Unifying Large Language Models and Knowledge Graphs: A Roadmap

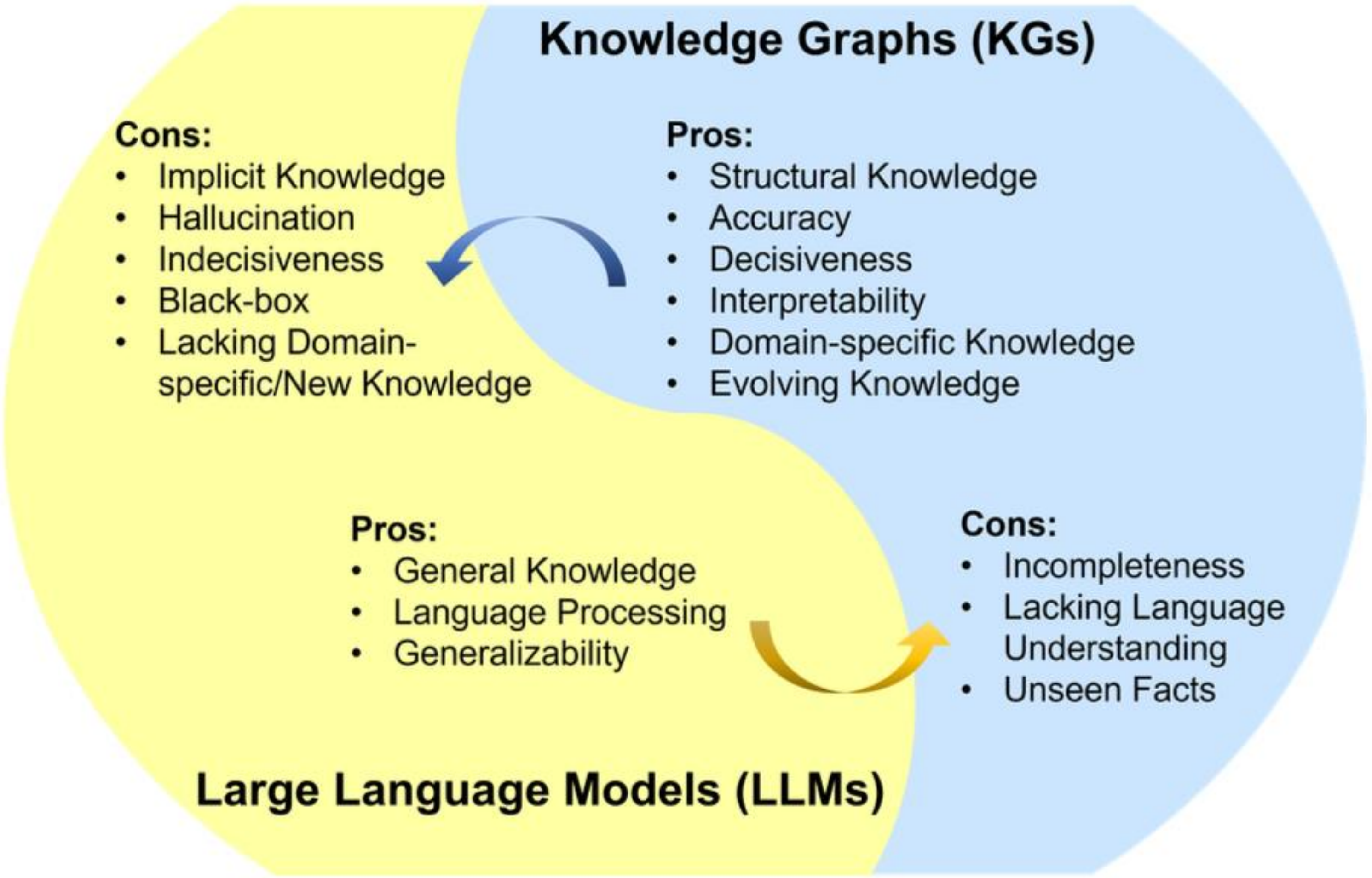
Shirui Pan, *Senior Member, IEEE*, Linhao Luo,
Yufei Wang, Chen Chen, Jiapu Wang, Xindong Wu, *Fellow, IEEE*

+ 50% Some Insights of mine

Interpreted by:

John Tan Chong Min

Knowledge Graphs and LLMs



Retrieval Augmented Generation (RAG)

- <Context 1>
- <Context 2>
- <Context 3>

- <Query>

Problems with LLMs

- Base LLM may not be able to recall all knowledge it is trained on

- May not be able to retrieve the right context using RAG
  - Embedding vector may not be comprehensive enough for specific use case

- Knowledge Graphs may be useful to retrieve the right context by searching the right keywords and retrieving the right subgraphs

ChatGPT may output differently with different languages

C Who is Singapore's President?

As of my last update in September 2021, the President of Singapore was Halimah Yacob. However, please note that political positions can change, and it's essential to verify this information with a current and reliable source, as my data may be outdated. I recommend checking the latest news or official Singaporean government sources for the most up-to-date information on Singapore's President.

C 新加坡的总统是谁?

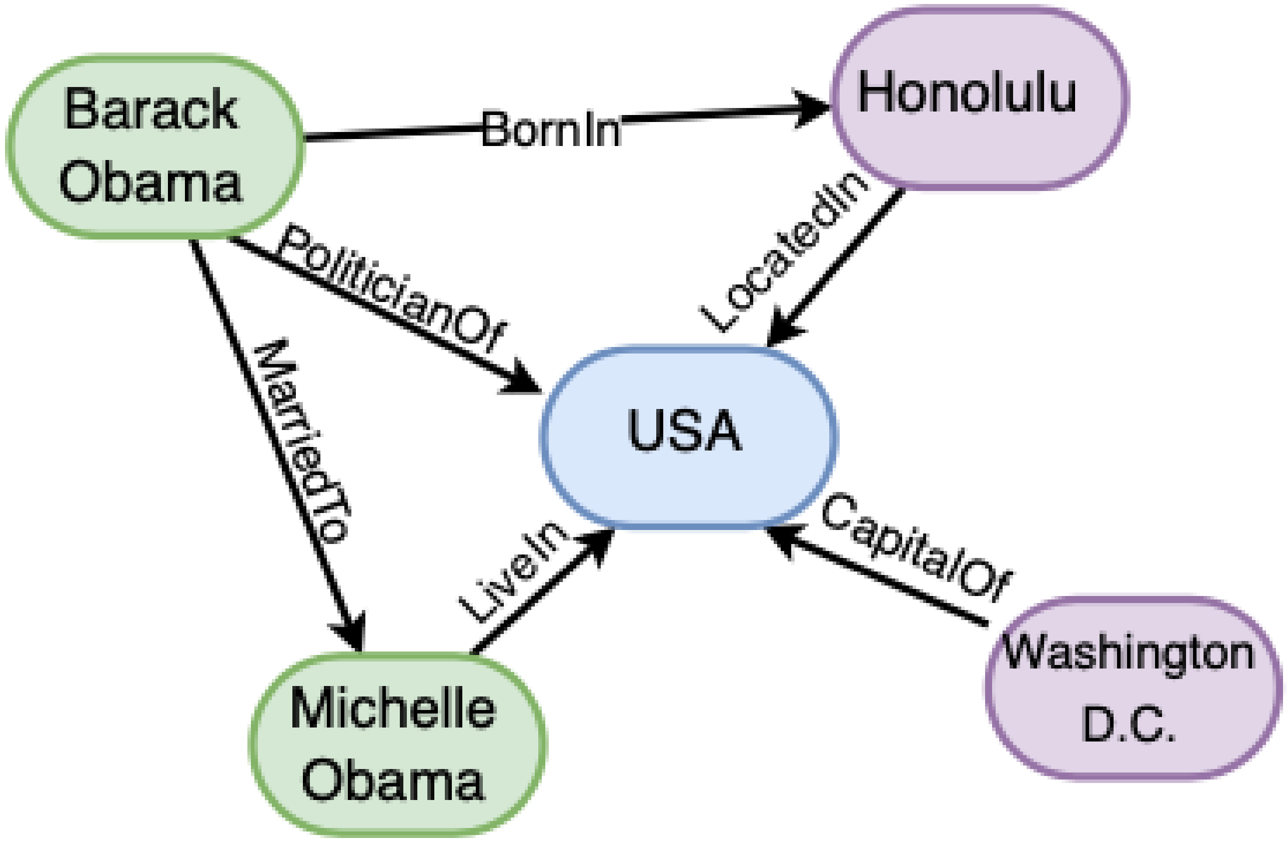
截至我所掌握的信息，新加坡没有总统。新加坡是一个共和制国家，其政府体系是议会制，国家元首是总统，而政府首脑是总理。新加坡的最高行政职位是总统，但总统的角色主要是象征性的，职责相对有限。新加坡的实际政府运作由总理领导。

