

# Semantic Cataloging of Models using Large Language Models and Model Cards

*Master Thesis*

Vishal Sehgal  
2109374

Supervisors:  
Dr. I. (Indika) P.K. Weerasingha Dewage  
Prof. Georgios Manias

s-Hertogenbosch, The Netherlands  
July 2025

# Abstract

This thesis addresses the challenge of inconsistent and incomplete documentation in model cards, which detail critical aspects of machine learning models such as performance, biases, limitations, and ethical considerations. Despite the increasing use of platforms like Hugging Face, only 44.2% of models have comprehensive model cards, leaving stakeholders without crucial information about the models they are working with. This lack of standardization hinders effective model management and informed decision-making.

In response, this thesis proposes an ontology-based approach to automate the generation and discovery of semantic model cards. By leveraging Large Language Models (LLMs), the developed system extracts relevant metadata from Hugging Face model repositories, generating model cards that explicitly address biases, ethical concerns, environmental impact, and other important factors. Additionally, the system converts the existing Hugging Face model catalog into a structured, queryable knowledge graph using an ontology-based model catalog.

This work contributes to improving the standardization and accessibility of model documentation by automating the process of generating detailed and semantically rich model cards, which are essential for effective model governance and informed stakeholder engagement.

# Preface

This thesis represents the culmination of my research and marks an important milestone in my academic journey. I would like to first express my heartfelt gratitude to Dr. Indika Kumara for his continuous support, enthusiasm, energy, and expert guidance. His mentorship has been instrumental in shaping this research. I would also like to deeply thank my family for their unwavering support, love, and encouragement throughout this process. Their belief in me provided the strength I needed to keep going, especially during challenging moments. I am also immensely grateful to the ontology experts who provided invaluable insights and feedback, helping to refine the conceptual and technical aspects of this work. Their expertise has greatly enriched this thesis. Finally, I am grateful to my friends for their understanding, motivation, and companionship. Their presence was a source of comfort and inspiration during this journey.

**Vishal Sehgal**

# Acronyms

## Core AI/ML & Semantic Technologies

AI - Artificial Intelligence  
BERT - Bidirectional Encoder Representations from Transformers  
BLEU - Bilingual Evaluation Understudy (evaluation metric)  
CURIE - Compact URI  
G-RAG - Graph Retrieval-Augmented Generation  
GPT - Generative Pre-trained Transformer  
IAO - Information Artifact Ontology  
KG - Knowledge Graph  
KGQA - Knowledge Graph Question Answering  
LLM - Large Language Model  
MCRO - Model Card Report Ontology  
ML - Machine Learning  
NL - Natural Language  
NLU - Natural Language Understanding  
NLP - Natural Language Processing  
OWL - Web Ontology Language  
PROV - Provenance Ontology  
RAG - Retrieval-Augmented Generation  
RDF - Resource Description Framework  
SKOS - Simple Knowledge Organization System  
SPARQL - SPARQL Protocol and RDF Query Language  
SPDX - Software Package Data Exchange (license standard)  
SWO - Software Ontology  
ViT - Vision Transformer

## Technical Methods & Metrics

ANN - Approximate Nearest Neighbor (search)  
APOC - Awesome Procedures on Cypher (Neo4j plugin)  
COR - Correct Output Rate  
FN - False Negative  
FP - False Positive  
F1-score - Harmonic mean of Precision and Recall  
MSA - Multiple Sequence Alignment (protein folding)  
Q/A - Question Answering  
RDFLib - RDF Library for Python  
SCA - Schema Compliance Accuracy  
SVR - Syntax Validity Rate  
TP - True Positive

## Tools & Platforms

GraphDB - Graph Database for RDF (Ontotext)  
HF - Hugging Face

---

HfApi - Hugging Face API  
LangChain - Framework for LLM applications  
Neo4j - Graph Database  
Protg - Ontology Editor

#### **Miscellaneous**

JSON - JavaScript Object Notation  
URI - Uniform Resource Identifier  
URL - Uniform Resource Locator  
YAML - YAML Ain't Markup Language

#### **Dataset References**

CC-News - Common Crawl News Dataset  
ImageNet - Image Database for Visual Recognition  
OpenWebText - Open-source Web Text Dataset

# Contents

<b>Contents</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Research Context . . . . .	1
1.2 Scientific Relevance . . . . .	2
1.3 Practical Relevance . . . . .	3
1.4 Research Questions . . . . .	3
1.5 Thesis Structure . . . . .	4
<b>2 Literature Review</b>	<b>6</b>
2.1 Model Cards . . . . .	6
2.2 Ontology . . . . .	7
2.2.1 Triplestores . . . . .	8
2.2.2 Model Card Report Ontology . . . . .	8
2.2.3 SPARQL . . . . .	9
2.3 Knowledge Graph . . . . .	10
2.3.1 KG Construction from Unstructured Sources . . . . .	10
2.4 Model Discovery and Semantic Augmentation . . . . .	10
2.4.1 Model Cards as Metadata Sources . . . . .	10
2.5 Large Language Models . . . . .	11
2.5.1 Transformer Architecture and Self-Attention . . . . .	11
2.5.2 Pre-Training and Knowledge Acquisition . . . . .	11
2.5.3 Adaptive Learning Capabilities . . . . .	11
2.5.4 Applications in Structured Knowledge Generation . . . . .	11
2.6 Text-to-SPARQL Semantic Parsing . . . . .	12
2.6.1 Prompt Engineering . . . . .	13
2.7 Retrieval-Augmented Generation (RAG) . . . . .	15
2.7.1 Graph Retrieval-Augmented Generation . . . . .	15
2.7.2 Embedding and Indexing in Graph RAG . . . . .	16
2.8 Related Work . . . . .	16
2.8.1 LLMs for Model Card Metadata Extraction . . . . .	16
2.8.2 RDF Triples and Knowledge Graph Construction . . . . .	17
2.8.3 Graph-Based Retrieval-Augmented Generation (Graph-RAG) . . . . .	17
2.8.4 SPARQL Querying and Knowledge Graphs . . . . .	17
2.9 Research Gaps and Goals . . . . .	17
<b>3 Methodology</b>	<b>19</b>
3.1 Research Questions . . . . .	19
3.1.1 Sub-questions . . . . .	19
3.2 System Architecture . . . . .	20
3.2.1 Pipeline Descriptions . . . . .	21
3.3 Dataset . . . . .	22
3.3.1 Data Source Collection . . . . .	22
3.3.2 Triple Extraction, Conversion and Storage . . . . .	23
3.4 Knowledge Graph Ingestion . . . . .	25
3.4.1 Ingestion into GraphDB . . . . .	25

3.4.2	Ingestion into Protege . . . . .	25
3.4.3	Ingestion into Neo4j . . . . .	25
3.5	Pipeline 1: ModelCards to Knowledge Graph . . . . .	26
3.5.1	Objective and Scope . . . . .	26
3.5.2	Preprocessing Model Cards . . . . .	27
3.5.3	Metadata Extraction and Prompting via LLM . . . . .	27
3.5.4	RDF Triple Generation . . . . .	29
3.5.5	Prompting Strategies . . . . .	31
3.5.6	Evaluation Metrics and Process . . . . .	35
3.6	Pipeline 2: Model Search NL Queries to SPARQL . . . . .	38
3.6.1	Objective and Scope . . . . .	38
3.6.2	LLM Integration . . . . .	39
3.6.3	Evaluation Metrics and Process . . . . .	41
3.7	Pipeline 3: Graph RAG Pipeline for Model Search . . . . .	44
3.7.1	Objective and Scope . . . . .	44
3.7.2	Preprocessing Knowledge Graph . . . . .	46
3.7.3	Evaluation Metrics and Process . . . . .	47
<b>4</b>	<b>Results</b>	<b>48</b>
4.1	RQ1: Impact of LLM-Based Ontology Extraction on Model Card Metadata Quality . . . . .	48
4.1.1	Quantitative Evaluation on 10-Model Dataset . . . . .	48
4.1.2	Expert Review and Qualitative Assessment . . . . .	49
4.1.3	Generalizability on 100-Model Dataset . . . . .	49
4.1.4	Summary of Findings . . . . .	50
4.2	RQ2: Effectiveness of LLMs in Natural Language Querying of Semantic Catalogues . . . . .	50
4.2.1	Quantitative Results . . . . .	50
4.2.2	Detailed Analysis of Generated Queries . . . . .	51
4.2.3	Qualitative Feedback from Expert Review . . . . .	62
4.2.4	Summary of Evaluation Findings . . . . .	63
4.3	RQ3: Effectiveness of G-RAG Compared to SPARQL for Model QA . . . . .	63
4.3.1	Quantitative Analysis . . . . .	63
4.3.2	Qualitative Analysis . . . . .	65
4.3.3	Summary of Findings . . . . .	68
<b>5</b>	<b>Discussions</b>	<b>70</b>
5.1	Ontology-Guided Metadata Structuring with LLMs . . . . .	70
5.2	LLMs for Natural Language to SPARQL Translation . . . . .	70
5.3	Retrieval Augmentation and SPARQL Approaches . . . . .	71
5.4	Cross-Cutting Insights . . . . .	71
5.5	Threats to Validity . . . . .	72
5.5.1	Internal Validity . . . . .	72
5.5.2	External Validity . . . . .	72
<b>6</b>	<b>Conclusions</b>	<b>74</b>
6.1	Answers to Research Questions . . . . .	74
6.2	Research Contributions . . . . .	75
6.3	Industrial Recommendations . . . . .	76
6.3.1	Implement Ontology-Driven Metadata Extraction for Model Cards . . . . .	76
6.3.2	Enhance Natural Language Querying Over Semantic Model Catalogues . . . . .	76
6.3.3	Invest in Performance . . . . .	76
6.4	Academic Recommendations . . . . .	76
<b>Bibliography</b>		<b>78</b>
<b>List of Figures</b>		<b>86</b>
<b>List of Tables</b>		<b>87</b>
<b>Listings</b>		<b>88</b>

<b>Appendix</b>	<b>89</b>
.1 Tripples . . . . .	89
.2 Result . . . . .	89
.2.1 Reasearch Question2 - Q2 . . . . .	89
.2.2 Reasearch Question2 - Q6 . . . . .	90
.3 Methodology . . . . .	91
.3.1 3.6.3 Evaluation Metrics and Process . . . . .	91

# Chapter 1

## Introduction

### 1.1 Research Context

As machine learning (ML) continues to penetrate critical domains such as healthcare, education, finance, and law, the need to ensure that models are well-documented, traceable, and auditable has never been more urgent. Documentation formats, such as model cards, introduced by Mitchell et al. (2019) [70], were designed to support this need by detailing a models purpose, performance characteristics, training data, limitations, and ethical considerations. However, in practice, model cards remain inconsistently used and lack both syntactic and semantic standardization. A 2023 analysis of nearly 75,000 AI models hosted on Hugging Face provides empirical evidence for this gap [58].

As illustrated in Figure 1.1, the number of models on Hugging Face has grown exponentially, with a weekly growth rate of 3.16% and a doubling time of approximately 22 weeks. By October 1st, 2022, the platform hosted 74,970 model repositories. Despite this surge, only 42.8% of these repositories (32,111 models) included any form of model card [58]. Nevertheless, these documented models accounted for over 90.5% of the total download traffic, underscoring the strong practical demand for well-documented models. This disparity indicates a pressing need to automate and standardize the creation and structuring of model cards to ensure broad adoption and usability. The figure also highlights trends in community focus across time and provides a reference example of a model cards key components. The exponential growth of AI model repositories compounds this challenge. Platforms like Hugging Face have become central to the ML ecosystem, hosting millions of models across diverse tasks, modalities, and domains. According to a 2024 report by Gartner [32], over 70% of enterprise AI projects now rely in some part on pre-trained or open-source foundation models. As a result, stakeholders ranging from developers to policymakers must regularly evaluate, compare, and verify models using inconsistent documentation that varies in format, detail, and availability.

Meanwhile, semantic technologies have gained traction as a powerful solution for organizing heterogeneous and unstructured information. Ontologies and knowledge graphs are increasingly used to create interoperable, machine-readable representations of domain knowledge [49]. In parallel, advances in Large Language Models (LLMs), such as GPT, Gemini, and LLaMA, have unlocked new potential for the automated extraction and structuring of knowledge from unstructured text [2]. These models are capable of interpreting domain-specific language, identifying entities and relationships, and aligning natural language inputs with formal schema making them ideal candidates for automating metadata extraction from model cards. Studies by Norouzi et al., 2024 [78] and Kommineni et al., 2024 [48] have already demonstrated the viability of using LLMs for ontology population, RDF triple extraction, and schema-guided information retrieval. The integration of LLMs with semantic web technologies represents a frontier of innovation in AI infrastructure. Prior works have explored the use of LLMs to populate knowledge graphs, generate SPARQL queries from natural language, and even guide reasoning over symbolic structures [78]. Yet, few studies have targeted the specific challenge of transforming model documentation into structured semantic resources [47]. This thesis addresses that gap by developing an end-to-end framework that extracts ontologically aligned metadata from Hugging Face model cards, converts it into RDF triples, and stores it in a queryable knowledge graph.

The ability to query model metadata using natural language introduces further opportunity for demo-

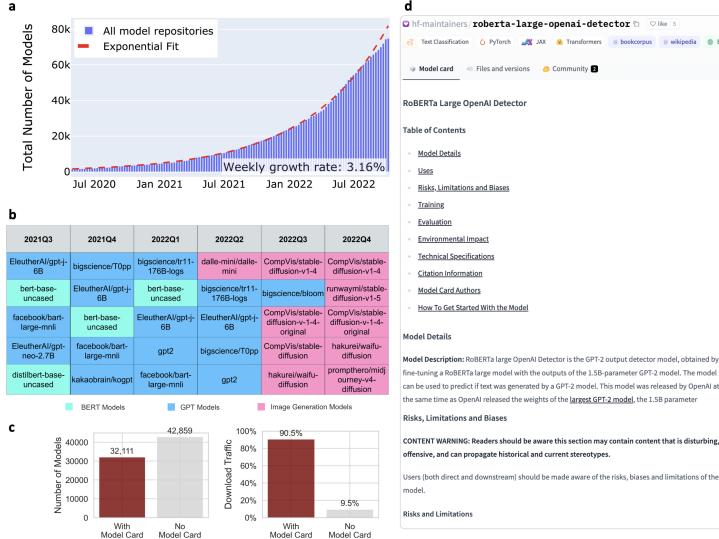


Figure 1.1: Figure 1: Overview of Hugging Face Model Repositories and Model Card Adoption.(a) Exponential Growth of Model Repositories: The number of models hosted on Hugging Face has grown rapidly, with a weekly growth rate of 3.16% and a doubling time of 22 weeks. As of October 1st, 2022, there are 74,970 models available. (b) Temporal Trends in Model Popularity: Analysis of the top-5 trending model repositories reveals shifts in focus from language models to image generation models over time. (c) Model Card Adoption and Download Traffic: While only 42.8% of models (32,111 out of 74,970) feature model cards, these models account for 90.5% of the total download traffic, emphasizing the critical role of documentation in model usage. (d) Model Card Example: An illustrative example of a typical model card, showing key metadata elements including model purpose, performance, and limitations [58]

cratization. While SPARQL is powerful, it is not user-friendly for non-technical stakeholders. Research demonstrates that LLMs can translate natural language queries into executable SPARQL, though challenges remain in terms of schema alignment, syntactic correctness, and interpretability [25]. Additionally, recent innovations in retrieval-augmented generation (RAG), particularly the Graph-RAG, have shown promise for enhancing LLM reasoning with contextual grounding from structured sources such as knowledge graphs [55][7].

In this context, the present thesis introduces a unified system that automates the creation of a semantic model catalogue by extracting and organizing model card data, and enables intuitive access via both SPARQL-based semantic parsing and G-RAG pipelines. This is not only a technical advancement, but a timely response to the broader AI governance landscape, where explainability, auditability, and provenance are becoming non-negotiable. The system is designed to serve both expert users who may prefer precision querying and non-experts, who benefit from natural language interfaces grounded in graph-structured knowledge.

Ultimately, this research is driven by a central question: How can LLMs and model documentation be leveraged to automatically generate, catalog, complete, compare, and discover semantic model cards? This question lies at the core of the thesis, which aims to develop a framework for automating metadata extraction, structuring model documentation, and enabling efficient querying through knowledge graphs. In answering this, the thesis contributes to the field of semantic AI infrastructure, supporting the creation of reliable, interoperable, and reusable AI models for both research and industry.

## 1.2 Scientific Relevance

Although model cards have emerged as an important tool for documenting AI models, the majority of prior research on this topic assumes an idealized scenario in which model metadata is manually curated and easily accessible in a structured format [80, 58]. In contrast, real-world applications often involve large, heterogeneous collections of models and associated metadata, which are scattered, inconsistent,

and not easily comparable. This poses a significant challenge in both research and industry, where the lack of structured and interoperable model documentation impedes reliable discovery, comparison, and oversight of AI systems.

Recent work on knowledge graphs and LLM-based querying has underscored the need for more scalable and robust methods of automating model documentation and metadata extraction [2]. However, the existing literature rarely addresses how to integrate such techniques in a way that supports the generation, cataloging, completion, and comparison of model cards at scale. As highlighted by recent surveys, advancing the use of retrieval-augmented generation (RAG) and ontology-driven approaches remains a critical priority for making model card systems more reliable and practical in diverse deployment settings [30, 76].

This thesis aims to address these gaps by leveraging LLMs to automatically generate, catalog, complete, compare, and discover semantic model cards from existing model documentation. By utilizing ontology-based knowledge graphs, the research proposes a framework for automatically extracting metadata, structuring model documentation, and enabling efficient querying of model cards. The approach not only improves the consistency and completeness of model documentation but also enhances the ability to compare AI models and track their evolution. By doing so, this thesis contributes to the growing field of semantic AI infrastructure, providing a scalable, interoperable, and machine-readable solution for managing AI model metadata in both research and industry settings.

