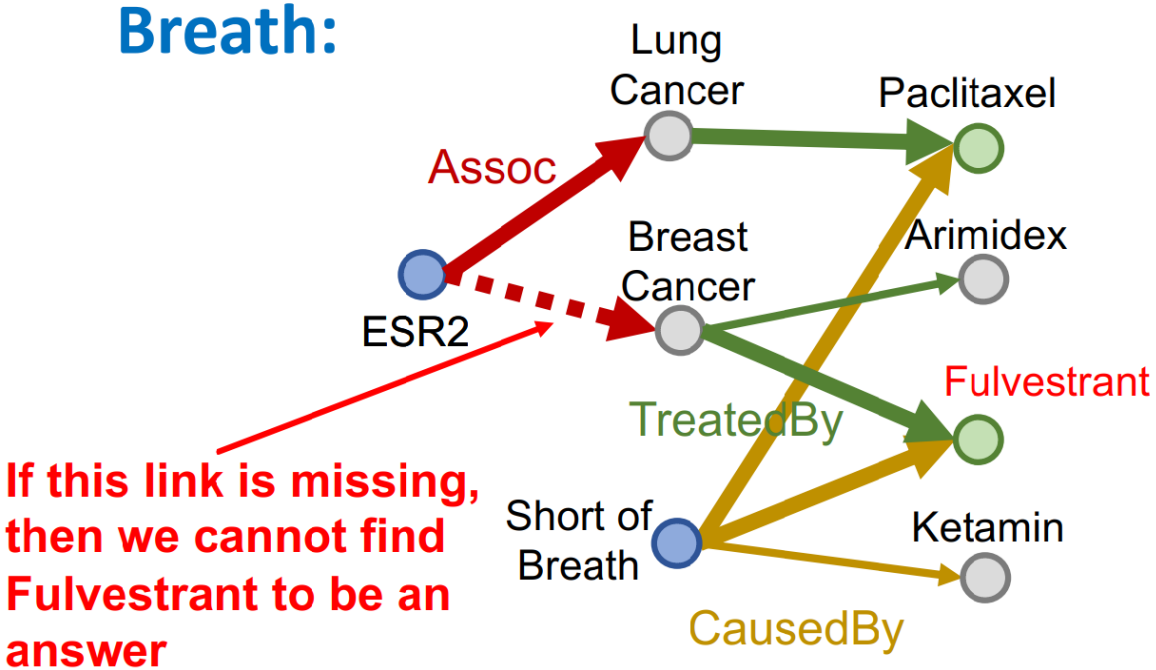


We take intersection between the two sets and get the answers {"Fulvestrant", "Paclitaxel"}

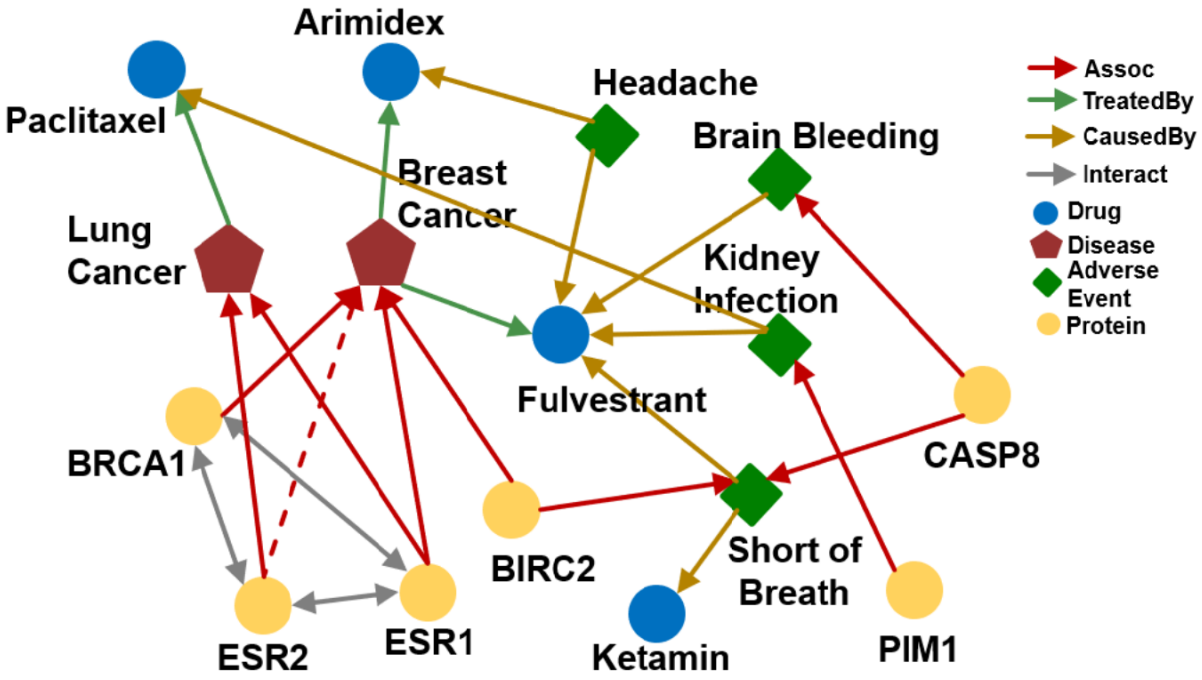
Traversing KG for Conjunctive Queries

Conjunctive Queries: "*What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?*".
((**e:ESR2**, (r:Assoc, r:TreatedBy)), (**e:Short of Breath**, (r:CausedBy))

Traverse KG from **anchor nodes**: **ESR2** and **Short of Breath:**



Traversing KG for Conjunctive Queries



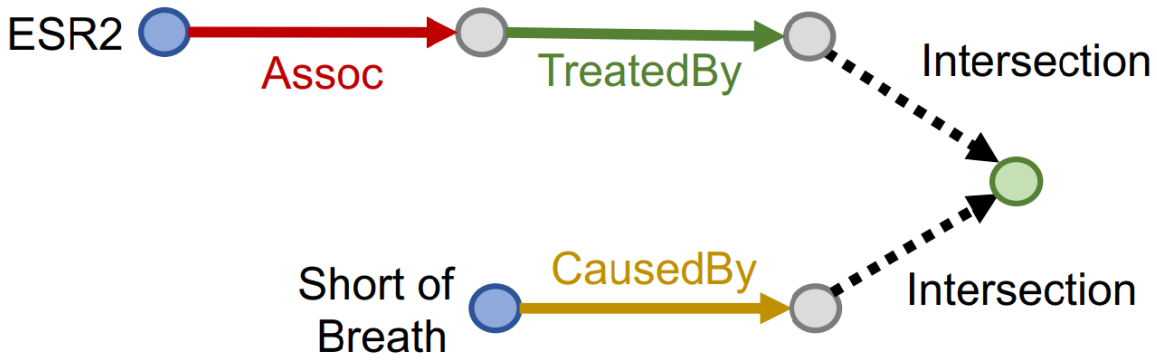
- How can we use embeddings to implicitly impute the **missing (ESR2, Assoc, Breast Cancer)?**
- **Intuition:** **ESR2** interacts with both **BRCA1** and **ESR1**. Both proteins are associated with **breast cancer**.

Traversing KG for Conjunctive Queries

Conjunctive Queries: "*What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?*".

((**e:ESR2**, (r:Assoc, r:TreatedBy)), (**e:Short of Breath**, (r:CausedBy))

Query plan:



Each intermediate node represents a set of entities, how do we represent it? How do we define the intersection operation in the latent space?

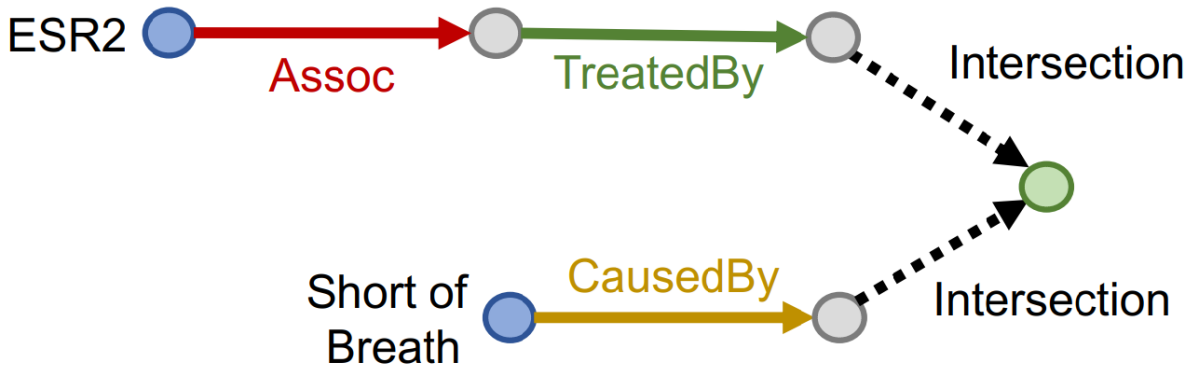
Neural Reasoning for Natural Language Query:

Query2Box – Reasoning over Knowledge Graphs using Box Embeddings

Conjunctive Queries

How can we answer **more complex queries** with logic conjunction operation?

Query plan:



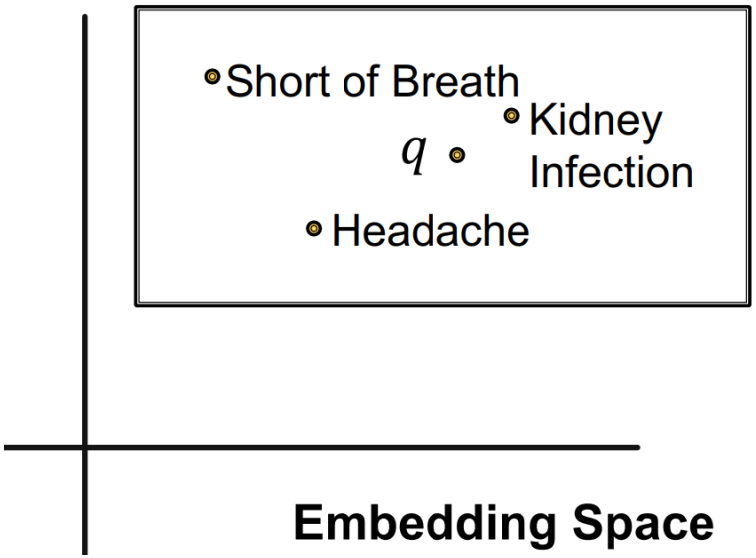
(1) Each intermediate node represents a set of entities; how do we represent it?

(2) How do we define the intersection operation in the latent space?

Box Embeddings

- Embed queries with **hyper-rectangles (boxes)**

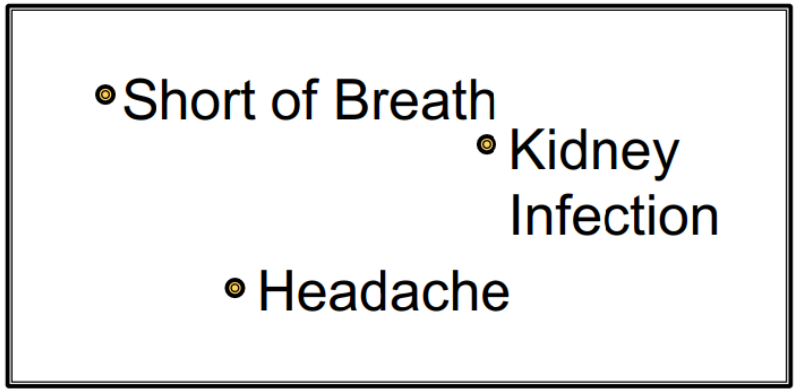
$$\mathbf{q} = (Center(q), Offset(q))$$



For example, we can embed the adverse events of Fulvestrant with **a box that enclose all the answer entities**.

Key Insight: Intersection

- Intersection of boxes is well-defined!
- When we traverse the KG to find the answers, each step produces a set of reachable entities.
- **How can we better model these sets?**
  - Boxes are a **powerful abstraction**, as we can project the center and control the offset to model the set of entities enclosed in the box.



Embed with Box Embeddings

Things to figure out:

- **Entity embeddings** (# params: $d|V|$):
  - Entities are seen as zero-volume boxes
- **Relation embeddings** (# params $2d|R|$)
  - Each relation takes a box and produces a new box
- **Intersection operator f :**
  - New operator, inputs are boxes and output is a box
  - Intuitively models intersection of boxes

Notation
d: out degree
|V|: # entities
|R|: # relations

Embed with Box Embeddings

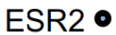
Embed queries in vector space: "*What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?*".
((**e:ESR2**, (r:Assoc, r:TreatedBy)), (**e:Short of Breath**, (r:CausedBy))

Traverse KG from **anchor nodes**: **ESR2** and **Short of Breath:**

Query plan

?

Embedding Space



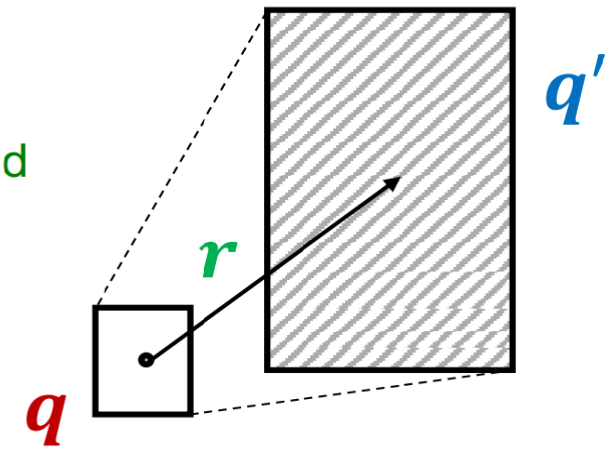
Projection Operator

Projection Operator $\mathcal{P}$

- **Intuition**:
  - Take the current box as input and use the relation embedding to **project and expand** the box!
- $\mathcal{P}$: Box × Relation → Box

$$Cen(q') = Cen(q) + Cen(r)$$
$$Off(q') = Off(q) + Off(r)$$

"×" (cross) means the projection operator is a **relation** from any box and **relation** to a new box



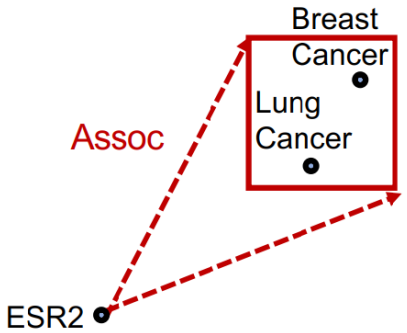
Embed with Box Embeddings

Embed queries in vector space: "*What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?*".
((**e:ESR2**, (r:Assoc, r:TreatedBy)), (**e:Short of Breath**, (r:CausedBy))

- Traverse KG from **anchor nodes**: **ESR2** and **Short of Breath:**
- Use **projection operator** again following the query plan.

Query Plan

Embedding Space

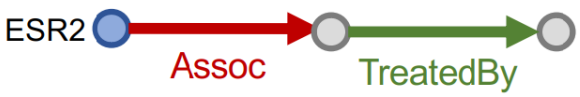


Embed with Box Embeddings

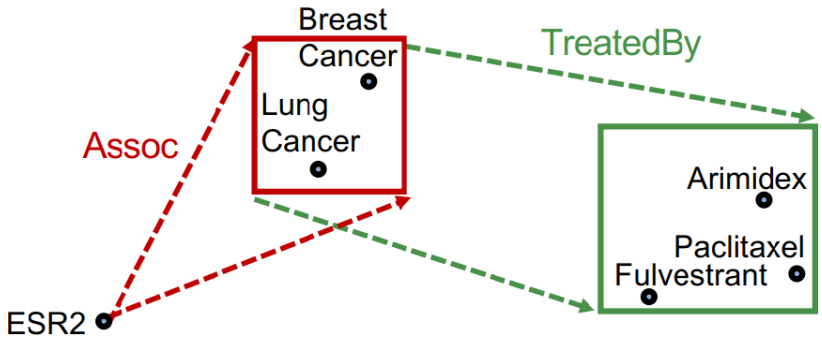
Embed queries in vector space: "*What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?*".
((**e:ESR2**, (r:Assoc, r:TreatedBy)), (**e:Short of Breath**, (r:CausedBy))

- Use **projection operator** again following the query plan.

Query Plan



Embedding Space

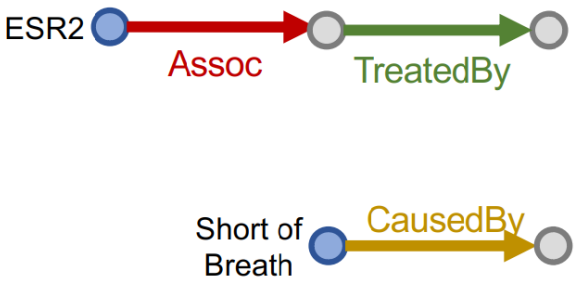


Embed with Box Embeddings

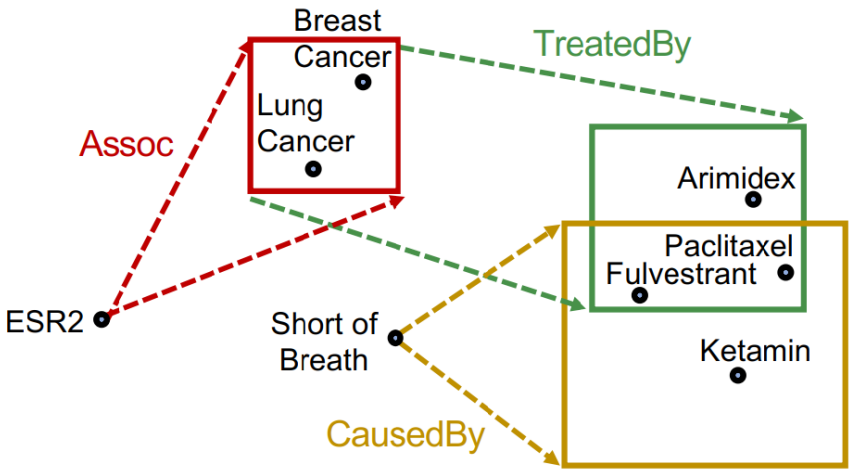
Embed queries in vector space: "*What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?*".
((**e:ESR2**, (r:Assoc, r:TreatedBy)), (**e:Short of Breath**, (r:CausedBy))

- Use **projection operator** again following the query plan.