然而，请注意我的知识截止日期是2021年9月，情况可能发生了变化。如果你对当前的新加坡政治局势感兴趣，建议查阅最新的来源以获取最新信息。

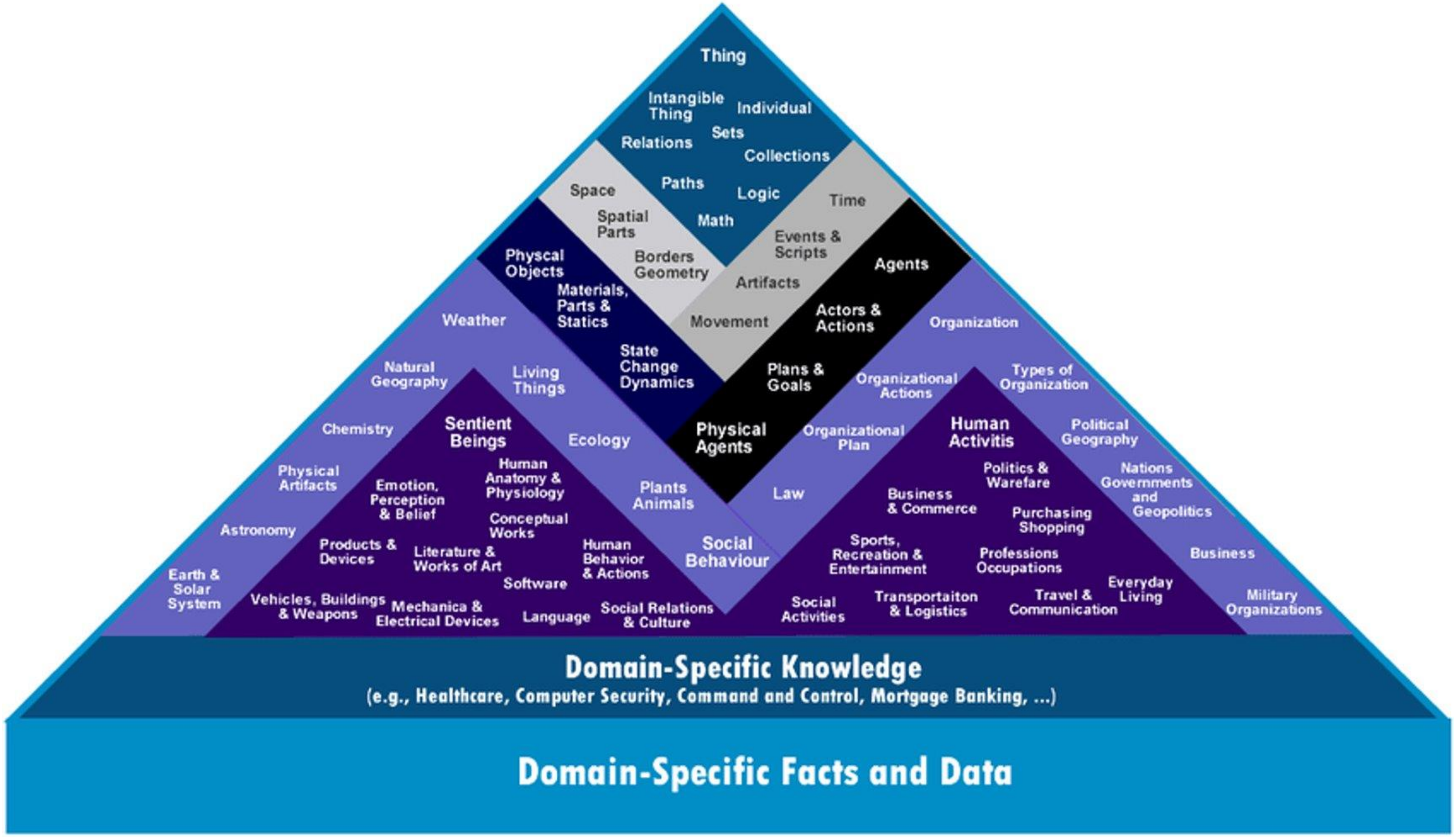
Adapted from: Keynote, Knowledge Graph Conference 2023, Denny Vrandečić
The Future of Knowledge Graphs in a World of Large Language Models
https://www.youtube.com/watch?v=WqYBx2gB6vA

What is a Knowledge Graph?

- Triplets:
  - {Source, Destination, Relation}
- Typically a Directed Graph



Hierarchy in Knowledge Graphs

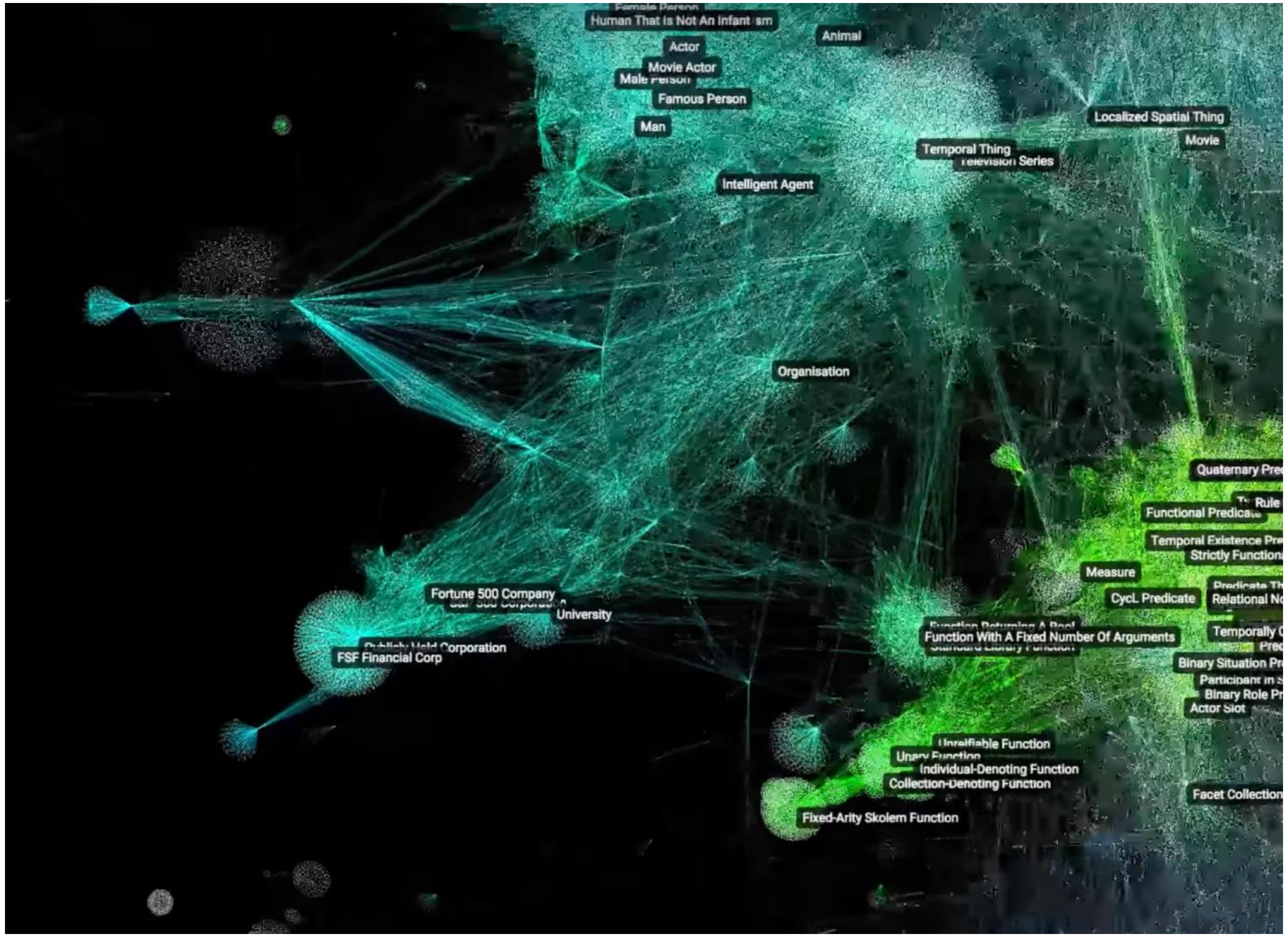


Cyc Knowledge Base.

