Okay, so I asked Indika about this `"obo:IAO_0000310"` kind of stuff. He told me he checked the ontology and found that `obo:IAO_*` are annotations meant for describing properties, not instances. Real object properties are under `obo:MCRO_*`. So, the triples generated earlier might be using some incorrect CURIEs from the IAO namespace as object properties.

So maybe during the mapping process, when Gemini suggests a CURIE like `swo:SWO_000002`, it is incorrect. The problem is twofold: (1) make sure Gemini maps model card fields only to MCRO, and (2) filter the generated triples accordingly. Since Gemini is an LLM, it might not strictly follow the ontology. Therefore, I need to do post-processing to validate that all predicates are from MCRO.

Even though I am uploading the ontology (`mcro.ttl`) to Gemini, and it should theoretically know the valid properties, it might not be able to interpret it properly. So basically, even after providing the model, it can make mistakes. That's why I think it's better to add a validation layer in the code to filter out invalid predicates.

What I can do is:

1. Parse `mcro.ttl` and extract all valid object properties (under `obo:MCRO_*`).
2. When Gemini returns triples, validate each predicate against this list.
3. If a predicate is invalid, either skip it or attempt to find a better replacement.

So the thing is how to do this programmatically. First, I can run a SPARQL query to get all properties that are instances of `owl:ObjectProperty` and belong to the MCRO namespace. Then, during the post-processing of the JSON triples, if a predicate matches a valid MCRO property, it will be allowed—otherwise, it will be discarded and a warning will be logged. Anything that starts with `obo:IAO` will be excluded. The final output will be returned in RDF Turtle format.

Now the issue came is Gemini only again incorrectly mapping all metadata to `prov:hasTextValue` instead of MCRO ontology properties like `prov:hasTextValue` and `mcro:HasDescription` etc . So the thing is when i print how many valid mcro properties are loaded it says 0 means sparql query in `load_valid_mcro_properties()` failed to find object properties in MCRO ontology. So i modified the Sparql query to find all the MCRO namespace regardless the type. So i put code to check the path of the name space and all of them uses ""http://purl.obolibrary.org/obo/MCRO_..." instead of "http://sbmi.uth.edu/ontology/mcro#" which is given to prompt so this can be the reason that sparql query is not finding the correct property. so to fix this i think change the prefix_map in mcro_ns in code to use obo purl instead of sbmi uri . Changing namespace in file and prompt didn't work . Change the prompt to obo style CURIE's

`prompt = f"""
Using the attached MCRO ontology {mcro_file.uri}, extract the following metadata fields:

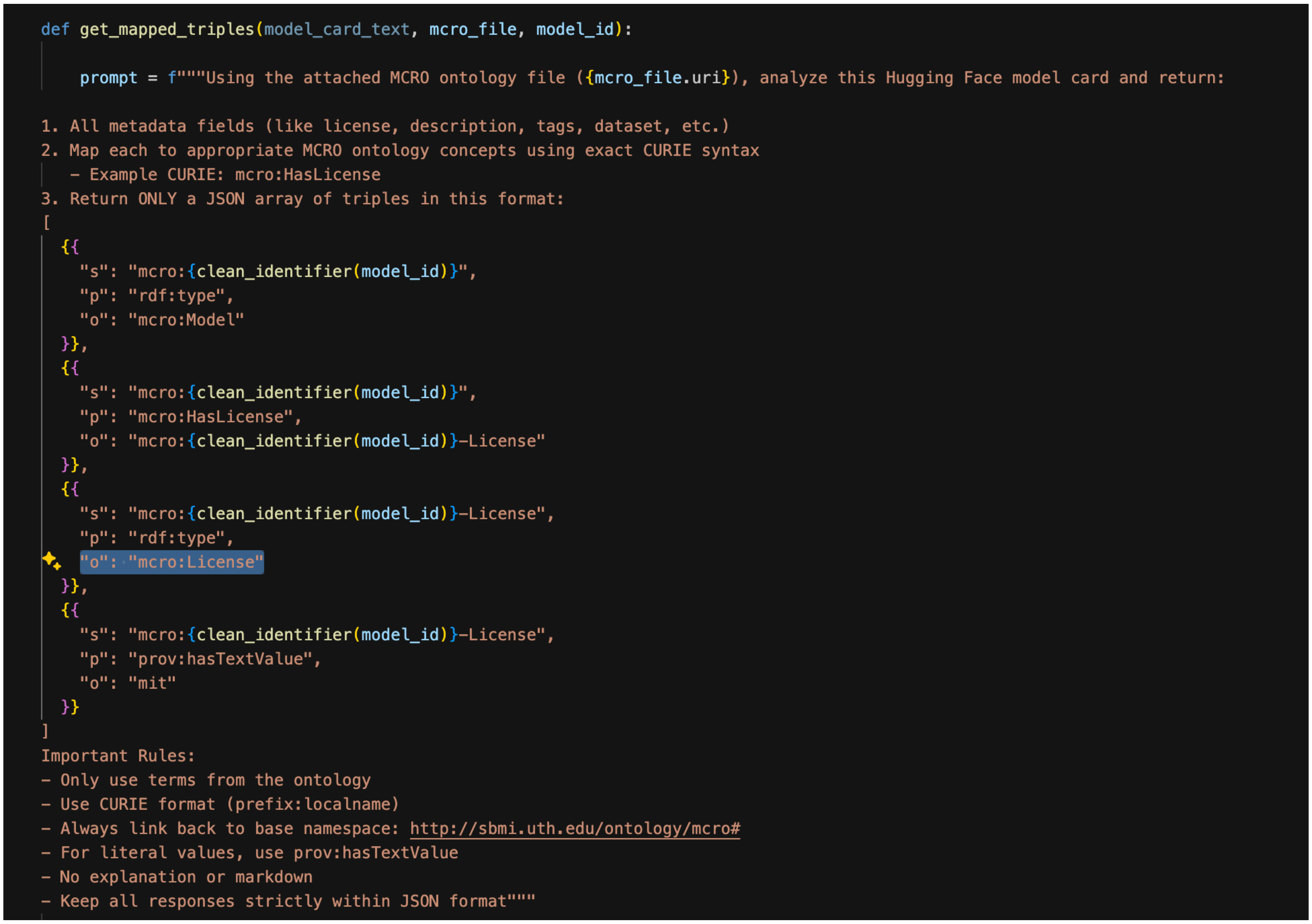
- License → http://purl.obolibrary.org/obo/MCRO_0000014
- Description → http://purl.obolibrary.org/obo/MCRO_0000003
- Tags → http://purl.obolibrary.org/obo/MCRO_0000029
- Dataset → http://purl.obolibrary.org/obo/MCRO_0000008
- Task → http://purl.obolibrary.org/obo/MCRO_0000028
- Language → http://purl.obolibrary.org/obo/MCRO_0000017
- Intended Use → http://purl.obolibrary.org/obo/MCRO_0000016
- Performance Metrics → http://purl.obolibrary.org/obo/MCRO_0000022
- Ethical Considerations → http://purl.obolibrary.org/obo/MCRO_0000009
  Return only a JSON array of triples using these exact CURIEs. DO NOT use prov:hasTextValue for these fields.
  """ `==after changing the prompt i'm getting 0 triples so maybe not the good option but this type it is Loading 52 valid MCRO properties which is a good sign == ohh i remember earlier i reject some of the curies maybe thats an issue to fix this i change the` is_valid_predicate() `to handle obo style properties. ![[Screenshot 2025-04-30 at 17.02.37.png]] so based on above indika's text i checked the smaple and they have the different namespace as .ttl file ie. mco not mcro and individual in this like mco:mc1_model_details instance of mco:Model_Detail have data properties like mco:name_text or mc:documentation which contain actual value. so triple should link via object properties but in current code its looking for object properties to map model metadata but in mco one structure is different. So as of now i change mcro namespace to mco. MCO ontology imports other othologies like IAO,SWO and prov, so i donwloaded it and imported it in code. Ok so i found .ttl file contains invalid turtle syntax uses` @prefix : `instead of` @prefix :mco `or` @prefix :mcro`.