Query Plan



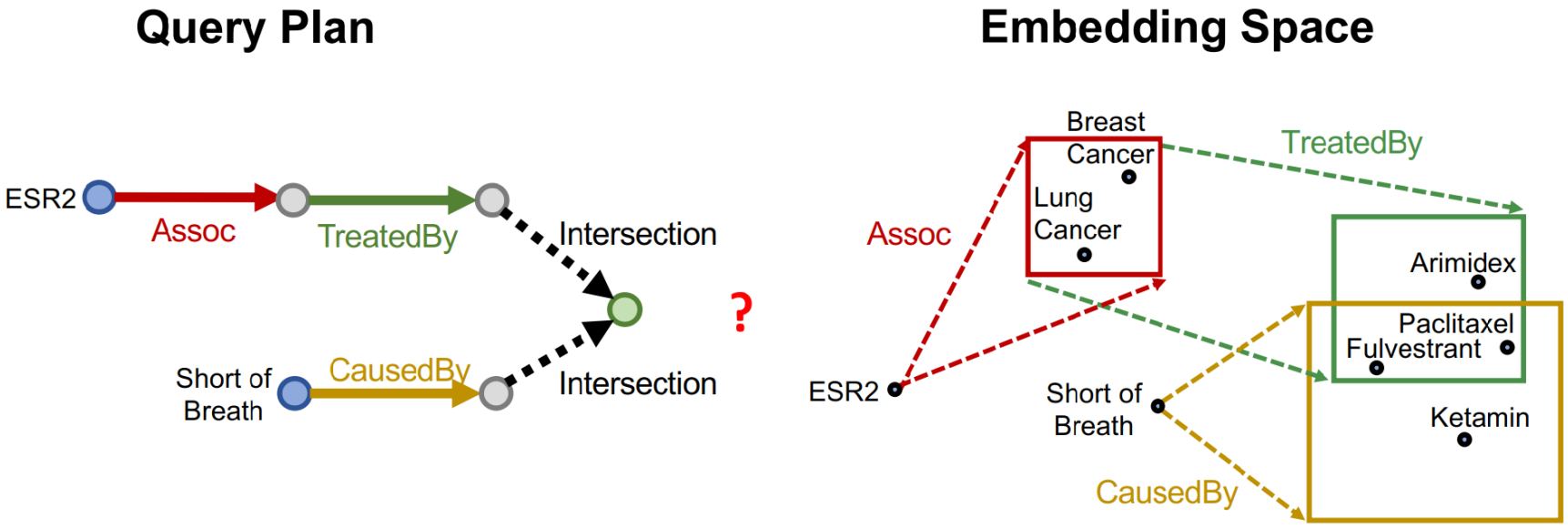
Embedding Space



Embed with Box Embeddings

Embed queries in vector space: "*What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?*".
((**e:ESR2**, (r:Assoc, r:TreatedBy)), (**e:Short of Breath**, (r:CausedBy))

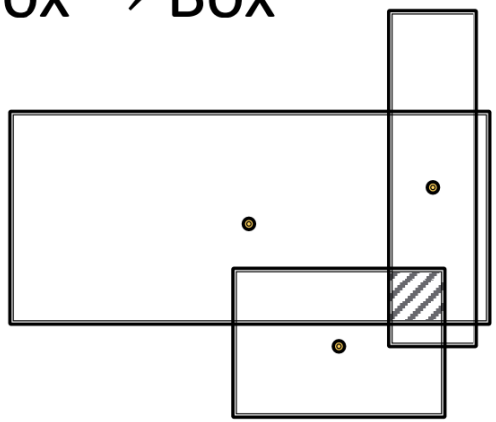
- **How do we take intersection of boxes?**



Intersection Operator

Geometric Intersection Operator $\mathcal{I}$

- Take multiple boxes as input and produce the intersection box
- **Intuition:**
  - The center of the new box should be **"close"** to the centers of the input boxes
  - The offset (box size) should **shrink** (since the size of the intersected set is **smaller** than the size of all the input set)
- $\mathcal{I}$: Box $\times \cdots \times$ Box $\rightarrow$ Box



Intersection Operator

Geometric Intersection Operator $\mathcal{I}$

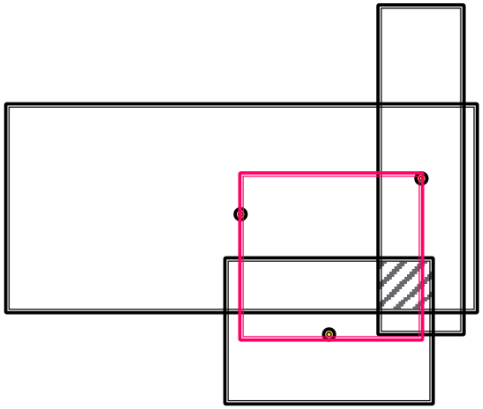
- $\mathcal{I}$: Box $\times \cdots \times$ Box $\rightarrow$ Box

Hadamard product
(element-wise product)

$$Cen(q_{inter}) = \sum_i \boldsymbol{w}_i \odot Cen(q_i)$$

$$\boldsymbol{w}_i = \frac{\exp(f_{cen}(Cen(q_i)))}{\sum_j \exp(f_{cen}(Cen(q_j)))}$$

$$Cen(q_i) \in \mathbb{R}^d$$
$$\boldsymbol{w}_i \in \mathbb{R}^d$$



Intuition: The center should be in the red region!
Implementation: The center is a **weighted sum** of the input box centers

$\boldsymbol{w}_i \in \mathbb{R}^d$ is calculated by a neural network f_{cen} (with trainable weights)

$\boldsymbol{w}_i$ represents a "self-attention" score for the center of each input $Cen(q_i)$.

Intersection Operator

Geometric Intersection Operator $\mathcal{I}$

- $\mathcal{I}$: Box $\times \cdots \times$ Box $\rightarrow$ Box

$$Off(q_{inter}) = \min\big(Off(q_1), \ldots, Off(q_n)\big) \odot \sigma(f_{off}(Off(q_1), \ldots, Off(q_n)))$$

guarantees shrinking

Sigmoid function: squashes output in (0,1)

f_{off} is a neural network (with trainable parameters) that extracts the representation of the input boxes to increase expressiveness

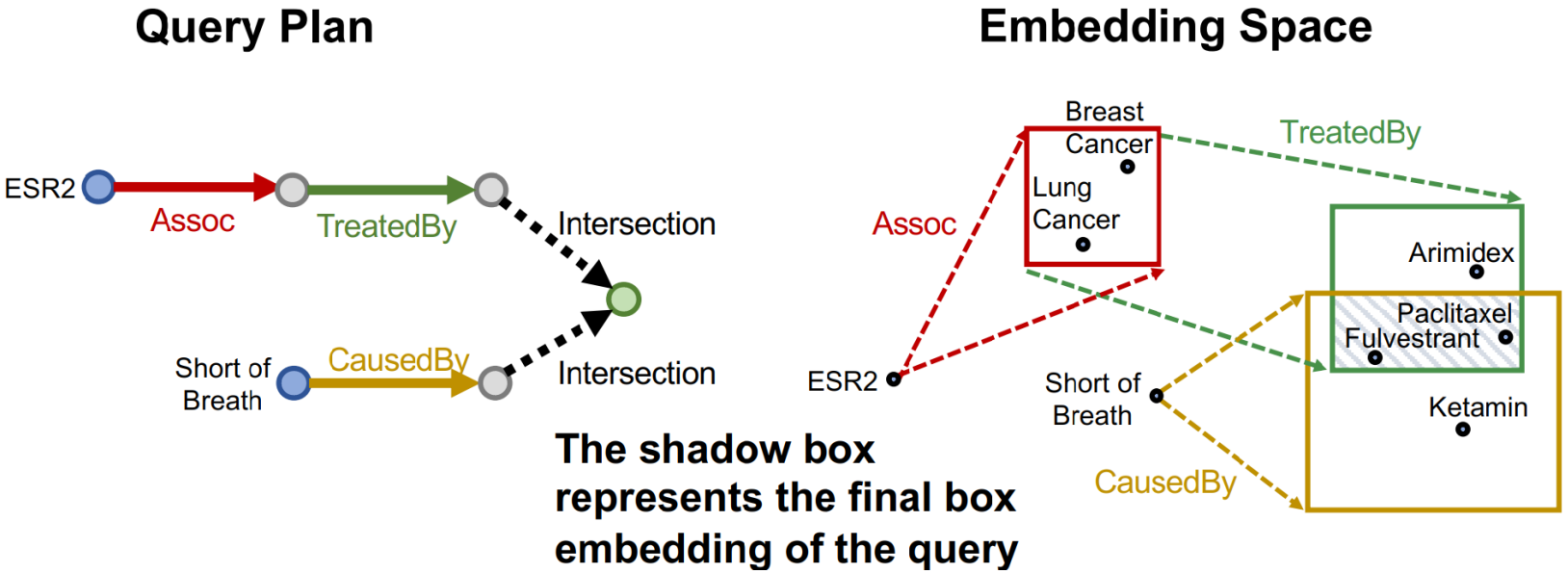
Intuition: The offset should be smaller than the offset of the input box

Implementation: We first **take minimum** of the offset of the input box, and then we make the model more expressive by introducing a new function f_{off} to extract the **representation** of the input boxes with a **sigmoid function** to **guarantee shrinking**.

Embed with Box Embeddings

Embed queries in vector space: "*What are drugs that cause Short of Breath and treat diseases associated with protein ESR2?*".
((**e:ESR2**, (r:Assoc, r:TreatedBy)), (**e:Short of Breath**, (r:CausedBy))

- **Use box intersection operator**



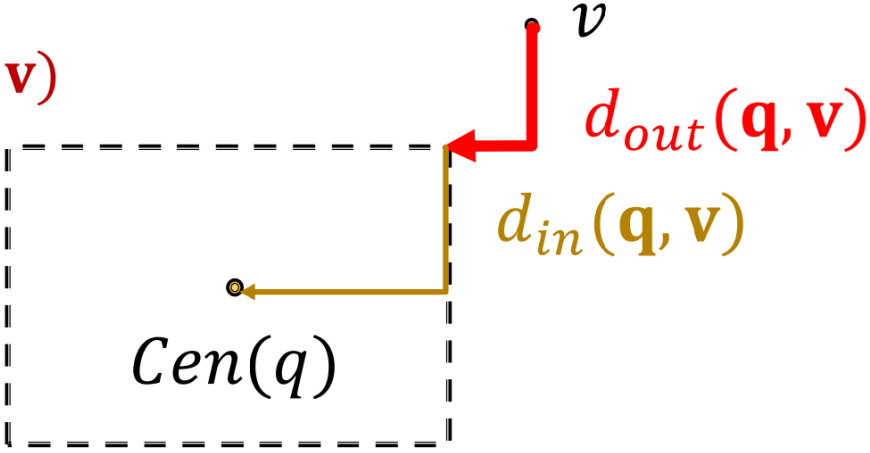
Entity-to-Box Distance

- **How do we define the score function $f_q(v)$ (negative distance)?**

  ($f_q(v)$ captures inverse distance of a node v as answer to q)
- Given a query box $\mathbf{q}$ and entity embedding (box) $\mathbf{v}$,

$$d_{box}(\mathbf{q}, \mathbf{v}) = d_{out}(\mathbf{q}, \mathbf{v}) + \alpha \cdot d_{in}(\mathbf{q}, \mathbf{v})$$

  where $0 < \alpha < 1$.
- **Intuition**: if the point is enclosed in the box, the distance should be **downweighted**.
- $f_q(v) = -d_{box}(\mathbf{q}, \mathbf{v})$



Extending to Union Operator

- Can we embed complex queries with union? E.g.: "What drug can treat breast cancer or lung cancer?"
- **Conjunctive queries** + **disjunction** is called **E**xistential **P**ositive **F**irst-**o**rder (EPFO) queries. We'll refer to them as **AND-OR** queries.
- **Can we also design a disjunction operator and embed AND-OR queries in low-dimensional vector space?**

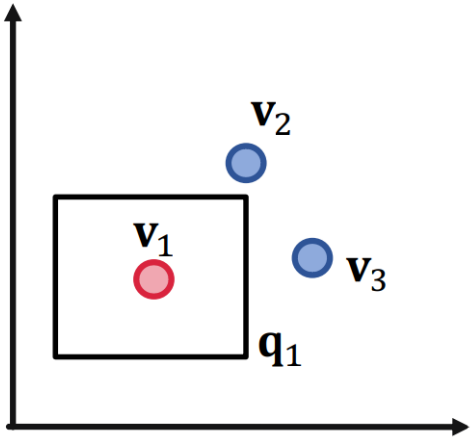
Embedding AND-OR Queries

- **Can we embed AND-OR queries in a low-dimensional vector space?**
- **No! Intuition**: Allowing **union** over **arbitrary queries** requires **high-dimensional** embeddings!

- **Example:**
  - Given 3 queries q_1, q_2, q_3, with answer sets:
  - $[\![q_1]\!] = \{v_1\}, [\![q_2]\!] = \{v_2\}, [\![q_3]\!] = \{v_3\}$
  - If we allow union operation, can we embed them in a **two-dimensional** plane?

Embedding AND-OR Queries

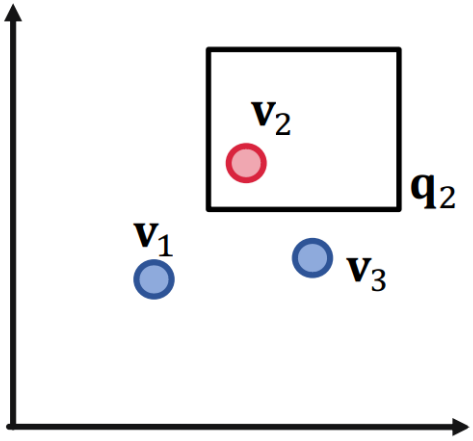
- Example
  - Given 3 queries q_1, q_2, q_3, with answer sets:
  - $[\![q_1]\!] = \{v_1\}, [\![q_2]\!] = \{v_2\}, [\![q_3]\!] = \{v_3\}$
  - If we allow union operation, can we embed them in two-dimensional plane?



We want **red dots (answers)** to be in the box while the **blue dots (negative answers)** to be outside the box

Embedding AND-OR Queries

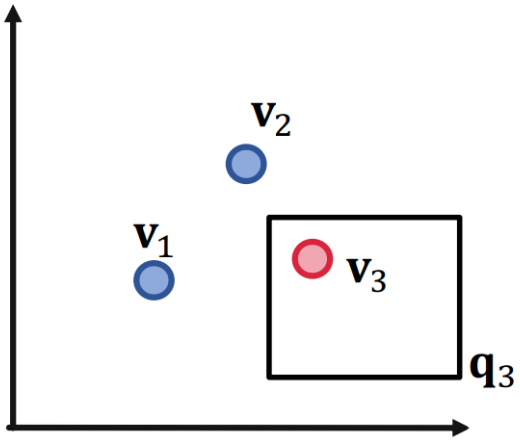
- Example
  - Given 3 queries q_1, q_2, q_3, with answer sets:
  - $[\![q_1]\!] = \{v_1\}, [\![q_2]\!] = \{v_2\}, [\![q_3]\!] = \{v_3\}$
  - If we allow union operation, can we embed them in two-dimensional plane?



We want **red dots (answers)** to be in the box while the **blue dots (negative answers)** to be outside the box

Embedding AND-OR Queries

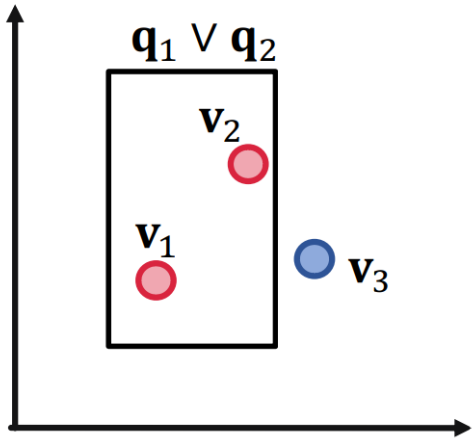
- Example
  - Given 3 queries q_1, q_2, q_3, with answer sets:
  - $[\![q_1]\!] = \{v_1\}, [\![q_2]\!] = \{v_2\}, [\![q_3]\!] = \{v_3\}$
  - If we allow union operation, can we embed them in two-dimensional plane?



We want **red dots (answers)** to be in the box while the **blue dots (negative answers)** to be outside the box

Embedding AND-OR Queries

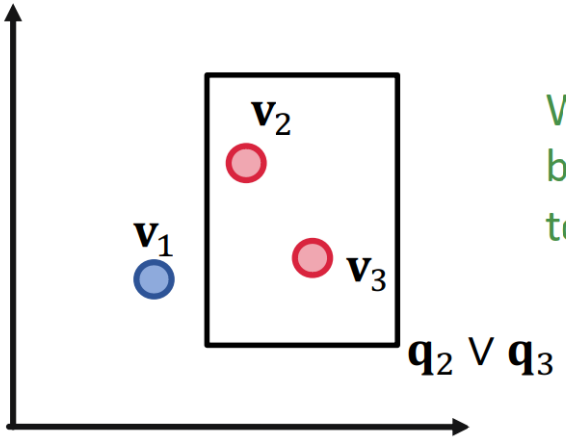
- Example
  - Given 3 queries q_1, q_2, q_3, with answer sets:
  - $[\![q_1]\!] = \{v_1\}, [\![q_2]\!] = \{v_2\}, [\![q_3]\!] = \{v_3\}$
  - If we allow union operation, can we embed them in two-dimensional plane?



We want **red dots (answers)** to be in the box while the **blue dots (negative answers)** to be outside the box

Embedding AND-OR Queries

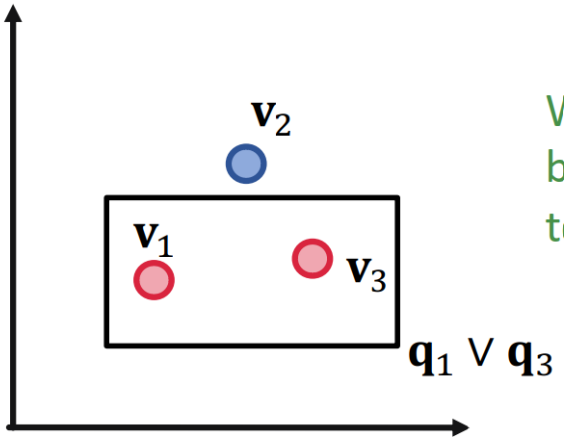
- Example
  - Given 3 queries q_1, q_2, q_3, with answer sets:
  - $[\![q_1]\!] = \{v_1\}, [\![q_2]\!] = \{v_2\}, [\![q_3]\!] = \{v_3\}$
  - If we allow union operation, can we embed them in two-dimensional plane?



We want **red dots (answers)** to be in the box while the **blue dots (negative answers)** to be outside the box

Embedding AND-OR Queries

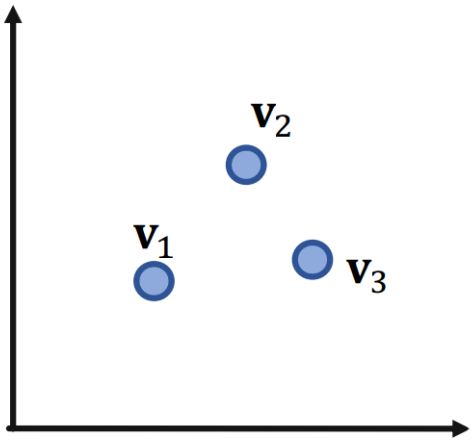
- Example
  - Given 3 queries q_1, q_2, q_3, with answer sets:
  - $[\![q_1]\!] = \{v_1\}, [\![q_2]\!] = \{v_2\}, [\![q_3]\!] = \{v_3\}$
  - If we allow union operation, can we embed them in two-dimensional plane?



We want **red dots (answers)** to be in the box while the **blue dots (negative answers)** to be outside the box

Embedding AND-OR Queries

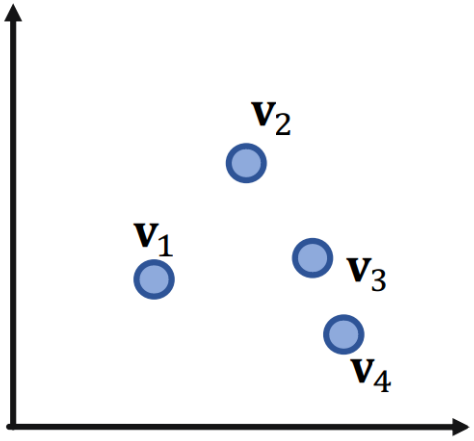
- Example
  - Given 3 queries q_1, q_2, q_3, with answer sets:
  - $[\![q_1]\!] = \{v_1\}, [\![q_2]\!] = \{v_2\}, [\![q_3]\!] = \{v_3\}$
  - If we allow union operation, can we embed them in two-dimensional plane?



For 3 points, 2-dimension is okay!
How about 4 points?