Computational Analysis on a Corpus of Political Satire Articles: A Theoretical and Experimental Study.

Stingo et al. 2016.

Cyc Knowledge Graph – Common-sense reasoning across nodes



"A frightened person" in CycL:

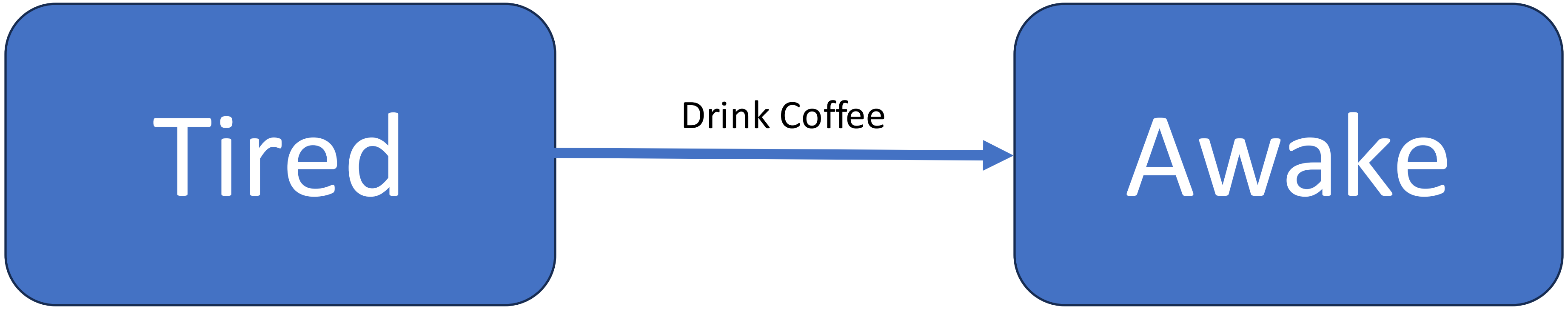
(#$and
  (#$isa ?x #$Person)
  (#$feelsEmotion ?x #$Fear #$High))

Can we leverage the power of LLMs to improve upon this?

https://www.youtube.com/watch?v=8NRvhGm9EOg

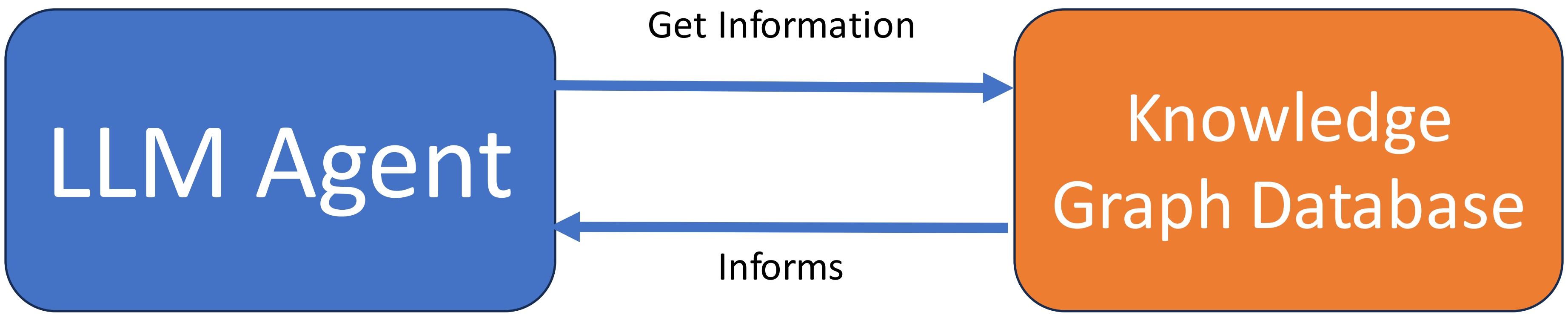
Knowledge Graphs and Causal Relations

- The relation between Source and Destination can be a causal link

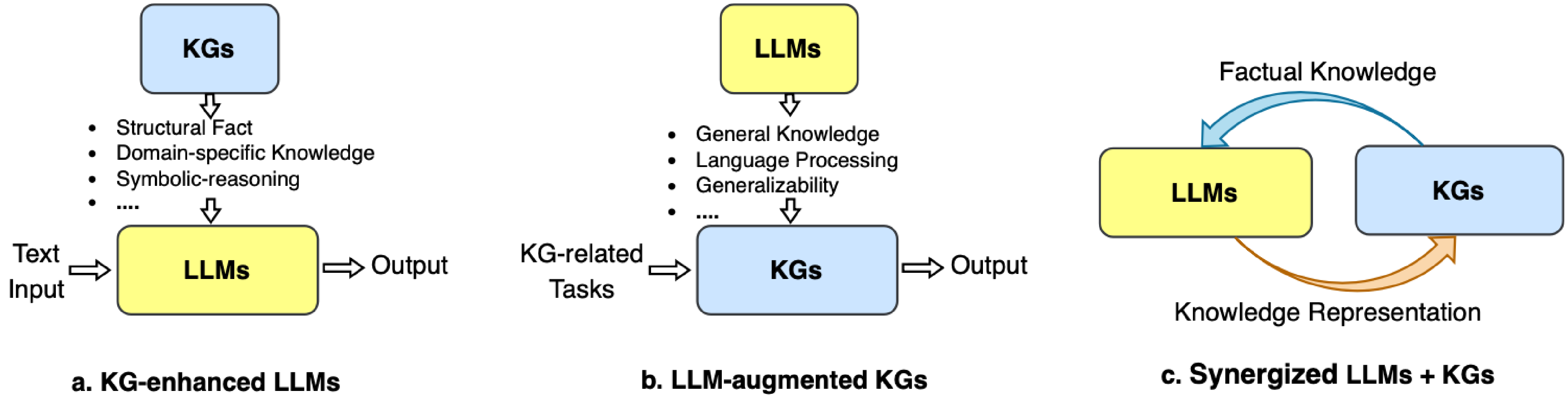


Can Knowledge Graph be viewed as a tool/memory?

- Perhaps a more efficient way to get data than querying large corpuses of text
    - Extracting relevant parts of a Knowledge Graph can serve as a way to retrieve context
- May embody causal relations easily
- Knowledge Graphs can grow dynamically, much like memory