OK [All above is kinda useles but use in report]
So i go through the indika messages again and went in the earlier code , change my prompt a little and seems like now its working. The things i changed i avoided `mcro:License` as i saw it does not exist in the ontology so the correct one is `mcro:LicenseInformationSection` now it matches the ontology defination

```
obo:MCRO_0000016 rdf:type owl:Class ; rdfs:label "License Information Section" .
```

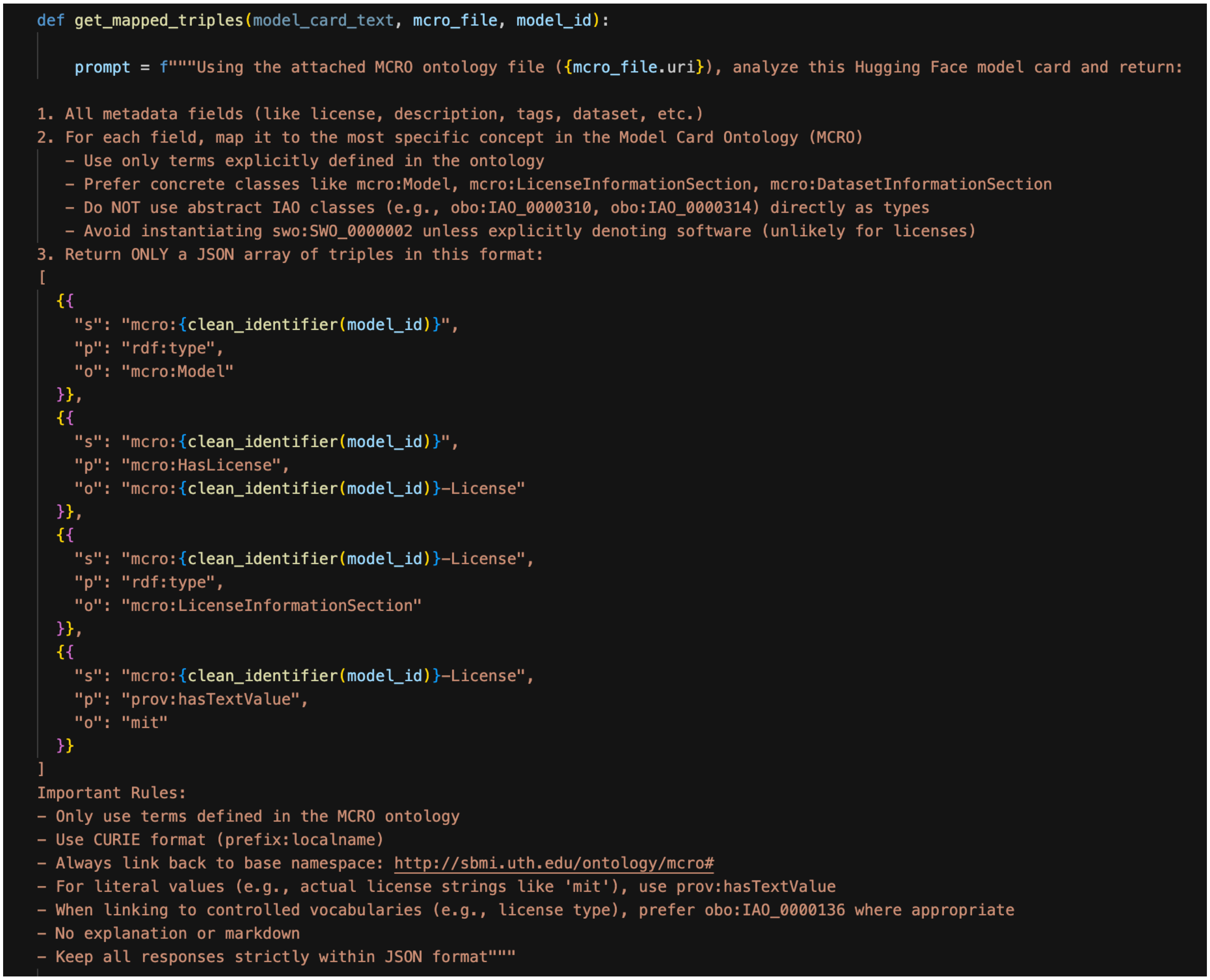
this will avoid instantiation abstract IAO classes like `swo:SWO_0000002` (software) and if i want to link to license it will change `swo:SWO_0000045` = `"MIT License"` from SWO.