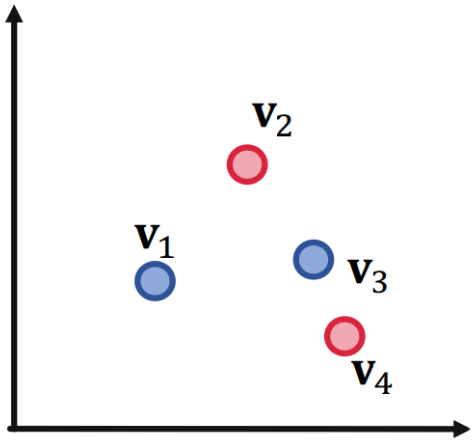
Embedding AND-OR Queries

- Example 2
  - Given 4 queries q_1, q_2, q_3, q_4 with answers:
  - $[\![q_1]\!] = \{v_1\}, [\![q_2]\!] = \{v_2\}, [\![q_3]\!] = \{v_3\}, [\![q_4]\!] = \{v_4\},$
  - If we allow union operation, can we embed them in two-dimensional plane?



Embedding AND-OR Queries

- Example 2
  - Given 4 queries q_1, q_2, q_3, q_4 with answers:
  - $[\![q_1]\!] = \{v_1\}, [\![q_2]\!] = \{v_2\}, [\![q_3]\!] = \{v_3\}, [\![q_4]\!] = \{v_4\},$
  - If we allow union operation, can we embed them in two-dimensional plane?



We cannot design a box embedding for $q_2 \vee q_4$, that only v_2 and v_4 are in the box but v_1 and v_3 are outside the box.

Embedding AND-OR Queries

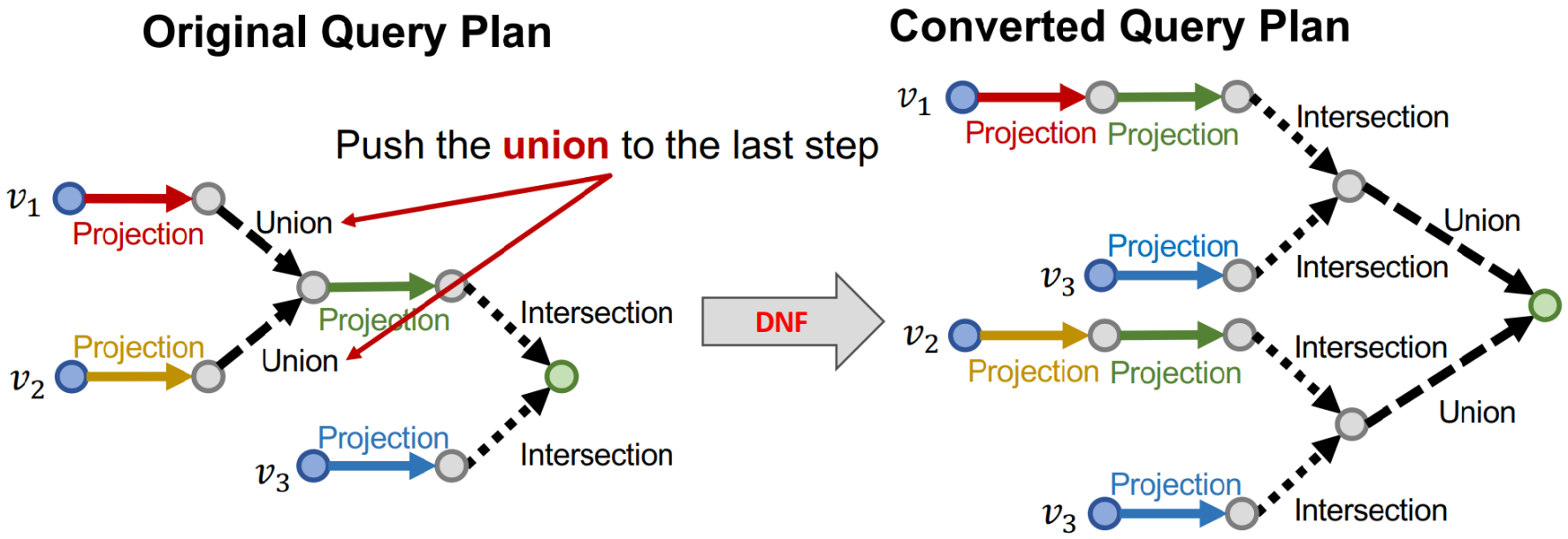
Can we embed AND-OR queries in low dimensional vector space?

- **Conclusion**: Given any M conjunctive queries $q_1, \dots, q_M$ with non-overlapping answers, we need dimensionality of $\Theta(M)$ to handle all OR queries.
  - For real-world KG, such as FB15k, we find $M \geq 13{,}365$, where $|V| = 14{,}951$.
  - Remember, this is for arbitrary OR queries.

Embedding AND-OR Queries

Since we **cannot embed** AND-OR queries in low dimensional space, can we still handle them?

- **Key idea:** take all unions out and only do union **at the last step**!



Disjunctive Normal Form

- Any **AND-OR query** can be transformed into equivalent DNF, i.e., **disjunction of conjunctive queries**.

- **Given any AND-OR query q,**

$$q = q_1 \vee q_2 \vee \cdots \vee q_m$$

where q_i is a **conjunctive query**.

- Now we can first embed each q_i and then "**aggregate**" at the last step!

Distance between q and an Entity

- **Distance** between entity embedding and a DNF $q = q_1 \vee q_2 \vee \cdots \vee q_m$ is defined as:

$$d_{box}(\mathbf{q}, \mathbf{v}) = min(d_{box}(\mathbf{q}_1, \mathbf{v}), \ldots, d_{box}(\mathbf{q}_m, \mathbf{v}))$$

- **Intuition:**
  - As long as v is the answer to one conjunctive query q_i, then v should be the answer to q
  - As long as $\mathbf{v}$ is close to one conjunctive query $\mathbf{q}_i$, then $\mathbf{v}$ should be close to $\mathbf{q}$ **in the embedding space**

Distance between q and an Entity

- **Distance** between entity embedding and a DNF $q = q_1 \vee q_2 \vee \cdots \vee q_m$ is defined as:

$$d_{box}(\mathbf{q}, \mathbf{v}) = min(d_{box}(\mathbf{q}_1, \mathbf{v}), \ldots, d_{box}(\mathbf{q}_m, \mathbf{v}))$$

- **The process of embedding any AND-OR query q**
  1. Transform q to **equivalent DNF** $q_1 \vee \cdots \vee q_m$
  2. **Embed** q_1 to q_m
  3. Calculate the (box) distance $d_{box}(\mathbf{q}_i, \mathbf{v})$
  4. Take the **minimum** of all distance
  5. **The final score** $f_q(v) = -d_{box}(\mathbf{q}, \mathbf{v})$

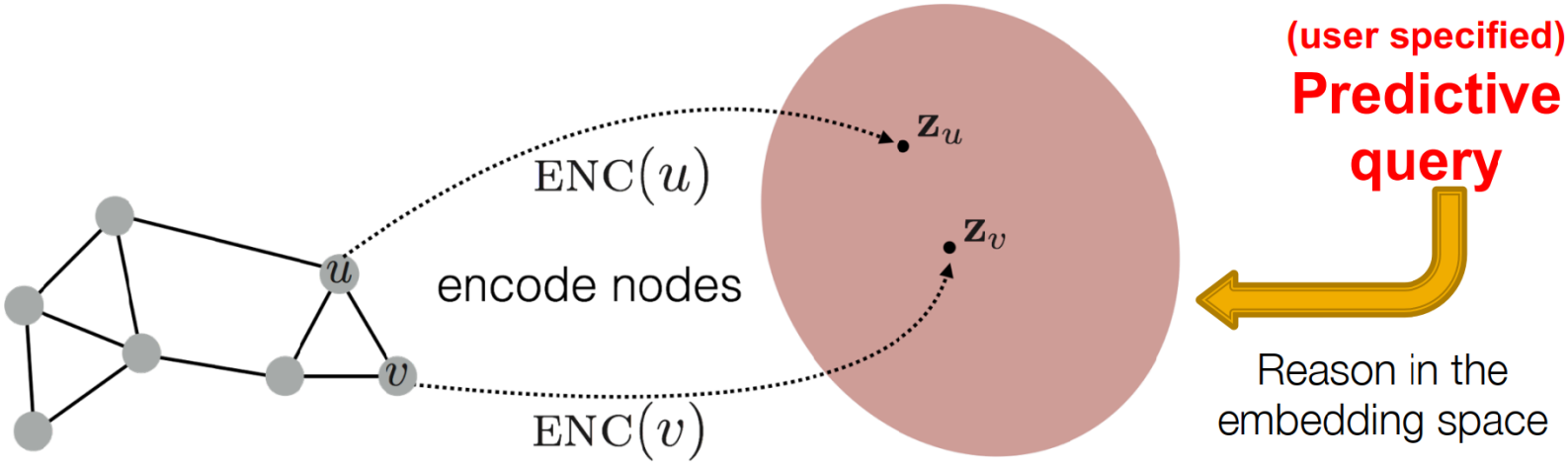
Neural Reasoning for Natural Language Query:

How to Train Query2Box

Training Overview

- **Overview and Intuition** (similar to KG completion):
  - Given a query embedding $\mathbf{q}$, maximize the score $f_q(v)$ for answers $v \in [\![q]\!]$ and minimize the score $f_q(v')$ for negative answers $v' \notin [\![q]\!]$
- Trainable parameters:
  - Entity embeddings with $d|V|$ # params
  - Relation embeddings with $2d|R|$ # params
  - Intersection operator
- **How to achieve a query, its answers, its negative answers from the KG to train the parameters?**
- **How to split the KG for query answering?**

Training Overview



Generate a set of training queries (q, v, v').
Train entity embeddings and operators to minimize the loss (i.e., to answer the training queries correctly).

Each training query provides a "constrain" on the embeddings of entities.

Training loop:

1) Get query (q, v, v')
2) Using current operators, embed q.
3) Compute the loss to update entity embs. and operators

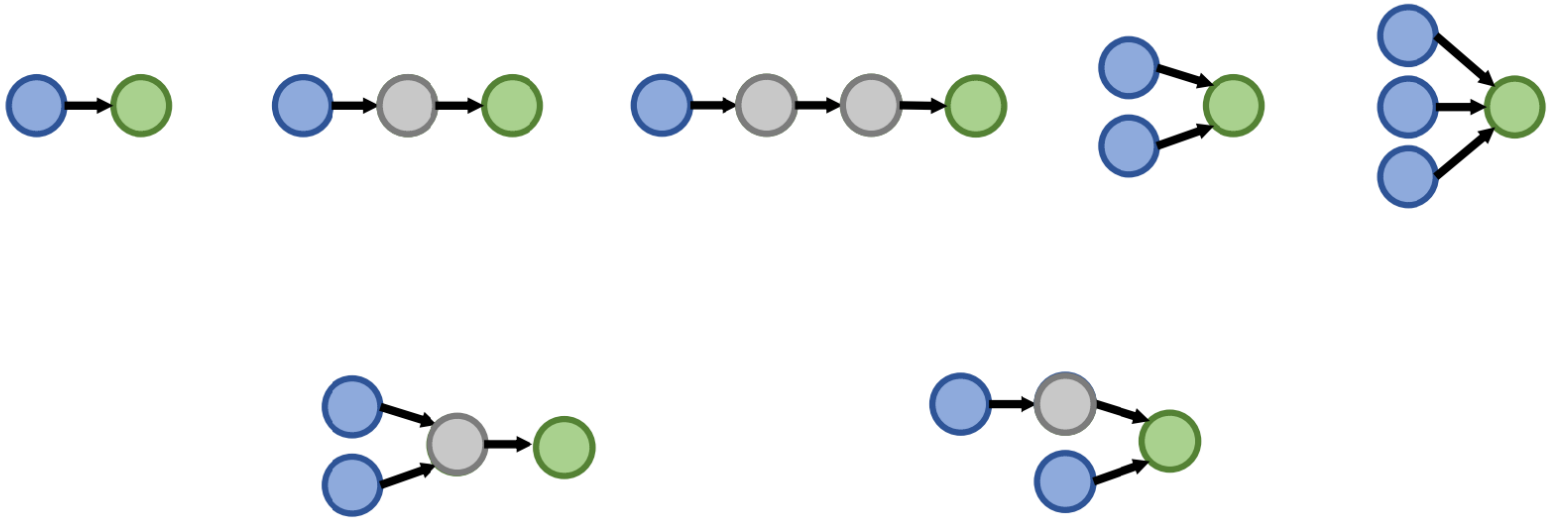
Training: Details

- **Training:**
  1. Sample a query q from the training graph G_{train}, answer $v \in [\![q]\!]_{G_{train}}$, and non-answer $v' \notin [\![q]\!]_{G_{train}}$
  2. Embed the query $\mathbf{q}$.
     - Use current operators, to compute query embedding.
  3. Calculate the score $f_q(v)$ and $f_q(v')$.
  4. Optimize embeddings and operators to minimize the loss ℓ (maximize $f_q(v)$ while minimize $f_q(v')$):

$$\ell = -\log \sigma\left(f_q(v)\right) - \log(1 - \sigma(f_q(v')))$$

Query Generation from Templates

- **Generate queries from multiple query templates:**

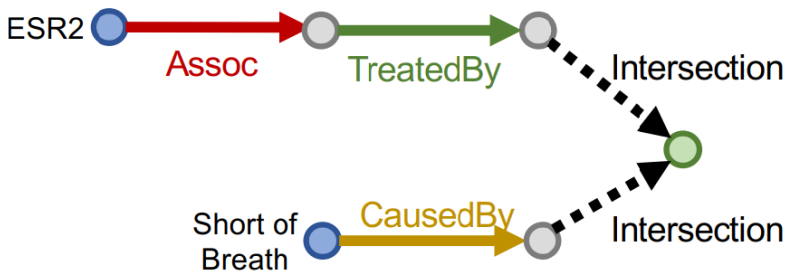


Query Generation from Templates

- How can we generate **a complex query**?
- We start with a **query template**
- **Query template** is an abstraction of the query
- We generate a query by instantiating every variable with a concrete entity and relation from the KG
  - E.g., instantiate **Anchor1** with **ESR2** (a node on KG)
  - E.g., instantiate **Rel1** with **Assoc** (an edge on KG)
- **How to instantiate query template given a KG?**

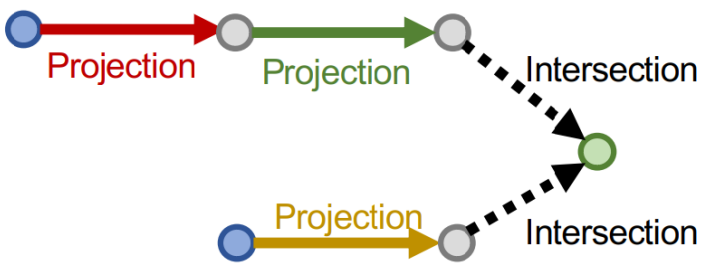
Query

((e:ESR2, (r:Assoc, r:TreatedBy)), (e:Short of Breath, (r:CausedBy))



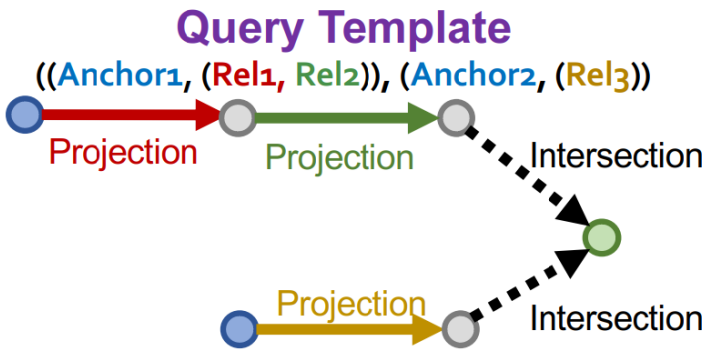
Query Template

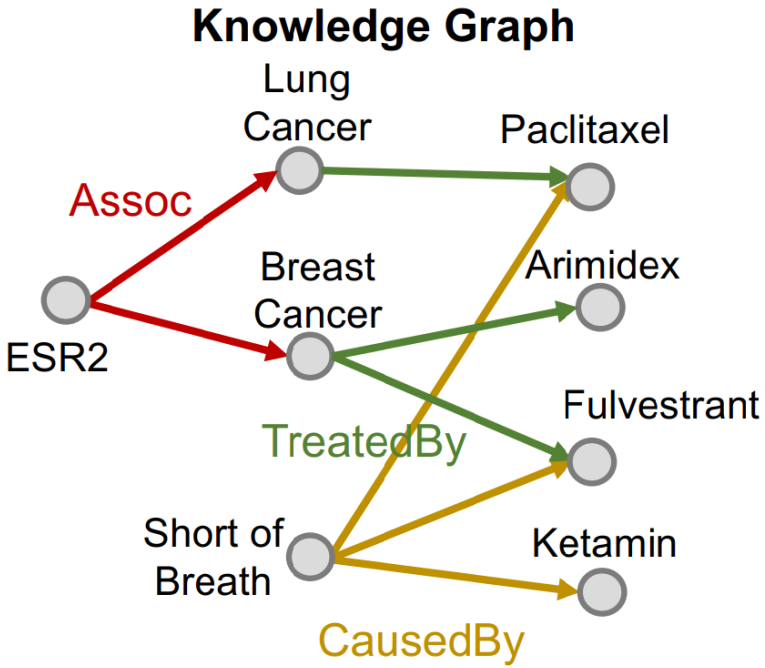
((Anchor1, (Rel1, Rel2)), (Anchor2, (Rel3))



Query Generation from Templates

- **How to instantiate a query template given a KG?**

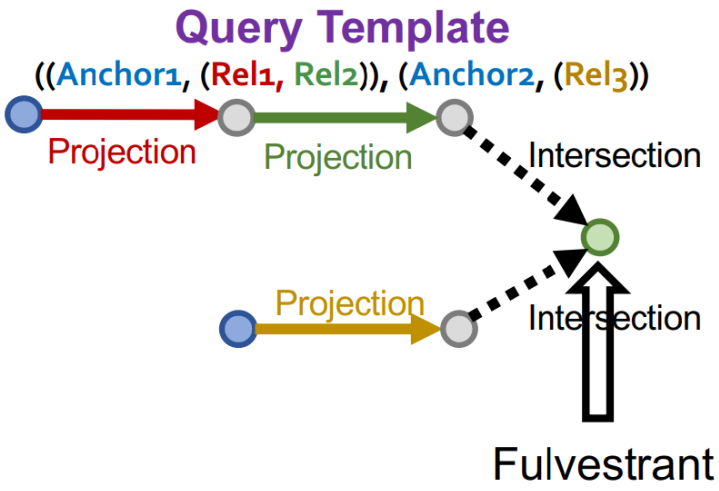


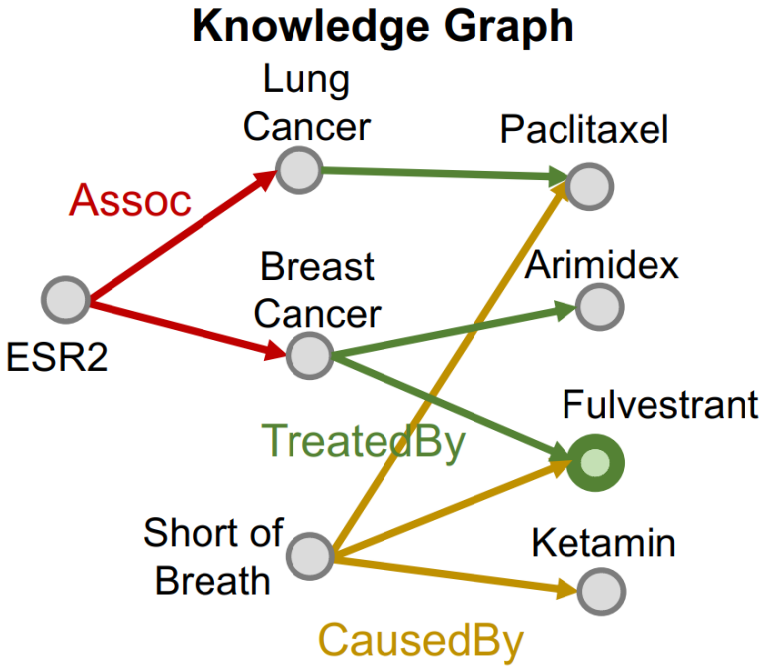


Overview:
Start from instantiating the **answer node** of the query template and then iteratively instantiate the other edges and nodes until we ground **all the anchor nodes**