### 1.3 Practical Relevance

The practical implications of using LLMs for automating model card generation and semantic cataloging are substantial, particularly as AI systems become more integrated into both research and industry. As organizations and researchers continue to deploy an increasing number of AI models, the need for structured and standardized model documentation becomes critical for ensuring model transparency, accountability, and effective use. However, current practices for model card creation are often fragmented, inconsistent, and time-consuming, making it difficult to track and compare key information regarding AI models. This is particularly problematic in high-stakes domains such as healthcare, finance, and autonomous systems, where decisions based on AI models can have significant ethical and safety consequences [59]. In real-world applications, AI developers, stakeholders, and regulatory bodies need accurate, reliable, and easily accessible information to make informed decisions about model deployment, bias mitigation, and ongoing model monitoring. Yet, the lack of consistent and comprehensive documentation can hinder decision making and increase the risk of deploying models with unaddressed limitations or biases [109]. Furthermore, as AI models evolve rapidly with frequent updates and new versions, manually maintaining up-to-date model documentation becomes increasingly impractical, especially across large-scale AI infrastructures. The complexity and diversity of modern AI deployments further exacerbate this issue, as many models are deployed in multi-cloud or hybrid environments, where manually tracking model details and versions is inefficient and prone to error. In such dynamic settings, the inability of current systems to provide automated, structured, and machine-readable model documentation represents a significant challenge for organizations aiming to ensure the responsible use and oversight of AI systems [109].

By developing a pipeline to automate the generation and semantic cataloging of model cards using LLMs, this thesis seeks to overcome these practical challenges. The goal is to establish a scalable, consistent, and transparent process for model documentation that facilitates better decision-making, comparison, and oversight of AI systems, ultimately leading to more trustworthy and reusable models across various sectors.

### 1.4 Research Questions

This thesis investigates the limitations and opportunities of large language model-driven frameworks for the automatic structuring, querying, and discovery of machine learning models using model cards. Despite the emergence of model cards as a critical tool for documenting model performance, intent, and limitations, their adoption remains inconsistent, and their structure rarely conforms to any formal schema. As platforms like Hugging Face continue to scale, the need for machine-readable and semantically enriched model metadata becomes increasingly urgent for auditing, reuse, and governance.

Building on recent advances in ontology-aligned information extraction, SPARQL generation, and graph-augmented LLM reasoning, we structure our investigation around three core research questions:

**RQ1. To what extent can leveraging LLMs for ontology-driven metadata extraction improve the consistency, completeness, and machine-readability of model card information?**

A central hypothesis of this thesis is that large language models, when guided by a formal domain ontology such as the Model Card Report Ontology (MCRO) can produce structured RDF triples from unstructured or inconsistently formatted model cards. Prior work in ontology population and knowledge graph construction [58, 87] has demonstrated the potential of LLMs in translating free-text into structured semantic representations. However, these efforts are often constrained by narrow domains, manually curated rules, or pre-aligned datasets [117]. This thesis instead evaluates the feasibility of the LLM-based approach for generating ontology-compliant triples directly from publicly available model cards on Hugging Face, where metadata is noisy, missing, or inconsistently labeled. Through expert evaluation and schema alignment metrics, we assess the reliability and accuracy of LLM outputs in terms of completeness, semantic conformance, and overall machine readability.

**RQ2. To what extent can LLMs support natural language (NL) querying over a semantic model catalogue?**

As the generated RDF triples are integrated into a knowledge graph, enabling end users, particularly non-technical stakeholders, to interact with the graph becomes a key concern. This question investigates whether LLMs can accurately translate natural language questions into syntactically correct and semantically meaningful SPARQL queries, a task that combines aspects of semantic parsing, schema linking, and controlled generation. Previous research has shown that while LLMs are capable of generating executable queries in structured languages, issues such as hallucination, schema misalignment, and query invalidity remain significant challenges [52, 3]. Our evaluation includes expert-guided prompt refinement, automatic syntactic validation, and correctness checks against a gold-standard ground truth, with a focus on both generalizability and precision in NL-to-SPARQL conversion across unseen queries.

**RQ3. To what extent does Graph Retrieval-Augmented Generation (G-RAG) compare to SPARQL in querying a semantic model catalogue?**

Traditional SPARQL-based querying requires users to be familiar with both the syntax and structure of the underlying ontology. To bridge this gap, recent research has proposed retrieval-augmented generation (RAG) frameworks, particularly Graph-RAG, which leverages graph-based representations as a retrieval layer to ground LLM responses [56, 16]. This question explores whether G-RAG can serve as a viable alternative to direct SPARQL querying for supporting natural language interaction with the semantic model catalogue. We compare the two approaches across multiple dimensions: usability for non-experts, accuracy of retrieved information, resilience to query ambiguity, and consistency of responses. The system incorporates vector-based retrieval over knowledge graph embeddings using Neo4j and Cypher, combined with Gemini-powered generation pipelines via LangChain. Both qualitative feedback and quantitative performance metrics (e.g., answer correctness, relevance, latency) are used to evaluate the relative strengths, weaknesses, and trade-offs of G-RAG versus symbolic querying.

Collectively, these research questions aim to assess the viability of a fully automated, LLM-powered framework for transforming fragmented model documentation into a structured, semantic catalog, while enabling intelligent and accessible interactions for diverse user groups.

## 1.5 Thesis Structure

This thesis is structured as follows:

Chapter 2 provides a comprehensive review of existing literature on model cards, ontology-based metadata extraction, knowledge graphs, large language models, semantic parsing, SPARQL querying, and retrieval-augmented generation. It identifies research gaps and synthesizes theoretical insights relevant to the thesis.

Chapter 3 describes the research design and the system architecture. It introduces the components responsible for metadata extraction, semantic modeling, and natural language querying, and details the datasets and evaluation methods used throughout the study.

Chapter 4 presents the findings of empirical evaluations corresponding to each research question. Metrics such as schema compliance, syntax validity, accuracy, expert feedback, and system performance are reported and analyzed.

Chapter 5 reflects on the implications of the results, highlighting contributions to theory and practice. It also addresses limitations of the current approach, potential risks (e.g., LLM hallucination), and considerations for real-world deployment.

Finally, Chapter 6 summarizes the thesis's contributions, addresses the research questions, and proposes recommendations for future work in both academic and industrial contexts.

# Chapter 2

## Literature Review

This chapter provides the essential background for understanding the contributions of this thesis, which utilizes Large Language Models (LLMs) for automated ontology learning, knowledge graph (KG) construction from model cards, and text-to-SPARQL query generation for question answering (Q/A) systems. It traces the development of techniques in ontology learning, semantic parsing, and natural language interfaces to structured data, highlighting the transformative impact of LLMs. Core topics include ontology learning, knowledge graphs, prompt engineering, retrieval-augmented generation (GraphRAG), error taxonomies, data augmentation, and two-stage SPARQL generation. By integrating progress in these fields, this chapter sets the stage for the thesis innovations in database-agnostic KG construction and advanced query generation.

The structure is as follows: Section 2.1 introduces model cards, examining their purpose and key components. Section 2.2 discusses ontology, including ontology storage, model card ontologies, and the role of SPARQL. Section 2.3 focuses on knowledge graphs and their construction techniques. Section 2.4 explores model discovery and semantic augmentation strategies. Section 2.5 delves into large language models, highlighting their architecture and capabilities. Section 2.6 presents text-to-SPARQL semantic parsing approaches. Section 2.7 reviews Retrieval-Augmented Generation (RAG) and Graph RAG. Section 2.8 covers related work on LLMs for knowledge graph construction, Graph-RAG and SPARQL generation. Finally, Section 2.9 identifies research gaps and outlines the goals of this study.

### 2.1 Model Cards

Model cards are a documentation format created to increase the accountability and transparency of machine learning models [69]. The key aspects of a model, such as performance, training data, capabilities, and potential ethical implications, as illustrated in Figure 2.1, can be effectively conveyed through them [69]. By making comprehensive and easily comprehensible information available to a broad spectrum of stakeholders, including developers, data scientists, policymakers, and end-users, Model cards primarily aim to ensure the responsible use of machine learning models. Having such thorough information allows for more informed decisions about the deployment and use of the model [79].

A typical Model Card contains several key sections that cover the essential aspects of the model: ModelDetails, Intended Use, Data, Evaluation, Ethical Considerations and Caveats. (1) **Model Details:** This section includes technical information about the model, such as its type, architecture, and training methodology. It provides context on how the model was developed and what makes it suitable for certain tasks. (2).**Intended Use:** This section defines the specific tasks and applications for which the model is designed, clarifying the target users and the contexts in which the model is expected to perform well. (3) **Data:** The dataset used to train the model is described in this section. Information about the data sources, data composition, and any preprocessing steps is provided. This helps users assess whether the models data aligns with their application needs and ensures that the model is not biased. (4) **Evaluation:** This section presents the models performance across different datasets and tasks, providing evaluation metrics such as accuracy, precision, recall, and other relevant measures. (5) **Ethical Considerations:** This section addresses potential risks, such as biases in the model, fairness

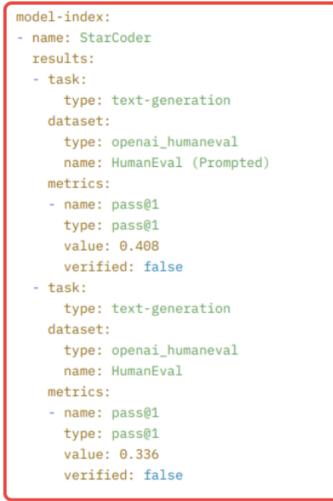


Figure 2.1: Model Card metadata of Model StarCoder from HuggingFace

concerns, and other ethical issues. It helps ensure that the model is used responsibly and without causing harm. (6) **Caveats:** Known limitations and failure modes of the model are outlined here. This section ensures that users are aware of the models boundaries and performance under certain conditions [69].

Even though Model Cards promote greater transparency, their use is currently constrained by a number of issues: Lack of Standardization, Inconsistent Detail, and Human-Readable Formats. (1) **Lack of Standardization:** Regarding the format and content of Model Cards, there is no widely recognized standard. It is challenging for users to consistently compare models because different researchers and organizations employ varying formats and levels of detail. This inconsistency results in information gaps and a lack of model comparability. (2) **Inconsistent Detail:** Model Cards vary greatly in the level of detail they provide. Important details may be left out of some Model Cards, while others may offer comprehensive information about model performance and data. The dependability of model cards as a consistent tool for model evaluation is diminished by this variability. (3) **Human-Readable Formats:** The majority of Model Cards are not machine-readable, even though they are written in human-readable formats (like Markdown). This restricts their application in automated systems where it would be possible to extract and compare model metadata more effectively. Better integration into model repositories and automated assessment of model transparency would be made possible by a machine-readable format [58].

## 2.2 Ontology

Ontologies are an essential component of knowledge representation, especially in the Semantic Web, where they provide a structured framework for defining concepts, properties, and relationships within a domain [62, 37]. This thesis employs ontologies as the formal backbone to enable automated ontology learning, knowledge graph construction, and SPARQL (SPARQL Protocol and RDF Query Language)<sup>1</sup> query generation via LLMs. Ontologies are formal representations of knowledge within a specific domain, defined as a “formal, explicit specification of a shared conceptualization” [73]. They are made up of concepts (or classes), properties, and relationships, which form a common vocabulary for data interpretation and reasoning [73]. This is critical for semantic web applications, which require data integration and interoperability. For example, in the domain of machine learning models, an ontology could define “Model” and “Dataset” classes, with properties like “hasArchitecture” and relationships like “isTrainedOn,” to aid machine understanding and automated inference, as shown in Listing 2.1 [6].

---

```

Class: ModelCard
hasModelDetails: ModelDetails
hasIntendedUse: IntendedUse

```

<sup>1</sup>SPARQL is the common query language and protocol for Linked Open Data and RDF databases. It has been built to query a wide range of data and can easily retrieve information hidden in non-uniform data stored in diverse formats and sources.[86]

```

hasTrainingDataset: Dataset
hasEvaluation: Evaluation
hasEthicalConsideration: EthicalConsideration
hasCaveatsAndRecommendations: Caveats

Class: ModelDetails
hasName: string
hasArchitecture: Architecture
hasVersion: string
developedBy: Organization/Person

Class: IntendedUse
intendedForUseIn: ApplicationDomain
notIntendedForUseIn: ApplicationDomain
intendedUsers: UserGroup

Class: Dataset
hasName: string
hasSize: integer
hasPreprocessing: string
dataSource: DataSource

Class: Evaluation
evaluatedOn: Dataset
hasMetric: Metric
hasResult: float

Class: EthicalConsideration
hasBiasRisk: string
fairnessAssessment: string
privacyRisk: string

Class: Caveats
hasLimitation: string
knownFailureModes: string

```

Listing 2.1: Sample of Model Card Ontology

### 2.2.1 Triplestores

Ontologies are typically written using standardized languages such as RDF [9] and OWL [38]. RDF expresses knowledge as triples, each including a subject, predicate, and object (e.g., `<ex:model1, ex:hasName, "BERT">`), creating a graph structure that captures relationships between things. OWL expands on RDF by providing a broader vocabulary for creating ontologies, including class hierarchies, property constraints, and logical axioms, which enables more expressive and complex knowledge representations. These formalisms render ontologies machine-readable and interoperable.

Triplestores are used to store and maintain RDF-based ontologies as well as the data connected with them. Triplestores are specialized databases designed to handle massive numbers of RDF triples, allowing for efficient storing, retrieval, and reasoning on graph-structured data. They store both the ontological schema (class and property definitions) and the instance data (particular models and their attributes), making them a scalable solution for knowledge base management. Triplestores include Apache Jena [28], and GraphDB [82]. Triplestores provide the framework for storing ontologies and Knowledge Graphs created using LLMs from model cards, allowing for further querying and analysis [112] [125].

### 2.2.2 Model Card Report Ontology

The Model Card Report Ontology (MCRO) is an OWL 2-encoded ontology used for representing model card reports for machine learning models [6]. The purpose of MCRO is to provide a standardized framework for describing machine learning models, enabling the exchange of precise details such as trade-offs, intended users, licensing, and other relevant information. MCRO uses an ontology-based structure to provide semantic expression, standardization, and computable benefits of model card reports, ensuring the transparency and traceability of machine learning models [6].

MCRO is based on several foundational ontologies, and a sample prefix declaration of the MCRO ontology is shown Listing 2.2. These include: Information Artifact Ontology, Software Ontology, Provenance

Ontology , Provenance Ontology (1) **Information Artifact Ontology (IAO)**<sup>2</sup>: a framework for characterizing artifacts such as reports and models. (2) **Software Ontology (SWO)**<sup>3</sup>: Uses a subset of this ontology, focused on elements useful to machine learning models . (3) **Provenance Ontology (PROV)**<sup>4</sup>: Records the provenance or history of data, models, and other objects. (4) **Simple Knowledge Organization System (SKOS)** <sup>5</sup>: Helps standardize the concepts and terminologies used in model descriptions [6].

---

```

@prefix : <http://sbmi.uth.edu/ontology/mco#> .
@prefix mco: <http://sbmi.uth.edu/ontology/mco#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@base <http://sbmi.uth.edu/ontology/mco> .

<http://sbmi.uth.edu/ontology/mco> rdf:type owl:Ontology ;
    owl:imports <https://raw.githubusercontent.com/UTHealth-Ontology/MCO/main
                /import/swo_import_final.owl> ,
        <http://purl.obolibrary.org/obo/iao/2020-12-09/iao.owl> ,
        <http://www.w3.org/2004/02/skos/core> ,
        <http://www.w3.org/ns/prov-o-20130430> ;
    <http://purl.org/dc/terms/creator> "Tuan Amith and Cui Tao" ;
    <http://purl.org/dc/terms/description> "An ontology representing the
        model card structure" ;
    <http://purl.org/dc/terms/title> "Model Card Ontology" ;
    <http://www.w3.org/2004/02/skos/core#note> "encoding of sample model card
        as an instance" .

```

---

Listing 2.2: Prefix Sample of MCRO

### 2.2.3 SPARQL

SPARQL (SPARQL Protocol and RDF Query Language) [90], RDF's standard query language, facilitates easier access to and manipulation of data in triplestores. SPARQL enables users to create expressive queries that retrieve and manipulate data based on patterns in the RDF graph. A typical SPARQL query consists of a SELECT clause, which provides the variables to return, and a WHERE clause, which defines the triple patterns to match. SPARQL, for example, can query both instance data and ontological structures, making it an effective tool for extracting model insights from a knowledge graph, such as tasks or creators. They can be used in more complex ways by utilizing large language models to generate SPARQL queries from natural language inputs, thereby enhancing the capabilities of question-and-answer (Q/A) systems.

For instance, the following SPARQL query retrieves all models along with their creators:

---

```

PREFIX ex: <http://example.org/ontology#>
PREFIX dc: <http://purl.org/dc/terms/>

SELECT ?model ?creator
WHERE {
    ?model a ex:Model ;
           dc:creator ?creator .
}

```

---

<sup>2</sup>The Information Artifact Ontology is a new ontology for information entities that was inspired by work done by the OBI digital entity and realizable information entity branches [18, 81]

<sup>3</sup>The Software Ontology is a resource for describing software tools, including their types, tasks, versions, provenance, and related data. It contains detailed information on licensing, formats, and software applications that are primarily (but not exclusively) relevant to the bioinformatics community [65].