Prompt used earlier:



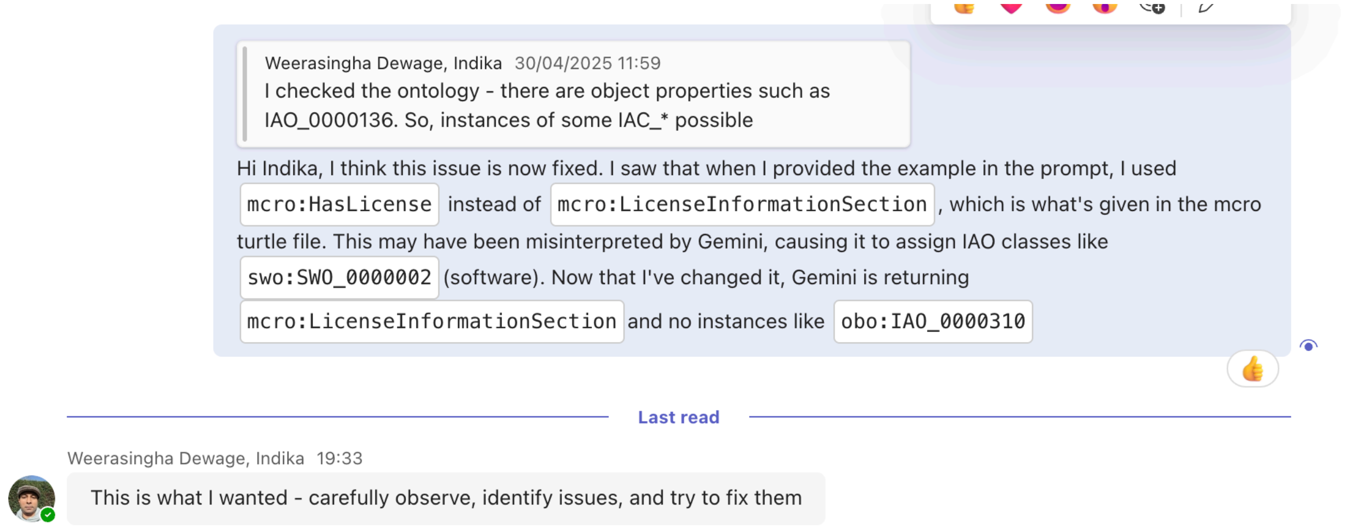
```
def get_mapped_triples(model_card_text, mcro_file, model_id):

    prompt = f"""Using the attached MCRO ontology file ({mcro_file.uri}), analyze this Hugging Face model card and return:

1. All metadata fields (like license, description, tags, dataset, etc.)
2. Map each to appropriate MCRO ontology concepts using exact CURIE syntax
   - Example CURIE: mcro:HasLicense
3. Return ONLY a JSON array of triples in this format:
[
  {{
    "s": "mcro:{clean_identifier(model_id)}",
    "p": "rdf:type",
    "o": "mcro:Model"
  }},
  {{
    "s": "mcro:{clean_identifier(model_id)}",
    "p": "mcro:HasLicense",
    "o": "mcro:{clean_identifier(model_id)}-License"
  }},
  {{
    "s": "mcro:{clean_identifier(model_id)}-License",
    "p": "rdf:type",
    "o": "mcro:License"
  }},
  {{
    "s": "mcro:{clean_identifier(model_id)}-License",
    "p": "prov:hasTextValue",
    "o": "mit"
  }}
]
Important Rules:
- Only use terms from the ontology
- Use CURIE format (prefix:localname)
- Always link back to base namespace: http://sbmi.uth.edu/ontology/mcro#
- For literal values, use prov:hasTextValue
- No explanation or markdown
- Keep all responses strictly within JSON format"""
```

prompt used after and getting better results just by chaging license check detail above



```
def get_mapped_triples(model_card_text, mcro_file, model_id):

    prompt = f"""Using the attached MCRO ontology file ({mcro_file.uri}), analyze this Hugging Face model card and return:

1. All metadata fields (like license, description, tags, dataset, etc.)
2. For each field, map it to the most specific concept in the Model Card Ontology (MCRO)
   - Use only terms explicitly defined in the ontology
   - Prefer concrete classes like mcro:Model, mcro:LicenseInformationSection, mcro:DatasetInformationSection
   - Do NOT use abstract IAO classes (e.g., obo:IAO_0000310, obo:IAO_0000314) directly as types
   - Avoid instantiating swo:SWO_0000002 unless explicitly denoting software (unlikely for licenses)
3. Return ONLY a JSON array of triples in this format:
[
  {{
    "s": "mcro:{clean_identifier(model_id)}",
    "p": "rdf:type",
    "o": "mcro:Model"
  }},
  {{
    "s": "mcro:{clean_identifier(model_id)}",
    "p": "mcro:HasLicense",
    "o": "mcro:{clean_identifier(model_id)}-License"
  }},
  {{
    "s": "mcro:{clean_identifier(model_id)}-License",
    "p": "rdf:type",
    "o": "mcro:LicenseInformationSection"
  }},
  {{
    "s": "mcro:{clean_identifier(model_id)}-License",
    "p": "prov:hasTextValue",
    "o": "mit"
  }}
]
Important Rules:
- Only use terms defined in the MCRO ontology
- Use CURIE format (prefix:localname)
- Always link back to base namespace: http://sbmi.uth.edu/ontology/mcro#
- For literal values (e.g., actual license strings like 'mit'), use prov:hasTextValue
- When linking to controlled vocabularies (e.g., license type), prefer obo:IAO_0000136 where appropriate
- No explanation or markdown
- Keep all responses strictly within JSON format"""
```

Mail to indika :



> Weerasingha Dewage, Indika 30/04/2025 11:59
> I checked the ontology - there are object properties such as IAO_0000136. So, instances of some IAC_* possible

Hi Indika, I think this issue is now fixed. I saw that when I provided the example in the prompt, I used `mcro:HasLicense` instead of `mcro:LicenseInformationSection`, which is what's given in the mcro turtle file. This may have been misinterpreted by Gemini, causing it to assign IAO classes like `swo:SWO_0000002` (software). Now that I've changed it, Gemini is returning `mcro:LicenseInformationSection` and no instances like `obo:IAO_0000310`

Weerasingha Dewage, Indika 19:33

This is what I wanted - carefully observe, identify issues, and try to fix them

So now asi know what i have to fix now i can also assign values in prompt for `"obo:MCRO_0000006"` , `"obo:MCRO_0000016"` ,`"obo:MCRO_0000001" etc.

With this new prompt i checked it from hugging face there are 100% accuracy in this .

Now i have added .ttl file extraction in `triple_creation.ipynb` file so now we will also get json and .ttl ,therefore we don't need `model_card_knowledge.ipynb`.