Knowledge Graphs and LLMs: 3 Approaches



Approach 1

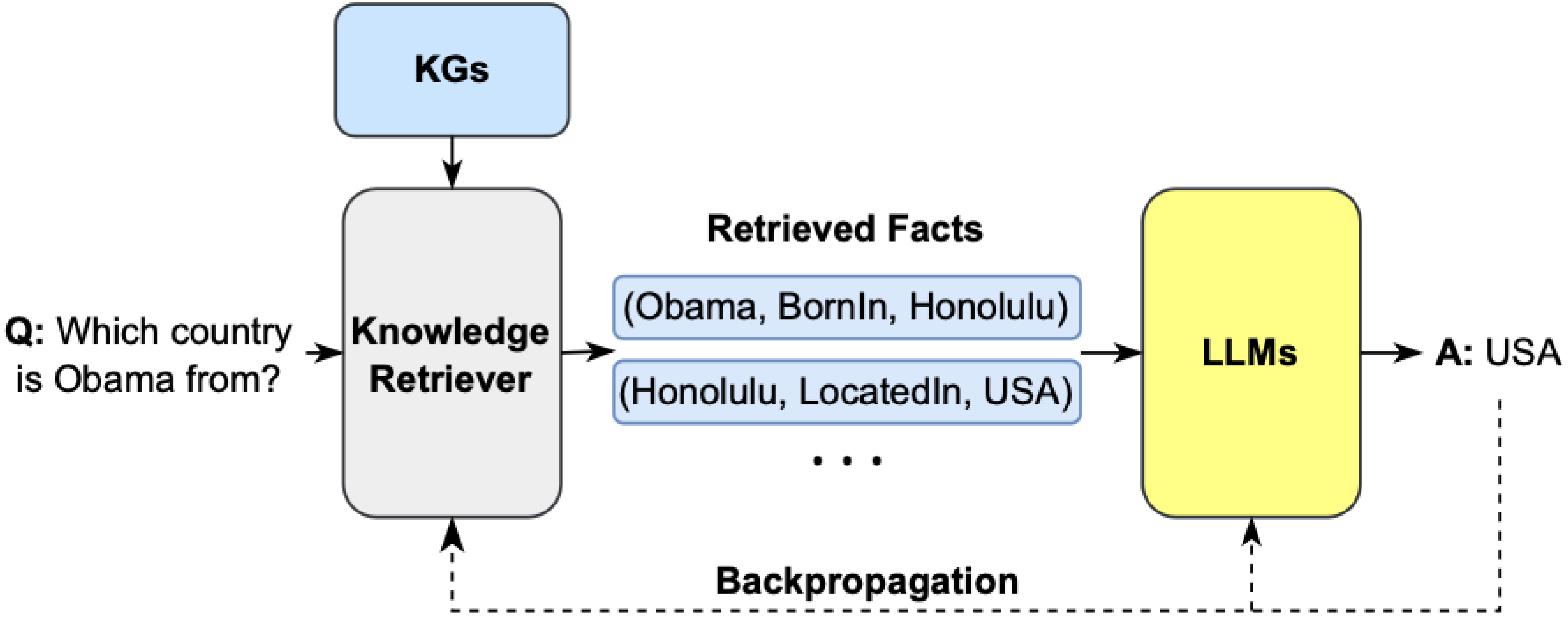
KG-Augmented LLMs

Approach 1: KG-augmented LLMs

- **KG as Text:** Pass in Knowledge Graph (Generative Agents: Interactive Simulacra of Human Behavior, Park et al, 2023) as text

- **KG as Object:** Treat KG as an object and pass it into the model
  - Typically uses Graph Neural Networks to embody KG information

KG to retrieve text for LLMs



Or non-gradient methods (aka Memory)

Recursive prompting using JSON structure to ground actions

[Agent's Summary Description]
Eddy Lin is currently in The Lin family's house: Eddy Lin's bedroom: desk

Related areas in The Lin family's house: Mei and John Lin's bedroom, Eddy Lin's bedroom, common room, kitchen, bathroom, and garden.

Eddy Lin knows of the following areas: The Lin family's house, Johnson Park, Harvey Oak Supply Store, The Willows Market and Pharmacy, Hobbs Cafe, The Rose and Crown Pub.

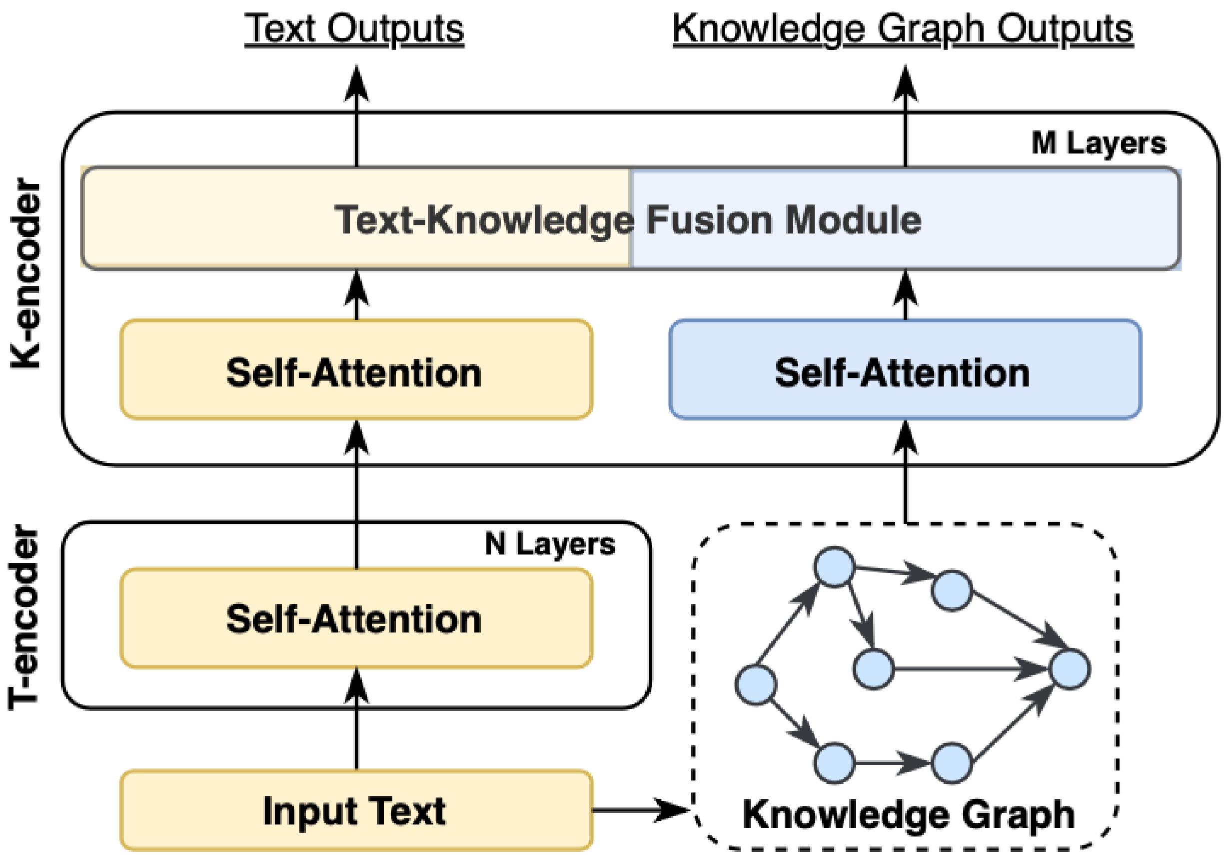
* Prefer to stay in the current area if the activity can be done there.
Eddy Lin is planning to take a short walk around his workspace.
Which area should Eddy Lin go to?

- Recursively use prompting to prompt for lower and lower sub-areas based on the JSON tree
- Ground the agent in feasible actions via prompting
- JSON structure to ground agent's actions is like grounding with a Knowledge Graph

Generative Agents: Interactive Simulacra of Human Behavior. Park et al. 2023.