<sup>4</sup>The PROV Ontology expresses the PROV Data Model in the OWL2 Web Ontology Language It defines a set of classes, properties, and restrictions that can be used to represent and exchange provenance data generated by different systems and contexts. It can also be used to create new classes and properties that represent provenance information for various applications and domains. [71]

<sup>5</sup>SSKOS is a field of study that develops specifications and standards to support the use of knowledge organization systems (KOS) such as thesaurus, classification schemes, subject heading systems, and taxonomies within the context of the Semantic Web [68]

---

Listing 2.3: Sample SPARQL query to retrieves all models with their creators

This query selects all resources of type ex:Model and returns their associated creators using the dc:creator property from the Dublin Core vocabulary.

## 2.3 Knowledge Graph

Knowledge Graphs (KGs) are organized representations of knowledge that connect elements (nodes) via semantic relationships (edges), resulting in a graph-based model of a given topic [102]. They offer a machine-interpretable semantic layer that enables applications like search, recommendation, and question answering. KGs are frequently used to integrate disparate data sources and facilitate reasoning over domain-specific knowledge [22].

A standard KG comprises three core components: entities (instances), classes (types), and relations (edges), typically represented as RDF triples. A basic example is:

---

```
ex:modelA ex:hasArchitecture ex:Transformer.
```

---

Listing 2.4: Basic example of KG Components

Here, `<ex:modelA>` is the subject (entity), `<ex:hasArchitecture>` is the predicate (relation), and `<ex:Transformer>` is the object (which can be another entity or a literal). This triple structure forms the basis for semantic graphs that capture complex interdependencies among concepts.

### 2.3.1 KG Construction from Unstructured Sources

Traditional KG construction has depended on rule-based pipelines, template-based extraction, or manual curation [123]. These approaches frequently lack scalability and generalization, despite being successful in clearly defined contexts. Recent studies have investigated the use of large language models to directly extract structured knowledge from unstructured and semi-structured sources, including dataset descriptions, model documentation, and scientific papers, to overcome these limitations. LLM-based methods typically map natural language text to semantic structures, producing ontology-aligned RDF triples. Techniques such as prompt engineering, in-context learning, and fine-tuning are frequently employed to achieve this [112]. These techniques offer a more scalable alternative to conventional information extraction by reducing the reliance on manually created rules or domain-specific feature engineering.

Using contextual awareness and pretrained knowledge, LLMs can capture domain-specific semantics and implicit relations, increasing the expressiveness and coverage of automatically generated KGs [112]. Additionally, ensuring schema conformance and facilitating integration with larger knowledge ecosystems are two benefits of aligning extracted triples with current ontologies.

## 2.4 Model Discovery and Semantic Augmentation

Artifacts such as model cards have been developed in response to the growing demand for standardized documentation and the discoverability of model characteristics, driven by the increasing adoption of machine learning models [69]. These artifacts provide detailed coverage of the design, functionality, constraints, and intended application of models or datasets. The unstructured or semi-structured formats in which this data is frequently presented, however, make automated comprehension and reuse difficult.

### 2.4.1 Model Cards as Metadata Sources

Model cards contain information about the architecture, training data, evaluation metrics, intended use cases, and ethical considerations. Model cards support documentation procedures, but because they are free-text, it is challenging to parse them programmatically. Recent research has explored the use of large language models and natural language processing to extract semantic representations from model card content, aiming to address this issue [110]. Structured metadata aligned with pre-established schemas or

ontologies has been created from free-text descriptions using methods such as entity recognition, relation extraction, and sentence-level classification. Model discoverability is enhanced, integration into searchable repositories is facilitated, and downstream applications, such as model selection and automated reporting, are supported.

## 2.5 Large Language Models

Large Language Models (LLMs), underpinned by the Transformer architecture, have become foundational in tasks such as natural language understanding, reasoning, and code generation [45]. Due to their extensive training on large datasets, LLMs have exceptional natural language creation and comprehension skills [11]. Their ability to perform complex language tasks with minimal supervision makes them highly effective in automating semantic tasks, such as ontology population, triple generation, and query synthesis. These capabilities have catalyzed their adoption in building knowledge-centric systems, including the construction of automated knowledge graphs and the development of natural language interfaces to structured data.

### 2.5.1 Transformer Architecture and Self-Attention

LLMs derive their capabilities from the Transformer architecture introduced by Vaswani et al., illustrated in Figure 2.2 [115]. The self-attention mechanism, at its core, enables the contextual encoding of tokens by dynamically computing dependencies across an entire input sequence. This enables the model to effectively capture hierarchical and cross-sentence relationships, which are essential when extracting RDF triples or aligning text with ontology schemas. While self-attention enables powerful representations, it also introduces interpretability challenges, as the mapping from attention patterns to semantic outputs remains difficult to trace [121].

### 2.5.2 Pre-Training and Knowledge Acquisition

Pre-training on large, diverse corpora equips LLMs with broad linguistic and factual knowledge, making them suitable for tasks such as concept grounding, relation extraction, and entity linking, cornerstones of semantic graph construction. During pre-training, LLMs internalize co-occurrence patterns and semantic structures that can be later leveraged for extracting ontology-aligned information from free text. However, this implicit learning can also encode biases or spurious associations, which complicates the reliability of generated triples and necessitates alignment with formal ontologies to ensure semantic validity.

### 2.5.3 Adaptive Learning Capabilities

LLMs demonstrate remarkable adaptability through mechanisms such as in-context learning, zero-shot prompting, and few-shot learning [13]. These paradigms allow models to generalize across domains and tasks without task-specific retraining. In semantic augmentation workflows, this adaptability supports entity typing, property linking, and constraint satisfaction based on few-shot examples aligned with ontology structure [11]. Moreover, in-context techniques facilitate SPARQL query generation by framing queries as structured outputs conditioned on prior samples or ontology schemas.

### 2.5.4 Applications in Structured Knowledge Generation

Recent advances have showcased the use of LLMs in generating structured knowledge representations, such as RDF triples and SPARQL queries, directly from text. This includes transforming model documentation, dataset descriptions, or question prompts into formal triples or queries aligned with domain-specific ontologies. For instance, models like GPT, Codex, and LLaMA variants are increasingly used for zero-shot triple extraction and program synthesis. These use cases illustrate the potential of LLMs as general-purpose engines for semantic data processing, enabling scalable knowledge graph population and natural language querying over structured stores.

Despite their promise, LLMs pose significant challenges regarding transparency, explainability, and reliability. Phenomena like hallucination, where plausible but incorrect outputs are generated, pose risks when LLMs are used for ontology-aligned knowledge extraction [91]. Furthermore, tracing the provenance of generated triples or queries is non-trivial due to the opaque nature of attention weights and

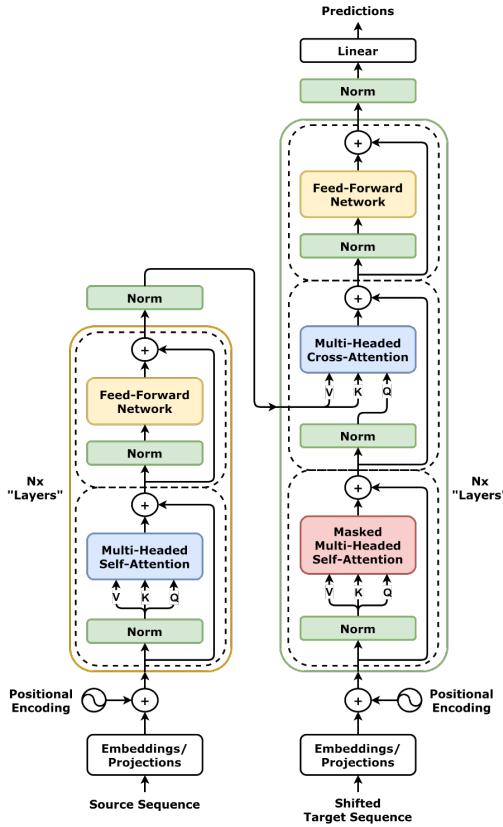


Figure 2.2: A key component of large language models is the Transformer architecture. Every input token is embedded and sent through multi-head attention stacking encoder-decoder blocks.

token-level reasoning. Transparency techniques such as prompt engineering, attention visualization, chain-of-thought prompting, and post-hoc evaluation with knowledge constraints aim to mitigate these limitations. In ontology-based systems, additional semantic validation layers can enforce schema compliance, offering a hybrid approach that combines the generative flexibility of LLMs with the structural rigor of formal ontologies [40].

## 2.6 Text-to-SPARQL Semantic Parsing

Text-to-SPARQL semantic parsing is a specialized subfield of semantic parsing that focuses on translating natural language questions or statements into SPARQL queries executable over RDF knowledge graphs, shown in Figure 2.3, which illustrates the flow of converting a natural language query into a SPARQL query. This capability is essential for building natural language interfaces to structured data, allowing non-expert users to retrieve precise answers from ontologically grounded knowledge stores. Early approaches to semantic parsing employed sequence-to-sequence (seq2seq) models with attention mechanisms, where the input is a natural language query and the output is a SPARQL string. These models treat SPARQL generation as a translation task, encoding the input into a latent vector space and decoding it into query syntax [104]. However, the rigid structure of SPARQL and the need for schema alignment often result in syntactic errors or semantic mismatches. Improvements, such as pointer networks and constrained decoding, have been introduced to enhance the syntactic validity of the output [100].

To ensure well-formedness and adherence to ontology-specific constraints, grammar-constrained models incorporate SPARQL grammar rules during the generation process. These methods use predefined production rules or templates to restrict the search space of the decoder, improving precision and reducing syntactic errors. Template-based models, on the other hand, rely on learned or manually created SPARQL skeletons that are populated based on identified entities and predicates, offering higher control but reduced generalization. A critical component of text-to-SPARQL systems is accurate entity linking—the process of mapping mentions in natural language to entities and relations in the KG. Schema-

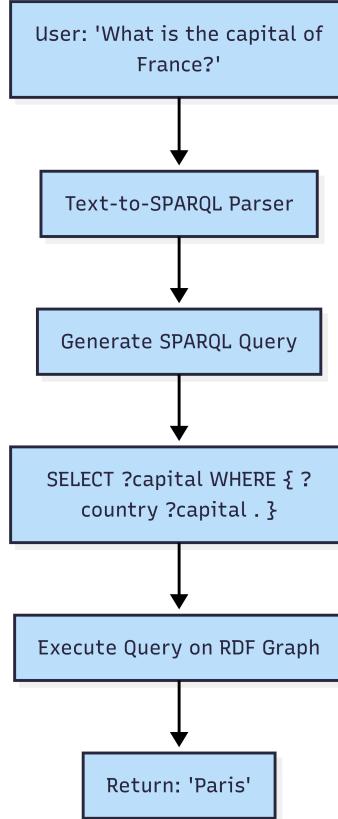


Figure 2.3: Process of Converting a Natural Language Query to a SPARQL Query

aware parsers enhance the parsing process by incorporating domain-specific ontologies or KG schema information during generation. These models often encode the schema (classes, properties, constraints) along with the input query, enabling the model to produce queries that are semantically compliant with the underlying graph.

Recent advancements leverage LLMs, such as T5, GPT, and LLaMA, fine-tuned on NL-SQL or NL-SPARQL datasets. These models show strong zero-shot and few-shot performance in generating structured queries, particularly when combined with prompt engineering or retrieval-augmented generation. Chain-of-thought prompting and schema priming are frequently used to guide the models outputs toward syntactically valid and semantically relevant SPARQL queries. Despite their promise, these models still suffer from occasional hallucinations or semantic drift, emphasizing the need for post-processing and query validation layers. Standard metrics for evaluating text-to-SPARQL systems include exact match accuracy, BLEU score, Schema Adherence, and syntactic Correctness. More nuanced metrics, such as schema compliance accuracy and logical form similarity, have been proposed to assess whether the generated query is not only executable but also semantically correct with respect to the KG schema [52]. Evaluation often involves expert annotation or automated testing against ground truth query-answer pairs.

### 2.6.1 Prompt Engineering

Prompt engineering is essential for guiding LLMs to extract structured information from unstructured text, such as model cards. Effective prompt engineering ensures that LLMs can transform free-text descriptions into machine-readable data, facilitating seamless integration into structured formats [14, 119]. There are several types of prompts used to extract information efficiently. Task-specific prompts direct the models focus on particular aspects of the model card [35], such as What is the model architecture? or What are the performance metrics of the model? These prompts ensure that the models output is targeted and relevant. Zero-shot prompting asks the model to extract information without providing examples, as shown in Figure 2.4.

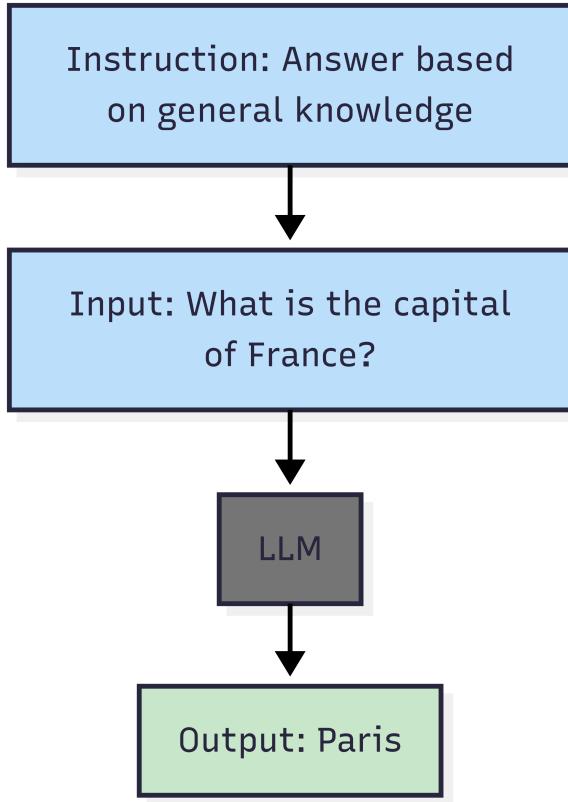


Figure 2.4: Zero-shot learning: The LLM is given only an instruction and input without examples.

This relies on the models pre-existing knowledge to identify and retrieve relevant data. For instance, a zero-shot prompt like Extract the training datasets used for this model tests the models ability to comprehend and extract specific details from the text. In contrast, few-shot prompting provides the model with a few examples to guide it in understanding the format and type of output required, as shown in Figure 2.5 [11]. This method is particularly useful when dealing with diverse formats or unclear sections within the model card.

Additionally, in-context learning involves including relevant examples directly within the prompt to provide context. By embedding snippets of text from the model card, this technique enables the model to understand the structure and extract the desired information more accurately [101]. It is especially beneficial when the documentation is complex or inconsistent, allowing the model to make more informed decisions about how to process the text and generate structured metadata. Refining prompts is often necessary to achieve high-quality outputs. Initially, a prompt may produce vague or incomplete information, and adjustments are made to increase its specificity or clarify the task [96, 14]. For example, a general prompt such as What is the models architecture? can be refined to something more precise, like What is the architecture of the model described in this card? This iterative refinement helps ensure that the extracted data aligns more closely with the requirements, improving both accuracy and relevance. The process of fine-tuning prompts is crucial in transforming raw, unstructured information into well-defined, machine-readable formats suitable for constructing knowledge graphs.

However, prompt engineering does come with challenges. One of the main difficulties lies in the variability of model card formats. Since model cards often differ in structure and presentation, creating a single prompt that works universally can be difficult. This variability necessitates adjusting the prompt to suit the language and format of each specific card. Additionally, certain aspects of model documentation, such as technical performance metrics or ethical considerations, can require more complex prompts or multi-step processes to ensure complete data extraction. The balance between specificity and generalization is also a challenge. Overly specific prompts may restrict the models ability to extract a broader range of information, while overly general prompts may lead to incomplete or ambiguous responses [97].

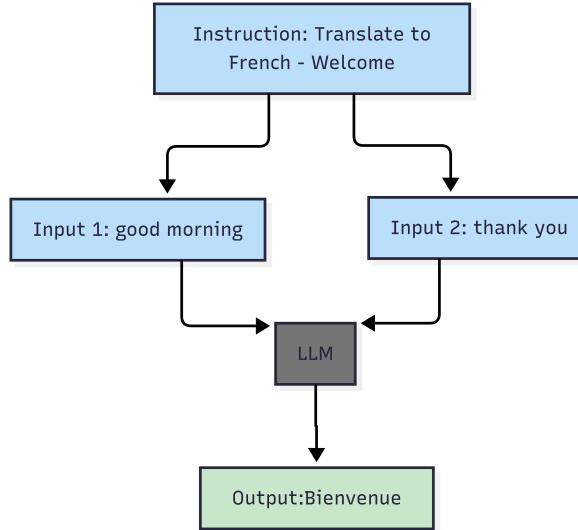


Figure 2.5: Few-shot learning: The LLM receives two examples and an instruction to perform the task.

## 2.7 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) is a powerful technique that enhances the performance of large language models by integrating external knowledge retrieval into the generation process [54]. The concept behind RAG is to combine the strengths of generative models with information retrieval, enabling the model to generate more informed and accurate responses by leveraging external knowledge sources [77]. The process consists of two main components: the retriever and the generator. The retriever searches external document corpora or knowledge bases to identify relevant information, which is then passed to the generator. The generator combines the retrieved information with the original query to produce a response that is not only contextually relevant but also grounded in real-world knowledge [15].

A typical RAG system illustrates in Figure 2.6, works by first embedding both the input query and the knowledge base or document corpus into dense vector representations using models such as transformers [31]. The retriever uses these embeddings to find relevant pieces of information from external sources, typically through similarity search methods such as cosine similarity or nearest neighbor search. Once the relevant information is retrieved, the generator synthesizes the input query along with the retrieved data to generate a response [31]. This approach ensures that the response is both relevant to the query and based on knowledge that goes beyond the models training data.