Now i am working on NL -> SPARQL
Created repo, and imported my KG turtle file in import with a base uri `http://purl.obolibrary.org/obo/MCRO_` ok Created a script which can find generate the sparql query based on the data in graphdb but we need it for big data like what if the data is terabytes in size we don't have that much space in graphdb so it should create sparql for that too as they have same schema. For this i need to figure with graphdb configuration and llm prompting.
So Indika gave some paper i am using the paper `Generating SPARQL Queries over CIDOC-CRM Using a Two-Stage Ontology Path Patterns Method in LLM Prompts` by `Michalis Mountantonakis` Changes i did in my code is i added 2 stage process like 1st prompt predicts relevant classes/properties using only triple patterns and second prompt generates sparql using filtered pattern based on predictions. Why this code will build on two stage architecture stage1: class/property prediction using triple patterns(A<=1) Stage2: Sparql generation with filtered path patterns

So i started with GRAPH RAG i generated nodes and relationships using python from the turtle file and now i am in neo4j i created constrains `CREATE CONSTRAINT FOR (n:Entity) REQUIRE n.id IS UNIQUE;` copy file from folder to neo4j import folder `v@Vishals-MacBook-Air dbms-eeca9e10-0a6b-4501-881c-09f44555b1db % cp /Users/v/Documents/Thesis/automation-model-cards/nodes.csv "/Users/v/Library/Application Support/Neo4j Desktop/Application/relate-data/dbmss/dbms-eeca9e10-0a6b-4501-881c-09f44555b1db/import/nodes.csv" then gave readable permissions` v@Vishals-MacBook-Air dbms-eeca9e10-0a6b-4501-881c-09f44555b1db % chmod 644 "/Users/v/Library/Application Support/Neo4j Desktop/Application/relate-data/dbmss/dbms-eeca9e10-0a6b-4501-881c-09f44555b1db/import/nodes.csv"` and loaded the nodes ! [[Screenshot 2025-05-04 at 01.28.24.png]] Now i have to generate embeddings for nodes description using gemini (RAG/g_embeddings.py) , had some warning related to ssl i suppressed it`(.venv) v@Vishals-MacBook-Air automation-model-cards % /Users/v/Documents/Thesis/automation-model-cards/.venv
/bin/python /Users/v/Documents/Thesis/automation-model-cards/RAG/g*embeddings.py*
*/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-packages/urllib3/***init***.py:35: NotOpenSSLWarning: urllib3 v2 only supports OpenSSL 1.1.1+, currently the 'ssl' module is compiled with 'LibreSSL 2.8.3'. See:*
[*https://github.com/urllib3/urllib3/issues/3020*](https://github.com/urllib3/urllib3/issues/3020)
warnings.warn(
*Exception ignored in: <function Driver.***del*** at 0x1209c00d0>*
Traceback (most recent call last):
File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-

packages/neo4j/_sync/driver.py", line 554, in ***del***
File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-packages/neo4j/_sync/driver.py", line 650, in close
TypeError: catching classes that do not inherit from BaseException is not allowed `after this work i create a vector index in Neo4j now the embedddings and vector index are setup im gonna build graph rag pipeline using langchain (RAG/hybrid+gem_retrieval.py) run the code got apoc error` == *(.venv) v@Vishals-MacBook-Air automation-model-cards % /Users/v/Documents/Thesis/automation-model-cards/.venv/bin/python /User==*
s/v/Documents/Thesis/automation-model-cards/RAG/hybrid+gem_retrieval.py
*/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-packages/urllib3/**init**.py:35: NotOpenSSLWarning: urllib3 v2 only supports OpenSSL 1.1.1+, currently the 'ssl' module is compiled with 'LibreSSL 2.8.3'. See: https://github.com/urllib3/urllib3/issues/3020*
warnings.warn(
/Users/v/Documents/Thesis/automation-model-cards/RAG/hybrid+gem_retrieval.py:5: LangChainDeprecationWarning: The class `Neo4jGraph` *was deprecated in LangChain 0.3.8 and will be removed in 1.0. An updated version of the class exists in the :class:*`~langchain-neo4j package and should be used instead. To use it run` *pip install -U :class:*`~langchain-neo4j` *and import as* `from :class:` *~langchain_neo4j import Neo4jGraph``.*
graph = Neo4jGraph(url="neo4j://localhost:7687", username="neo4j", password="12345678")
Traceback (most recent call last):
File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-packages/langchain_community/graphs/neo4j_graph.py", line 429, in ***init***
self.refresh_schema()
File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-packages/langchain_community/graphs/neo4j_graph.py", line 511, in refresh_schema
for el in self.query(
File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-packages/langchain_community/graphs/neo4j_graph.py", line 467, in query
==*data, , _* = self._driver.execute_query(==
File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-packages/neo4j/_sync/driver.py", line 970, in execute_query
return session._run_transaction(
File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-packages/neo4j/_sync/work/session.py", line 583, in _run_transaction
result = transaction_function(tx, *args, **kwargs)
File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-packages/neo4j/_work/query.py", line 144, in wrapped

```
return f(*args, **kwargs)
File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-
packages/neo4j/_sync/driver.py", line 1306, in _work
res = tx.run(query, parameters)
File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-
packages/neo4j/_sync/work/transaction.py", line 206, in run
result._tx_ready_run(query, parameters)
File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-
packages/neo4j/_sync/work/result.py", line 177, in _tx_ready_run
self._run(query, parameters, None, None, None, None, None, None)
File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-
packages/neo4j/_sync/work/result.py", line 236, in _run
self._attach()
File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-
packages/neo4j/_sync/work/result.py", line 430, in _attach
self._connection.fetch_message()
File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-
packages/neo4j/_sync/io/_common.py", line 184, in inner
func(*args, **kwargs)
File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-
packages/neo4j/_sync/io/_bolt.py", line 864, in fetch_message
res = self._process_message(tag, fields)
File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-
packages/neo4j/_sync/io/_bolt5.py", line 500, in _process_message
response.on_failure(summary_metadata or {})
File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-
packages/neo4j/_sync/io/_common.py", line 254, in on_failure
raise self._hydrate_error(metadata)
neo4j.exceptions.ClientError: {code: Neo.ClientError.Procedure.ProcedureNotFound}
{message: There is no procedure with the name apoc.meta.data registered for this database
instance. Please ensure you've spelled the procedure name correctly and that the procedure is
properly deployed.}

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
File "/Users/v/Documents/Thesis/automation-model-cards/RAG/hybrid+gem_retrieval.py", line
5, in module>
graph = Neo4jGraph(url="neo4j://localhost:7687", username="neo4j", password="12345678")
File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-
packages/langchain_core/_api/deprecation.py", line 224, in warn_if_direct_instance
```