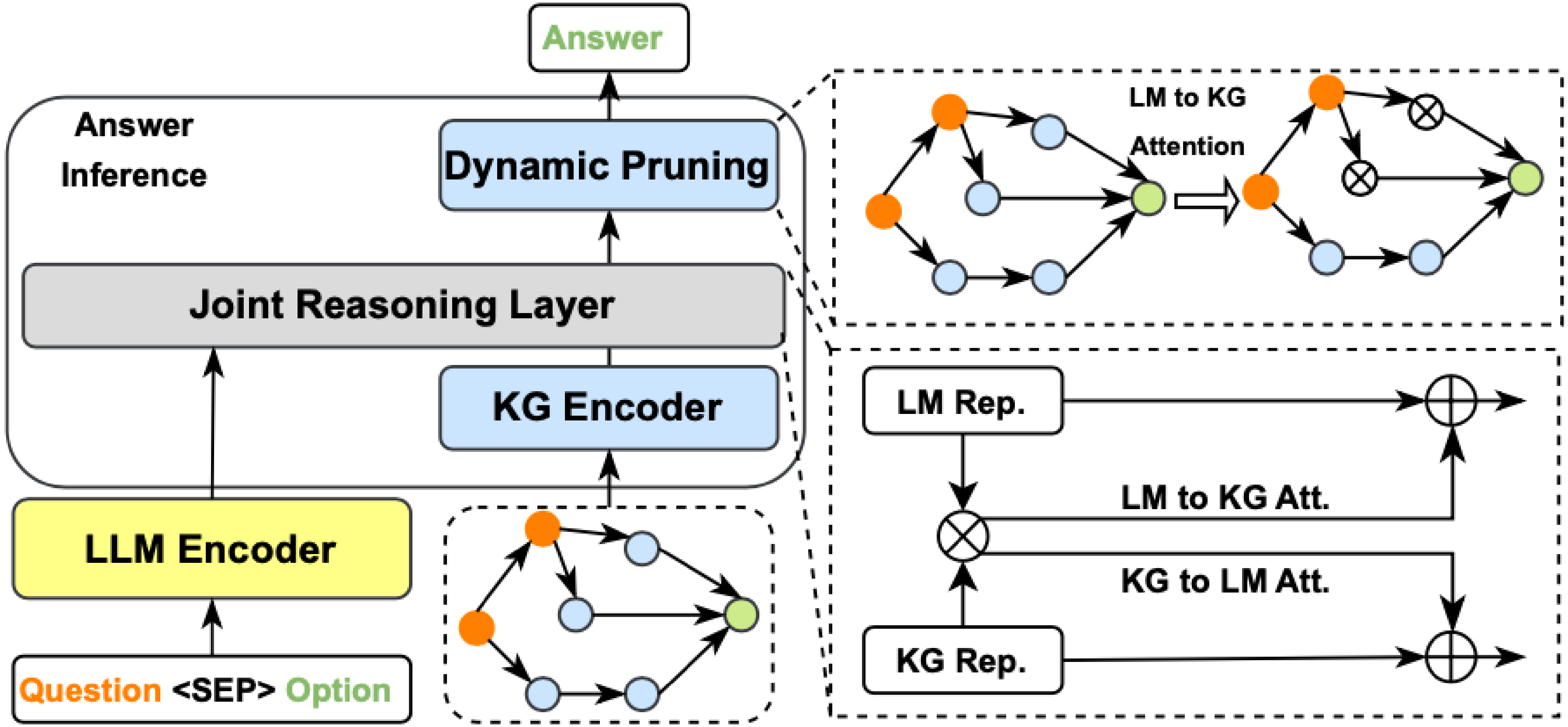
ERNIE – Hybrid Input/Output

- LM and KG are both jointly used for processing



QA-GNN – Two-way interaction between LM and KG

- LLM and KG influences each other



Approach 2

LLM-Augmented KGs

Approach 2: LLM-augmented KG

- Use LLM to generate KG via zero-shot/few-shot prompting
  - e.g. LangChain (not very good)

- LLMs as text encoders (embedding space) to enrich KG representations

Use LLMs to generate Knowledge Graphs

- LLMs can be more versatile and can be few-shot prompted to generate relations!
- Better than SpaCy for Named-Entity Recognition (NER) for out-of-distribution datasets (LLMs are superb learners if prompted well)

LLM zero-shot/few-shot prompting for KG generation

You are a networked intelligence helping a human track knowledge triples about all relevant people, things, concepts, etc. and integrating them with your knowledge stored within your weights as well as that stored in a knowledge graph.

Extract all of the knowledge triples from the text.

A knowledge triple is a clause that contains a subject, a predicate, and an object.

The subject is the entity being described, the predicate is the property of the subject that is being described, and the object is the valuae of the property.

LLM zero-shot/few-shot prompting for KG generation

EXAMPLE

It's a state in the US. It's also the number 1 producer of gold in the US.

Output: (Nevada, is a, state), (Nevada, is in, US), (Nevada, is the number 1 producer of, gold)

END OF EXAMPLE

EXAMPLE

I'm going to the store.

Output: NONE

END OF EXAMPLE

EXAMPLE

Oh huh. I know Descartes likes to drive antique scooters and play the mandolin.

Output: (Descartes, likes to drive, antique scooters),(Descartes, plays, mandolin)

END OF EXAMPLE

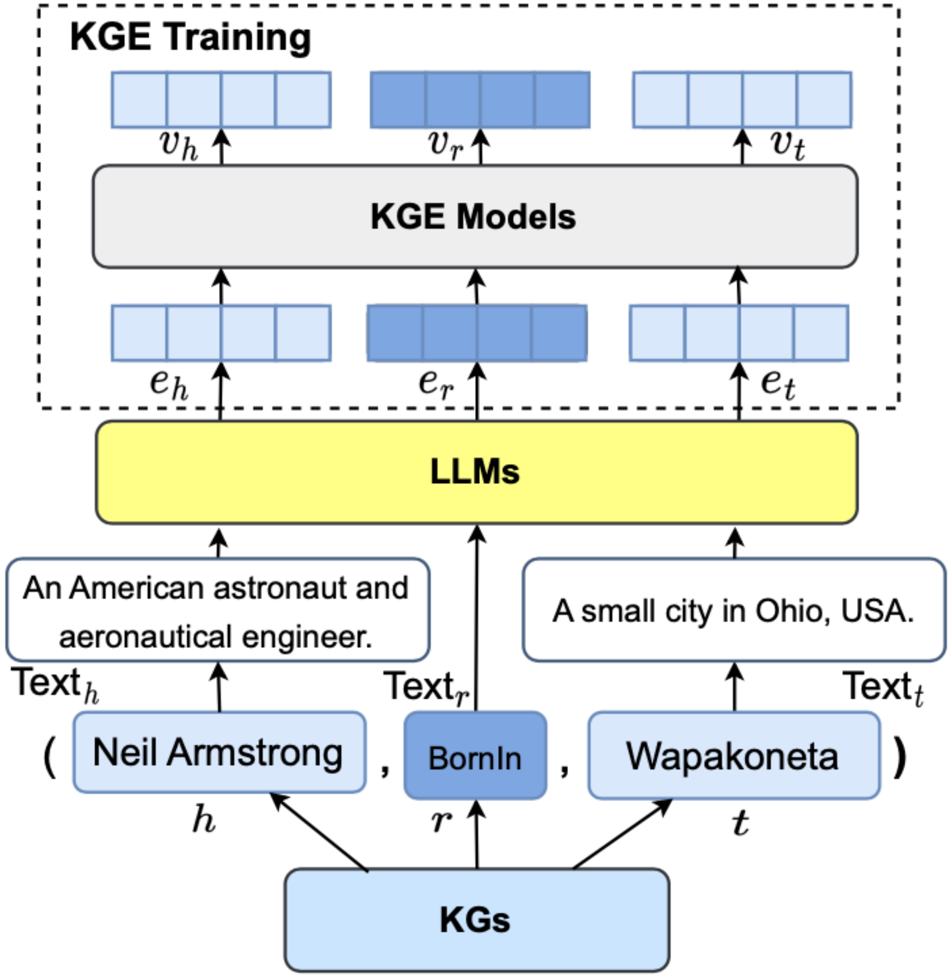
EXAMPLE

{text}

Output:

LLMs as text encoders for Knowledge Graph Embeddings

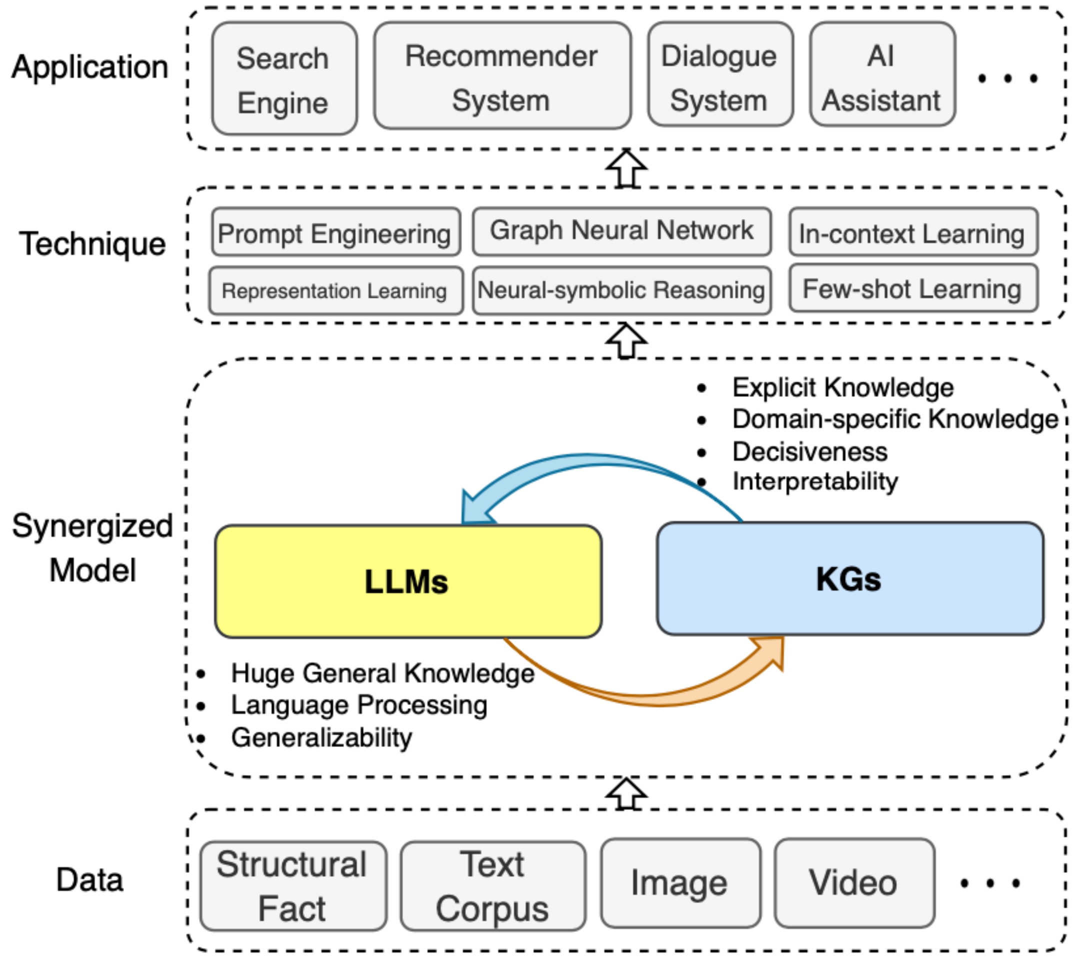
- KG built based on embedding space



Approach 3

LLMs and KG two-way interaction

Approach 3: Synergy between Knowledge Graphs and LLMs



Knowledge Graphs for Fact-Checking!