### 2.7.1 Graph Retrieval-Augmented Generation

Graph Retrieval-Augmented Generation (G-RAG) [21] extends the concept of RAG by leveraging knowledge graphs as the retrieval source instead of traditional text-based corpora. Knowledge graphs represent relationships between entities in a structured, graph-based format, where nodes represent entities, and edges represent the relationships between them [88]. In Graph RAG, the retrieval component focuses on extracting relevant subgraphs or graph-based knowledge, allowing the model to reason over more structured and interconnected data.

Graph RAG systems are particularly useful in tasks that require reasoning about relationships between multiple entities, such as knowledge graph completion, question answering, and domain-specific reason-

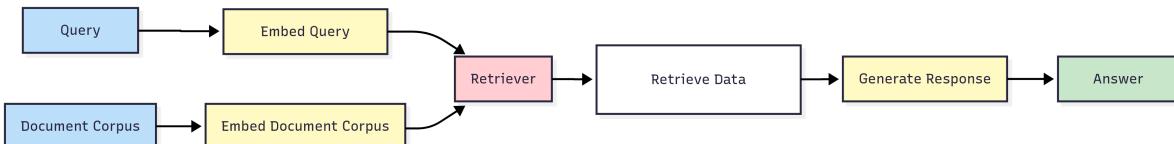


Figure 2.6: Typical RAG workflow

ing [15]. The structure of knowledge graphs provides a richer semantic context than traditional unstructured text, enabling the generator to produce more informed and precise outputs.

In the context of Graph RAG, the retriever queries the knowledge graph based on the input query and retrieves relevant subgraphs that contain information about the entities or relations mentioned in the query. These subgraphs are then passed to the generator, which uses them along with the input query to generate an output that is not only contextually relevant but also semantically rich [88]. This combination of retrieval and generation allows Graph RAG to handle more complex tasks that involve reasoning over structured data.

### 2.7.2 Embedding and Indexing in Graph RAG

To efficiently retrieve relevant knowledge from large-scale knowledge graphs, embedding and indexing techniques play a crucial role in optimizing the process. Embeddings are used to represent both entities and relations in the graph as dense vectors, capturing their semantic meaning and relationships [39]. Techniques such as graph neural networks or transformer-based models are used to learn these embeddings, ensuring that the vector representations accurately reflect the structure and semantics of the graph. Once the embeddings are generated, indexing is used to efficiently search for relevant subgraphs or entities during the retrieval phase. By leveraging techniques like Approximate Nearest Neighbor (ANN) search<sup>6</sup> [64], Graph RAG systems can quickly identify the most relevant knowledge from the graph, even in large-scale graphs with millions of nodes and edges [106]. Indexing ensures that the retrieval process is fast and scalable, which is critical for applications that require real-time or high-throughput performance.

The combination of embedding and indexing techniques allows Graph RAG systems to perform complex queries and generate responses that are semantically grounded in the knowledge graph, while also being computationally efficient.

## 2.8 Related Work

This section explores relevant research in areas that contribute both methodologically and conceptually to the challenges addressed in this thesis. The key themes covered include model metadata extraction, RDF triple generation, knowledge graph construction, SPARQL querying, and the integration of G-RAG techniques. Each of these research areas plays a vital role in the automation of model metadata documentation, structure, and querying, addressing the core challenge of improving AI transparency and model discovery.

### 2.8.1 LLMs for Model Card Metadata Extraction

Model cards, as a documentation tool for machine learning models, typically contain unstructured text that needs to be systematically extracted and converted into structured data for easier analysis. The challenge this thesis addresses is automating this extraction process using Large Language Models (LLMs), a methodology that significantly reduces manual effort.

Previous works have introduced methods for extracting model metadata from unstructured model card descriptions. For instance, Tian et al. (2024) [110] proposed a system to automatically generate LLM model cards using Named Entity Recognition (NER)<sup>7</sup> and Relation Extraction (RE)<sup>8</sup> methods. These approaches rely on LLMs ability to understand and process natural language to identify key model characteristics such as training data, architecture, intended usage, and performance metrics. This capability directly addresses the challenge of automating metadata extraction in this thesis. The methodologically relevant aspect of these studies lies in their use of LLMs to parse complex text and convert it into structured data, an approach that underpins the framework used in this thesis for extracting RDF triples from model cards.

<sup>6</sup> ANN Search is a well-designed index to find approximate nearest neighbors in a high-dimensional dataset for a given query [116].

<sup>7</sup>Entity recognition (NER), also known as entity chunking or entity extraction, is a natural language processing (NLP) technique that detects preset types of objects in a body of text [41].

<sup>8</sup>Relationship extraction is a technique used in natural language processing (NLP) to understand the relationships between items stated in text [33].

### 2.8.2 RDF Triples and Knowledge Graph Construction

The generation of RDF triples from unstructured text data is foundational to this thesis approach of structuring model metadata into knowledge graphs. RDF triples provide a formalized, machine-readable representation of relationships between entities. The process of generating these triples from model card metadata is a critical component of the pipeline developed in this thesis.

Recent Research has explored the use of NLP techniques to automatically extract RDF triples from model descriptions. Regino et al. (2022) [92] proposed the QART framework, which uses templates and few-shot prompts with models like T5 and GPT Neo to generate RDF triples from e-commerce product Q/A to populate knowledge graph. These efforts align conceptually with the goal of this thesis, which aims to use RDF triples to represent model metadata in a structured, queryable form. The methodological approach employed in these studies, including the use of LLMs and prompt engineering, serves as a foundation for the triple generation process in this work. These prior studies also highlight the challenge of ensuring high accuracy in RDF triple extraction, which is crucial for creating reliable knowledge graphs for model comparison and selection.

### 2.8.3 Graph-Based Retrieval-Augmented Generation (Graph-RAG)

Graph-RAG combines retrieval-based and generation-based techniques, enabling models to query a knowledge graph to retrieve relevant information and generate outputs based on that context [120]. This hybrid approach enhances the models ability to reason about data, improving the accuracy and relevance of its outputs.

The concept of G-RAG, as explored by Zhu et. al. [124], is highly relevant to this thesis as it provides a mechanism to integrate knowledge graphs with LLM-based generation. The challenge addressed here is ensuring that the generated content is informed by structured knowledge, such as model metadata stored in RDF triples. In the context of this thesis, Graph-RAG allows the model to retrieve pertinent metadata from the knowledge graph before generating structured model explanations or summaries. This methodologically aligns with the design of the system developed in this work, where knowledge graphs are used to guide model explanations and SPARQL queries are employed to retrieve specific triples for generation tasks.

### 2.8.4 SPARQL Querying and Knowledge Graphs

The use of SPARQL to query knowledge graphs is essential for extracting and interacting with model metadata stored as RDF triples. This querying capability enables users to retrieve specific pieces of information based on defined criteria, facilitating tasks like model comparison and discovery [7].

Previous studies have demonstrated how SPARQL can be used effectively to query knowledge graphs populated with model metadata. This research provides a methodological foundation for the SPARQL-based querying framework developed in this thesis [67]. The challenge this thesis addresses is enabling seamless, efficient querying of large knowledge graphs using LLMs. The integration of SPARQL with LLM-based systems enables complex querying and extraction of relevant model metadata, a central component of the system architecture developed in this work. Additionally, the research on improving SPARQL query efficiency and accuracy through optimization techniques directly contributes to refining the querying process in this thesis [44, 26].

## 2.9 Research Gaps and Goals

Despite growing attention to model documentation, a key research gap lies in the lack of standardization across model cards. Their diverse structure, ranging from concise bullet points to lengthy narratives, creates ambiguity and impedes automated metadata extraction. This inconsistency results in poor coverage and accuracy when attempting to extract structured knowledge from these documents. To address this, the goal is to build a robust, format-agnostic extraction pipeline using LLMs and prompt engineering. This pipeline will adapt dynamically to diverse card structures and reliably identify metadata attributes such as model purpose, inputs, limitations, and evaluation results [27, 34, 110].

Another pressing challenge is transforming unstructured textual metadata into machine-interpretable semantic representations. While prior work in RDF triple extraction has shown promise, most existing

systems are either rule-based or narrowly trained, and they perform poorly on domain-specific or noisy inputs, such as those found in real-world model cards. The goal here is to develop an LLM-driven RDF triple generation mechanism that aligns extracted information with a predefined ontology, ensuring semantic consistency and coverage across model documentation [20].

A further gap exists in the construction and scalability of the resulting knowledge graph. As more triples are generated, indexing and retrieval become performance bottlenecks. Traditional symbolic SPARQL engines struggle with responsiveness and scalability, especially when integrated with generation tasks. To overcome this, embedding-based graph representations will be used to support similarity-aware retrieval, and approximate nearest neighbor indexing methods will be employed to accelerate performance. These optimizations are essential to maintain fast, relevant access to metadata within a RAG pipeline [42, 66].

Typical RAG systems retrieve unstructured documents based on keyword similarity, often ignoring semantic structure. When adapted for semantic metadata retrieval, this introduces additional complexity. Retrieved triples must be coherent, meaningful, and contextually relevant to the query. The goal is to tightly integrate knowledge graph retrieval with generative answering via a filtered, graph-aware fusion mechanism that respects both structural and semantic proximity, reducing hallucinations and improving response fidelity. An equally important issue is the reliability and quality of SPARQL queries generated by the system. Automatically generated queries often suffer from syntax errors, schema violations, or irrelevant paths due to model misalignment with the graph schema. The goal is to introduce schema-constrained prompting and post-generation validation to ensure that generated queries are executable, accurate, and semantically meaningful. This also includes embedding schema-awareness into the LLM prompt design to improve the interpretability and correctness of SPARQL outputs [61, 63].

# Chapter 3

## Methodology

This methodology presents a concise framework aimed at improving semantic cataloging by automating the generation, management, and querying of semantic model cards derived from Hugging Face model cards [118]. It is structured around three core components, seamlessly integrated to address the challenges of organizing and accessing machine learning model information:

1. **Ontology Mapping:** This component leverages advanced LLMs to extract key metadata from model cards and organizes it into a standardized, machine-readable format using MCRO. The structured data is stored as RDF triples in both JSON & Turtle files.
2. **Knowledge Graph Query System:** Using the structured turtle schema, this component builds a knowledge graph in GraphDB<sup>1</sup> [83]. It employs LLMs to translate natural language questions into SPARQL queries, enabling precise retrieval of information from the graph, with results validated for accuracy.
3. **Retrieval-Augmented Generation:** This component enhances accessibility by integrating the data into a Neo4j<sup>2</sup> knowledge graph [43]. By combining similarity-based vector searches with Cypher queries<sup>3</sup> and leveraging LLMs and LangChain<sup>4</sup>, the system enables both technical and non-technical users to interact with it effortlessly.

Together, these components form a robust pipeline that automates the creation and querying of semantic model cards, making AI systems more transparent and approachable by bridging the gap between complex model documentation and user understanding.

### 3.1 Research Questions

Main Research Question:

**To what extent can LLMs and model documentation be leveraged to automatically generate, catalog, complete, compare, and discover semantic model cards?**

#### 3.1.1 Sub-questions

RQ1: **To what extent can leveraging LLMs for ontology-driven metadata extraction improve the consistency, completeness, and machine-readability of model card information?**

**Motivation:** Model cards are critical for documenting machine learning models, yet they often lack standardization, contain incomplete metadata, and are difficult to parse programmatically. This limits their utility for tasks such as model auditing, discovery, and reuse. Ontologies can provide a formal

<sup>1</sup>GraphDB is a highly efficient, scalable, and robust graph database that supports RDF and SPARQL queries. GraphDB is one of the few triplestores capable of performing real-time semantic inference at scale, allowing users to deduce new semantic facts from existing ones [84].

<sup>2</sup>Neo4j is the world's top graph database. It is a high-performance graph store that includes all of the features of a mature and robust database, such as a user-friendly query language and ACID transactions [74].

<sup>3</sup>Cypher is a declarative graph query language used by developers around the world. Neo4j's Cypher provides expressive and efficient queries for property graphs [74].

<sup>4</sup>LangChain: LangChain is a framework for creating applications using LLMs [51].

structure to address these issues, but manual annotation is not scalable [58]. LLMs offer strong capabilities in understanding and structuring natural language, making them suitable for automated metadata extraction guided by ontological constraints [87, 49].

**Aim:** This research investigates whether combining LLMs with an ontology (e.g., MCRO) can significantly enhance the consistency (alignment with a formal schema), completeness (coverage of key metadata fields), and machine-readability (conversion into structured RDF triples) of model card information. The goal is to evaluate the extent to which this approach can standardize and automate model documentation at scale [87].

**RQ2: To what extent can LLMs support natural language (NL) queries over a semantic model catalogue?**

**Motivation:** As ML models proliferate across domains, the need for transparency, reusability, and responsible deployment has led to the emergence of semantic model catalogues, repositories enriched with ontologically structured metadata (e.g., using RDF and OWL). While these structured representations offer rich querying capabilities through languages like SPARQL, their use remains largely inaccessible to non-expert users unfamiliar with formal query syntax or ontology schemas [52]. Recent advances in LLMs have demonstrated strong potential in translating NL into executable queries across various structured formats [114]. Leveraging LLMs to bridge the gap between human-intuitive NL and the formalism of semantic queries could democratize access to ML model metadata and foster broader participation in model selection, auditing, and reuse.

**Aim:** This study aims to investigate the extent to which LLMs can effectively support NL querying over semantically structured model catalogues. Specifically, it seeks to evaluate LLMs ability to translate user intents into accurate and ontology-compliant SPARQL queries, identify patterns in failure cases (e.g., ambiguity, ontology misalignment), and propose strategies to improve robustness and interpretability in NL-to-SPARQL generation.

**RQ3: To what extent does G-RAG compare to SPARQL in querying a semantic model catalogue?**

**Motivation:** Current semantic query systems like SPARQL enable structured querying of knowledge graphs but often require expert knowledge, limiting their accessibility. G-RAG systems, which combine vector-based retrieval with Cypher queries in databases such as Neo4j, promise enhanced flexibility and user-friendliness. However, the comparative performance, usability, and result relevance of G-RAG systems versus traditional SPARQL-based systems, particularly in the context of AI model querying, remain underexplored. Understanding these differences is crucial for advancing AI model catalogues and improving knowledge retrieval systems [56].

**Aim:** This study aims to evaluate the extent to which a G-RAG system, which integrates vector similarity search and Cypher queries in Neo4j, compares to SPARQL-based systems in querying AI model metadata. Specifically, the research will focus on assessing the systems performance in processing large datasets, usability for non-expert users in terms of ease of query formulation, and the relevance of the results in terms of accuracy and completeness. The ultimate goal is to understand the advantages and limitations of G-RAG systems for efficient, intuitive, and scalable querying of AI model catalogues.

## 3.2 System Architecture

This subsection presents a top-down architectural perspective of the framework, as depicted in Figure 3.1, which illustrates the whole pipeline for extracting, structuring, and querying AI model metadata from HF model cards. It addresses the challenges of processing unstructured or semi-structured data, aligning it with formal ontologies such as MCRO, and enabling semantic querying with contextual augmentation. To achieve these objectives, the system employs a modular architecture comprising three interoperable pipelines: (1) the Ontology Mapping Pipeline, (2) the Knowledge Graph Question Answering (KGQA) Pipeline, and (3) the Retrieval-Augmented Generation (RAG) Pipeline. This pipeline based design is both deliberate and essential, offering several key advantages that enhance **Semantic Layering**: Each pipeline operates at a distinct semantic level data structuration (Ontology Mapping), precise knowledge retrieval (KGQA), and contextual knowledge synthesis (RAG) ensuring clean and efficient transformations between different representations (text → RDF triples → SPARQL → NL responses + retrieved

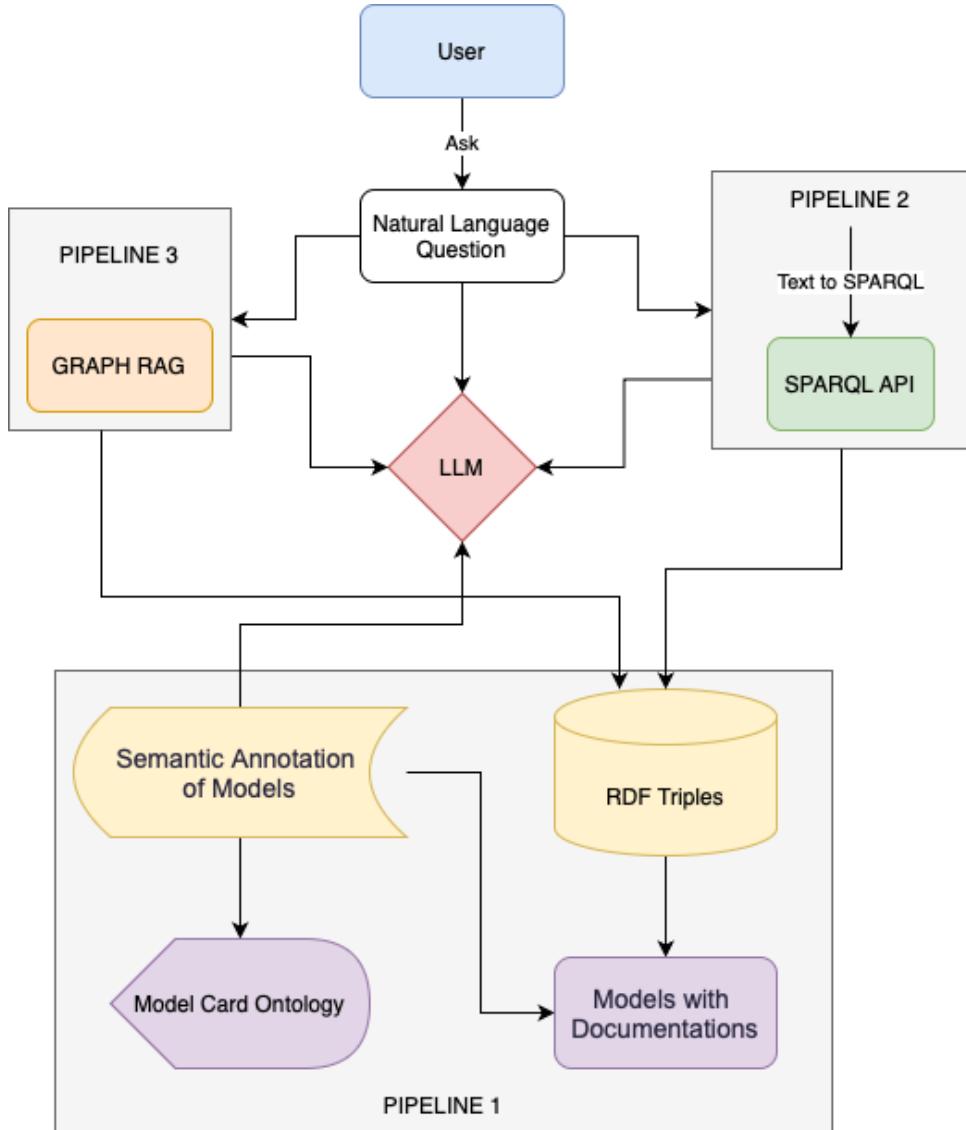


Figure 3.1: Overall System Architecture

context)., **Separation of Concerns**: Critical functionalities are isolated, preventing unintended interactions and simplifying design, implementation, and maintenance., **Scalability and Reusability**: Modules can be scaled independently and reused in different contexts., **Ease of Evaluation and Debugging**: Modularity allows for targeted evaluation and localized debugging.