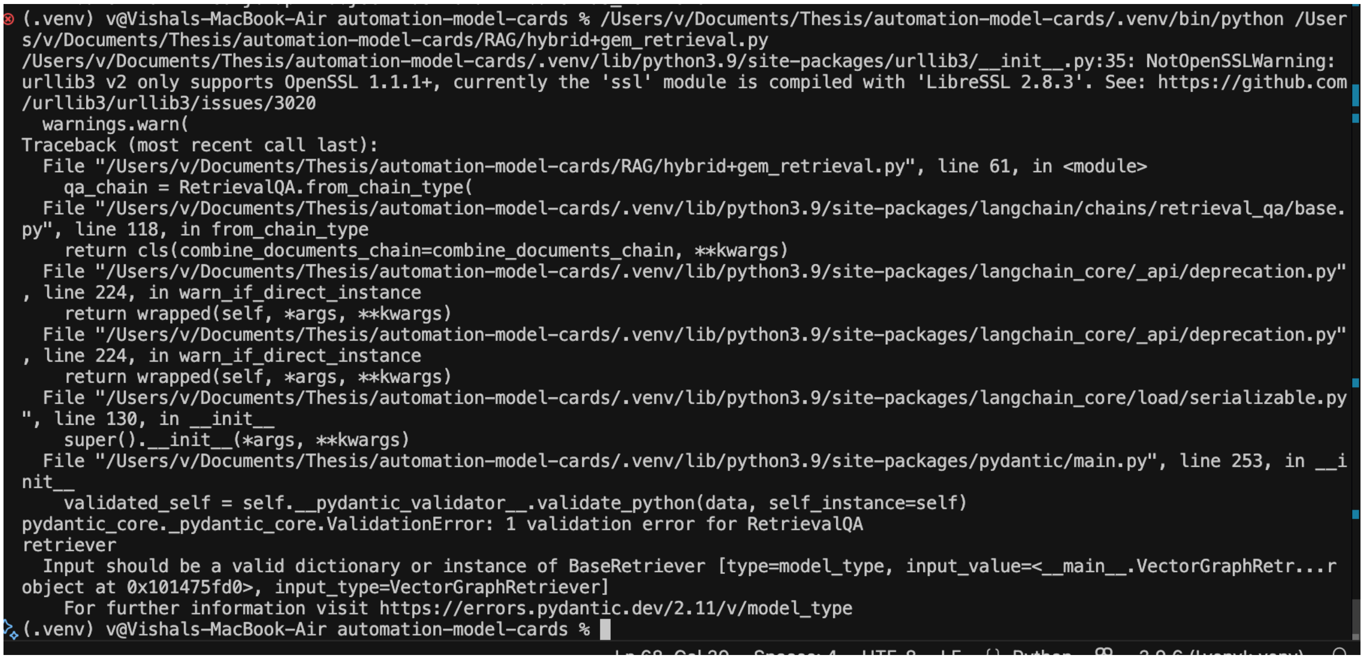
return wrapped(self, *args, **kwargs)

File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-packages/langchain_community/graphs/neo4j_graph.py", line 432, in **init**

raise ValueError(

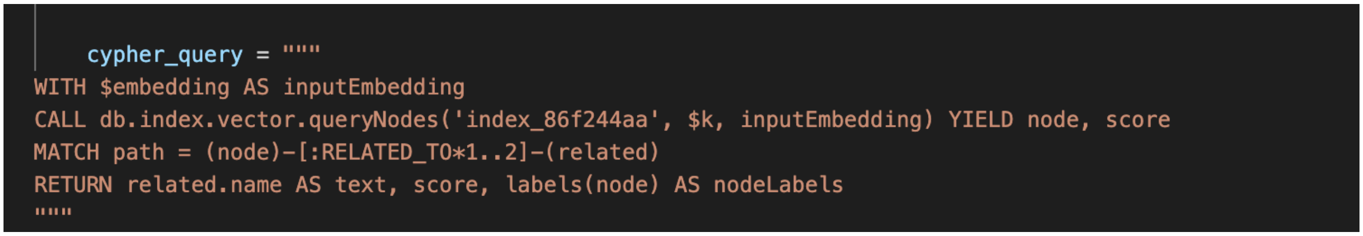
ValueError: Could not use APOC procedures. Please ensure the APOC plugin is installed in Neo4j and that 'apoc.meta.data()' is allowed in Neo4j configuration

`Fixed it by installing APOC plugin in neo4j, now it generate embeddingfor the query, uses gemini to convert the query into 768 dimensional vector, uses neo4j;s vector index to find the top k similar entities , traverses related_to relationships to expand context, converts neo4j results into lang chain compatible documents, uses the custom retriever to get context , passes it to gemini for answer generation Got another error ==This error is by using custom retriever` VectorGraphRetriever `which doesnt inherit from langchain's` BaseRetriever` causing a type validation faliure in langchanin 0.3.8+ and 1.0+==



```
⊗ (.venv) v@Vishals-MacBook-Air automation-model-cards % /Users/v/Documents/Thesis/automation-model-cards/.venv/bin/python /User
s/v/Documents/Thesis/automation-model-cards/RAG/hybrid+gem_retrieval.py
/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-packages/urllib3/__init__.py:35: NotOpenSSLWarning:
urllib3 v2 only supports OpenSSL 1.1.1+, currently the 'ssl' module is compiled with 'LibreSSL 2.8.3'. See: https://github.com
/urllib3/urllib3/issues/3020
  warnings.warn(
Traceback (most recent call last):
  File "/Users/v/Documents/Thesis/automation-model-cards/RAG/hybrid+gem_retrieval.py", line 61, in <module>
    qa_chain = RetrievalQA.from_chain_type(
  File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-packages/langchain/chains/retrieval_qa/base.
py", line 118, in from_chain_type
    return cls(combine_documents_chain=combine_documents_chain, **kwargs)
  File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-packages/langchain_core/_api/deprecation.py"
, line 224, in warn_if_direct_instance
    return wrapped(self, *args, **kwargs)
  File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-packages/langchain_core/_api/deprecation.py"
, line 224, in warn_if_direct_instance
    return wrapped(self, *args, **kwargs)
  File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-packages/langchain_core/load/serializable.py
", line 130, in __init__
    super().__init__(*args, **kwargs)
  File "/Users/v/Documents/Thesis/automation-model-cards/.venv/lib/python3.9/site-packages/pydantic/main.py", line 253, in __i
nit__
    validated_self = self.__pydantic_validator__.validate_python(data, self_instance=self)
pydantic_core._pydantic_core.ValidationError: 1 validation error for RetrievalQA
retriever
  Input should be a valid dictionary or instance of BaseRetriever [type=model_type, input_value=<__main__.VectorGraphRetr...r
object at 0x101475fd0>, input_type=VectorGraphRetriever]
    For further information visit https://errors.pydantic.dev/2.11/v/model_type
(.venv) v@Vishals-MacBook-Air automation-model-cards % 
```

fixed all errors now this is my cypher query



```
    cypher_query = """
WITH $embedding AS inputEmbedding
CALL db.index.vector.queryNodes('index_86f244aa', $k, inputEmbedding) YIELD node, score
MATCH path = (node)-[:RELATED_TO*1..2]-(related)
RETURN related.name AS text, score, labels(node) AS nodeLabels
"""
```

got an error one relation ship type doesnt exist then i remember i didnt put a relationships.csv file generated from RAG.ipynb