- We can perhaps use knowledge graphs to perform fact-checking
- Here it is done as pre-training, but we can/should also use KGs during inference

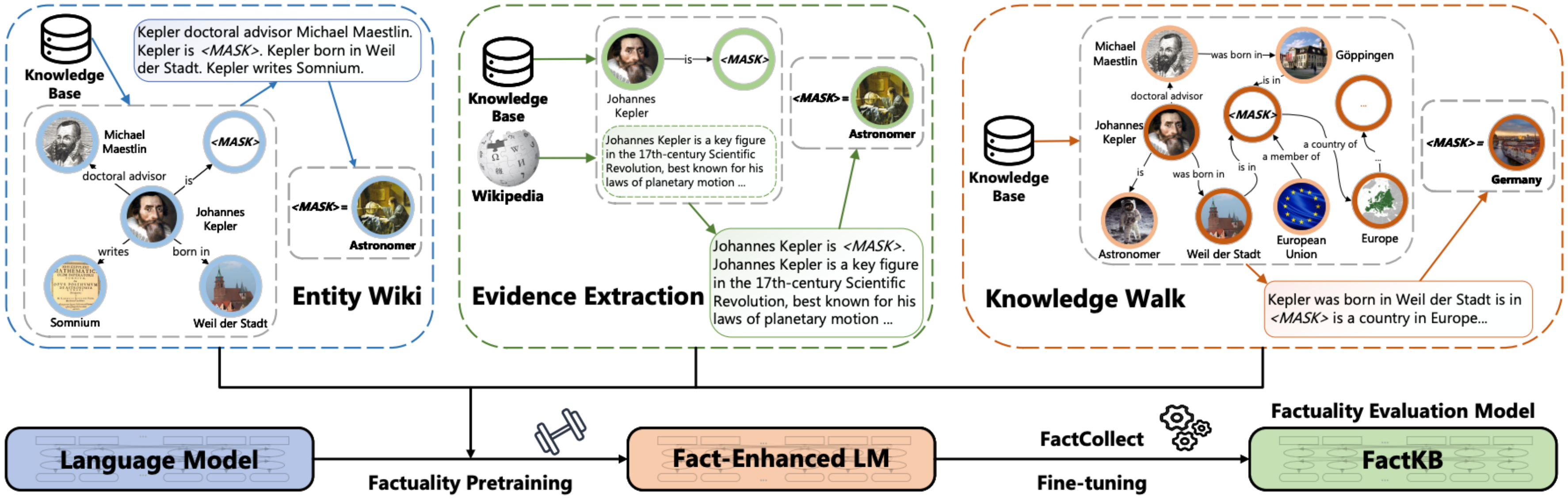


Figure 2: Overview of FACTKB. FACTKB pretrains LMs using three entity-centric pretraining strategies to improve fact representations. The objectives are designed to fill masked entities/relations in KB facts using i) *Entity Wiki* - direct facts about entities ii) *Evidence Extraction* - auxiliary knowledge about entities and iii) *Knowledge Walk* - compositional knowledge from the KB. The pretrained LMs are then fine-tuned for robust factuality evaluation.

FACTKB. Feng et. al. 2023.

LangChain Graph QA Example

LangChain Graph Creation and Utilisation Process

- Step 1: Generate triplets from context
- Step 2: Entity Extraction from query
- Step 3: Use entities to extract relevant triplets
- Step 4: Using triplets to answer question

https://github.com/langchain-ai/langchain/blob/master/libs/langchain/langchain/indexes/graph.py
https://github.com/langchain-ai/langchain/blob/master/libs/langchain/langchain/chains/graph_qa/base.py

Step 1: Generate Triplets from Context

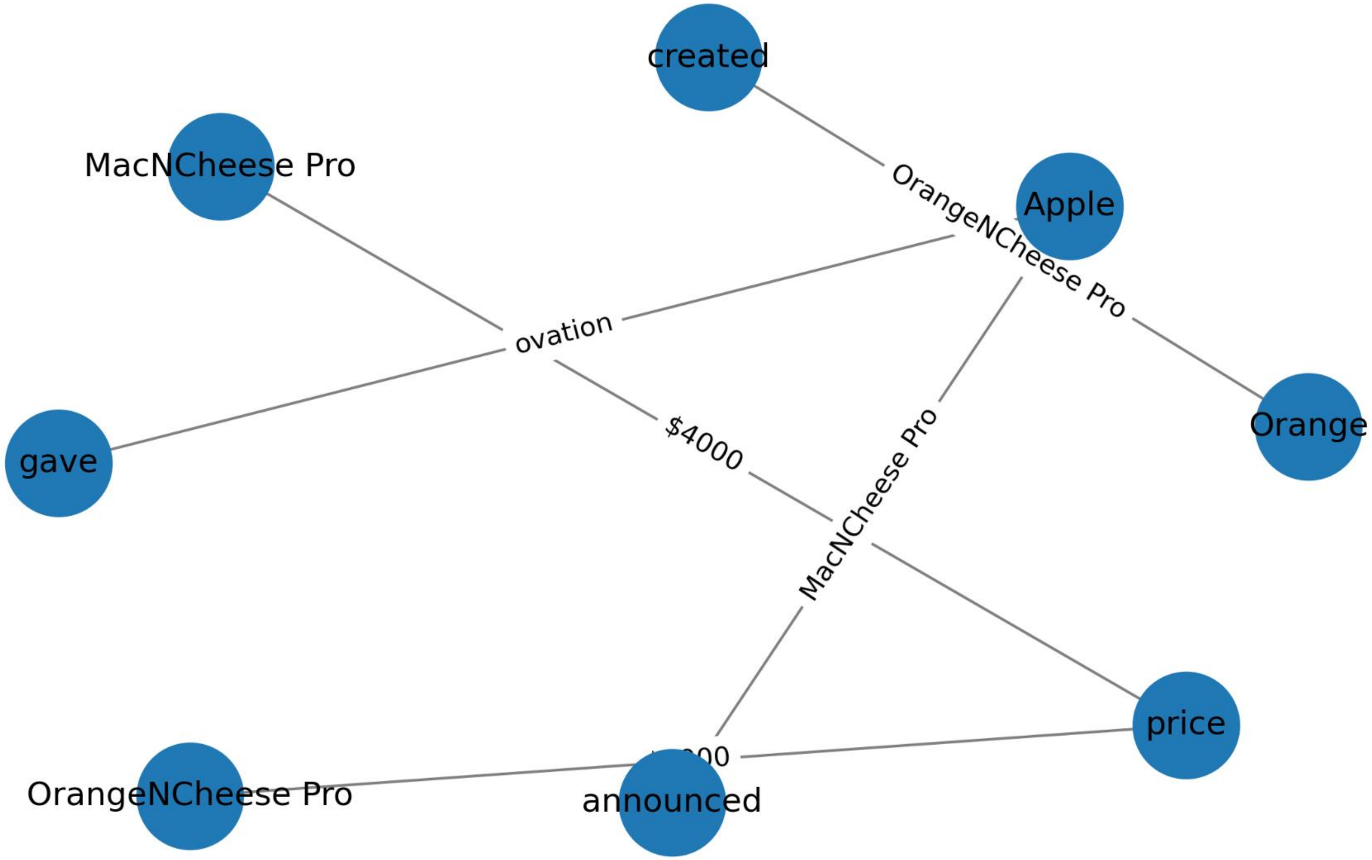
```
from langchain.llms import OpenAI
from langchain.indexes import GraphIndexCreator
from langchain.chains import GraphQAChain

context ='''Apple announced the MacNCheese Pro in 2025. It proved a big hit.
Apple gave Cheese a rousing ovation in 2026 after he invented the MacNCheese Pro in 2024.
Orange created a competing product called the OrangeNCheese Pro.
It's price was slightly higher at $5000, compared to Apple's $4000.'''

index_creator = GraphIndexCreator(llm=OpenAI(temperature=0))
graph = index_creator.from_text(context)
graph.get_triples()
```

```
[('Apple', 'MacNCheese Pro', 'announced'),
 ('Apple', 'Cheese', 'gave'),
 ('Apple', 'ovation', 'gave'),
 ('MacNCheese Pro', '$4000', 'price'),
 ('Orange', 'OrangeNCheese Pro', 'created'),
 ('OrangeNCheese Pro', '$5000', 'price')]
```