### 3.2.1 Pipeline Descriptions

This pipeline processes raw text from HF model cards and associated datasets, extracting metadata entities and relationships. It employs a “Text to Triples” process to align this data with MCRO, generating RDF triples that populate a KG. A semantic augmentation process is applied, as shown in Figure 3.2. This step enriches the extracted metadata by assigning ontological meaning and aligning it with the underlying knowledge model. The semantic augmentation involves: Entity recognition and linking , Relation Extraction, Ontology Alignment and Triple Normalisation and deduplication. (1) Entity Recognition and Linking - Identifying and linking mentions to ontology classes or individuals. (2) Relation Extraction - Structuring unstructured phrases into RDF triple patterns. (3) Ontology Alignment - Ensuring concepts match those in the MCRO-based schema. (4) Triple Normalisation and Deduplication - Removing redundant triples and converting predicates to canonical forms.

This process enables the population of a consistent and queryable RDF knowledge graph from noisy LLM-generated outputs. It enhances both syntactic structure and semantic richness, improving downstream

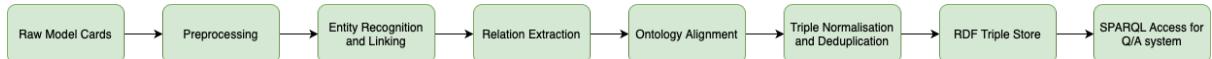


Figure 3.2: Semantic augmentation pipeline applied to LLM-extracted metadata

```

"s": "mcro:Falconsainsfwimedetection-DatasetInformationSection",
"p": "prov:hasTextValue",
"o": "Datasets: ImageNet-21k, proprietary dataset"
  
```

Figure 3.3: Generating normalized RDF triples

tasks like SPARQL-based QA and semantic search.

These triples are stored in an RDF Triple Store, forming a structured catalog of semantic annotations accessible for querying as shown in Figure 3.3. The KGQA Pipeline facilitates user interaction by interpreting natural language queries via a Natural Language Interface. It translates these queries into SPARQL, executes them against the RDF Triple Store, and returns structured, precise answers. This pipeline leverages the SPARQL API to ensure seamless communication between the user and the knowledge base. For queries requiring broader context beyond the KG, the RAG Pipeline integrates retrieval and generation capabilities. It employs vector similarity search to retrieve relevant passages from external documents or datasets and uses an LLM to synthesize comprehensive responses. The LLM is informed by ontologies, which enhance its reasoning and annotation capabilities, while G-RAG coordinates the interplay between retrieval and generation, ensuring that responses are both factually grounded and contextually rich.

The subsequent subsections will provide a detailed exposition of each pipeline, dissecting their internal components, algorithms, and specific contributions to the overall framework’s capability to transform unstructured model metadata into actionable, queryable knowledge.

### 3.3 Dataset

The dataset consists of structured RDF triples extracted from Hugging Face model cards, transformed into semantic RDF knowledge graphs explicitly aligned with the MCRO. A custom pipeline, integrating the Google Gemini<sup>5</sup> and the Hugging Face API, was employed to extract and process these triples. These triples are specifically extracted from Hugging Face model cards to check the AI models and contain raw documentation from the model cards, making them ideal for evaluating transparency. The triples contain all the classes defined in the MCRO ontology, and constraints were enforced using zero-shot demonstrations and instructions to ensure they fit the required structure.

#### 3.3.1 Data Source Collection

The dataset’s source was selected after evaluating multiple platforms, including GitHub<sup>6</sup>, Hugging Face, and Kaggle<sup>7</sup>, for their suitability in providing model metadata. Hugging Face was chosen due to its ease of access, robust API integration, and extensive repository of millions of models, which includes both model cards and data models for potential future expansion. The platform requires only an API key and a URL, enabling seamless data retrieval and integration into the research pipeline. Its vast and diverse model collection ensured a rich dataset, critical for producing statistically meaningful results. In contrast, Kaggle’s model repository, with only 463 models and 3,041 variations, lacked the breadth necessary for generalizable insights, despite available API access. GitHub, while increasingly utilized for hosting AI models, was deemed impractical due to its immature model hosting infrastructure and the additional configuration required to extract metadata. Thus, Hugging Face emerged as the most practical and comprehensive source.

The model cards were accessed programmatically using the Hugging Face API. Specifically, the HfApi was utilized to retrieve a list of models sorted by download frequency, and the ModelCard was employed to load the content of each individual model card. The model cards are stored in Markdown format,

<sup>5</sup>Gemini: A LLM developed by Google [17].

<sup>6</sup>GitHub is a proprietary developer platform for creating, storing, managing, and sharing code [36].

<sup>7</sup>Kaggle is a platform for data science and machine learning, where users can compete, collaborate, and learn [46].

containing semi-structured text with various metadata fields such as model name, license, architecture, and datasets used.

The dataset was constructed by curating 1,000 model cards from the Hugging Face Model Hub, selected based on download frequency to ensure diversity and representativeness. The raw text of each model card was extracted and processed as the foundation of the dataset. This sampling strategy balanced computational feasibility with statistical robustness, employing a retry fallback method to ensure no model was excluded. During the data collection process, models without model cards were encountered. The pipeline was designed to handle such cases gracefully, skipping models where the model card could not be loaded or where triple generation failed, ensuring that only valid data was included in the dataset. The dataset included seven models without model cards, reflecting real-world inconsistencies: approximately 70% of the model cards featured incomplete metadata, while 30% provided comprehensive documentation [58]. Each model card contained semi-structured markdown content, including metadata such as model name, license type, architectural details, training datasets, and performance metrics. These attributes made the dataset well-suited for studying semantic cataloging and supporting semantic query systems. Raw markdown texts were retrieved using the Hugging Face API (HfApi and ModelCard classes), filtered, and deduplicated to eliminate broken or redundant metadata.

To evaluate the system’s ability to handle model card-related queries, the dataset also includes QA data in the form of prompts (questions) that relate directly to the data in the triples. These prompts are designed to assess the system’s performance in retrieving and reasoning about the information stored in the knowledge graph, with a focus on semantic cataloging.

The questions are categorized into three types:

- Description - These questions aim to extract explicit facts about models, datasets, or metrics from the model cards. An example would be: “What is the use case of DeBERTaDecodingenhanced-BERTwithDisentangledAttention?”
- Comparison - These questions are designed to compare different models, datasets, or metrics. An example would be: “Which model has fewer than 1 million parameters?”
- Reason - These questions focus on understanding causal relationships or design choices. An example would be: “What is the primary tradeoff of using LTXVideo?”

These questions are used for both the KGQA and G-RAG components of the system, allowing for a comprehensive evaluation of its capabilities in promoting transparency in AI models.

### 3.3.2 Triple Extraction, Conversion and Storage

RDF triples were extracted from the normalized model card text using the Google Gemini Large Language Model (LLM), specifically the Gemini 2.0 Flash variant. The choice of LLM followed an evaluation of several models, including Llama-2-7b-chat-hf, which was initially explored (the implementation code is available in the repositorys dump folder). However, after comparing Llama-2-7b-chat-hf and Gemini 2.0 Flash with Ground truth from 2 semantic web experts, Llama-2-7b-chat-hf exhibited missing rdf types and non compatibility with MCRO Ontology, elevated error rates, and significant computational overhead, undermining efficiency and reliability as seen on the right in Figure 3.4 [52, 94]. In contrast, Gemini 2.0 Flash offered faster execution, improved accuracy, reduced hallucinations, compatibility with MCRO and provided the types which are necessary for the project with compatibility with tools like GraphDB and Neo4j, making it ideal for processing the large-scale Hugging Face dataset as seen on the left in Figure 3.4. Therefore, the Gemini 2.0 flash is used in all the pipelines.

```
[{"s": "mcro:Falconsainsfwimagedetection", "p": "rdf:type", "o": "mcro:Model"}, {"s": "mcro:Falconsainsfwimagedetection", "p": "mcro:hasModelDetail", "o": "mcro:Falconsainsfwimagedetection-ModelDetailSection"}, {"s": "mcro:Falconsainsfwimagedetection", "p": "mcro:hasUseCase", "o": "mcro:Falconsainsfwimagedetection-UseCaseInformationSection"}, {"s": "mcro:Falconsainsfwimagedetection", "p": "mcro:hasLimitations", "o": "mcro:Falconsainsfwimagedetection-LimitationsInformationSection"}, {"s": "mcro:Falconsainsfwimagedetection", "p": "mcro:hasTrainingData", "o": "mcro:Falconsainsfwimagedetection-TrainingDataInformationSection"}, {"s": "mcro:Falconsainsfwimagedetection", "p": "mcro:hasReference", "o": "mcro:Falconsainsfwimagedetection-ReferenceInformationSection"}, {"s": "mcro:Falconsainsfwimagedetection-ModelDetailSection", "p": "rdf:type", "o": "mcro:ModelDetailSection"}, {"s": "mcro:Falconsainsfwimagedetection-ModelDetailSection", "p": "prov:hasTextValue", "o": "Model Description: The Fine-Tuned Vision Transformer (ViT) is a variant of the transformer encoder architecture, similar to BERT, that has been adapted for image classification tasks. This specific model, named \"google/vit-base-patch16-224-in21k\", is pre-trained on a substantial collection of images in a supervised manner, leveraging the ImageNet-21k dataset. The images in the pre-training dataset are resized to a resolution of 224x224 pixels, making it suitable for a wide range of image recognition tasks.;Hyperparameter Settings: The model was fine-tuned with a judiciously chosen batch size of 16. To facilitate this fine-tuning process, a learning rate of 5e-5 was employed.;Training Process: This training phase was executed using a proprietary dataset containing an extensive collection of 80,000 images, each characterized by a substantial degree of variability. The dataset was thoughtfully curated to include two distinct classes, namely \"normal\" and \"nsfw.\". This diversity allowed the model to grasp nuanced visual patterns, equipping it with the competence to accurately differentiate between safe and explicit content."}, {"s": "mcro:Falconsainsfwimagedetection-UseCaseInformationSection", "p": "rdf:type", "o": "mcro:UseCaseInformationSection"}, {"s": "hf:falconsainsfw_image_detection", "p": "modelcard:hasArchitecture", "o": "hf:architecture-transformer"}, {"s": "hf:architecture-transformer", "p": "dul:hasParameterValue", "o": "Transformer"}, {"s": "hf:falconsainsfw_image_detection", "p": "modelcard:hasTask", "o": "hf:task-image-classification"}, {"s": "hf:task-image-classification", "p": "dul:hasParameterValue", "o": "Image Classification"}, {"s": "hf:falconsainsfw_image_detection", "p": "modelcard:hasDataset", "o": "hf:dataset-imagenet"}, {"s": "hf:dataset-imagenet", "p": "dul:hasParameterValue", "o": "Imagenet"}, {"s": "hf:falconsainsfw_image_detection", "p": "modelcard:hasFramework", "o": "hf:framework-huggingface"}, {"s": "hf:framework-huggingface", "p": "dul:hasParameterValue", "o": "Huggingface"}, {"s": "hf:falconsainsfw_image_detection", "p": "modelcard:hasModalities", "o": "hf:modalities-image"}, {"s": "hf:modalities-image", "p": "dul:hasParameterValue", "o": "Image"}]
```

Figure 3.4: Comparing Gemini (Left) and LLama (Right) Output

## 3.4 Knowledge Graph Ingestion

In this subsection, we outline the process of ingesting the approximately 45,000 RDF triples, previously extracted from Hugging Face model cards and stored in Turtle format, into three distinct systems: GraphDB, Neo4j, and Protege. This ingestion process was critical for constructing and validating the knowledge graphs underpinning the systems evaluated in this thesis.

### 3.4.1 Ingestion into GraphDB

The Turtle files containing the RDF triples were ingested directly into GraphDB, a triplestore optimized for managing RDF data. During ingestion, schema validation was enforced to ensure compliance with the MCRO ontology, thereby preserving the semantic integrity of the knowledge graph. Additionally, CURIE (Compact URI) resolution was applied to correctly expand prefixed URIs, maintaining consistency across all identifiers within the graph. After successful ingestion, SPARQL queries generated by the system were executed against the knowledge graph and their results were rigorously compared to a gold standard (i.e., a manually curated ground truth developed by 2 semantic web experts). This comparison enabled a systematic evaluation of the models ability to generate accurate and semantically valid queries.

### 3.4.2 Ingestion into Protege

Additionally, the Turtle files were loaded into Protege, an ontology editor, to visualize and validate the MCRO ontology structure. This step enabled a visual inspection of the ontology and instance data, ensuring that the extracted triples correctly instantiated the MCRO ontology. Here, the structure of Model BGE-M3 got validated and checked for inconsistencies with reasoner HermiT<sup>8</sup>, showing in Figure 3.5.

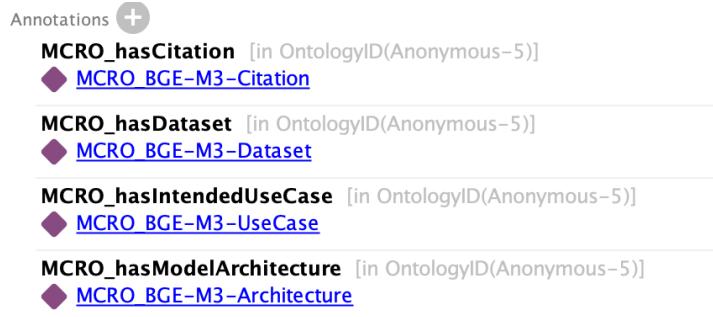


Figure 3.5: BGE-M3 Structure Validate using property Assertion

### 3.4.3 Ingestion into Neo4j

The knowledge graph is constructed from a set of RDF triples, where nodes represent mcro entities such as Model, ModelArchitectureInformationSection, and ReferenceInformationSection. Each triple contributes to either defining a relationship between two entities or assigning properties to a specific node. For instance, a triple like:

---

```
(model_GeminiPro, mcro:hasConsideration, model_GeminiPro_Consideration)
```

---

is transformed into a relationship in the graph, where “model\_GeminiPro” (label: Model) is linked to “model\_GeminiPro\_Consideration” (label: ConsiderationInformationSection) via the relationship HAS-CONSIDERATION. Properties, such as textual content or framework details, are attached directly to nodes. These values are cleaned for Cypher compatibility using a custom preprocessing function that removes LaTeX artifacts, braces, and special characters to ensure the integrity of Neo4j ingestion.

The triples are automatically converted into a Cypher script, which includes:

- CREATE statements for labeled nodes with attached properties.

---

<sup>8</sup>HermiT is a reasoner for ontologies written in the Web Ontology Language. HermiT can check an OWL file to see if the ontology is consistent, identify subsumption relationships between classes, and much more [108].

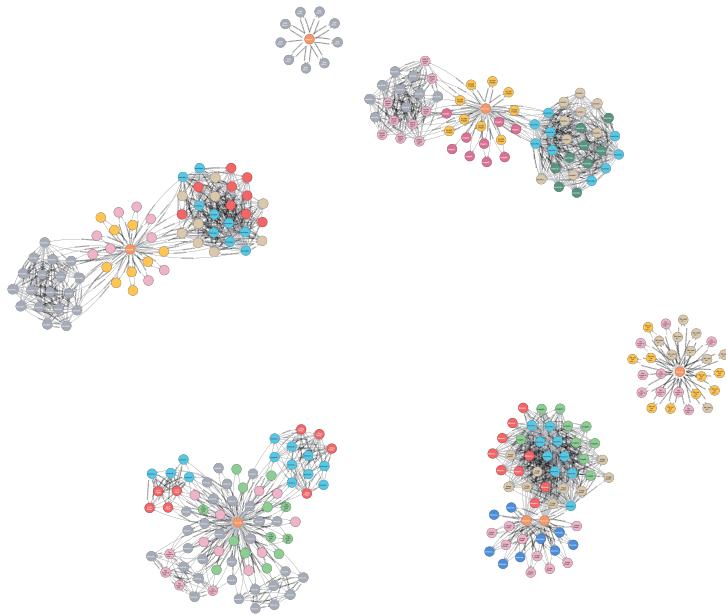


Figure 3.6: Sample of Extracted\_triples.ttl in Neo4j

- MATCH and CREATE statements for relationships between nodes.