Query Generation from Templates

- **How to instantiate a query template given a KG?**



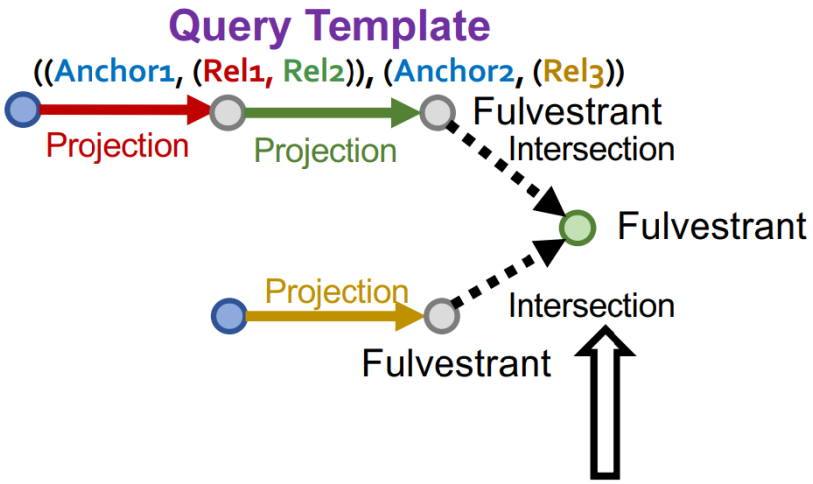


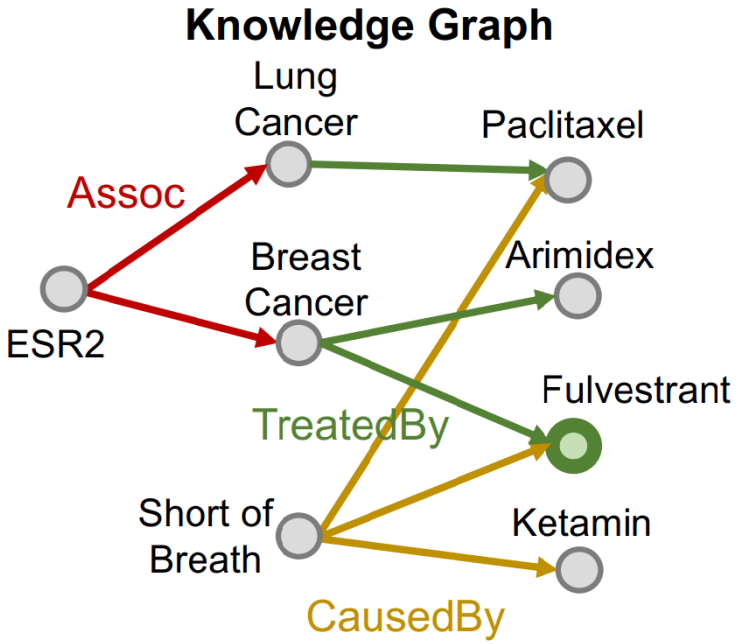
Start from instantiating the **root node** of the query template.

Randomly pick one entity from KG as the root node, e.g., we pick **Fulvestrant**.

Query Generation from Templates

- **How to instantiate a query template given a KG?**

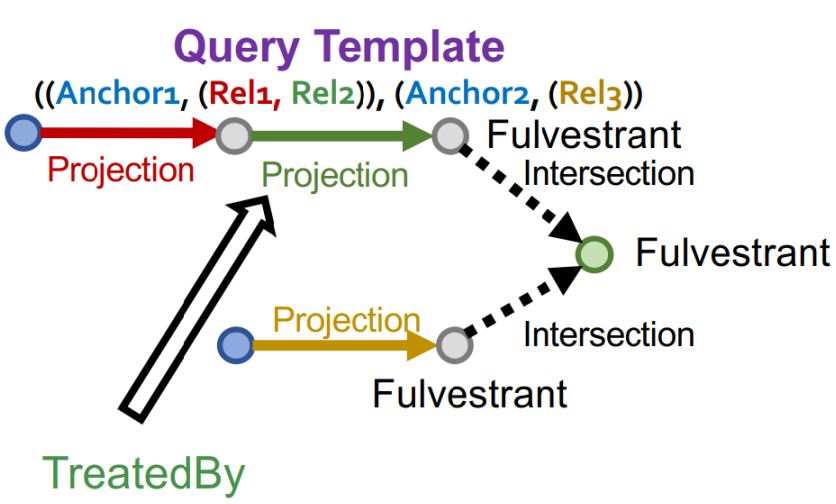


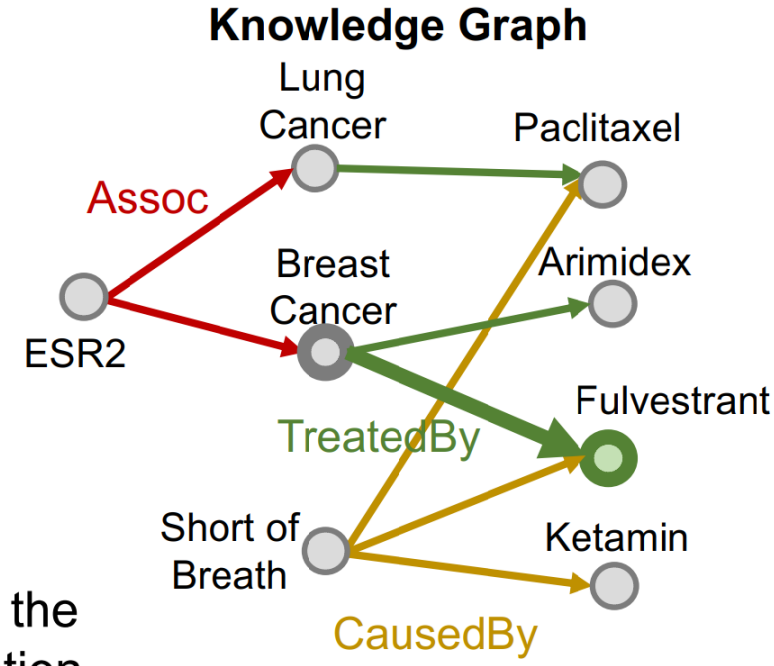


Now we look at intersection. What we have is that the intersection of the sets of entities is **Fulvestrant**, then naturally the two sets should also contain **Fulvestrant**.

Query Generation from Templates

- **How to instantiate a query template given a KG?**

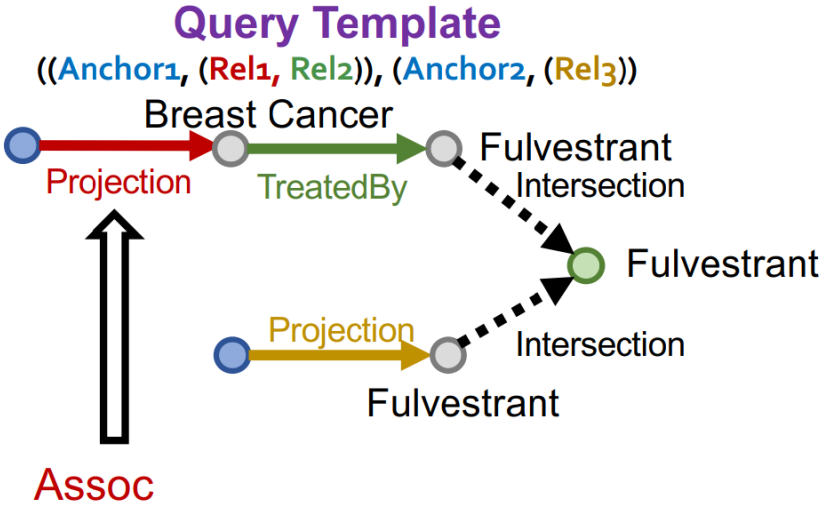


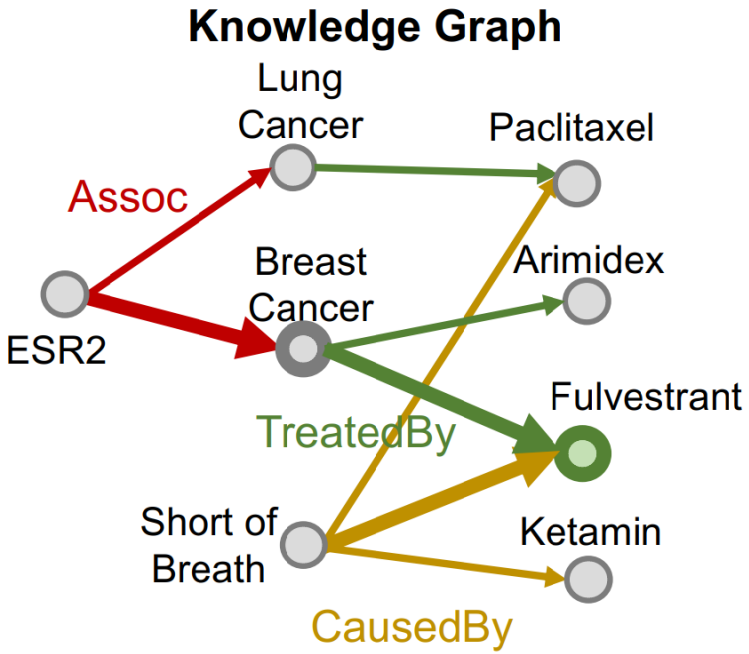


We instantiate the **Projection edge** in the template by randomly sample one relation associated with the current entity **Fulvestrant**. For example, we may select relation **TreatedBy**, and check what entities are connected to **Fulvestrant** with **TreatedBy**: {**Breast Cancer**}.

Query Generation from Templates

- **How to instantiate a query template given a KG?**

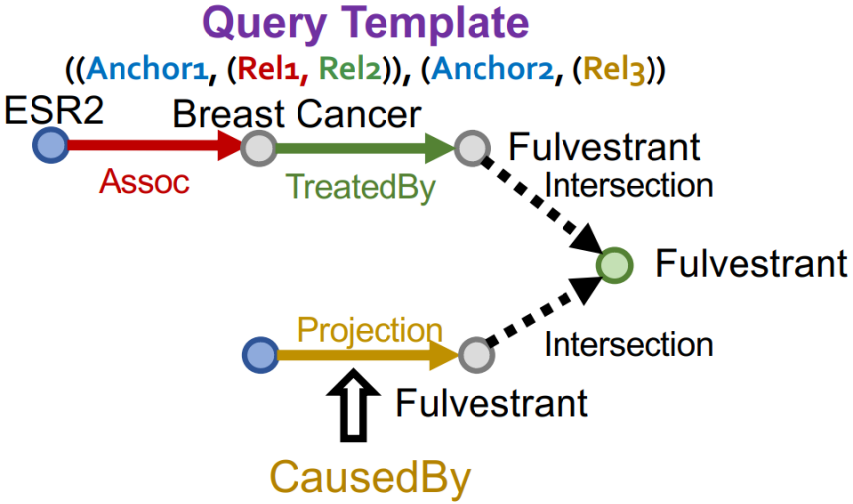


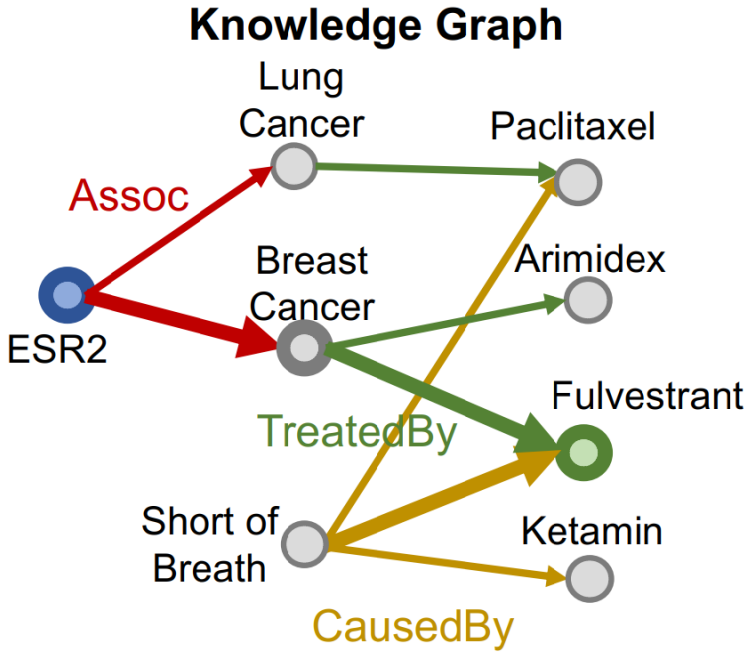


We first look at one branch and ground the **Projection edge** with the relation associated with **Breast Cancer**, e.g., **Assoc**. Then we check what entities are connected to **Breast Cancer** with **Assoc**: {**ESR2**}.

Query Generation from Templates

- **How to instantiate a query template given a KG?**

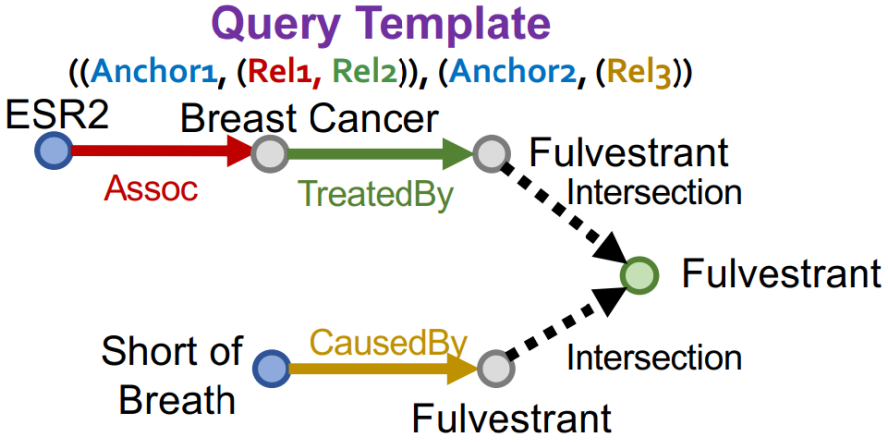


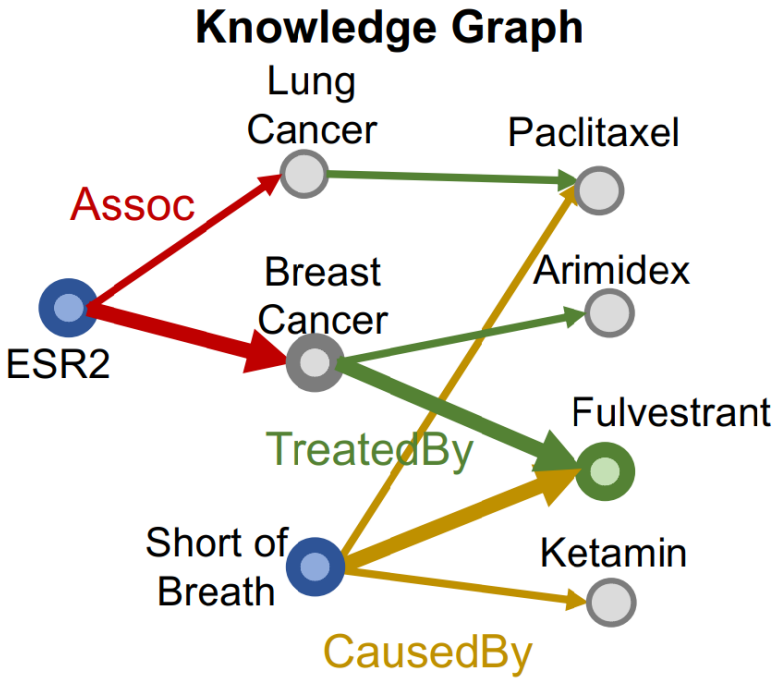


Then we look at the second branch and ground the **Projection edge** with the relation associated with **Fulvestrant**, e.g., **CausedBy**. Then we check what entities are connected to **Fulvestrant** with **CausedBy**: {**Short of Breath**}.

Query Generation from Templates

- **How to instantiate a query template given a KG?**

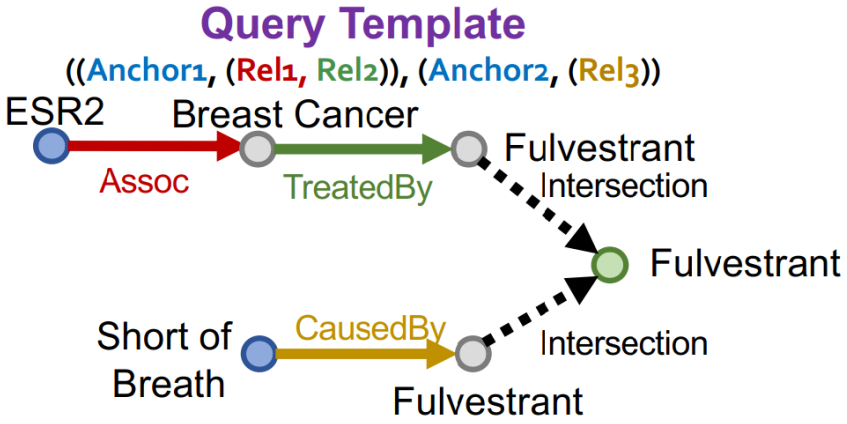




We select entity from {**Short of Breath**}, set it as the anchor node.