I decided to generate embeddings in json file as putting them in neo4j instead of writing 768 embedding manually.

```
╒════════════════════════╤══════════════════╕
│name                    │score             │
╞════════════════════════╪══════════════════╡
│"MCROclip"              │0.9997787475585938│
├────────────────────────┼──────────────────┤
│"MCROclipmodel"         │0.9756255149841309│
├────────────────────────┼──────────────────┤
│"MCROclipmodelcitation1"│0.9630792140960693│
├────────────────────────┼──────────────────┤
│"MCROclipmodelusecase"  │0.962954044342041 │
├────────────────────────┼──────────────────┤
│"MCROclipmodelcitation2"│0.9616742134094238│
└────────────────────────┴──────────────────┘
```

The question is why i am using embedding in general because my topic is more related to semantic understanding so basically it will convert raw data to vector to capture meaning and relationships. but what about vector databases , so for this im kind of using it but not fully im using neo4j it stores node and relationships and embedding as node properties. hybrid+gem_retrieval.py - i changed `googlegenerativeai` from `langchain_google_genai`, added proper lanchain llm wrapper configuration, fixed the chain initialisation parameters . I am still getting neo4j index error. Ok so i made it work with fixed cypher query basically means without prompt. Added prompt Logs so that i can log the prompts and their results and issue.

`prompt1 = f"""Generate a Cypher query for Neo4j that: 1. Starts with vector search using index 'index_86f244aa' and $embedding. 2. Traverses relationships with OPTIONAL MATCH (1-2 hops). 3. Returns: 3.1 Combined text from node/related nodes. 3.2 Score 3.3 Node labels 3.4 Relationship types
`Schema: {schema}
`Question: {query}
`Use $embedding and $k as parameters. Return ONLY the Cypher query."""

`
`Enter your question (type 'exit' to quit): list all the models.
`Retrieval failed: 'str' object is not callable Answer: I need the context to answer the question "list all the models". Please provide the context. Sources:`

```
prompt2 = f"""You're a Neo4j expert. Generate a query that:
1. Uses vector search with index 'index_86f244aa' and $embedding
2. Finds related nodes (1-2 hops)
3. Returns:
- Combined text as 'text'
- Score
- Node labels as 'labels'
- Relationships as 'relationships'
```

```
        ==Current schema:==
        =={graph.get_schema()}==

        ==User question: {query}==

        ==Use exactly these parameters: $embedding and $k==
        ==Return ONLY the Cypher code with no comments."""==
```

```
Enter your question (type 'exit' to quit): list all the models
Retrieval failed: 'str' object is not callable
Answer: I need the question to answer it. You haven't provided a question, only the context and the instruction to list all the models. I don't know what models you are referring to.

Sources:
```

Now i seprated embedding and query generation clients , and made a function to clean the query.

```
class VectorGraphRetriever(BaseRetriever):
def _get_relevant_documents(self, query: str) -> List[Document]:
try:
# Initialize Gemini clients
embed_client = genai.GenerativeModel('models/embedding-001')
query_client = genai.GenerativeModel('gemini-pro')
```

```
        ==# Get query embedding==
        ==query_embedding = embed_client.embed_content(==
            ==content=query,==
            ==task_type="retrieval_query"==
        ==)['embedding']==

        ==# Generate Cypher with proper client initialization==
        ==prompt = f"""Generate a Neo4j Cypher query that:==
        ==1. Uses vector search with index 'index_86f244aa' and $embedding==
        ==2. Returns: text, score, labels, relationships==
        ==3. Follows this schema:==
        =={graph.get_schema()}==

        ==Query: {query}==
        ==Return ONLY valid Cypher with $embedding and $k parameters."""==

        ==# Generate and clean query==
        ==cypher_response = query_client.generate_content(prompt)==
        ==cypher_query = self._clean_cypher(cypher_response.text)==

        ==# Execute query==
        ==results = graph.query(==
            ==cypher_query,==
            ==params={"embedding": query_embedding, "k": 5}==
        ==)==

        ==return [Document(==
            ==page_content=r["text"],==
            ==metadata={==
                =="score": r["score"],==
                =="labels": r["labels"],==
                =="relationships": r["relationships"]==
            ==}==
        ==) for r in results if r["text"]]==

    ==except Exception as e:==
        ==print(f"Retrieval error: {e}")==
        ==return []==

==def _clean_cypher(self, text: str) -> str:==
    =="""Remove markdown formatting from generated query"""==
    ==return text.replace('```cypher', '').replace('```', '').strip()==
```

still getting same error

prompt = f"""Generate a Neo4j Cypher query that:

1. Starts with vector search: CALL db.index.vector.queryNodes('index_86f244aa', $k, $embedding)
2. Focuses on nodes with properties: [name, id]
3. Handles relationships like: `MCROhas*`, `prov*`, `22rdfsyntaxnstype`
4. Returns:
   - Combined text from name properties
   - Score
   - Node labels
   - Relationship types

Schema Overview:

- Nodes have ID and NAME (sometimes long text)
- Relationships use prefixes: MCRO*, prov*, 22*

Example Valid Query:

```
CALL db.index.vector.queryNodes('index_86f244aa', $k, $embedding)
YIELD node, score
OPTIONAL MATCH (node)-[r:MCROhasIntendedUseCase|provhasTextValue*1..2]-(related)
RETURN
coalesce(node.name, node.id) AS text,
score,
labels(node) AS labels,
type(r) AS relationships
```

Current Query Needs: {query}
Generate ONLY the Cypher code between `cypher markers:"""`

```
Ask about models (type 'exit' to quit): list all the models
Retrieval Error: 'str' object is not callable
Answer: I need the context to answer the question "list all the models". Please provide the context.
```

In RAG.py file i changed the code a bit bcoz the issue was there was low textual metadata and it failed to capture literal values(hasTextValue) as node properties treating them as relationships instead, ID were also meeaning less and had missing information that didnt preserve class/type labels (MCRO_Model,PerformanceMetric etc) losing crucial categorization data. and ignore anotation properties like licenses, citations and performance metrics. I saw some of the id's (n4db42cfb3dcb42c4a1433be245b60202b1) in nodes.csv appears blank which can be malformed or incomplete triples in turtle file.