Each nodes identifier is normalized (e.g., mcro:clip becomes mcro.clip) to comply with Cypher syntax. The resulting graph is imported into Neo4j, where it is visualized as a network of labeled entities and their typed connections.

In the Figure 3.6, it is worth noting that not all nodes are connected in a single component; some parts of the graph form isolated clusters or disconnected subgraphs. This fragmentation reflects gaps or silos in the original metadata and will be discussed in Detailed in Section 3.7.

Both the GraphDB and Neo4j knowledge graphs underwent manual testing to verify their reliability. Sample queries were executed on each system, and the correctness of the responses was confirmed. This step was essential to ensure that the graphs could support accurate and consistent querying, laying a solid foundation for their use in subsequent stages of the thesis. A single, consistent knowledge graph representation was selected based on its capacity to enable accurate and consistent querying across both GraphDB and Neo4j systems. This unified representation ensures that the knowledge graph remains robust and adaptable, regardless of the querying environment. Finally, knowledge graph were connected to a LLM, allowing users to interact with the graphs through natural language queries. This integration enhances the usability and accessibility of the dataset, facilitating intuitive exploration of the constructed knowledge graphs.

## 3.5 Pipeline 1: ModelCards to Knowledge Graph

### 3.5.1 Objective and Scope

This pipeline serves as the foundational data transformation layer within the framework architecture. Its primary objective is to convert unstructured textual content from Hugging Face model cards into rigorously structured RDF triples compliant with the MCRO. This transformation enables semantic querying, ontological reasoning, and interoperability of model metadata across heterogeneous AI ecosystems. A secondary objective ensures comprehensive alignment between model card sections (e.g., “Intended Use”, “Limitations”, “Training Data”) and corresponding MCRO entities and relationships, and compares it with the gold standard triples (ground truth). This mapping establishes a machine-readable semantic representation that facilitates efficient exploration, comparison, and integration of AI model characteristics at scale while preserving contextual nuances present in the original documentation.

### 3.5.2 Preprocessing Model Cards

The preprocessing pipeline transforms raw Hugging Face model cards into normalized, ontology-aligned inputs through a rigorous four-stage process as shown in Figure 3.7. (1) **Source acquisition** begins by programmatically retrieving model card Markdown (.md) files from Hugging Face repositories via their unique URLs, using the Hugging Face Hub API to fetch both YAML frontmatter metadata and free-form documentation. (2) **Normalization** then converts this Markdown to plain text through deterministic operations: YAML frontmatter is isolated and converted to JSON; Markdown syntax (code blocks, images, links, tables) is stripped while preserving semantic content; headers are standardized to consistent levels (e.g., “### Limitations” → “## Limitations”); and content is split into discrete sections using header boundaries, with key sections like “Intended Use,” “Limitations,” and “Training Data” explicitly segmented. (3) **Tokenization and noise removal** subsequently process each section: content undergoes sentence segmentation and word tokenization using Unicode-aware splitting, followed by aggressive noise filtration that removes non-alphanumeric tokens, URLs, email addresses, boilerplate phrases, and sentences with fewer than three meaningful tokens while preserving critical elements like model names and acronyms. Finally, (4) **Ontology term mapping** applies a bidirectional lexicon to normalize terminology, converting MCRO identifiers to human-readable labels (e.g., “MCRO:0000147” “licenseInformationSection”) and vice versa, with contextual disambiguation resolving term ambiguity (e.g., distinguishing between “model” as architecture vs. metadata). The output is a structured JSON representation per model, containing cleaned text, token arrays, and mapped MCRO terms per section, which creates standardized inputs for downstream knowledge extraction.

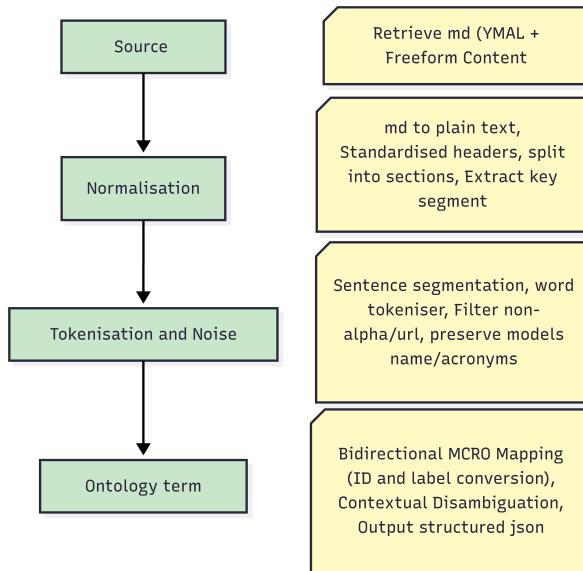


Figure 3.7: Preprocessing Pipeline of Ontology Mapper

### 3.5.3 Metadata Extraction and Prompting via LLM

The metadata extraction subsystem is implemented with Gemini 2.0 Flash to transform unstructured model card text into structured semantic representations. As depicted in Listing 3.1, this approach eliminates the need for task-specific training data by leveraging ontological grounding and rule-based constraints. The technical implementation utilizes Hugging Face’s ModelCard class for Markdown retrieval and Google’s generativeai module for API integration. Central to this architecture is the `get_mapped_triples()` function, which orchestrates a dual-context inference workflow:

- **Ontological Grounding:** The MCRO ontology provides semantic context, enabling Gemini to interpret metadata within a formal knowledge framework. This ensures that extracted information adheres to a standardized vocabulary, allowing for consistent RDF triple generation and accurate downstream querying [50].
- **Rule-Constrained Generation:** Thirteen deterministic rules are embedded in the LLM prompt to ensure semantic consistency and structural validity of the output. Key rules include:

- **Rule 1: Strict CURIE usage** - Enforces the exclusive use of MCRO ontology terms in CURIE format (e.g., `mcro:BiasMitigation`). This avoids semantic drift and ensures compatibility with the MCRO schema.
- **Rule 3: Structural integrity** Requires:
  - \* RDF classes to be explicitly typed using `rdf:type`.
  - \* Literal values (e.g., strings, metrics) to be expressed via `prov:hasTextValue`.
 This guarantees syntactic correctness and supports valid knowledge graph instantiation.
- **Rule 4: Taxonomic precision** Prohibits the instantiation of abstract MCRO classes (e.g., `mcro:MetadataSection`), ensuring that only concrete concepts are used in RDF output.
- **Rule 8: Identifier standardization** Enforces CamelCase for section identifiers (e.g., `DataLicense`, `ModelCardInfo`) to promote uniformity and improve query reliability.
- **Output Control:** The prompt explicitly includes directives such as “Return ONLY the JSON array” to suppress natural language completions. This ensures the LLM output remains clean, machine-readable, and directly parseable for RDF conversion.

The JSON triple definitions exhibit consistent schema adherence: license information maps to `mcro:LicenseInformationSection` (Rule 2), datasets follow `mcro:hasDataset`/`mcro:hasTrainingData` differentiation (Rule 13), and multi-value metrics merge into semicolon-delimited literals (Rule 7). Error analysis reveals that failure modes primarily involve underspecified architectural descriptions (e.g., “BERT variant” without versioning), which are mitigated through Rule 11’s descriptor prefixes [50].

---

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`
  - model detail - `mcro:ModelDetailSection`
  - limitations - `mcro:LimitationsInformationSection`
  - accuracy - `mcro:AccuracyInformationSection`
  - loss - `mcro:LossInformationSection`
  - metrics - `mcro:MetricsInformationSection`
  - training data - `mcro:TrainingDataInformationSection`
  - reference - `mcro:ReferenceInformationSection`
  - model card - `mcro:ModelCard`
  - model - `mcro:Model`
3. Use proper relationships:
  - `rdf:type` for types
  - `prov:hasTextValue` for textual values (e.g., `"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`, etc. Use CURIES instead.
6. Only the root model instance (e.g., `mcro:{clean_identifier(model_id)}`) should be assigned `rdf:type` `mcro:Model`.
  - DO NOT assign `rdf:type` `mcro:Model` to supporting entities like `mcro:{clean_identifier(model_id)}_ModelDetail`
  - Use specific types like `mcro:ModelArchitectureInformationSection` instead.
7. Merge multiple values using commas or semicolons in `prov:hasTextValue`.
8. Use CamelCase for instance identifiers (e.g., `mcro:{id}-LimitationsInformationSection`).
9. For dataset mentions, include a `mcro:DatasetInformationSection` and link via `mcro:hasDataset`.
10. If a section exists without clear text, include `prov:hasTextValue` with ‘Not specified’ or similar.
11. Use prefixed descriptors in `prov:hasTextValue` (e.g., `"Sample count: 80,000 images"`).
12. Avoid redundancy between sections. Reference instead of duplicating text.
13. Use `mcro:hasDataset` only for named datasets. Use `mcro:hasTrainingData` for descriptions.

Sample Output Format:

```
[  
 {  
   "s": "mcro:{clean_identifier(model_id)}",  
   "p": "rdf:type",  
   "o": "mcro:Model"  
 }]
```

```
"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}

---

Listing 3.1: Final Prompt - RDF Triple Extraction with MCRO Ontology

### 3.5.4 RDF Triple Generation

The RDF generation subsystem transforms JSON triple definitions into a queryable knowledge graph through the `convert_json_triples_to_turtle()` function. This process implements a structured five-stage materialization protocol, designed to ensure syntactic correctness, semantic validity, and structural consistency of the resulting RDF representation. Each stage is outlined below:

1. **Graph Initialization:** An empty RDFLib `Graph` object is instantiated as the container for all RDF triples. This stage also involves binding standard prefix mappings using the `prefix_map`. These prefixes (e.g., `rdf`, `prov`, `mcro`) enable CURIE-based triple construction and compact serialization. Namespace binding at this early stage ensures that all subsequently added triples will adhere to a consistent URI scheme, critical for ontology-aligned knowledge graph generation.

---

```
g = Graph()
for prefix, uri in prefix_map.items():
    g.bind(prefix, Namespace(uri))
```

---

2. **CURIE Expansion:** To support compact representation, the JSON inputs define RDF terms using CURIEs (e.g., `mcro:Model`). The expansion function converts each CURIE to a full URIRef by joining the prefix URI with the local identifier. This step is essential for generating globally unique RDF identifiers, while preserving readability and compliance with the ontology vocabulary.

---

```
def expand(curie):
    prefix, local = curie.split(":", 1)
    return URIRef(prefix_map[prefix] + local)
```

---

3. **Triple Instantiation:** The core logic parses each JSON triple and translates it into an RDF statement. The process includes:

- **Literal assignment:** Predicates such as `prov:hasTextValue` indicate that the object should be treated as a literal value.
- **Resource linking:** All non-literal objects are expanded into URIRefs using the CURIE expansion logic.
- **Blank node generation:** For nested model card sections or non-identifiable entities, blank nodes are created to preserve graph structure without requiring global identifiers.

This ensures semantic accuracy and prevents ontology violations during materialization.

- 4. Turtle Serialization:** Once all triples are instantiated, the graph is serialized to Turtle (.ttl) format. This human-readable format supports long-term storage, inspection, and compatibility with tools like GraphDB or Blazegraph. Encoding is handled to ensure UTF-8 compliance, allowing special characters and multilingual content to be preserved.

```
turtle_data = g.serialize(format="turtle", encoding="utf-8")
with open(ttl_path, "w") as f:
    f.write(turtle_data.decode())
```

- 5. Validation Framework:** Validation Framework: To ensure quality and correctness, the system implements a multi-layered validation pipeline. First, **syntactic validation** is performed using JSON parsing to verify that the input structure conforms to the expected format before the RDF transformation begins. Next, **logical validation** is enforced through prompt-injected rules or example, disallowing the instantiation of abstract classes and requiring the use of CURIEs to maintain ontological correctness and compliance with MCRO constraints. Finally, **structural validation** ensures consistent identifier schemes by enforcing prefix mappings and controlled URI construction, thereby eliminating ambiguity and enhancing SPARQL queryability across the knowledge graph. These checks serve to minimize downstream reasoning errors and improve the quality of the resulting semantic graph.

After the RDF triples were extracted from the processed model card metadata in JSON format, they were automatically converted into the Turtle (.ttl) serialization using the RDFlib library. These .ttl files conform to the MCRO ontology and serve as the final output of Pipeline 1 in Figure 3.8. The extracted triples were evaluated against a manually annotated ground truth dataset and achieved an extraction accuracy of 99%, demonstrating the effectiveness of the LLM-based metadata mapping approach. A small number of expected triples were not extracted, primarily due to inherent limitations in the LLMs ability to infer implicit metadata or handle edge-case phrasings. Nevertheless, the overall fidelity remained high. The resulting .ttl files were then passed as input to Pipeline 2: KGQA, where they were further processed. A comprehensive listing of all extracted RDF triples both in JSON and Turtle format is included in Appendix .1 for reference and reproducibility.

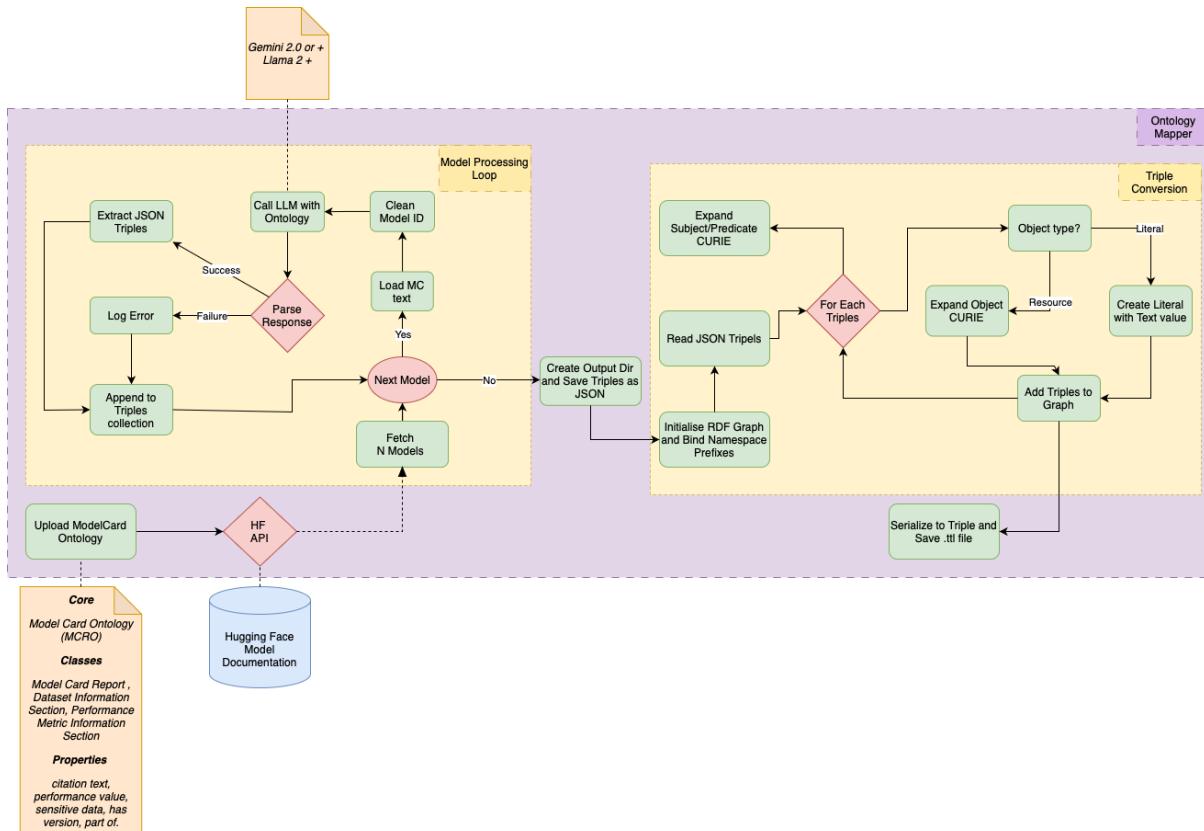


Figure 3.8: Ontology Mapper - ModelCards to KG

### 3.5.5 Prompting Strategies

A variety of prompts were designed and iteratively tested to evaluate different metadata extraction strategies from HF model cards. These prompts were categorized based on their extraction style, ranging from simple explicit extractions to ontology-grounded RDF triple generation. The goal was to understand the capabilities and limitations of different prompting paradigms in structuring metadata in a semantically meaningful, machine-consumable format. The design of these prompts was guided by the need to balance simplicity, accuracy, and semantic alignment. By exploring different approaches, the objective was to identify the most efficient strategy for extracting structured metadata that could be further integrated into automated systems for knowledge graph construction, ensuring scalability and consistency across diverse model cards. These prompt techniques were inspired by Ahmed et. al.(2025), [1], Sahbi et. al (2025) [95] and Saetia et. al (2025) [93], which laid the groundwork for understanding how to structure and map metadata within semantically rich ontologies. These evaluation criteria were selected to ensure that the extracted metadata met the projects primary goals: providing structured, machine-readable output that is consistent with the MCRO ontology and capable of seamlessly feeding into an automated knowledge graph pipeline. Adherence to MCRO ensures that the metadata aligns with established semantic frameworks, while the focus on structure and suitability for graph construction addresses the projects need for integration and scalability.

Below are the three major prompts that were initially considered, followed by a discussion of the final selected prompt and why it best met the requirements of the project. The following prompts were carefully chosen to test different strategies, allowing for an in-depth evaluation of various methods to identify the most effective one for automating model metadata extraction. Each approach provided insights into the strengths and weaknesses of different strategies, informing the decision on which prompt would best align with the projects objectives [12][111].

#### Prompt 2: Schema-Based Extraction with SPDX Validation

This prompt instructed the LLM to extract key metadata (e.g., model name, license type, architecture) using a predefined schema. A key feature was the validation of license types using the SPDX standard<sup>9</sup>. The output was expected in structured JSON format, enforcing constraints such as correct SPDX IDs for licenses, as shown below in Listing 3.2. The use of a predefined schema ensures that the extracted metadata is consistent and structured, facilitating the integration of this data into downstream systems like GraphDB<sup>10</sup>, which is used for constructing automated knowledge graphs.

---