Query Generation from Templates

- **How to instantiate a query template given a KG?**



Now, we instantiated a **query q**!

q: **((e:ESR2, (r:Assoc, r:TreatedBy)), (e:Short of Breath, (r:CausedBy))**

- The query q **must** have answers on the KG and one of the answers is the instantiated answer node: **Fulvestrant**.
- We may obtain the full set of answers $[\![q]\!]_G$ by **KG traversal**.
- We can sample negative answers $v' \notin [\![q]\!]_G$

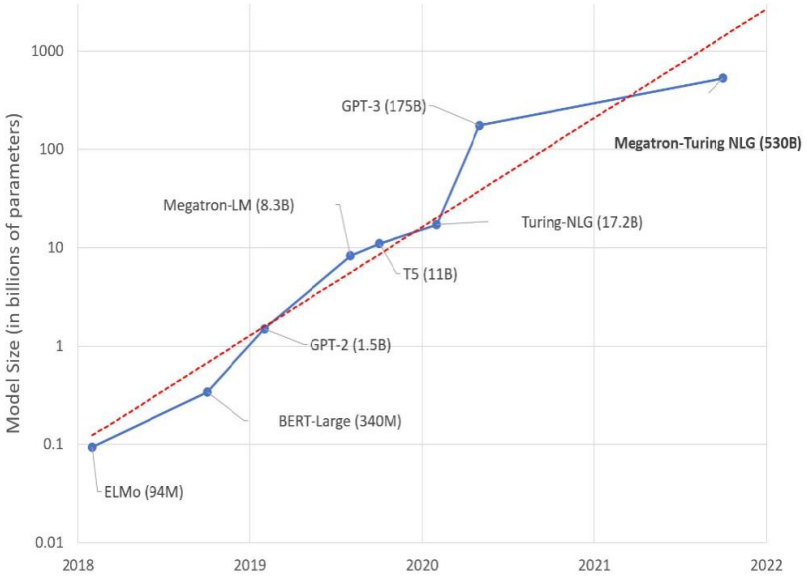
Knowledge Graph and Large Language Models

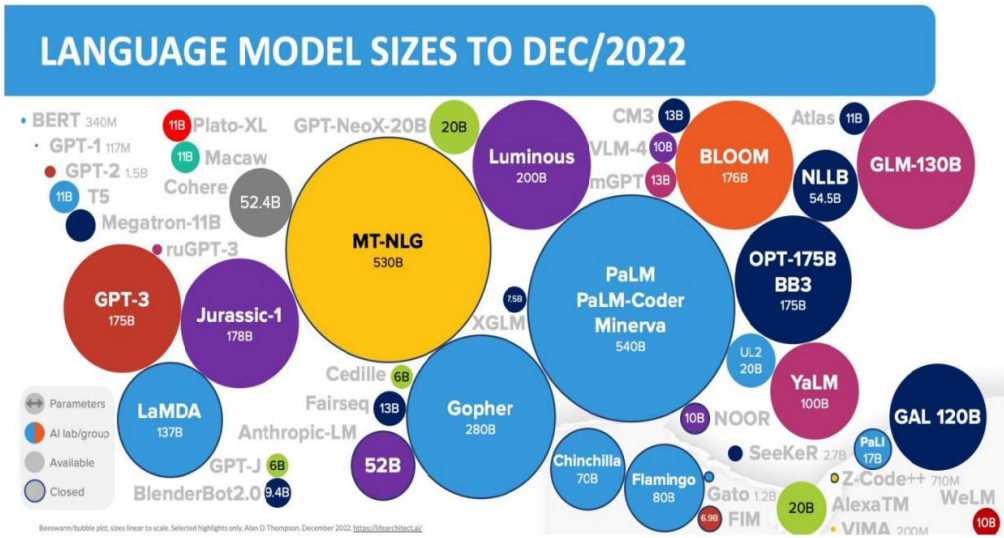
Large Language Models

- Language model: a probability distribution over strings of text
  - Sally fed my cat with meat: P(Sally, feed, my, cat, with, meat) = 0.03
  - My cat fed Sally with meat: P(My, cat, fed, Sally, with, meat) = 0.005
  - Fed cat meat my my with: P(Fed, cat, meat, my, my, with) = 0.0001
- Large language models (LLM): large, general-purpose language models can be pre-trained and then fine-tuned for specific purposes
  - Examples: GPT-3.5, GPT-4, GPT-4o, Llama2 and Llama3

Model Size of LLMs

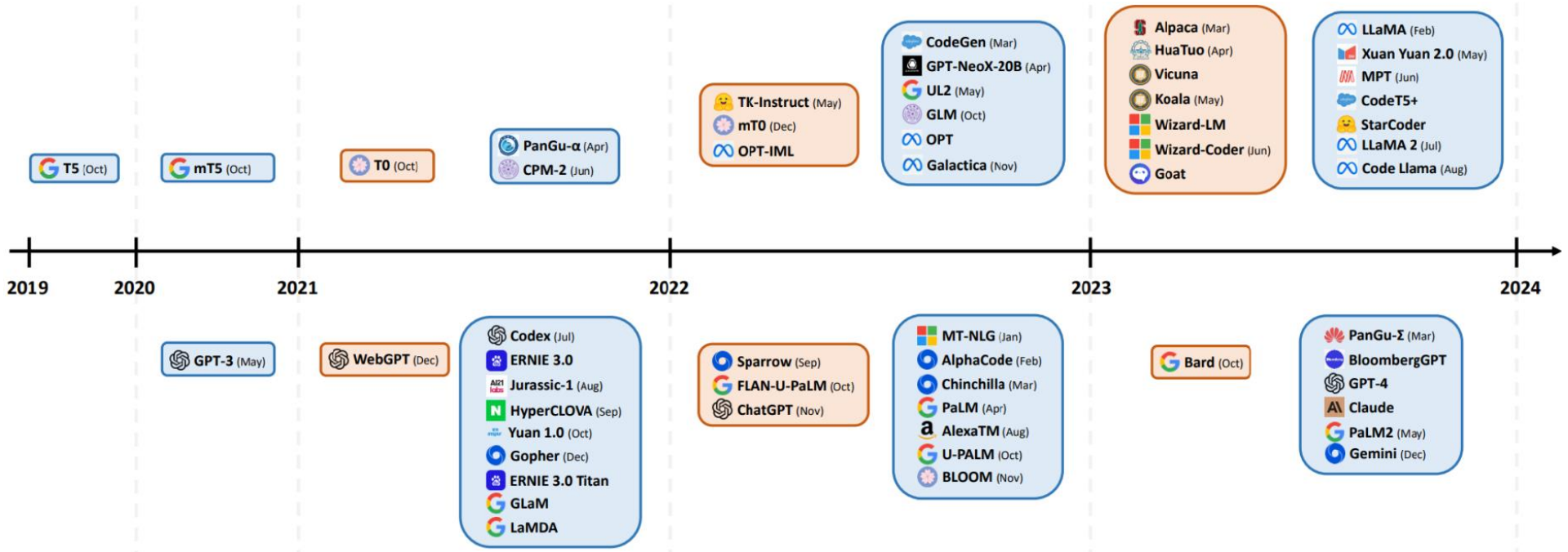
- Size: from millions of parameters to billions of parameters





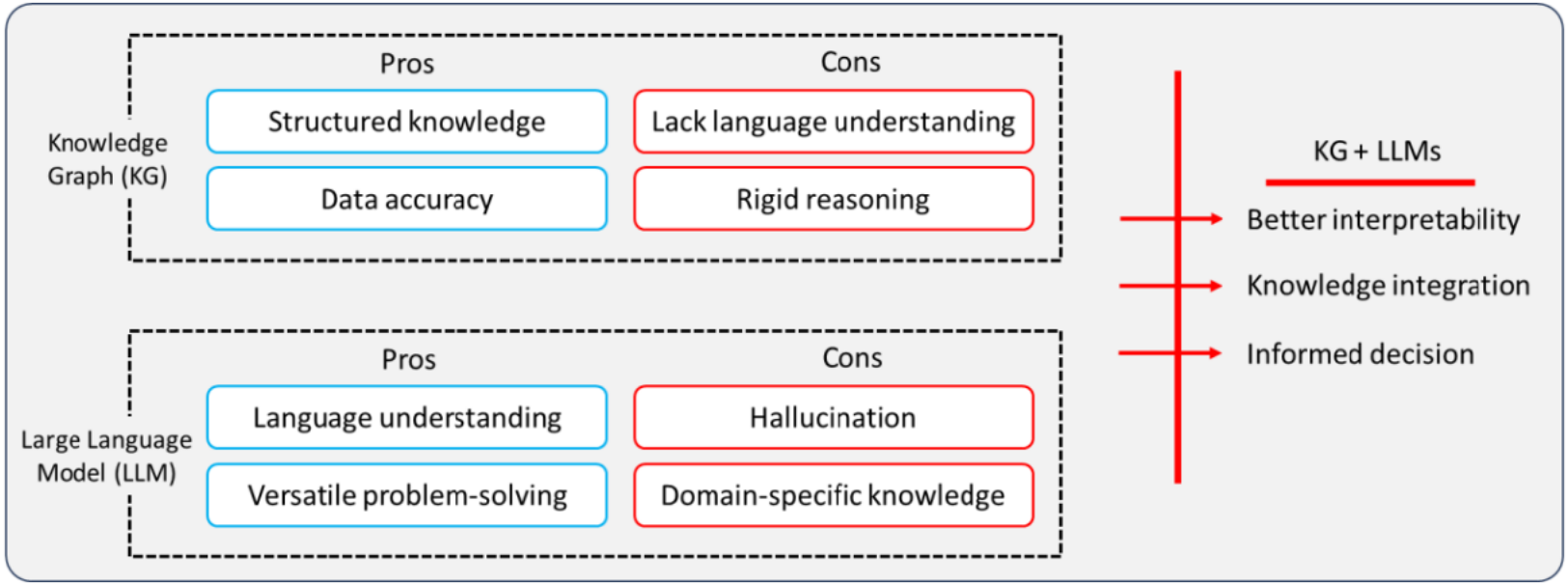
LLMs in Recent Literature

- Progression and improvement of large language models



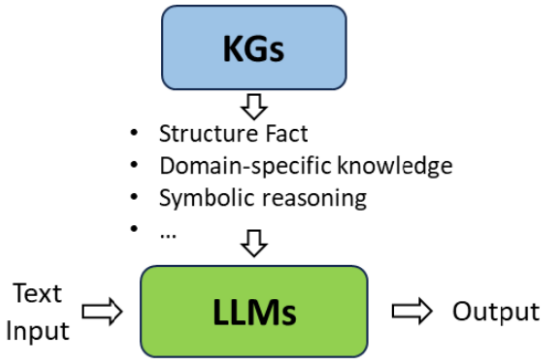
Knowledge Graphs vs LLMs

- Knowledge graph
  - Pros: accurate structural knowledge, interpretable, …
  - Cons: incomplete, lack language understanding
- LLMs
  - Pros: general knowledge, good at language understand,…
  - Cons: hallucination, lack interpretation, lacking new knowledge, …

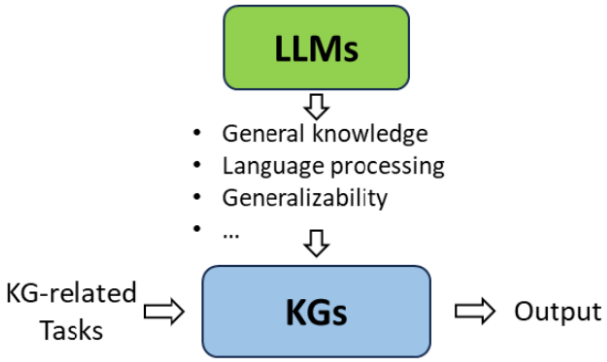


Combining Knowledge Graph with LLMs

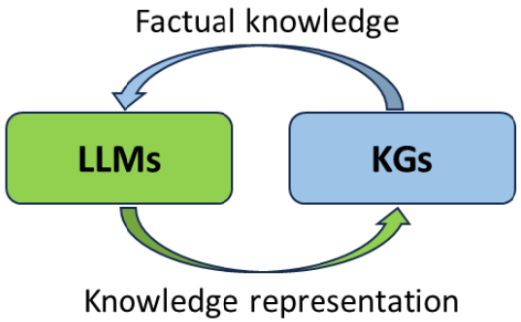
- Categorization
  - Knowledge graph-enhanced LLMs
  - LLMs-enhanced knowledge graph reasoning
  - Integrating knowledge graph reasoning with LLMs in a mutually beneficial way



KG-enhanced LLMs



LLMs-augmented KGs



Synergized LLMs + KGs

Paper: S. Pan, L. Luo,Y. Wang, C Chen. 2024. Unifying Large Language Models and Knowledge Graphs: A Roadmap (TKDE 24)

Knowledge Graph and Large Language Models

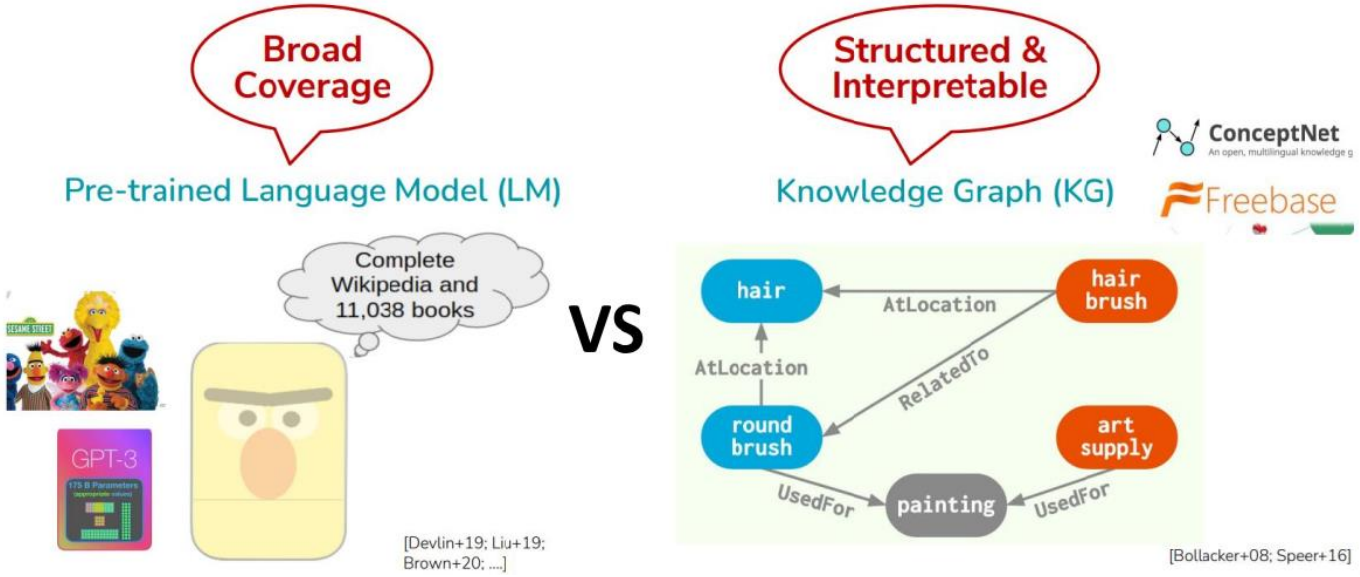
Knowledge Graph-Enhanced LLMs

QA-GNN: Reasoning with LLMs and KGs for Question Answering

- Goal: answer multi-choice question

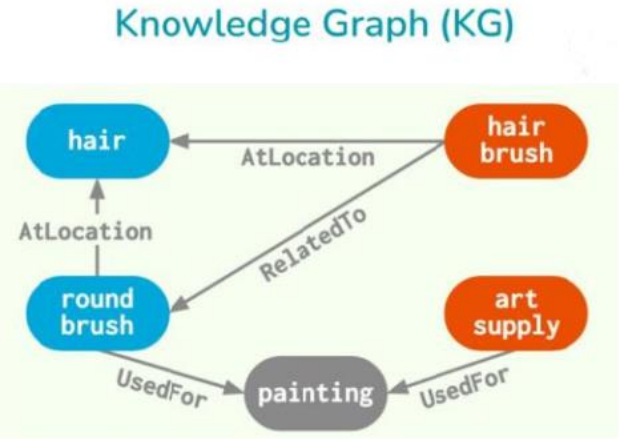
> If it is not used for **hair**, a **round brush** is an example of what?
>
> A. hair brush B. bathroom C. **art supplies*** D. shower

- Philosophy: the system needs to access a lot of knowledge and reason about it.



QA-GNN: Reasoning with LLMs and KGs for Question Answering

- Existing problem
  - Language models do not work well for interpretable or logical reasoning; they lack interpretation.
  - KG is incomplete and noisy
- Challenges
  - how to identify the relevant information in the knowledge graph
  - how to jointly reason over the text and the knowledge graph

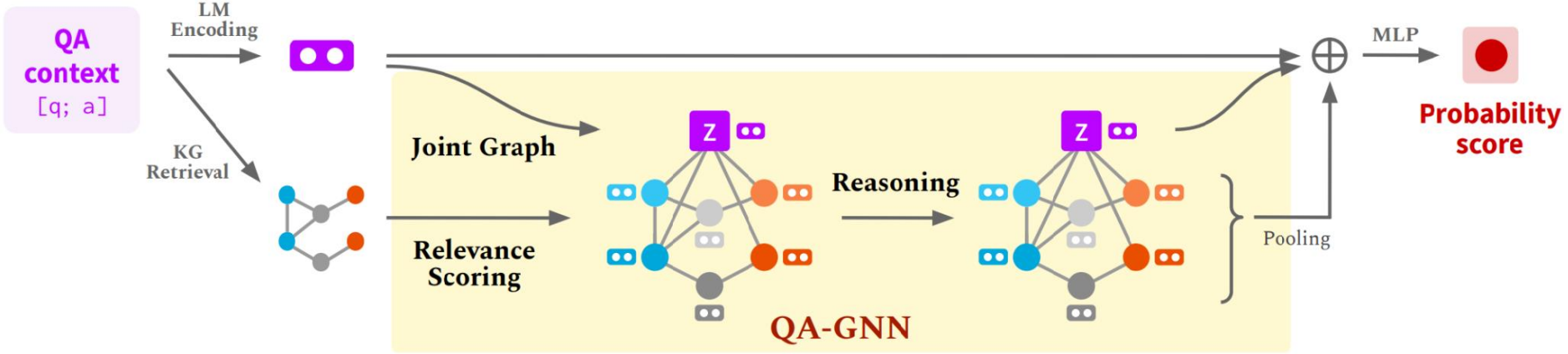


If it is not used for **hair**, a **round brush** is an example of what?

A. hair brush B. bathroom C. **art supplies*** D. shower

QA-GNN: Main Steps

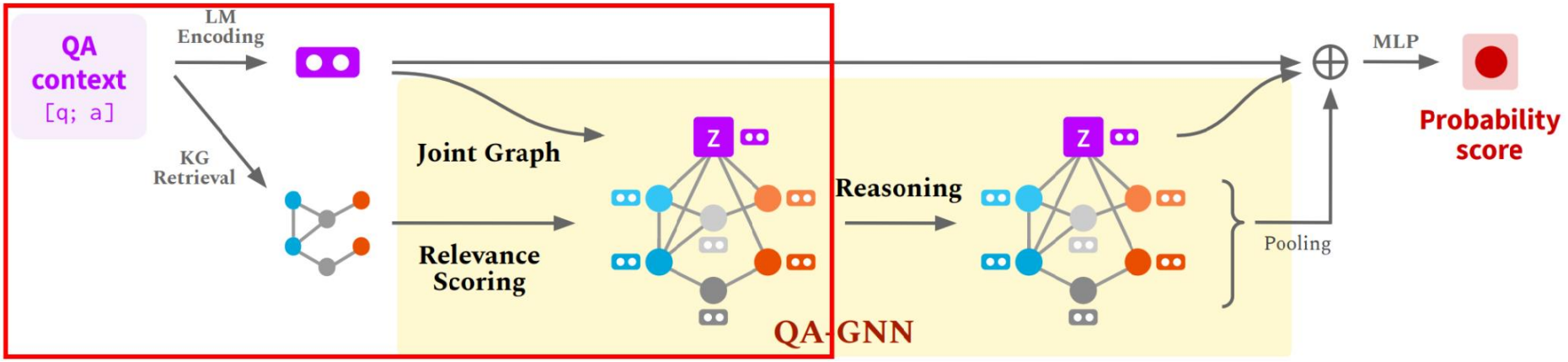
- Idea:
  - Language-conditioned KG node relevance scoring.
  - Joint Reasoning:
    - Connect text and KG to form a joint graph (working graph)
    - Mutually update their representations via Graph Neural Network (GNN)



Paper: M. Yasunaga, H. Ren, A. Bosselut, P. Liang, J. Leskovec. 2021. QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering (NAACL 21)

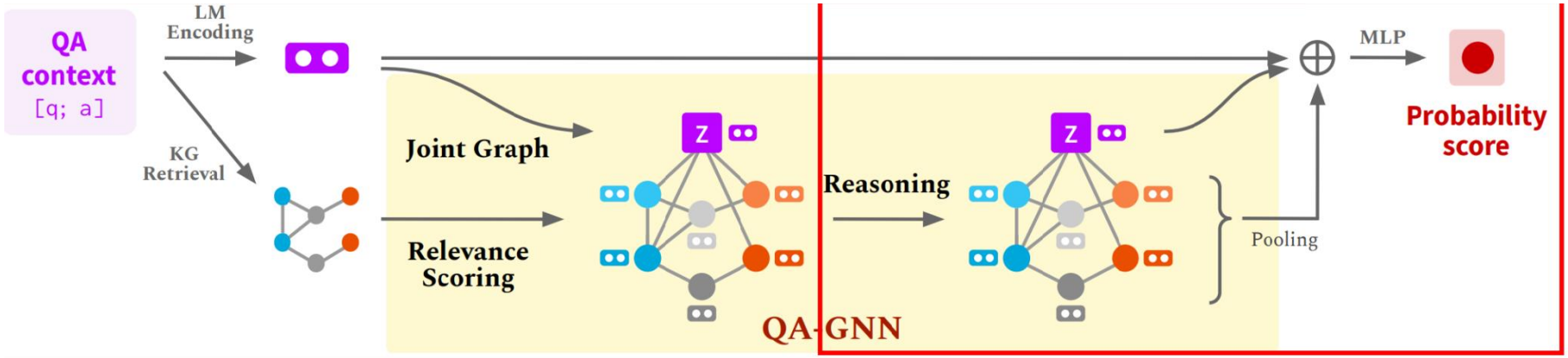
QA-GNN: Main Steps

- Idea:
  - Language-conditioned KG node relevance scoring.
  - Joint Reasoning:
    - Connect text and KG to form a joint graph (working graph)
    - Mutually update their representations via Graph Neural Network (GNN)