Generated Knowledge Graph



Steps 2-4: Extract and use relevant KG for query

```
chain = GraphQAChain.from_llm(OpenAI(temperature=0), graph=graph, verbose=True)
chain.run(question)
```

```
> Entering new GraphQAChain chain...
Entities Extracted:
 Apple, MacNCheese Pro
Full Context:
Apple announced MacNCheese Pro
Apple gave Cheese
Apple gave ovation
MacNCheese Pro price $4000

> Finished chain.
```

```
' Apple announced the MacNCheese Pro recently.'
```

Compare to just feeding in context directly

```python
# langchain QA

from langchain import PromptTemplate, OpenAI, LLMChain

template = """Context: {context}

Question: {question}

Answer: Let's think step by step."""
prompt = PromptTemplate(template=template, input_variables=["context", "question"])
llm_chain = LLMChain(prompt=prompt, llm=OpenAI(temperature=0), verbose=True)

llm_chain.run({"context": context, "question": question})
```

```
> Entering new LLMChain chain...
Prompt after formatting:
Context: Apple announced the MacNCheese Pro in 2025. It proved a big hit.
Apple gave Cheese a rousing ovation in 2026 after he invented the MacNCheese Pro in 2024.
Orange created a competing product called the OrangeNCheese Pro.
It's price was slightly higher at $5000, compared to Apple's $4000.

Question: When did apple announce the MacNCheese Pro?

Answer: Let's think step by step.

> Finished chain.
' Apple announced the MacNCheese Pro in 2025.'
```

How I would do it

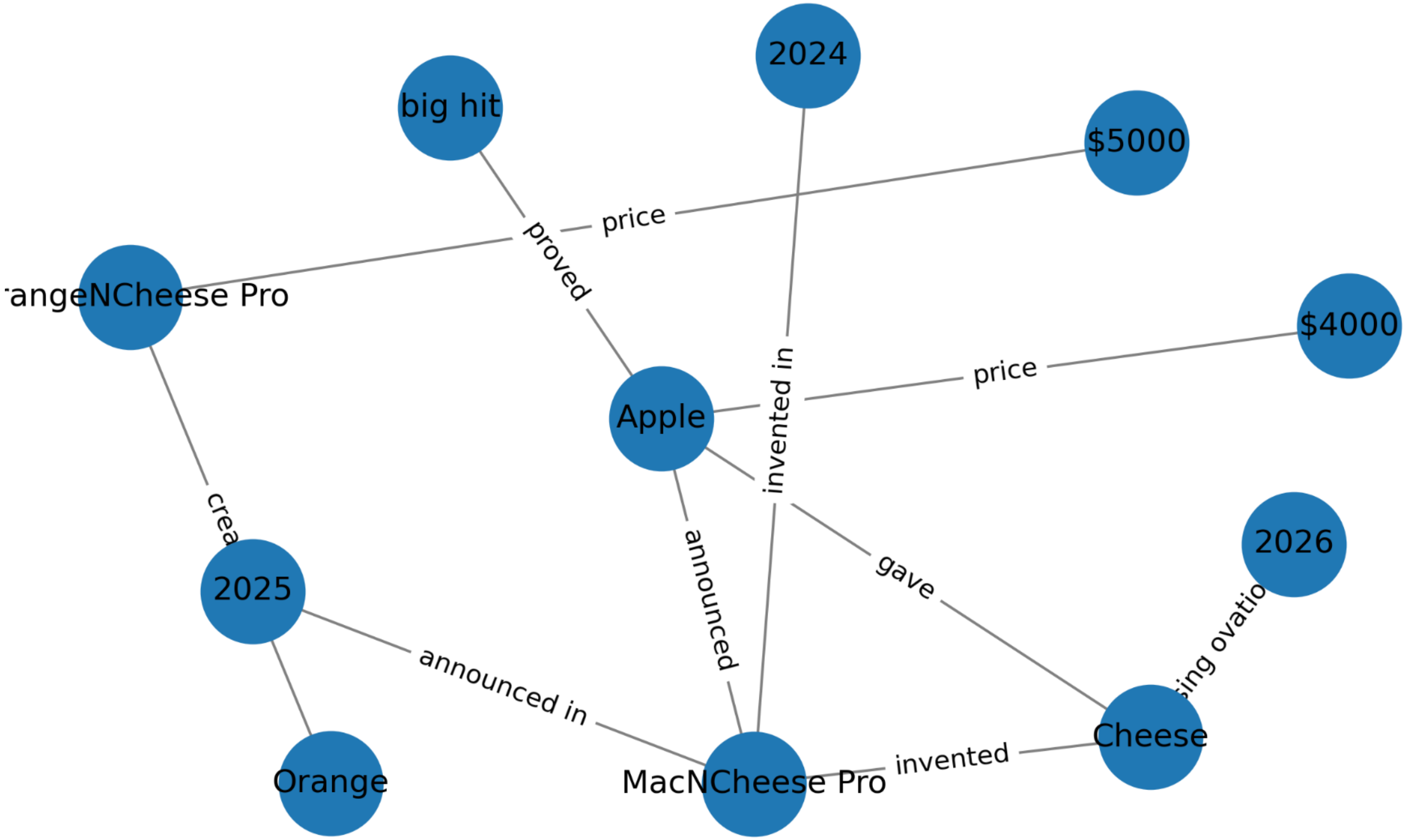
Strict JSON Framework

Step 1: Generate Knowledge Graph from context

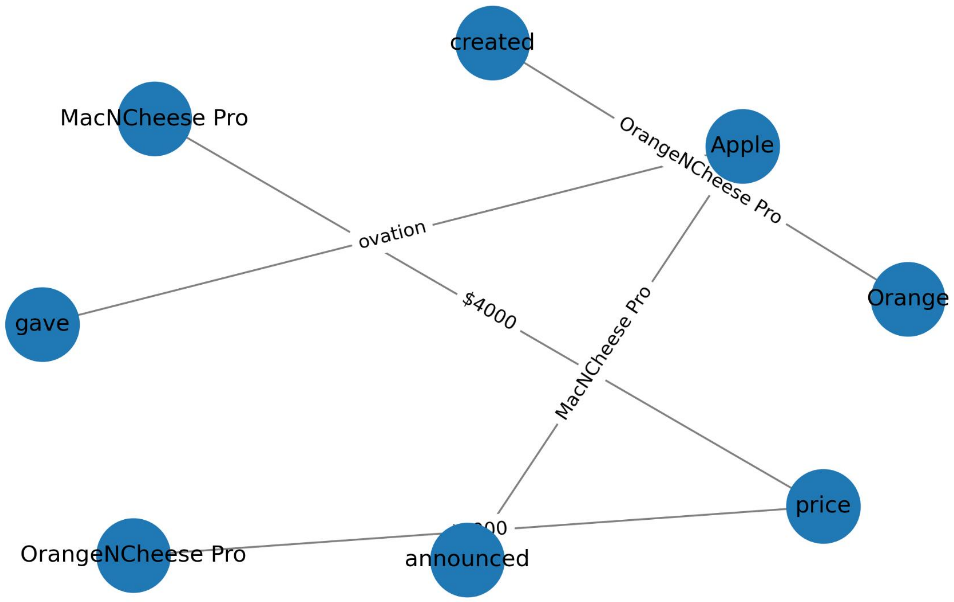
```
res = strict_output(system_prompt = '''You are a knowledge graph builder.
You are to output relations between two objects in the form [object_1, relation, object_2].
All information about dates must be included.
Example Input: John bought a laptop
Example Output: [['John', 'bought', 'laptop']]
Example Input: John built a house in 2019
Example Output: [['John', 'built', 'house'], ['house', 'built in', '2019']]''',
                    user_prompt = context,
                    output_format = {"Knowledge Graph": "List of relations of the form [object_1, relation, object_2]"}
print(res)
```

```
{'Knowledge Graph': [['Apple', 'announced', 'MacNCheese Pro'], ['MacNCheese Pro', 'announced in', '2025'], ['Apple', 'p
roved', 'big hit'], ['Apple', 'gave', 'Cheese'], ['Cheese', 'rousing ovation in', '2026'], ['Cheese', 'invented', 'MacN
Cheese Pro'], ['MacNCheese Pro', 'invented in', '2024'], ['Orange', 'created', 'OrangeNCheese Pro'], ['OrangeNCheese Pr
o', 'price', '$5000'], ['Apple', 'price', '$4000']]}
```