```
Extract model metadata as JSON with VALID SPDX LICENSE IDS:
```

```
Output Format:  
{  
    "Model name": "string",  
    "License type": "string (SPDX ID)",  
    "Architecture": "string",  
    "CO2 emitted": "float|null",  
    "Training data sources": "string",  
    "Datasets": ["string"],  
    "Languages": ["string"],  
    "Metrics": ["string"],  
    "Base model": "string",  
    "Pipeline tag": "string",  
    "Library name": "string"  
}  
  
Model Card Text:  
{text[:10000]}
```

---

Listing 3.2: Prompt - Schema-Based Extraction with SPDX Validation

#### Why include:

<sup>9</sup>The SPDX specification is an open standard for communicating bill of materials (BOM) information across multiple topic areas [105].

<sup>10</sup>GraphDB is an enterprise-ready Semantic Graph Database that follows W3C standards. Semantic graph databases (also known as RDF triplestores) serve as the foundation for solutions that prioritize modeling agility, data integration, relationship exploration, and cross-enterprise data publishing and consumption [85].

- Structured Metadata Extraction: By using a clear schema, the model produced consistent and repeatable JSON output with recognizable fields.
- License Validation: Ensuring SPDX compliance helped enforce data quality in the license field, a critical attribute for downstream applications.
- Progressive Complexity: This prompt introduced a more formal approach than basic entity extraction, showcasing the importance of guiding the model with structured expectations.

**Why Exclude:** Despite its structured format, this prompt lacked semantic grounding in ontologies. It operated purely at the schema level and did not map any extracted fields to ontology-based identifiers or relationships (e.g., CURIEs). As such:

- The output was in unstructured JSON format, not RDF triples.
- MCRO concepts were absent, making it challenging to integrate into a knowledge graph without extensive post-processing.
- The focus on license validation, while useful, applied to only a narrow slice of metadata and did not generalize to other fields.

While valuable in demonstrating schema-based validation, this prompt was ultimately rejected due to its lack of ontology grounding and incompatibility with the projects goal of directly generating RDF triples aligned with MCRO semantics.

### Prompt 3: Validation-Heavy Structured Extraction

This prompt was designed to extract only verifiable and validated fields strictly. It incorporated hard rules, such as ensuring numerical values had units (e.g., CO2 emissions must be in kg) and rejecting unverifiable or ambiguous claims from the model card, as shown below in Listing 3.3. The primary goal of this structured extraction was to ensure that only reliable and consistent information was captured from the model card, adhering to strict validation rules to ensure data integrity and accuracy. The use of validation-heavy rules was particularly important for this thesis, as the aim is to extract model metadata in a highly reliable and consistent manner for building an accurate and semantically valid knowledge graph. By enforcing constraints such as ensuring numerical values include units (e.g., for CO2 emissions) and rejecting unverifiable claims, this prompt structure ensures that only verifiable and precise information is included in the metadata, which is crucial for the semantic accuracy of the extracted data.

---

[STRICT MODE] Extract ONLY these verified fields from the text. Skip unverified information.

REQUIRED FIELDS:

1. License (SPDX format)
2. Model Architecture
3. Training Data Sources
4. CO2 Emissions (kg)

VALIDATION RULES:

- Must appear in ## Metadata section
- Must match Hugging Face conventions
- Numerical values require units
- Reject unverifiable claims

Text:

{text[:10000]}

---

Listing 3.3: Prompt - Validation-Heavy Structured Extraction

### Why Include:

- Strict Validation: This approach was ideal for ensuring high trust in the data being extracted. It enforced consistency and correctness, particularly in sensitive fields like energy usage.
- Rejection of Unverifiable Claims: It actively filtered out low-quality or unverifiable metadata, favoring quality over quantity.

- Structured Quality Control: Showcased a rigorous extraction process focused on validated fields only.

#### Why Exclude:

- This prompt, while excellent from a quality assurance perspective, did not align with the requirements of semantic knowledge graph construction:
- It prioritized correctness over completeness, potentially discarding fields that were partial or uncertain.
- It outputs JSON, not RDF triples, and lacks mappings to MCRO concepts.
- As in Prompt 2, ontology grounding was missing, and rejected data points were not tagged or contextualized (e.g., as unknown or refer to model card).

Although robust for data validation tasks, this strategy was incompatible with the goal of generating comprehensive RDF triples. The ontology gap and loss of partial metadata led to its rejection.

#### Prompt 4: Ontology-Based Extraction with RDF Triples

This prompt took a semantic approach to extraction by instructing the model to generate RDF triples using existing ontologies (e.g., MCRO, IAO<sup>11</sup>, BFO<sup>12</sup>). Metadata elements like model name, license, and dataset were mapped to ontology concepts and expressed using CURIE<sup>13</sup> syntax (e.g., mcro:ModelNameSection) as shown in Listing 3.4. This approach ensures that the extracted metadata is semantically rich, adhering to well-defined concepts within established ontologies.

The ontology-based extraction method was chosen for its ability to generate structured, machine-readable metadata that is inherently suitable for integration into a knowledge graph. By using RDF triples, the extracted metadata can be easily mapped to standardized ontology concepts, enabling precise relationships to be established between entities in the knowledge graph. By expressing metadata in RDF triples, this prompt also facilitates the interoperability of the extracted data with other systems and ontologies, which is vital for creating an extensible and reusable knowledge graph. GraphDB can then be used to store, query, and visualize the RDF triples, enabling efficient retrieval of model information and supporting SPARQL queries for advanced querying capabilities.

---

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 the 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",  
 }
```

---

<sup>11</sup>The Information Artifact Ontology is a new ontology for information entities that was inspired by work done by the OBI digital entity and realizable information entity branches [18, 81].

<sup>12</sup>The Basic Formal Ontology (BFO) is a small upper-level ontology designed to support information retrieval, analysis, and integration in scientific and other domains [8].

<sup>13</sup>Compact URI Reference (CURIE) is a shorthand notation for representing URIs in a more compact form using prefixes (e.g., mcro:ModelNameSection instead of the full URI) [10].

```

    "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/micro#
- For literal values, use prov:hasTextValue
- No explanation or markdown
- Keep all responses strictly within JSON format

```

Listing 3.4: Prompt - Ontology-Based Extraction with RDF Triples

**Why Include:**

- Semantic Structure: This prompt demonstrated the potential for extracting deep semantic metadata and translating it directly into knowledge graph formats.
- Advanced Representation: RDF triples inherently support complex relationships and are better suited for downstream querying and integration tasks.
- Ontology Mapping: It illustrates how domain ontologies can guide the generation of structured metadata.

**Why Exclude:** Despite its alignment with the knowledge graph approach, this prompt failed to enforce the correct use of MCRO-specific constraints:

- The model produced generic ontology terms, including instances like swo:SWO\_0000002 or abstract annotations such as obo:IAO\_0000310, which are not valid instance properties for metadata.
- obo:IAO\_\* terms are annotations used for describing properties, not instances, and should not appear as such in outputs.
- The prompt allowed ambiguous ontology mappings, missing identifier formatting, and inconsistent RDF rules (e.g., missing prov:hasTextValue).
- It also failed to restrict the usage of abstract or upper-level ontologies (e.g., BFO), which led to semantic drift and invalid triples.

While this RDF-based prompt was closer to the desired strategy, its lax enforcement of MCRO constraints, incorrect use of annotation properties, and allowance for vague or overly abstract ontological terms made it unsuitable. Only obo:MCRO\_\* terms\* should be used to extract true instance properties. These represent the actual object properties relevant to our task, unlike obo:IAO\_\*, which are annotations. We fixed this issue in the Final prompt by using *mcro:LicenseInformationSection* instead of *mcro:HasLicense*, which is what is specified in the MCRO Turtle file. This misinterpretation by Gemini, which used IAO classes like swo:SWO\_0000002 (software), has now been corrected. Therefore, this prompt was also rejected, and the issue related to the incorrect use of annotation properties has been fixed in the Final prompt.

**Final Selected Prompt: Ontology-Grounded RDF Triple Generation (MCRO-Enforced)**

The final and most effective prompt was explicitly designed to generate RDF triples grounded in the MCRO ontology by uploading an actual file to Gemini, as Gemini 2.0 Flash version does not recognize file URLs. It enforced strict rules around CURIE usage, class-instance mapping, identifier formatting, and output structure [50]. This approach was chosen because it directly addressed the key issues encountered in earlier prompts, including the improper use of annotation properties, vague ontological terms and incorrect triples extraction. By grounding the prompt in the MCRO ontology and using a structured RDF-based approach, it ensured that the metadata extraction was precise, consistent, and aligned with the projects semantic goals. The decision to upload the actual MCRO ontology file to Gemini was critical for ensuring that the system properly understood the ontologys structure and could accurately map model metadata to the appropriate CURIEs. According to forums, it seems that Gemini 2.0 Flash doesnt read links, which made this approach necessary. This method eliminated the misinterpretations

that had occurred when using indirect references or misaligned terms. Moreover, enforcing strict rules regarding CURIE usage, class-instance mapping, and identifier formatting ensured that the generated RDF triples were machine-readable, consistent, and ready for integration into the knowledge graph. This is particularly important for automating the extraction and structuring of metadata in a format that is semantically meaningful and suitable for further processing, as required by the thesis framework.

Why It Was Chosen:

- **Ontology-Aware:** The prompt strictly constrained the LLM to use only MCRO-defined object properties and classes, aligning the extraction process with a formal semantic model.
- **Correct Use of RDF Properties:** Properties like `rdf:type`, `prov:hasTextValue`, and `mcro: hasLicenseInformationSection` were enforced, ensuring semantic precision and machine readability.
- **Reusable Output:** The models output was JSON-encoded RDF triples, clean, consistent, and directly parseable by downstream tools (e.g., RDFLib, GraphDB).
- **Coverage of Edge Cases:** It handled multi-valued fields, merged split sections, formatted identifiers using naming conventions (e.g., CamelCase), and ensured fallback strategies like tagging unknown fields.
- **Designed for Knowledge Graph Construction:** This prompt directly supported the primary goal of building a structured, queryable metadata knowledge graph with full MCRO integration.

### 3.5.6 Evaluation Metrics and Process

The quantitative analysis approach is based on the study KOnPoTe vs. Claude by Sahbi et al. [95], while the qualitative analysis approach follows the methodology outlined in Lehmanns (2023) paper [52]. These methodologies were selected to ensure a rigorous and comprehensive evaluation of the data, combining both numerical precision and in-depth qualitative insights.

The evaluation of the Text to KG system focuses on measuring both the accuracy and structural validity of RDF triples generated from HF model cards, using a pipeline grounded in the MCRO. Two datasets were used: a 10-model dataset, including `Falconsainsfwimagedetection`, `allMiniLML6v2`, `YOLOv8DetectionModel`, and `resnet50a1in1k`, which consisted of 315 manually annotated RDF triples, and a broader 100-model dataset used for large-scale estimation and generalizability checks. The purpose of the evaluation is to determine the systems ability to extract relevant, correct, and ontologically valid metadata triples from semi-structured and unstructured sources.

For the 10-model dataset, formal evaluation metrics were computed against the ground truth. These metrics include precision, recall, and F1-score, which are standard in information extraction tasks. Precision measures the accuracy of the triples generated by the system. It evaluates how many of the triples produced are correct compared to the total number of triples extracted. In this case, precision is concerned with the relevance of the information, meaning it assesses how many of the generated triples are valid and accurate in the context of the model cards., i.e.,

$$\text{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

where TP represent the triples that were correctly extracted by the system. and FP represent the triples that were mistakenly generated but were not part of the ground truth. A **precision score of 89.84%** was obtained, indicating that 283 of the 315 system-generated triples matched the reference triples, while 32 were incorrectly predicted (false positive). These false positives were typically due to the system inferring architecture or dataset details not explicitly stated in the model card, suggesting a tendency toward overgeneration in ambiguous contexts.

Recall is a metric that evaluates the systems ability to correctly retrieve all relevant triples from the model cards. It is particularly important in tasks where missing relevant information is detrimental, such as extracting metadata that is critical for understanding a models license, architecture, and other key details. Recall focuses on identifying how well the system captures all expected or true triples (i.e., the ground truth). It is defined as the ratio of correctly extracted triples to all expected (ground truth)

triples. In short, it measures the proportion of actual triples that were retrieved by the system. The formula is:

$$\text{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

With FN being the number of false negatives, i.e., ground truth triples that the system does not predict. A **recall of 89.84%** was achieved, showing that 283 of the expected 315 triples were retrieved. Missed triples frequently pertained to licensing terms, usage guidelines, or evaluation procedures, often embedded in non-standard textual phrasing.

The F1-score is a crucial metric in evaluating the overall performance of an information extraction system, as it balances both precision and recall. It is particularly useful when the data may suffer from class imbalances, such as when there are many more correct extractions than errors (or vice versa). By combining precision and recall into a single value, the F1-score provides a more holistic view of the systems performance, highlighting both the ability to correctly extract metadata (precision) and the ability to capture all relevant metadata (recall). The F1-score is the harmonic mean of precision and recall and is calculated as follows:

$$F_1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

yielding an **F1-score of 89.84%**. This indicates a strong balance between under- and over-generation, reflecting the systems reliability in identifying relevant structured metadata from unstructured input.

Structural quality was assessed using two additional metrics: **schema adherence and syntax correctness**.

Schema adherence is a crucial metric that evaluates whether the extracted triples align with the structure and constraints defined by the ontology, in this case, the MCRO ontology. The objective is to ensure that the system generates structured metadata that conforms to the expected schema, ensuring consistency and compatibility with the ontology. Schema adherence measures the percentage of correctly structured triples relative to all triples extracted by the system. In the context of this work, it means ensuring that each extracted triple aligns with the structure, class, and properties defined by the MCRO ontology.

$$\text{Schema Adherence (\%)} = \frac{\text{Conforming Triples}}{\text{Total Triples}} \times 100 = \frac{299}{315} \times 100 \approx 94.92\%$$

For the 10-model dataset, 299 out of 315 triples conformed to the MCRO ontology, resulting in **95% schema adherence**. The 5% of non-conforming triples involved incorrect use of ontology-defined predicates or inappropriate subject-object relationships, such as assigning datatype properties to object URIs or confusing class-property domains.

Syntax correctness pertains to the technical validity of the RDF triples representation, independent of their semantic content. It ensures that the triples are formatted correctly according to RDF serialization standards, such as the Turtle format. This includes the use of proper prefixes, valid URIs, non-redundant subject-predicate-object relationships, and adherence to RDF syntactic rules.

Out of 315 triples, 299 were syntactically correct, yielding **95% syntax correctness**. The 16 incorrect triples exhibited issues such as malformed identifiers, repeated predicates for the same subject-object pair, or namespace errors.

$$\text{Syntax Correctness (\%)} = \frac{\text{Syntactically Valid Triples}}{\text{Total Triples}} \times 100 = \frac{299}{315} \times 100 \approx 94.92\%$$

Another key metric, Triple completeness is an important metric that measures the extent to which the system extracts all the expected metadata types defined in the ground truth. Unlike recall, which focuses on the correct extraction of individual triples, completeness evaluates how well the system covers all necessary metadata fields, regardless of whether the specific instances match. It is defined as the proportion of expected metadata fields that were successfully extracted from the input data, and is calculated as:

$$\text{Triple Completeness (\%)} = \frac{\text{Fields Extracted}}{\text{Fields in Ground Truth}} \times 100 = \frac{268}{315} \times 100 \approx 85.08\%$$

Based on coverage analysis, the system achieved **85% Triples completeness**, primarily missing metadata categories related to limitations (`micro:hasLimitation`) and hyperparameters (`micro:hasHyperparameter`). These omissions were largely due to the free-form nature of such content and inconsistent terminology in the source documents.

To complement the quantitative metrics, a qualitative expert evaluation was conducted by two experts in ontology and knowledge graphs. The two semantic Web experts were asked to review the semantic correctness, structural quality, and interpretability of the generated RDF triples in comparison to the ground truth.

Expert 1 provided practical feedback on URI naming conventions. They pointed out the use of hyphens (-) in URIs such as `Falconsainsfwimagedetection-DatasetInformationSection`, cautioning that some triple stores (e.g., GraphDB) may encounter parsing or syntactical errors when handling such symbols. It was recommended to replace hyphens with underscores (\_) for broader compatibility. This issue has since been resolved in the updated generation prompts.

A detailed content-level comparison was conducted by Expert 2 between the LLM-generated triples and the ground truth. Their evaluation noted several key points, Missing Architecture Details, Omitted license information, performance metrics, associated model variants, omission of references and link. (1) Missing Architecture Details: The LLM often generated missing architecture details. For example, Vision Transformer was missing for `Falconsai/nsfw_image_detection`, and Microsoft MPNet was missing for `sentence-transformers/all-mpnet-base-v2`. (2) Omitted License Information: Conversely, license information was frequently omitted in the LLM-generated versions. This was observed in models such as `dima806/fairface_age_image_detection`, `openai/clip-vit-large-patch14`, and others. (3) Performance Metrics: In some instances, performance metrics in the LLM output were more verbose or up-to-date than those in the ground truth. For example, this was the case with `openai/clip-vit-base-patch32`. (4) Associated Model Variants: For models like `Bingsu/adetailer`, the LLM captured additional associated model variants. These included models such as YOLO face and hand models, which were not explicitly mentioned in the ground truth. (5) Omissions of References and Links: Occasionally, references and links (such as training dataset citations) present in the ground truth were not replicated by the LLM. However, this was deemed acceptable given the higher fidelity of the ground truth. The evaluation noted several differences, but these were largely considered acceptable. The expert acknowledged that some information gaps were expected, given the inherent variability and incompleteness of source model cards.