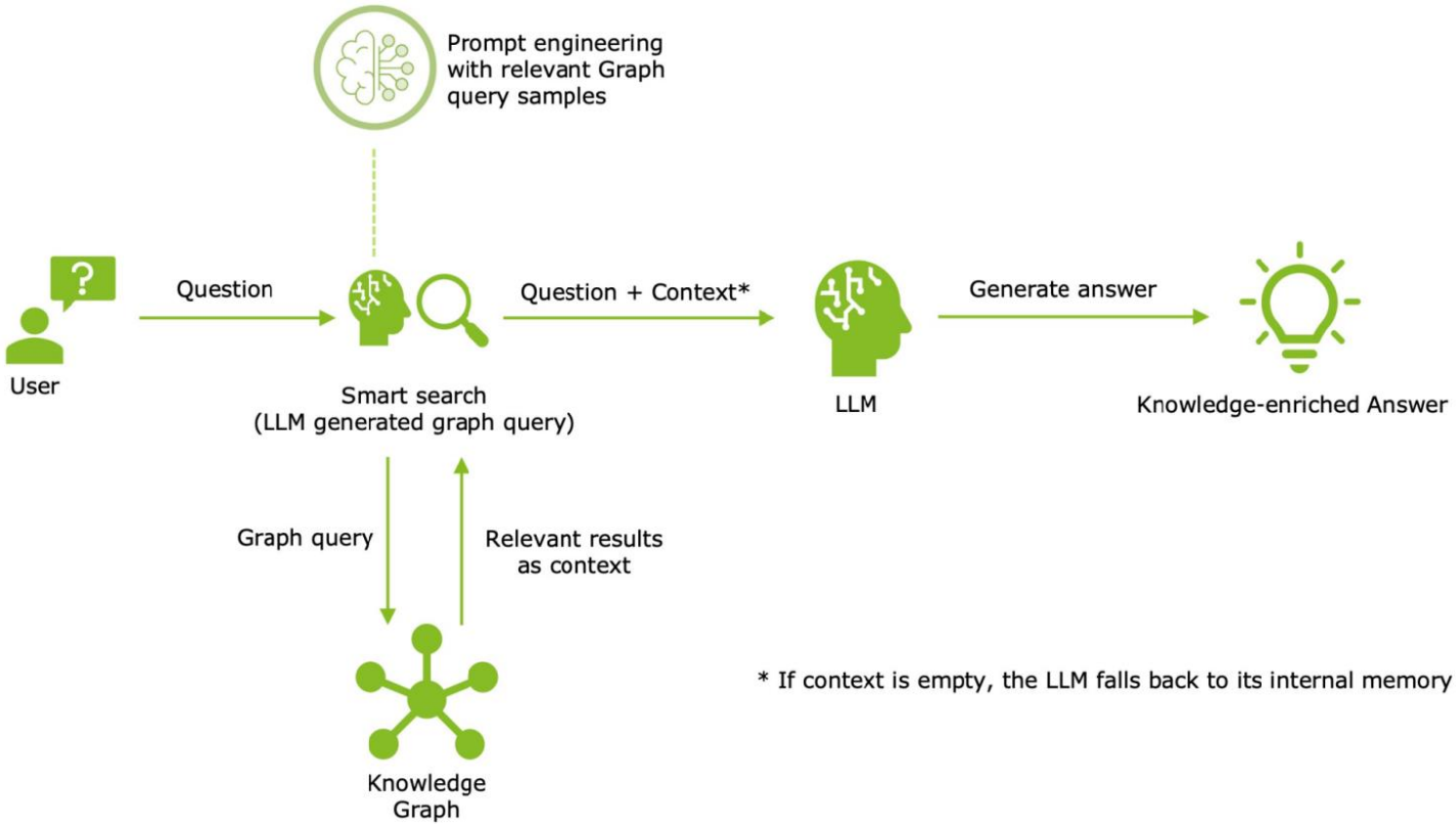
QA-GNN: Main Steps

- Idea:
  - Language-conditioned KG node relevance scoring.
  - Joint Reasoning:
    - Connect text and KG to form a joint graph (working graph)
    - Mutually update their representations via Graph Neural Network (GNN)



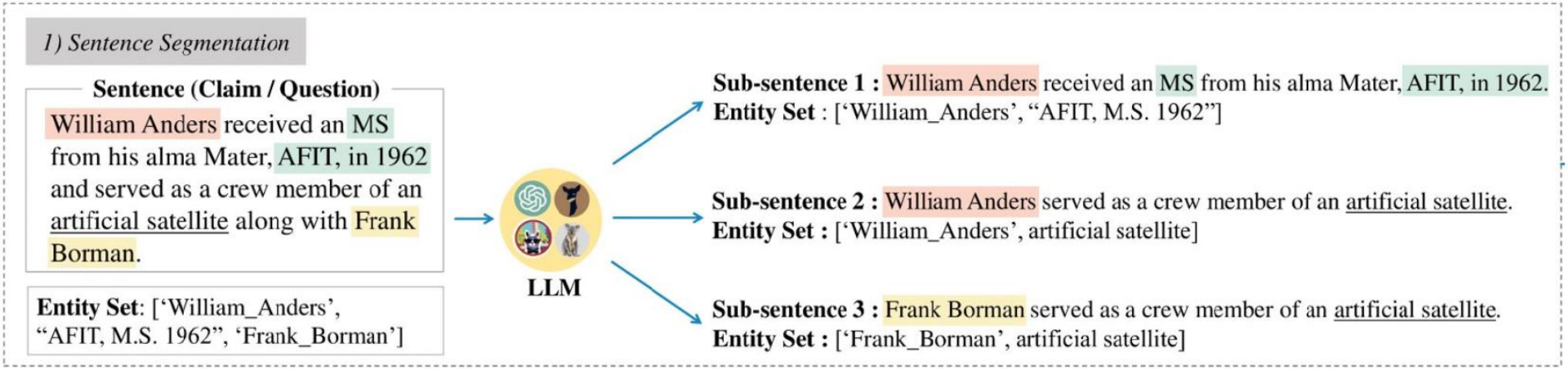
Retrieval-Augmented Generation (RAG) with Knowledge Graph

- Observation:
  - When the size of the model becomes large, retrain or fine-tune the model will be very time consuming



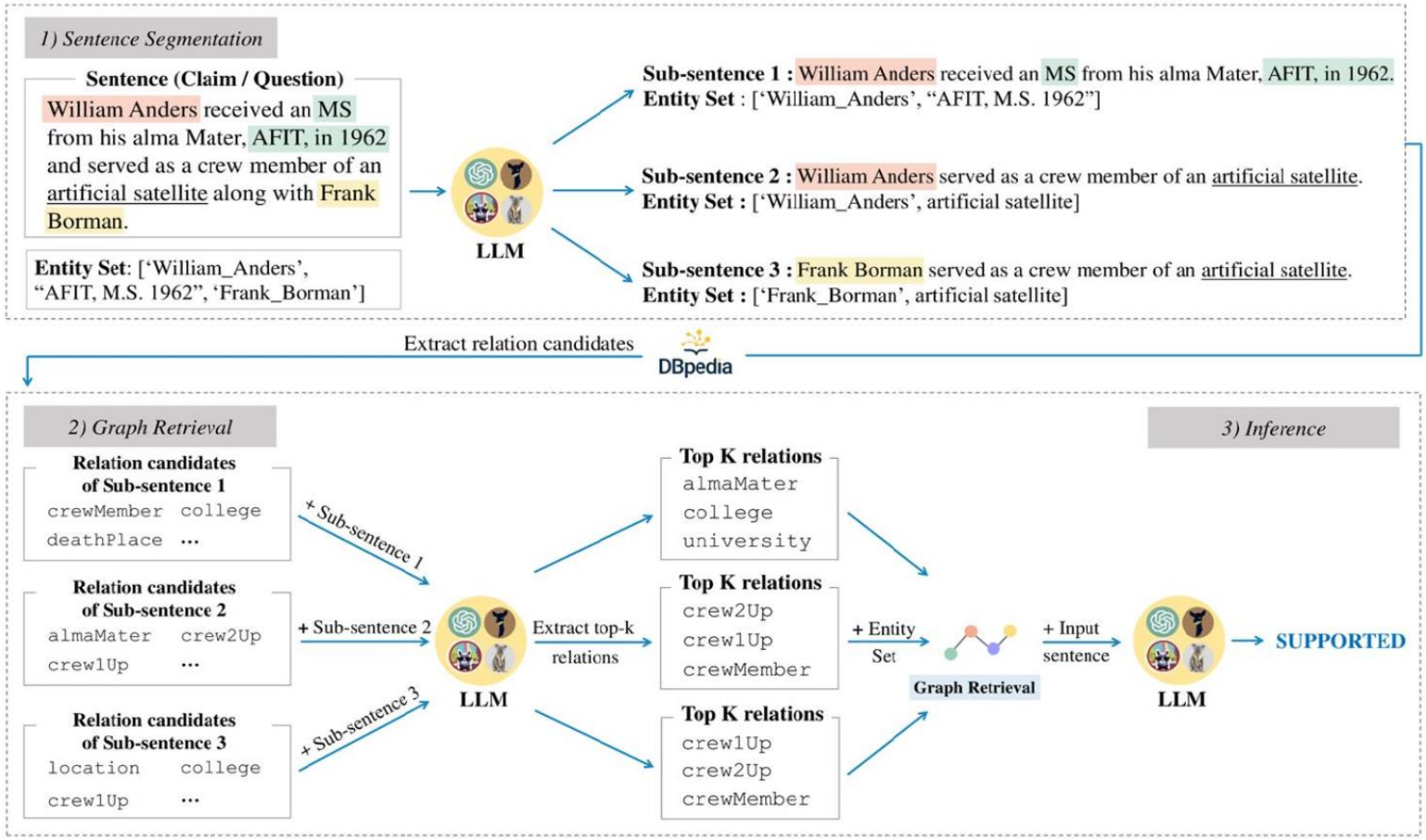
KG-GPT: A General Framework for Reasoning on KGs Using LLMs

- **Goal**
  - Utilize language models and knowledge graphs to answer a more complex natural language questions
- **Idea:**



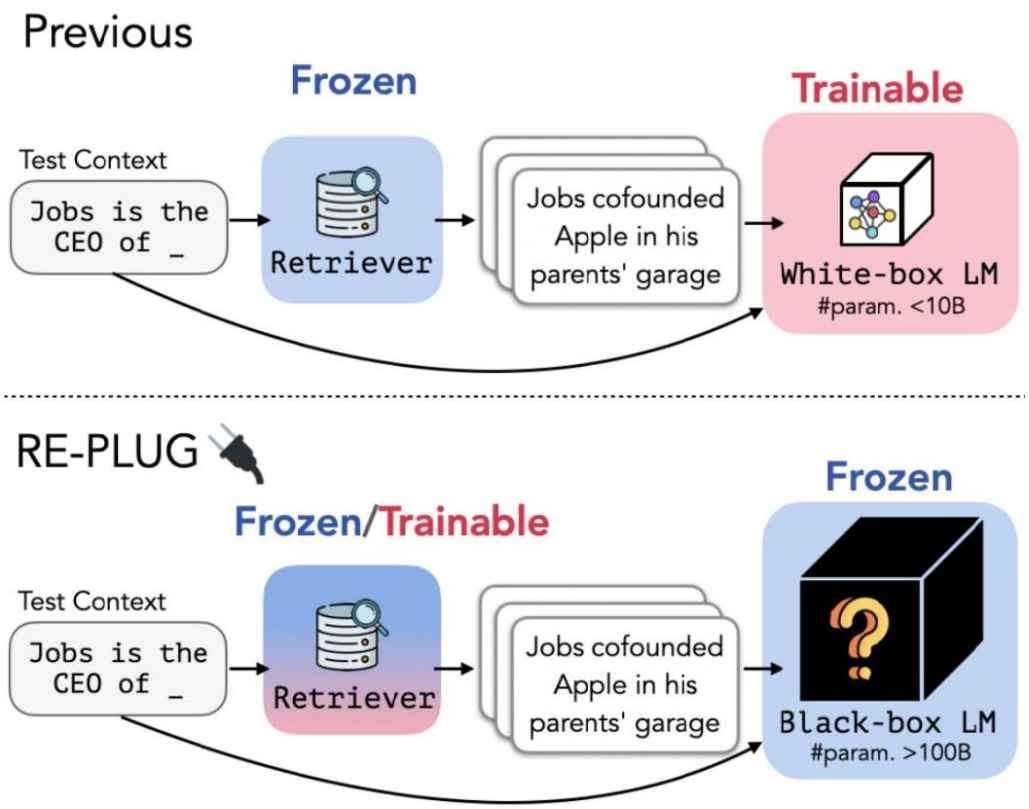
KG-GPT: A General Framework for Reasoning on KGs Using LLMs

- Sentence Segmentation
- Graph Retrieval: Extract relation candidates from KG
- Inference: Graph to prompt



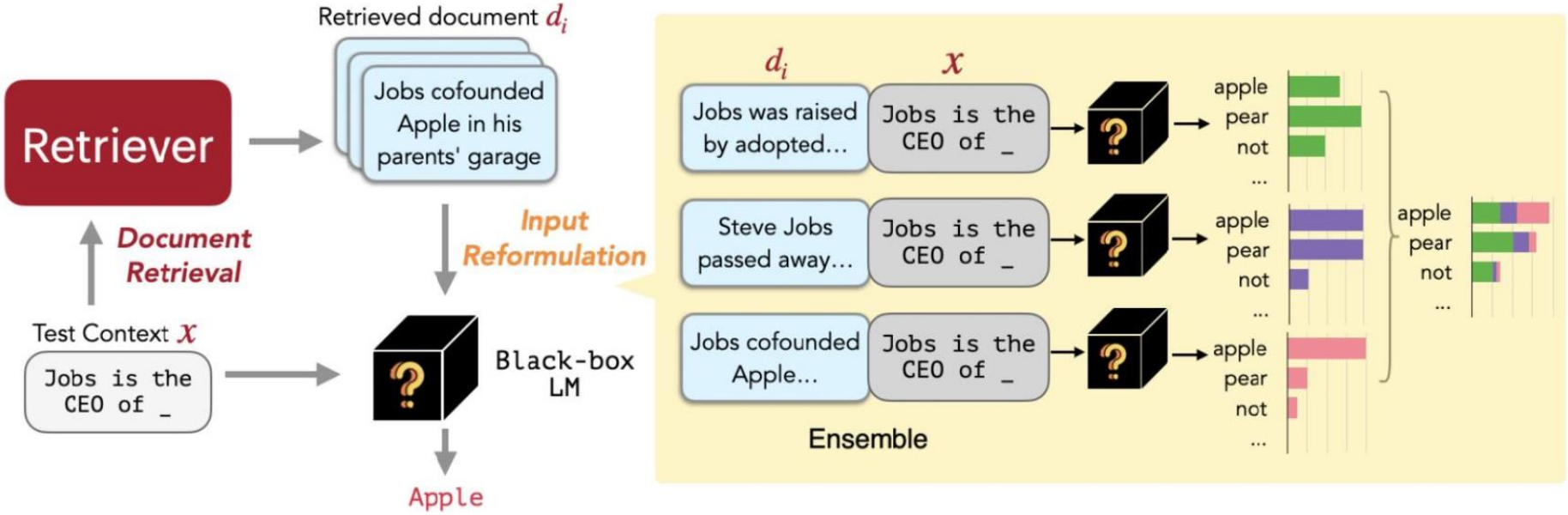
REPLUG: Retrieval-Augmented Black-Box Language Models

- Previous methods:
  - Enhance a language model with retrieval by updating the LM's parameters
- REPLUG:
  - Treats the language model as a black box
  - Augments it with a frozen or tunable retriever
  - Makes REPLUG applicable to large LMs, which are often served via APIs



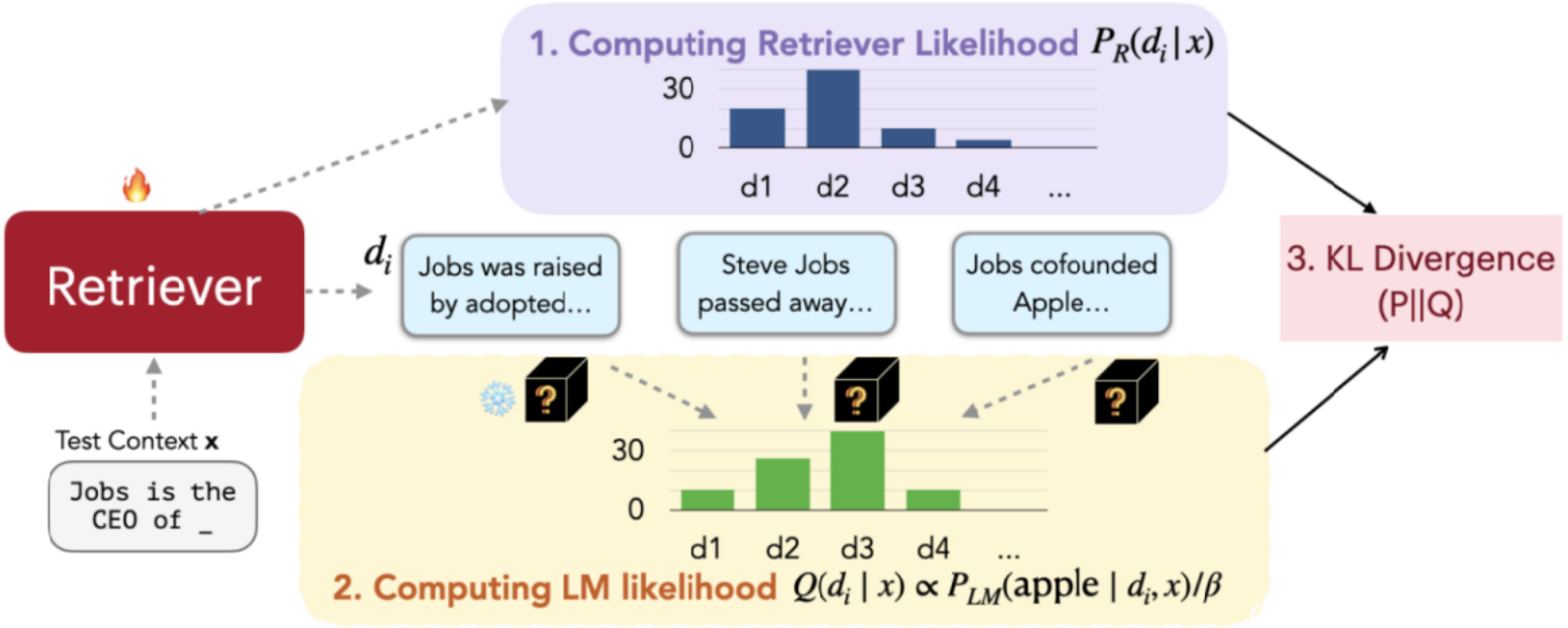
REPLUG: Retrieval-Augmented Black-Box Language Models