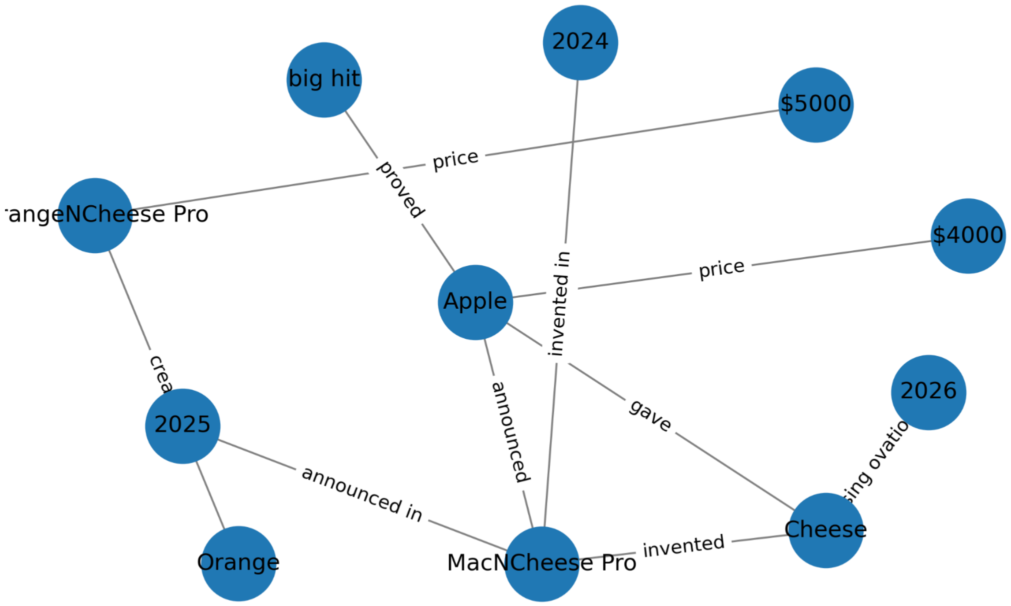
Generated Graph



Compare with LangChain



LangChain



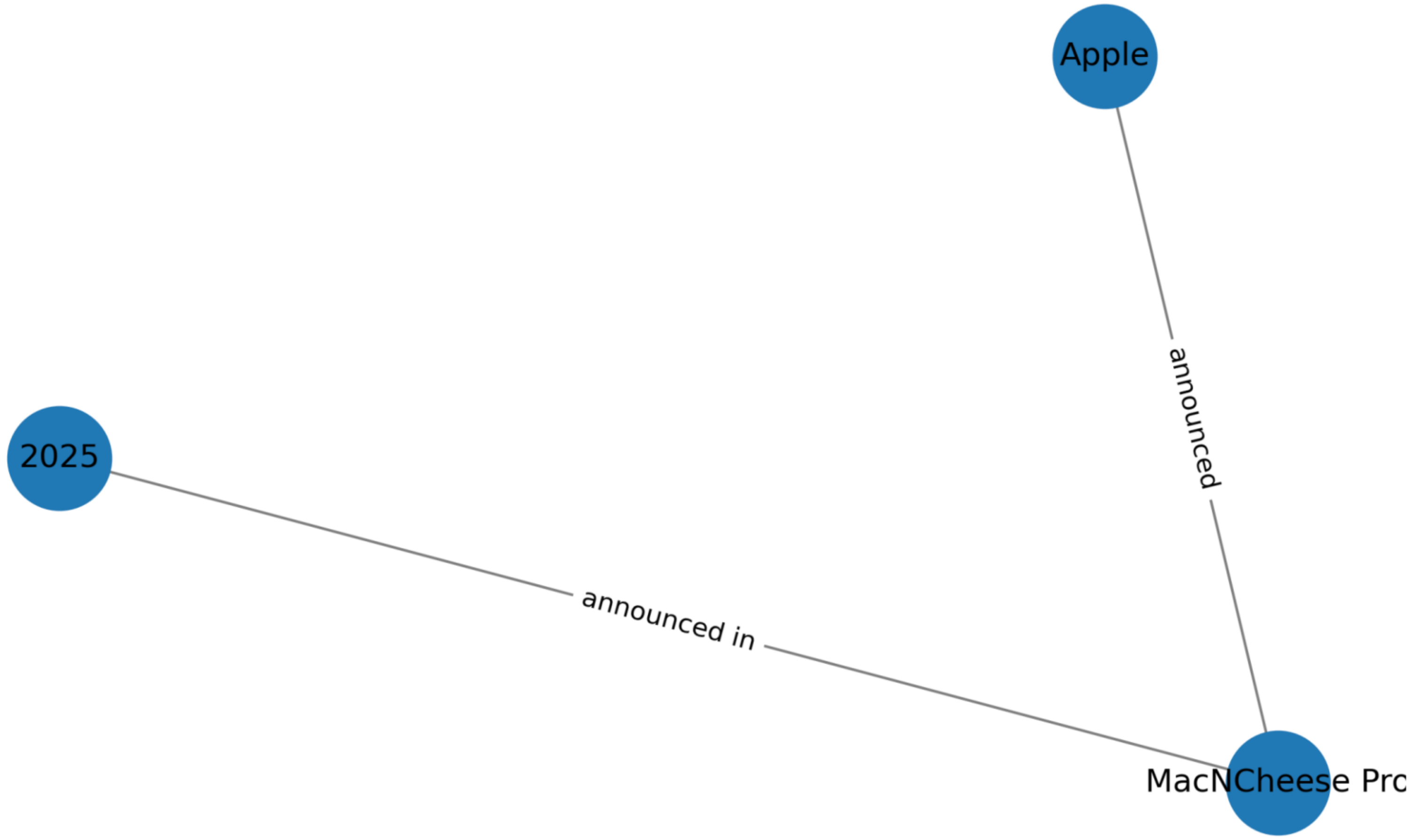
Custom StrictJSON Framework

Step 2: Flexible Knowledge Graph parsing

```python
res = strict_output(system_prompt = f'''You are a knowledge graph parser.
Only output the relations that are relevant to the question.
Knowledge graph: {kg}''',
                    user_prompt = f'''Question: {question}''',
                    output_format = {"Parsed Knowledge Graph": "List of relations of the form [object1, relation, object2]"})
print(res)
```

```
{'Parsed Knowledge Graph': [['Apple', 'announced', 'MacNCheese Pro'], ['MacNCheese Pro', 'announced in', '2025']]}
```

Parsed Knowledge Graph



Step 3: Use Parsed Knowledge Graph to Answer Question

```
res = strict_output(system_prompt = f'''Use the knowledge graph to answer the following question.
If you are unsure, output 'No Info'
Knowledge Graph: {parsed_kg}''',
                    user_prompt = f'''Question: {question}''',
                    output_format = {"Answer": "Answer question using knowledge graph"})
print('Question:', question)
print('Answer:', res['Answer'])
```

```
Question: When did apple announce the MacNCheese Pro?
Answer: 2025
```

Questions to Ponder

- What are the failure modes of using KG for context representation for LLMs?
- Should we utilise embedding space too for the Knowledge Graph, or just for the LLM?
- How can LLMs help with a more flexible interpretation/construction of a Knowledge Graph?
- How will we know what nodes are important to construct in a Knowledge Graph? Is it entirely learnable, or are there fixed biases?