Ok So i texted indika that i am using medium paper but he said use an actual research papers to justify the RAG architecture. so now im using this paper

KAG: Boosting LLMs in Professional Domains via Knowledge Augmented Generation

Lei Liang[*,1], Mengshu Sun[*,1], Zhengke Gui[*,1], Zhongshu Zhu[1], Ling Zhong[1], Peilong Zhao[1], Zhouyu Jiang[1], Yuan Qu[1], Zhongpu Bo[1], Jin Yang[1], Huaidong Xiong[1], Lin Yuan[1], Jun Xu[1], Zaoyang Wang[1], Zhiqiang Zhang[1], Wen Zhang[2], Huajun Chen[2], Wenguang Chen[1], Jun Zhou[†,1]

{leywar.liang, mengshu.sms, zhengke.gzk, jun.zhoujun}@antgroup.com

[1]**Ant Group Knowledge Graph Team,** [2]**Zhejiang University**

Github:**https://github.com/OpenSPG/KAG**

Abstract

The recently developed retrieval-augmented generation (RAG) technology has enabled the efficient construction of domain-specific applications. However, it also has limitations, including the gap between vector similarity and the relevance of knowledge reasoning, as well as insensitivity to knowledge logic, such as numerical values, temporal relations, expert rules, and others, which hinder the effectiveness of professional knowledge services. In this work, we introduce a professional domain knowledge service framework called Knowledge Augmented Generation (**KAG**). KAG is designed to address the aforementioned challenges with the motivation of making full use of the advantages of knowledge graph(KG) and vector retrieval, and to improve generation and reasoning performance by bidirectionally enhancing large language models (LLMs) and KGs through five key aspects: (1) LLM-friendly knowledge representation, (2) mutual-indexing between knowledge graphs and original chunks, (3) logical-form-guided hybrid reasoning engine, (4) knowledge alignment with semantic reasoning, and (5) model capability enhancement for KAG. We compared KAG with existing RAG methods in multihop question answering and found that it significantly outperforms state-of-the-art methods, achieving a relative improvement of 19.6% on hotpotQA and 33.5% on 2wiki in terms of F1 score. We have successfully applied KAG to two professional knowledge Q&A tasks of Ant Group, including E-Government Q&A and E-Health Q&A, achieving significant improvement in professionalism compared to RAG methods. Furthermore, we will soon natively support KAG on the open-source KG engine OpenSPG, allowing developers to more easily build rigorous knowledge decision-making or convenient information retrieval services. This will facilitate the localized development of KAG, enabling developers to build domain knowledge services with higher accuracy and efficiency.

1 Introduction

Recently, the rapidly advancing Retrieval-Augmented Generation (RAG)[1, 2, 3, 4, 5] technology has been instrumental in equipping Large Language Models (LLMs) with the capability to acquire

[1] *: These authors contributed equally to this work.

this paper allow using json triples so i think which is better and easy for me to use that.

OK now im back with triple_creation.py, i found the current prompt
`prompt = f"""Using the attached Model Card Ontology (MCRO) file ({mcro_file.uri}), analyze this Hugging Face model card text and return only RDF triples in JSON format. Follow these

strict rules:

`Rules for Mapping`

1. `Only use terms defined in the MCRO ontology.`
2. `Always map metadata fields to appropriate MCRO concepts **by their CURIEs**, such as:`
   - `license → mcro:LicenseInformationSection`
   - `dataset → mcro:DatasetInformationSection`
   - `model architecture → mcro:ModelArchitectureInformationSection`
   - `citation → mcro:CitationInformationSection`
   - `intended use case → mcro:UseCaseInformationSection`
3. `Use proper relationships:`
   - `rdf:type for types`
   - `prov:hasTextValue for textual values (like "mit", "CNN", "ImageNet")`
   - `Appropriate mcro:hasX properties for linking model to its sections`
4. `Never assign rdf:type to abstract IAO classes like obo:IAO_*.`
5. `Never directly type instances with obo:MCRO_0000004, obo:MCRO_0000016, etc. — always use CURIEs like mcro:CitationInformationSection, mcro:LicenseInformationSection.`

   `Sample Output Format:`

```
[
{{
"s": "mcro:{clean_identifier(model_id)}",
"p": "rdf:type",
"o": "mcro:Model"
}},
{{
"s": "mcro:{clean_identifier(model_id)}",
"p": "mcro:hasLicense",
"o": "mcro:{clean_identifier(model_id)}-License"
}},
{{
"s": "mcro:{clean_identifier(model_id)}-License",
"p": "rdf:type",
"o": "mcro:LicenseInformationSection"
}},
{{
"s": "mcro:{clean_identifier(model_id)}-License",
"p": "prov:hasTextValue",
"o": "mit"
}}
]
Important: Return ONLY the JSON array. No explanation. No markdown.
Input Text:
{model_card_text}
"""
```

having some issue while working with this .ttl file in GraphDB, I ran a query to retrieve a list of all models containing "vit". The result included 10 entries:

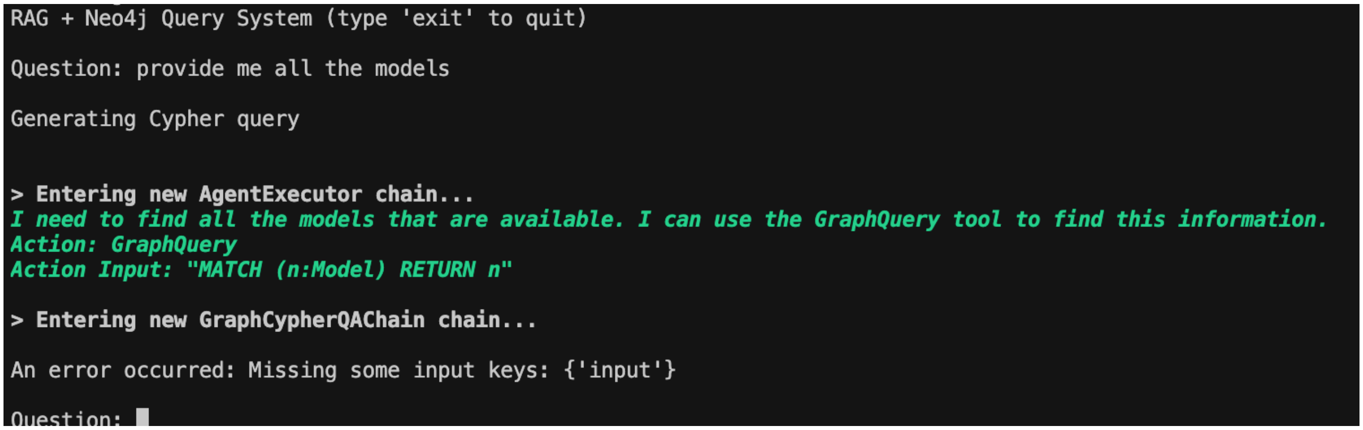
MCRO_CLIPViTB16LAION2B
MCRO_Salesforceblipbootstrapping
MCRO_Salesforceblipimagecaptioninglarge
MCRO_clip-ModelDetail
MCRO_clip-ModelDetailSection
MCRO_fashionclip-ModelDetail
MCRO_siglipso400mpatch14384-ModelDetail
MCRO_visiontransformerbase
MCRO_vit-face-expression
MCRO_vitmatte-ModelDetail
I noticed that some of the entries include both ModelDetail and ModelDetailSection. These appear to represent duplicate or closely related models, even though they have slightly different suffixes.

To improve accuracy in future queries, I'm going to refine the search rules in the prompt to avoid such duplicates or ambiguous results.

found another issue

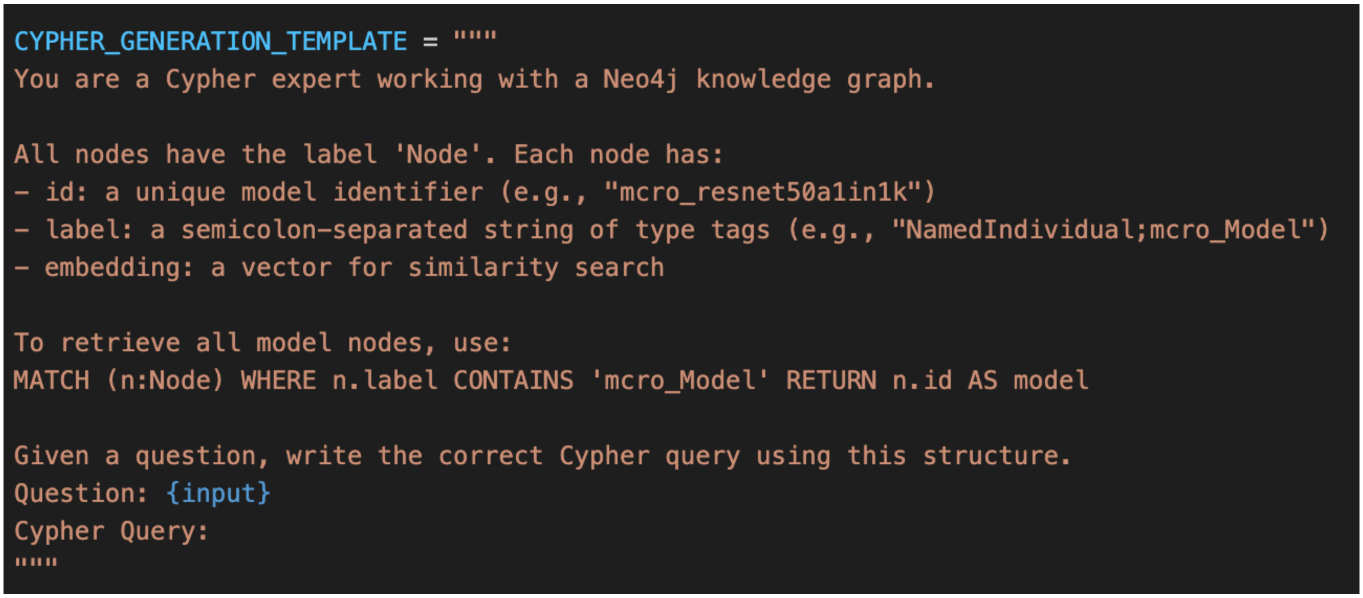
it keep giving me same data undtil i remove whole repo from graph db

in rag im gettting this error while exeuting query this means agent's llm still doesnt understands the schema. even though my cypher prompt in kg_query.py is correct, the agent is not using it directly and deciding the query text itself the issue here is prompt or less context to the agent telling it not to use :Model or intercept the query



```
RAG + Neo4j Query System (type 'exit' to quit)

Question: provide me all the models

Generating Cypher query


> Entering new AgentExecutor chain...
I need to find all the models that are available. I can use the GraphQuery tool to find this information.
Action: GraphQuery
Action Input: "MATCH (n:Model) RETURN n"

> Entering new GraphCypherQAChain chain...

An error occurred: Missing some input keys: {'input'}

Question: 
```

prompt used for above :



```
CYPHER_GENERATION_TEMPLATE = """
You are a Cypher expert working with a Neo4j knowledge graph.

All nodes have the label 'Node'. Each node has:
- id: a unique model identifier (e.g., "mcro_resnet50a1in1k")
- label: a semicolon-separated string of type tags (e.g., "NamedIndividual;mcro_Model")
- embedding: a vector for similarity search

To retrieve all model nodes, use:
MATCH (n:Node) WHERE n.label CONTAINS 'mcro_Model' RETURN n.id AS model

Given a question, write the correct Cypher query using this structure.
Question: {input}
Cypher Query:
"""
```

ok so idk why im getting this error but i changed :ID to :Model in csv file so that it detect model as its keep on writing model in cypher query but that gave me the same results but then i change {input} to {query} in prompt and KG_rag.py file `func=lambda q: chain.run({"query": q}),` this line and it worked :D

Do not assume one prompt will be able to extract all the metadata we need. We may need to use multiple prompts to extract and even correct different types of metadata. Also, different prompting strategies and data sources (data selection, schema, etc.). Also, LLMs can act as the domain expert (system or role prompt) of the dataset domain and can provide any missing information (which was one of the initial ideas in the thesis proposal). LLMs can do text augmentation, and then the augmented text can be used to generate triples, etc. Approach the problem as an investigator. Look for the solutions in the literature. Record all issues, solutions applied, and results. Also, try with a small dataset first. We need to try prompts, observe errors, document errors, and find ways to fix errors (repairing). You guys need to show the initiative `to look for solutions (mostly from literature), ways to do tasks systematically`

read KAG Graph + multimodal and Knowledge Graph(s) and