- First retrieves a small set of relevant documents from an external corpus
- Then pass the concatenation of each retrieved document with the input context through the LM in parallel, and ensemble the predicted probabilities



REPLUG-LSR: REPLUG with LM-Supervised Retrieval

- (1) Using retriever to computing the retrieval likelihood
- (2) Computing LM likelihood
- (3) Updating the retrieval model parameters by minimizing the KL divergence between the retrieval likelihood and the LM's score distribution

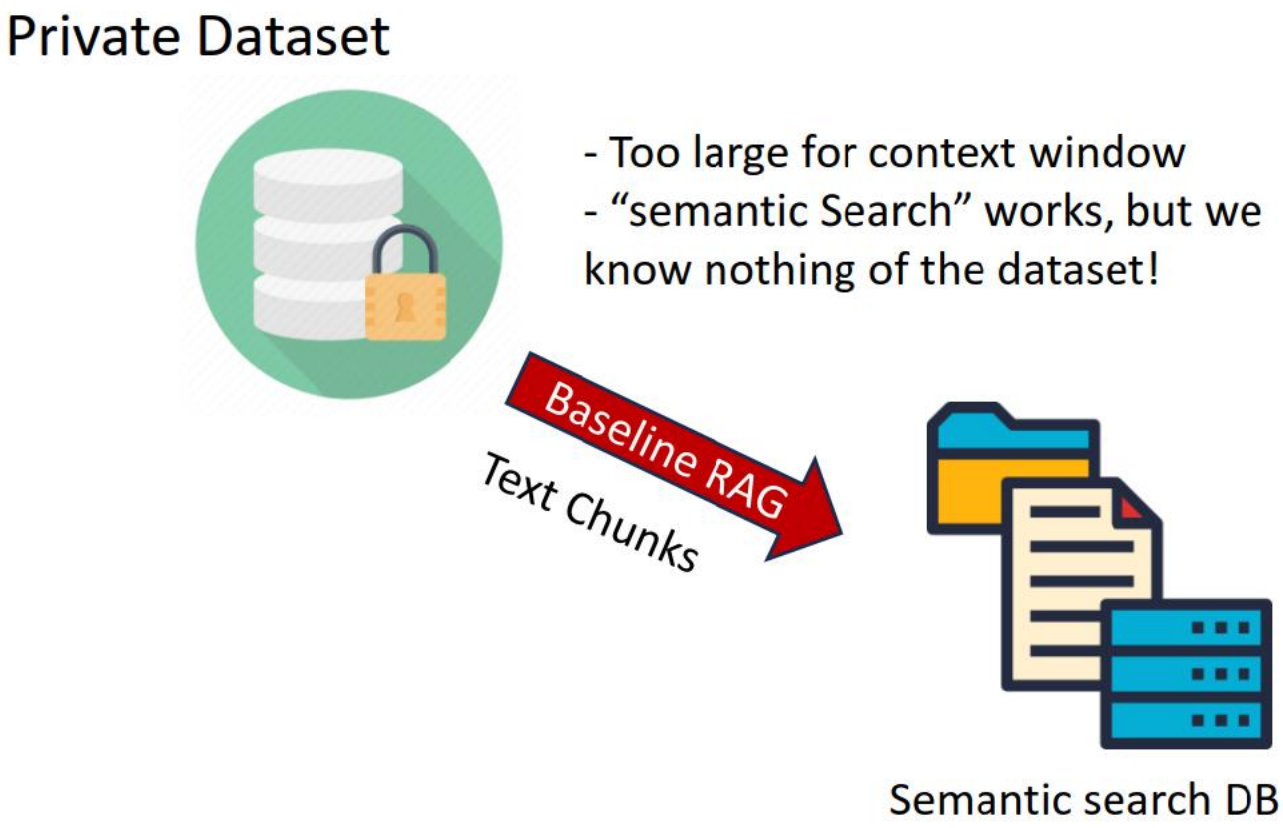


Paper: REPLUG: Retrieval-Augmented Black-Box Language Models

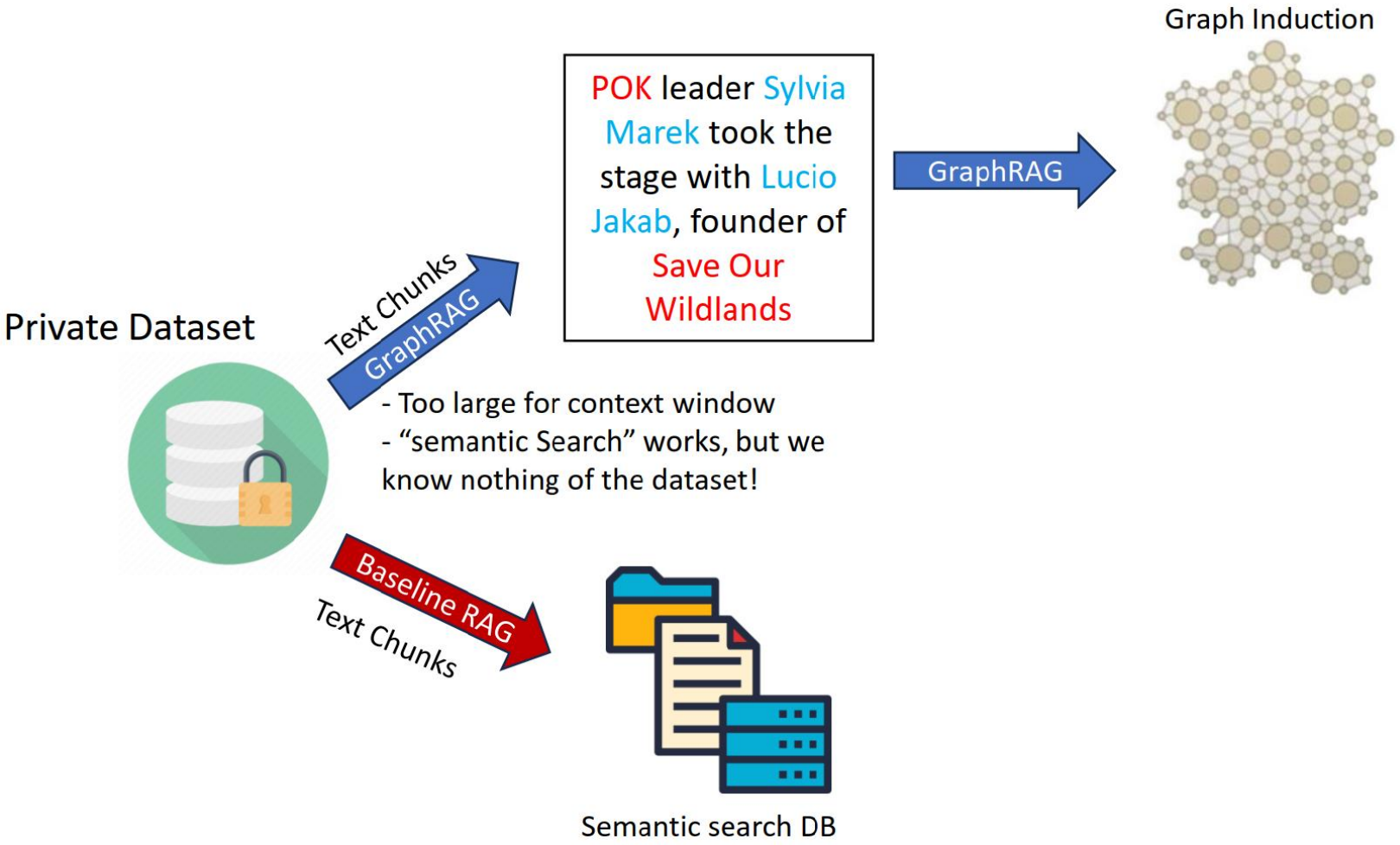
A Graph RAG Approach to Query-Focused Summarization

- Existing problem
  - Fail on global questions such as “what are the main themes in the dataset?”
- GraphRAG
  - Indexing process aimed at creating LLM Memory Representation on private data
    - Generating Knowledge Graphs
    - Dividing entities to different communities

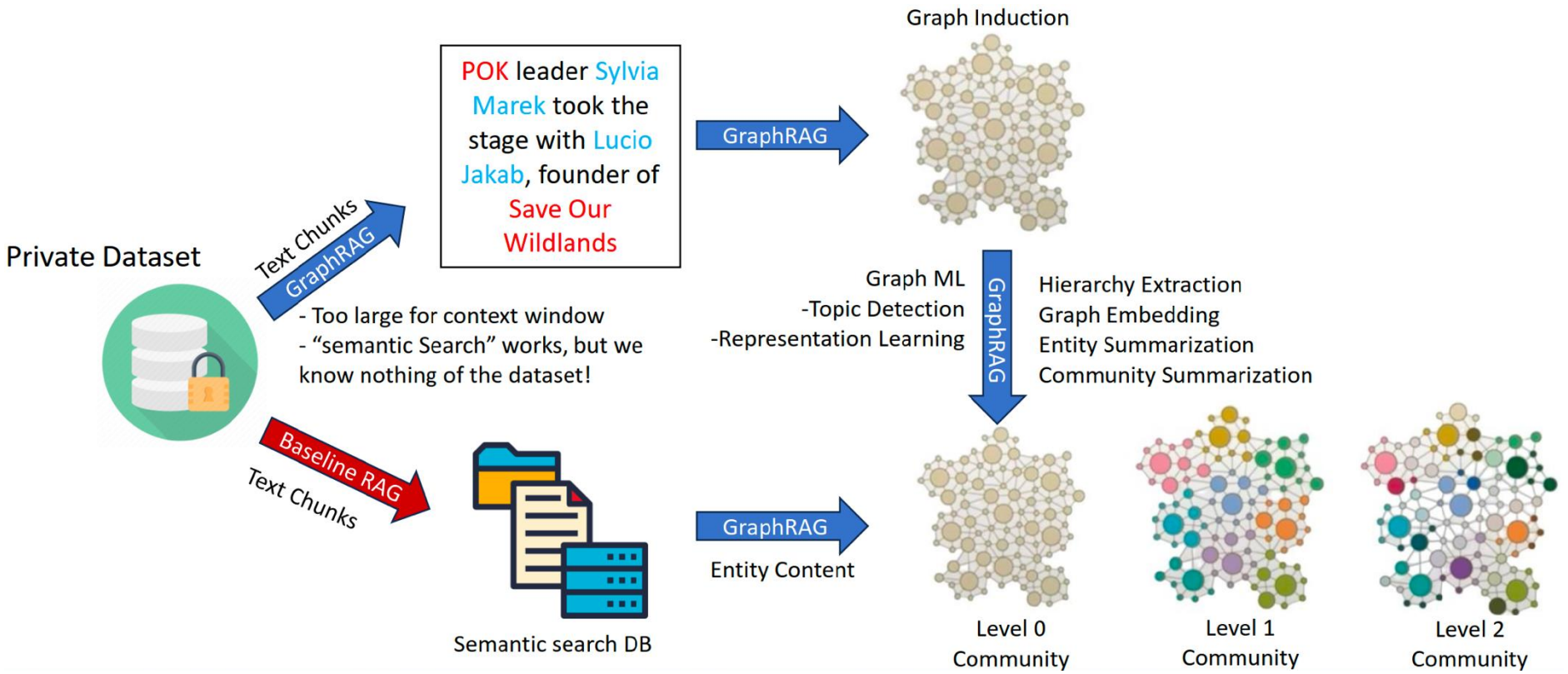
Preprocessing Pipeline- Graph Enabled RAG



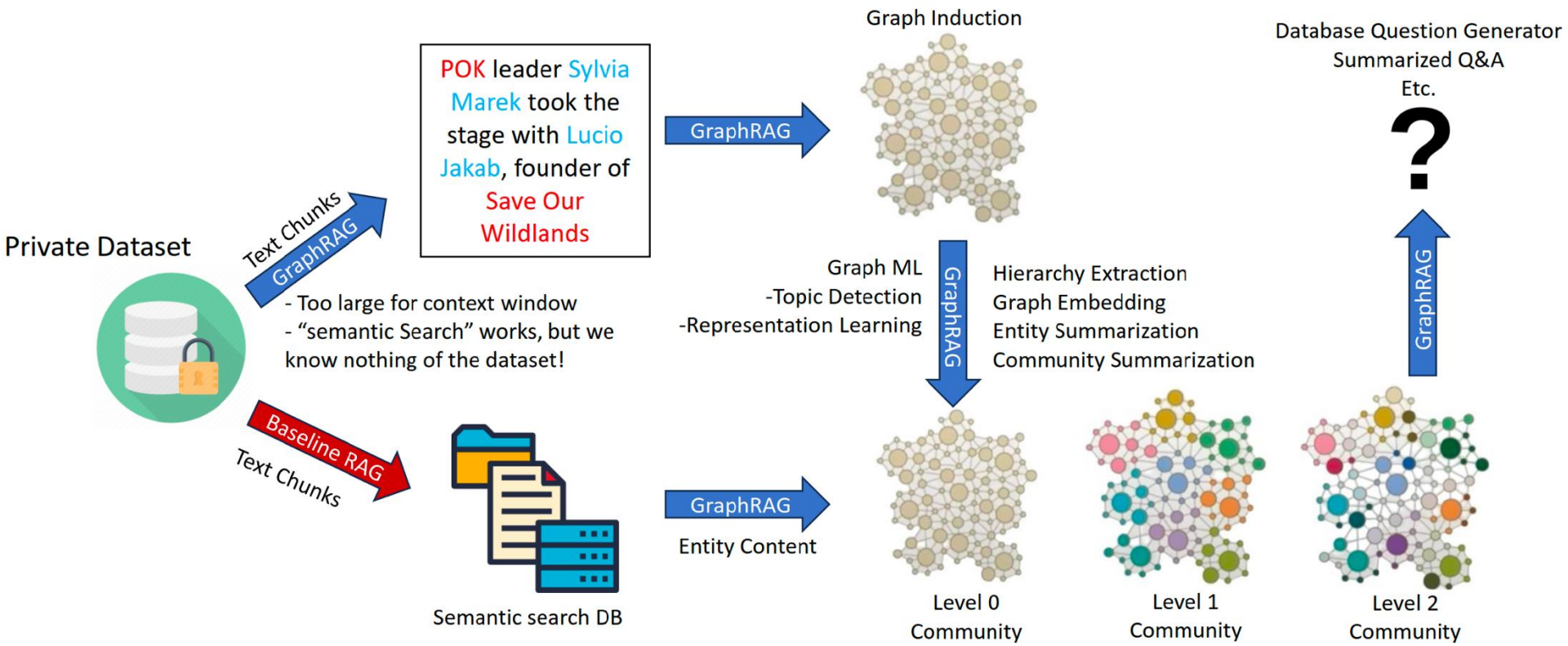
Preprocessing Pipeline- Graph Enabled RAG



Preprocessing Pipeline- Graph Enabled RAG



Preprocessing Pipeline- Graph Enabled RAG



Knowledge Graph and Large Language Models

LLMs - augmented KGs

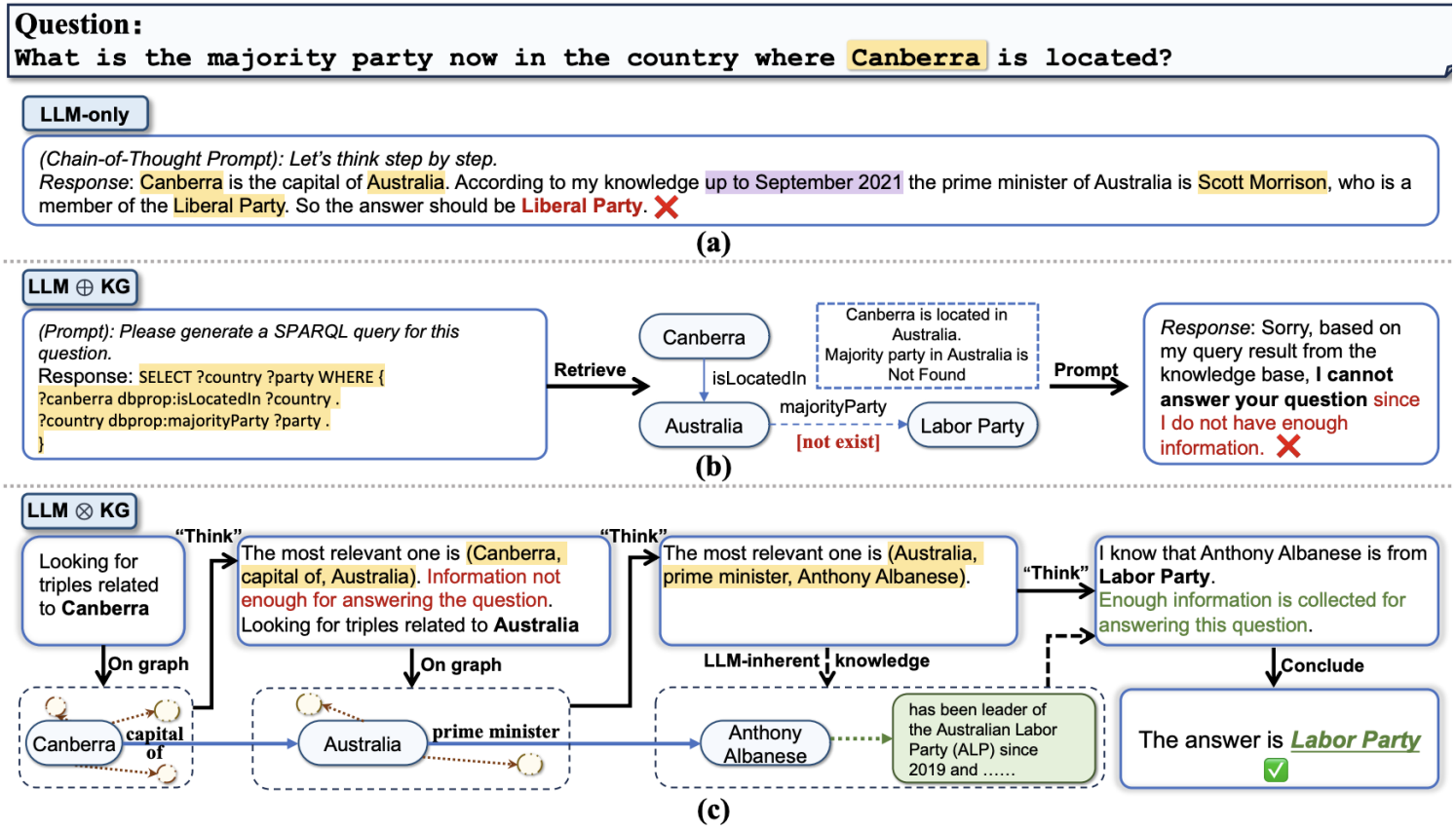
Think-on-Graph: Deep and Responsible Reasoning of LLM on Knowledge Graph

- Traditional methods
  - Retrieve information from KGs
  - Augment the prompt and feed it into LLMs
  - LLMs do not participate in the graph reasoning process directly
- Existing problem
  - Suffer from knowledge graph incompleteness

Paper: J. Sun, C. Xu, L. Tang, S. Wang, C. Lin (2024). Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph (ICLR 24)

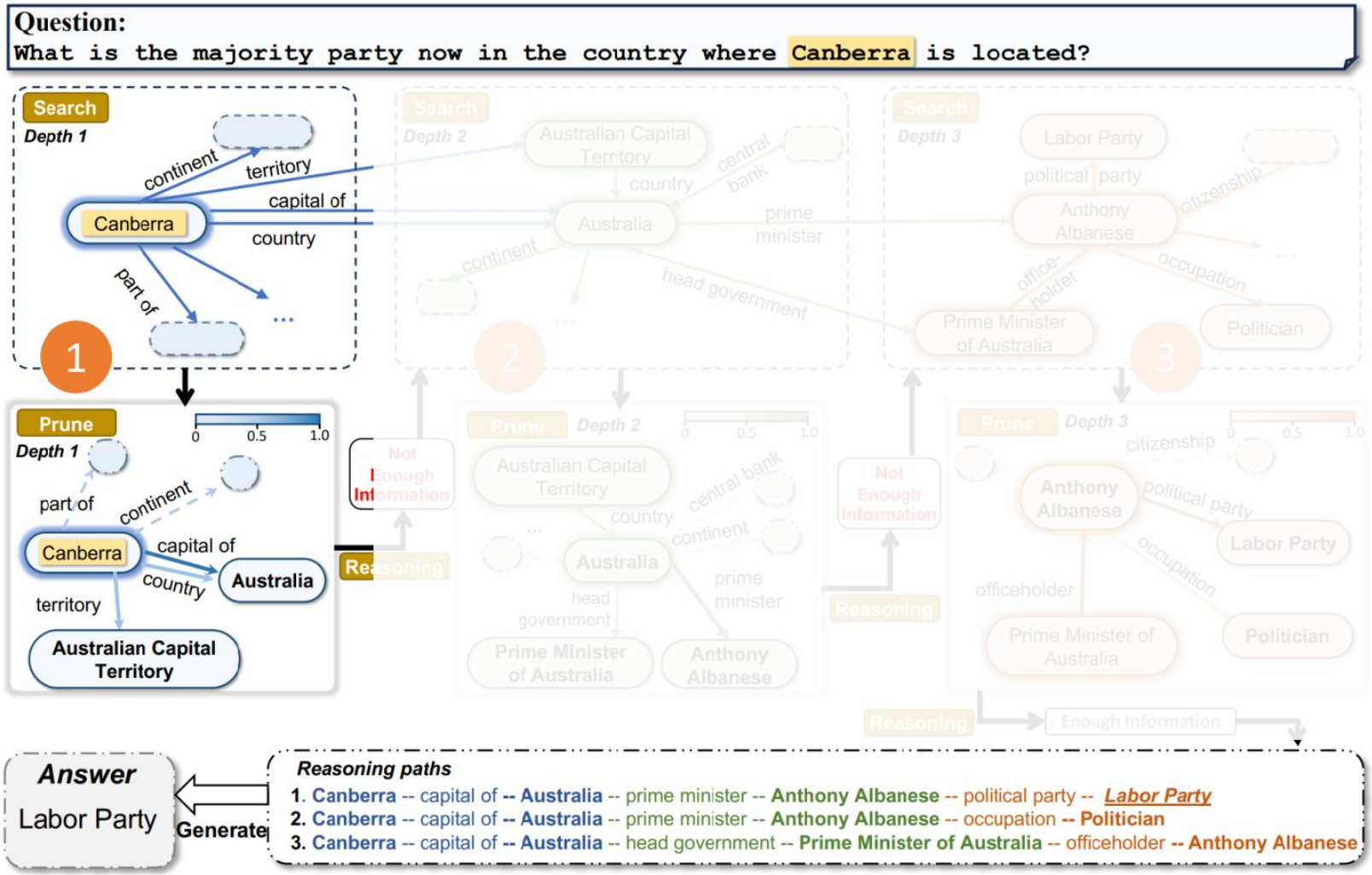
Think-on-Graph: Deep and Responsible Reasoning of LLM on Knowledge Graph

- **Goal:** treats the LLM as an agent to travel KGs and perform reasoning based on paths

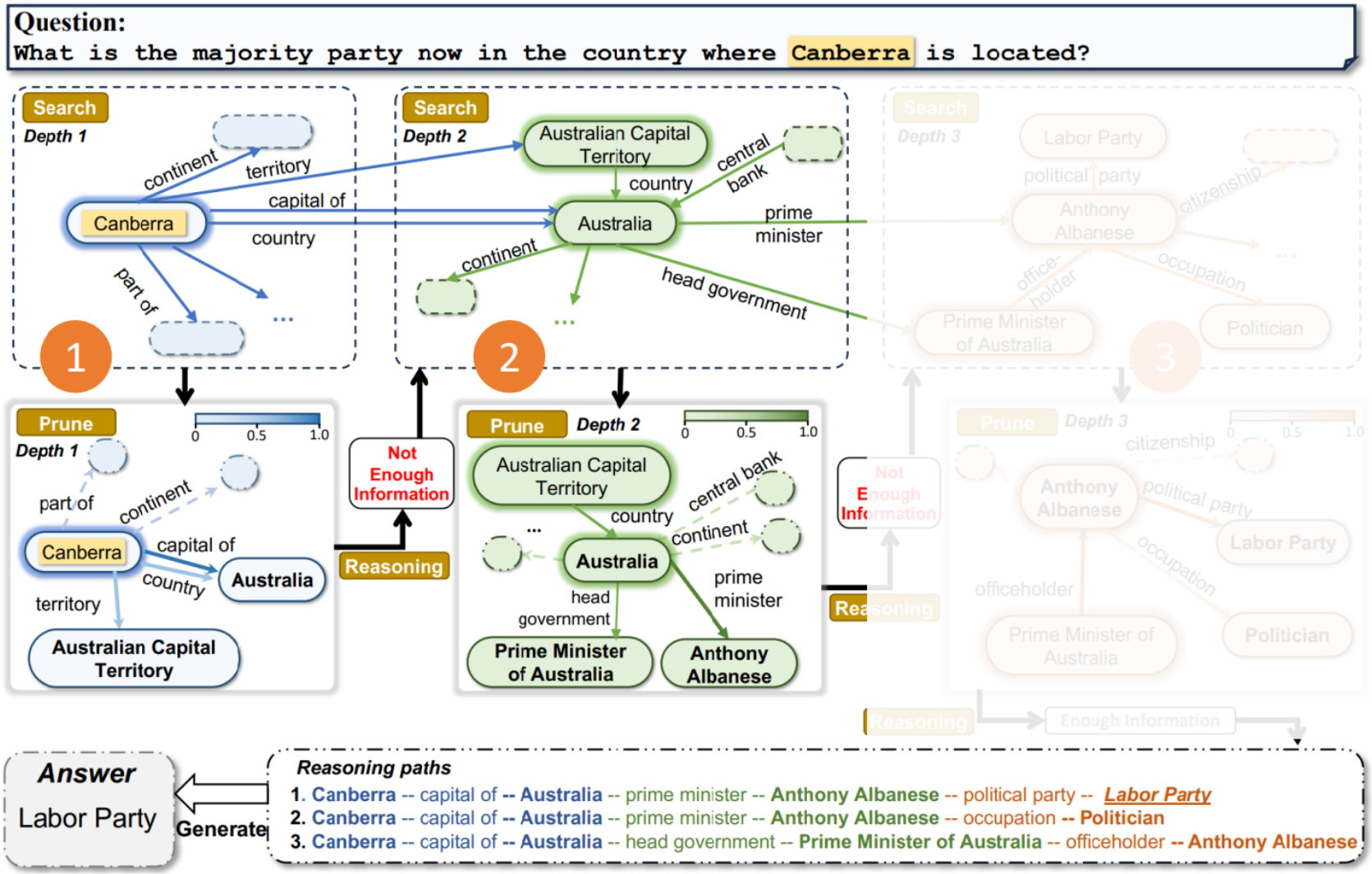


(a): LLM-only (e.g., Chain-of-Thought prompting), (b): LLM ⊕ KG (e.g., KBQA via LLM-generated SPARQL query), (c): LLM ⊗ KG (e.g., Think-on-Graph).

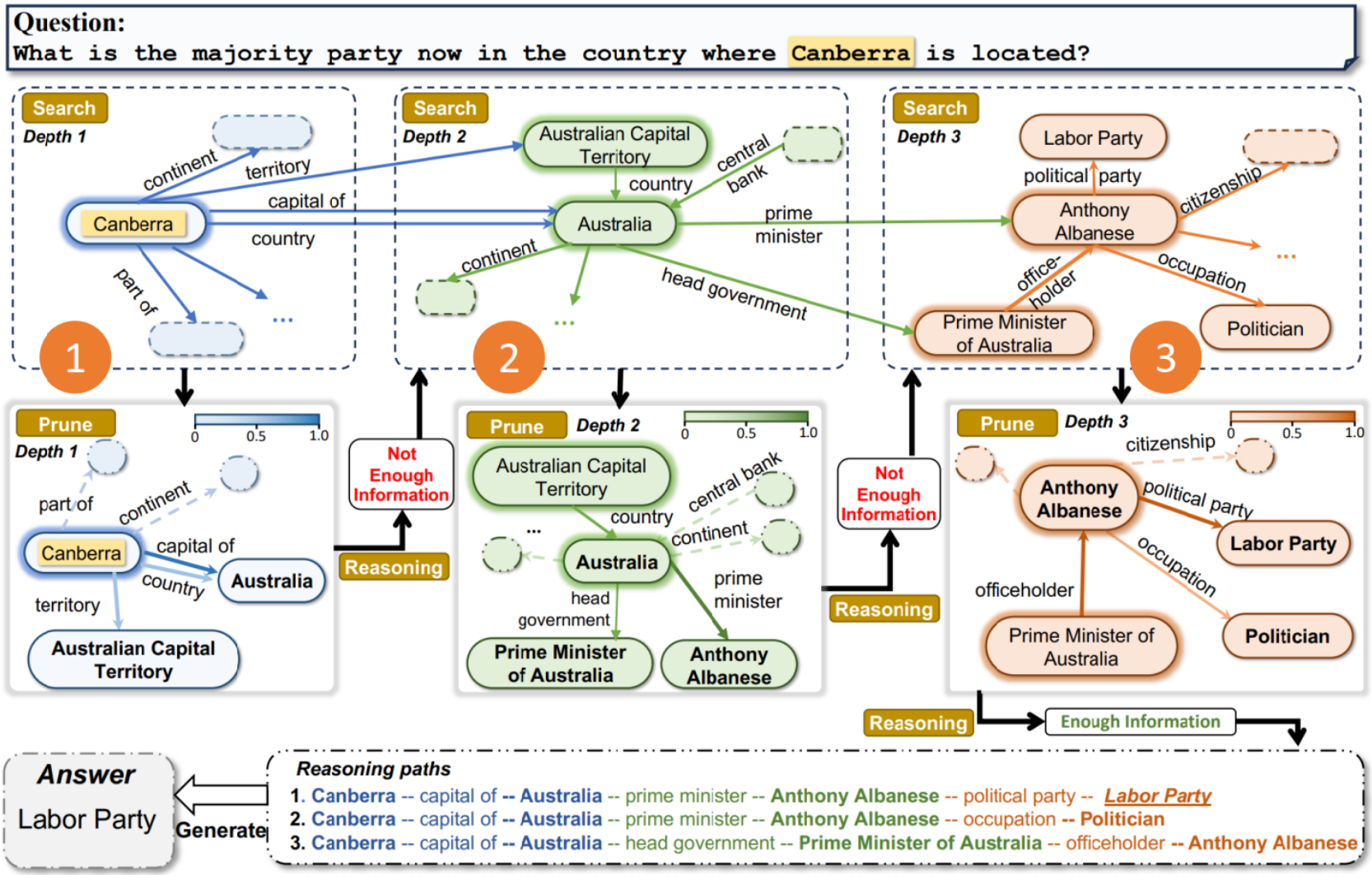
Think-on-Graph: Deep and Responsible Reasoning of LLM on Knowledge Graph



Think-on-Graph: Deep and Responsible Reasoning of LLM on Knowledge Graph



Think-on-Graph: Deep and Responsible Reasoning of LLM on Knowledge Graph

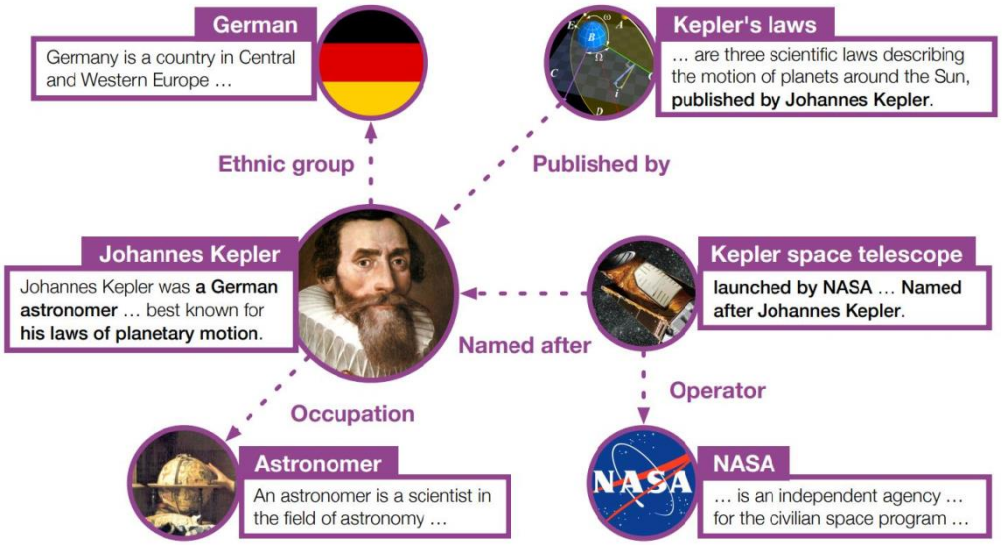


Knowledge Graph and Large Language Models

Synergized LLMs + KGs

KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation

- Motivation



Knowledge graph embedding

Infuse world facts

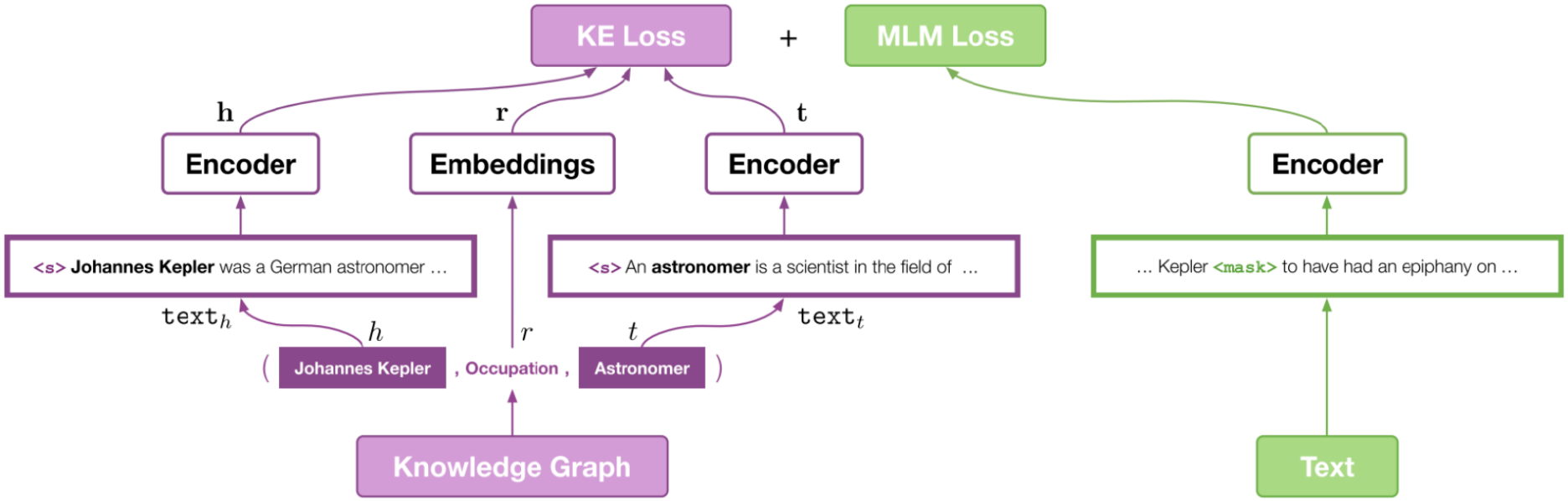
Leverage textual information do inductive KE

Pre-trained Language Models

Paper: X. Wang, T. Gao, Z. Zhu, Z. Zhang. 2021. KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation (ACL 21)

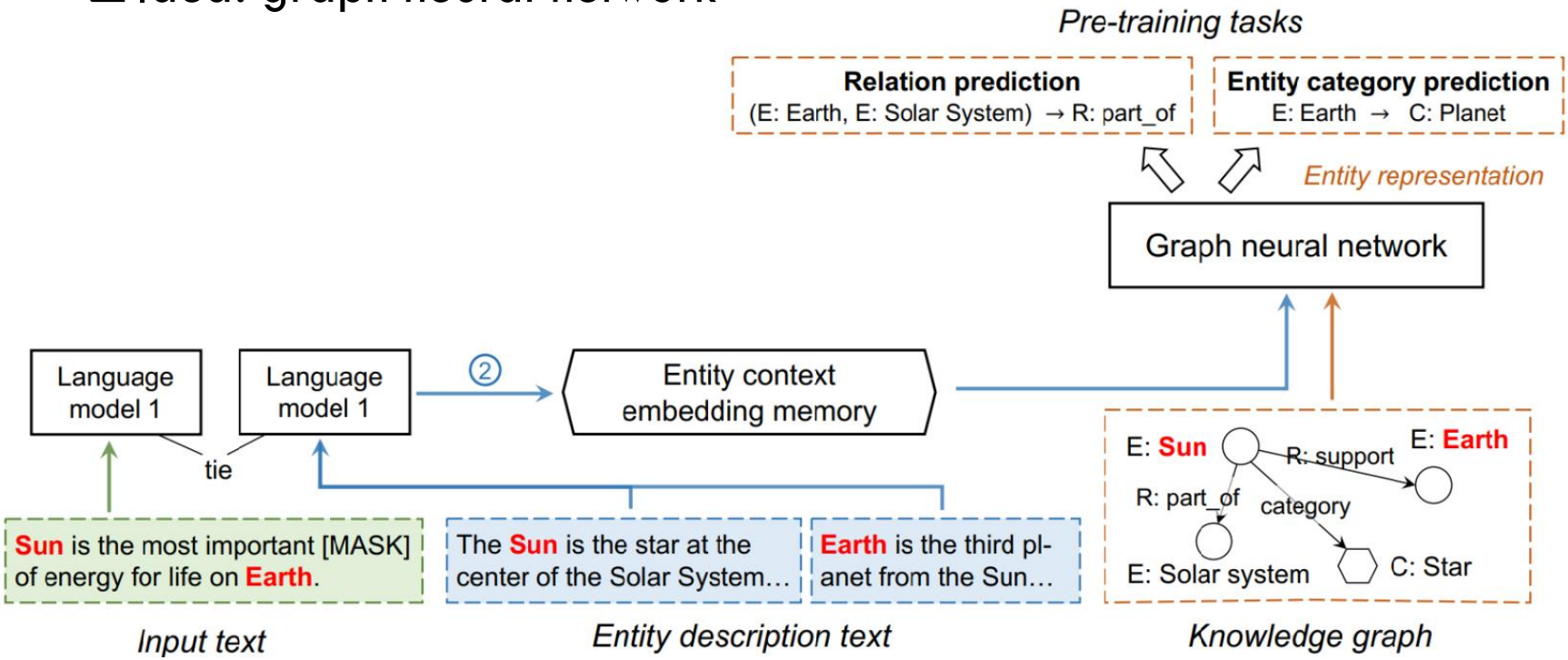
KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation

- **Goal:** joint optimizing knowledge graph embedding and masked language modeling (MLM) objectives



JAKET: Joint Pre-training of Knowledge Graph and Language Understanding

- **Goal:** Capture the knowledge graph structure information
  - TransE based method could not capture the structure information of KG
  - Idea: graph neural network



Paper: D. Yu, C. Zhu, Y. Yang, M. Zeng. 2022. JAKET: Joint Pre-training of Knowledge Graph and Language Understanding (AAAI 22)

JAKET: Joint Pre-training of Knowledge Graph and Language Understanding

- **Advantages:**
  - Eases the semantic matching between them
  - Solves the over-parameterization issue since entity embeddings are no longer part of the model's parameters