This expert evaluation confirms that the LLM-generated triples maintain a high degree of semantic coherence and are structurally sound, with differences primarily rooted in information availability or verbosity rather than correctness. These insights have informed updates to the prompting strategy and filtering mechanisms, enabling the capture of high-confidence, well-justified content more effectively.

The systems performance was tested on a broader dataset of 100 Hugging Face model cards, with 97 of these containing valid README content. Given the absence of a definitive ground truth for this larger dataset, precision and recall could not be computed directly. Instead, estimates were derived from structural properties and expert reviews. The methodology for evaluating the systems effectiveness was based on key metrics, such as schema adherence, syntax correctness, and triple completeness, aligned with the research questions.

The schema adherence dropped slightly to 88%, primarily due to the increased diversity in documentation styles across the 100 models. This drop is reflective of the model card structures varying more widely in the broader dataset. Methodologically, schema adherence was assessed by comparing the extracted RDF triples against the predefined schema used in the extraction process, with any structural mismatches

being flagged as errors. Syntax correctness across the 97-model dataset was estimated at 93%, slightly lower than the 10-model subset tested earlier. The decrease can be attributed to common errors like duplicated RDF statements and inconsistencies in namespace resolution, both of which were noted during expert review sampling. This aspect of the evaluation relied heavily on syntactical validation against the RDF format, with errors being categorized and reviewed based on their impact on the overall syntax. The triple completeness was estimated at 83%, a reflection of the broader distribution of model card structures and the less consistent documentation quality. Notably, models like llama-3, bloom, whisper, mistral-7b, and sdxl yielded a larger volume and variety of high-confidence triples. This result aligns with the methodology, where triple completeness was evaluated by assessing the volume of extracted triples that accurately represented the information in the model card, with higher-quality documentation leading to better extraction accuracy.

These results demonstrate the systems robustness and accuracy when applied to well-structured model documentation and its generalizability to a broader corpus, with acceptable degradation in structural metrics. The evaluation also highlights the impact of documentation quality and ontology coverage on system performance. Across both datasets, the system consistently maintained high structural validity and strong semantic alignment with MCRO, validating the effectiveness of the RDF generation pipeline built atop language models and ontology-driven design.

## 3.6 Pipeline 2: Model Search NL Queries to SPARQL

### 3.6.1 Objective and Scope

The objective of Pipeline 2 is to develop a robust and scalable natural language interface that enables users to perform semantic search over machine learning model documentation stored in a knowledge graph aligned to RQ2. Specifically, the system translates natural language questions into syntactically correct and semantically relevant SPARQL queries using a two-stage prompt-driven architecture, thereby facilitating intuitive access to structured metadata encoded using the MCRO.

This pipeline supports the broader thesis goal of by allowing both technical and non-technical users to query large-scale RDF knowledge graphs without requiring prior knowledge of SPARQL, RDF, or the underlying ontology schema.

The core scope includes:

- **Ontology-Constrained Query Generation:** The system assumes the presence of a structured schema based on the MCRO ontology and generates SPARQL queries compliant with its class-property design.
- **Two-Stage LLM Prompting Framework:** Inspired by recent work (e.g., Mountantonakis et al., 2023) [72], the system implements a two-stage prompting architecture:
  - Stage 1: Identifies relevant ontology path patterns using a constrained set of triple patterns.
  - Stage 2: Generates the final SPARQL query using the filtered set of relevant properties and classes derived in Stage 1.
- **Triplestore Compatibility:** Queries are designed for execution over a GraphDB and protege triplestore instance. While the current system runs over a local or hosted GraphDB with limited storage, the design is schema-centric and generalizable to distributed or sharded triple stores for large-scale datasets.
- **Scalability by Design:** Although the implementation currently targets manageable RDF data volumes, the pipeline is architected such that the same NL-to-SPARQL translation logic can be applied to terabyte-scale knowledge graphs with consistent schemas, assuming appropriate data partitioning and indexing on the backend.
- **Error Tolerance and Recovery:** The system includes iterative prompting and basic error handling to improve reliability, although advanced query correction or conversational repair is considered future work.

### 3.6.2 LLM Integration

In Pipeline 2, we utilized the same LLM as in Pipeline 1 to ensure a consistent and seamless process flow, minimizing the risk of discrepancies that could arise from employing different models. The primary goal was to maintain uniformity and automation across both pipelines, which is critical in ensuring that the entire system works together smoothly. By using the same model, we ensured that both pipelines follow a similar process for natural language understanding and SPARQL query generation, resulting in a more robust and reliable system. The LLM plays a central role in transforming natural language queries into structured SPARQL queries, which are subsequently executed against the knowledge graph in GraphDB. The prompt used for this task is meticulously structured to provide the necessary background, guiding the model toward generating accurate and effective queries. The prompt consists of several key elements: the MCRO ontology context, a set of few-shot examples, and the actual interactive user input query. By providing this structured input, we ensure that the model can infer the relationships and entities in the knowledge graph, making it more likely to generate accurate SPARQL queries. The entire architecture of this pipeline is shown in Figure 3.9, where the flow of the process is visually represented, from the initial user query to the final output of SPARQL query execution and result formatting which can help understand the Pipeline even better.

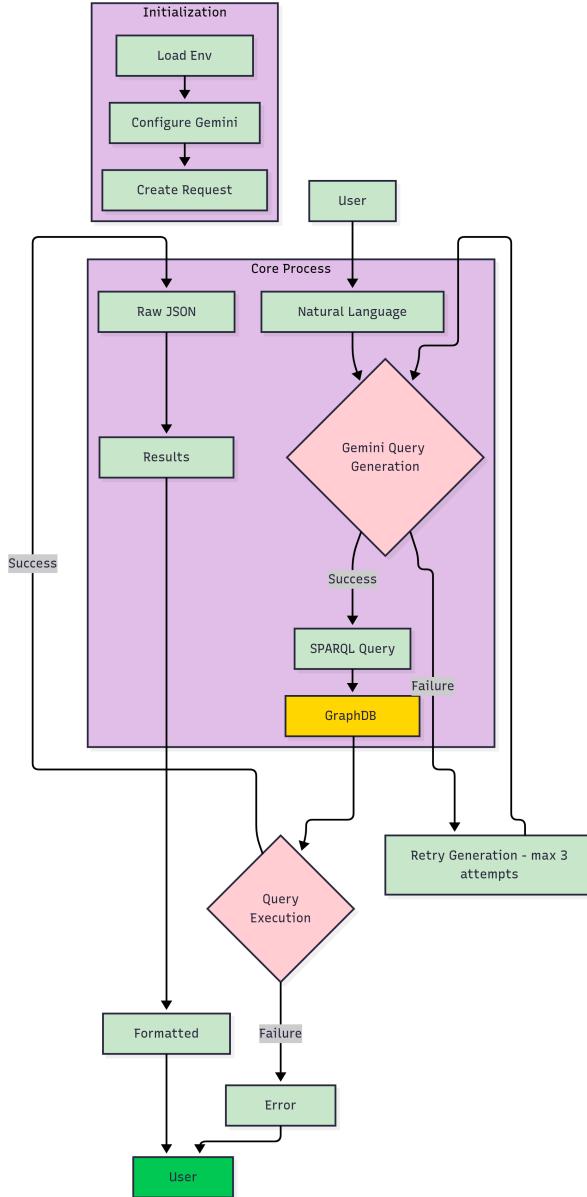


Figure 3.9: Knowledge Graph Question Answer Pipeline

The first component of the prompt, the **MCRO ontology context**, plays a crucial role in ensuring the LLM understands the semantics of the knowledge graph it is working with. The MCRO ontology includes relevant classes and properties such as mcro:hasUseCase, which links a model to its use case, and prov1:hasTextValue, which connects a use case to a specific description. By including these properties in the prompt, we guide the model to focus on the relevant aspects of the knowledge graph that are needed for generating the SPARQL query. To further enhance the models understanding, we also incorporate **few-shot examples** in the prompt shown in Listing 3.5. These examples demonstrate how to transform a natural language question into a SPARQL query, showing the LLM the specific structure it should follow. For instance, an example question like What are the use cases for the Falconsainsfwimagedetection model? is paired with its corresponding SPARQL query. This provides the model with concrete examples of what constitutes a valid query, reducing ambiguity and increasing the likelihood of generating correct SPARQL queries in response to user queries. These examples serve as a critical part of the LLMs training, helping it generalize the necessary patterns for query generation.

---

```
PREFIX mcro: <http://purl.obolibrary.org/obo/MCRO_>
PREFIX prov1: <https://www.w3.org/ns/prov#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
```

Convert the following question into a SPARQL query using the knowledge graph schema:

Known Model Properties  
 mcro:hasUseCase - links to use case information  
 mcro:hasTrainingData - links to training data details  
 mcro:hasModelArchitecture - links to architecture information  
 mcro:hasLimitation - links to limitations

Common Text Value Pattern  
 ?section prov1:hasTextValue ?text

Example:  
 mcro:Falconsainsfwimagedetection a mcro:Model ;  
 mcro:hasUseCase mcro:Falconsainsfwimagedetection-UseCase .

mcro:Falconsainsfwimagedetection-UseCase prov1:hasTextValue "NSFW Image Classification" .

Rules:  
 1. Always include the PREFIX declarations.  
 2. Use path patterns like:  
 ?model mcro:hasUseCase ?section .  
 ?section prov1:hasTextValue ?value .

Question: {question}

SPARQL:

---

Listing 3.5: Pipeline 2 : Final Prompt

The user input query itself is then processed and integrated into this structured prompt. For example, when a user asks about a models use cases or limitations, the LLM takes the users query and interprets it in the context of the MCRO ontology, using the examples and ontology elements to determine the most relevant classes and properties for constructing the query. The LLM then outputs a SPARQL query that is designed to extract the relevant data from the knowledge graph. In terms of the overall structure of the pipeline, the process begins when the user provides a natural language query. The LLM receives this query along with the MCRO ontology context and few-shot examples, then generates a corresponding SPARQL query. This generated query is automatically executed against the GraphDB endpoint, with the results retrieved and formatted for display. The system is designed to handle multiple user queries in an automated manner, allowing for an efficient, seamless user experience.

A key decision in this pipelines design was to retain the original prompt format rather than experimenting with different types of prompts. While multiple variations of prompts could be tested, the initial prompt consistently produced accurate and reliable results. This prompted us to focus on improving the existing prompt format rather than introducing additional complexity with new prompt types. By refining the original prompt and keeping the structure consistent, we were able to streamline the process and

maintain high-quality results without introducing potential errors or inconsistencies. The LLMs output is continually evaluated based on the success of the generated SPARQL queries, and this evaluation process ensures that the queries are both **syntactically correct and semantically meaningful**. Additionally, any failed queries are flagged for review, helping us further refine the prompt or model parameters. However, the decision to not explore many different types of prompts allows the system to remain simple and effective, minimizing complexity while maximizing the reliability of the SPARQL queries.

### 3.6.3 Evaluation Metrics and Process

The evaluation of Pipeline 2 was carried out through a structured process aimed at assessing the performance and correctness of the SPARQL queries generated by the LLM based question-answering system. The foundation on ground truth is inspired by Fu et. al. [29] in their IEEE paper Question Answering System for Knowledge Graphs on Datasets. In Section 4 they categorized natural language questions into three core types- **descriptive, comparative, and reasoning-based questions**, as outlined in Section 3.3.1 and shown in Appendix .3 These representative question types are also visually presented in Table 3.1, offering insight into the coverage and diversity of the query scenarios under test. Each question type was carefully selected to test different reasoning capabilities of the model, such as retrieving textual descriptions, comparing model attributes, or providing justifications for certain architectural or training decisions. The generated SPARQL queries were evaluated using three main criteria: **Schema Compliance Accuracy, Syntax Validity Rate, and Correct Output Rate**.

Schema Compliance Accuracy measures whether the structure and semantics of the generated SPARQL queries align with the MCRO ontology. This includes verifying the use of valid predicates (e.g., micro: hasUseCase, prov1:hasTextValue), correct entity referencing (e.g., micro:CLIP, micro:ESMFold), and properly typed RDF triples (e.g., ?section a micro:ModelArchitectureSection). A query was considered schema compliant if all of its predicates, classes, and instances existed in the MCRO ontology and were used in a semantically coherent manner. The Schema Compliance Accuracy (SCA) was computed using the formula:

$$\text{SCA} = \frac{\text{Number of Schema - Compliant Queries}}{\text{Total Queries Evaluated}} \times 100$$

Out of 12 evaluated queries, 9 adhered to the ontology structure, resulting in a **Schema Compliance Accuracy of 75%**. The remaining 25% failed due to issues such as referencing missing entities (e.g., RoBERTaLarge, DeBERTaDecodingenhancedBERTwithDisentangledAttention) or incorrect predicate mappings (e.g., retrieving License tags instead of LicenseText).

The Syntax Validity Rate (SVR) assessed whether the generated SPARQL queries were syntactically correct. This included checking for proper prefix declarations, balanced parentheses, clause order, and correct use of FILTER, OPTIONAL, and SELECT clauses. This is an important step to verify that the queries are not only semantically aligned with the ontology but are also structurally valid according to SPARQL syntax rules. The rate was calculated as:

$$\text{SVR} = \frac{\text{Number of Syntactically Valid Queries}}{\text{Total Queries Evaluated}} \times 100$$

In this evaluation, 11 out of 12 queries were syntactically valid, yielding a **Syntax Validity Rate of 91.7%**. The one invalid query involved a missing closing bracket in a FILTER clause used in a DeBERTa use case question. Another recurring issue was the overuse of FILTER(REGEX(...)), which sometimes led to false positives. For instance, a query attempting to find results for CLIP matched unrelated substrings.

The Correct Output Rate (COR) measured whether the execution of a query returned the expected answer as defined in the ground truth. A query could be syntactically valid and schema-compliant yet

still yield an incorrect result due to logical flaws or vague filtering criteria. This metric was calculated as:

$$\text{COR} = \frac{\text{Number of Queries Returning Correct Answers}}{\text{Total Queries Evaluated}} \times 100$$

Here, 8 of the 12 queries returned the expected results, producing a Correct Output Rate of 66.7%. Common causes of output errors included semantically vague filters, inadequate aggregation, and omission of relevant triples during query generation.

The qualitative analysis approach follows the methodology outlined in Lehmanns (2023) paper [52]. This methodologies were selected to ensure a rigorous and comprehensive evaluation of the data, combining both numerical precision and in-depth qualitative insights. An expert review was conducted by 2 Semantic web Experts, who provided additional qualitative feedback on the queries. While the majority were deemed well-formed, a few required optimization due to incorrect property targeting or ambiguous filtering. The expert suggested appending custom rules as a suffix in the prompt to handle complex queries more effectively. For example, the prompt can be structured as:

---

```
main_rules.prompt = "rules_for_prompt" + some_more_rules
some_more_rules = "input_for_complicated_cases"
```

---

This allows for modular prompt extension without disturbing the base instruction set, making the LLMs generation process more adaptable and scalable.

Table 3.1: Examples of Questions Used in KGQA

Question Type	Description	Example Questions
<b>Description</b>	Extract explicit facts about models, datasets, or metrics from model cards	<ul style="list-style-type: none"> <li>• What is the use case of DeBERTa-DecodingenhancedBERTwithDisentangledAttention?</li> <li>• What datasets were used to train RoBERTaLargeModel?</li> <li>• What tasks does CLIP support?</li> <li>• What is the architecture of deepseekr10528?</li> </ul>
<b>Comparison</b>	Compare models, datasets, or metrics	<ul style="list-style-type: none"> <li>• How does the use case of DeBERTa-DecodingenhancedBERTwithDisentangledAttention compare to that of ESMFold?</li> <li>• Which model has fewer than 1 million parameters?</li> <li>• How does the citation of HuggingFaceM4idefics28b compare to that of IDEAResearchgroundingdinobase?</li> <li>• How does the architecture of gpt2 compare to that of facebook-opt125m?</li> <li>• What are the differences in the use cases of vitmatte-model and llavah-flavav16mistral7bfh?</li> <li>• Which model has a more permissive license: FlagEmbedding or HuggingFaceH4zephyr7bbeta?</li> </ul>
<b>Reason</b>	Causal relationships or design choices	<ul style="list-style-type: none"> <li>• Why was the model architecture chosen for ESMFold?</li> <li>• What makes the dataset suitable for Fine-tunedT5SmallTextSummarizations use case?</li> <li>• What is the primary tradeoff of using LTXVideo?</li> </ul>

### 3.7 Pipeline 3: Graph RAG Pipeline for Model Search

### 3.7.1 Objective and Scope

The objective of the G-RAG pipeline is to evaluate the capabilities of a hybrid question-answering system that integrates semantic vector retrieval with Cypher-based graph traversal in a Neo4j knowledge graph. This system addresses RQ3: To what extent does G-RAG compare to SPARQL in answering AI model questions? The pipeline aims to assess performance, usability, and result relevance, particularly in facilitating access to AI model metadata for non-expert users. G-RAG combines the interpretability and structural precision of graph querying with the flexibility of language models to allow natural language interaction over formally structured data, similar to the knowledge-augmented generation approaches discussed by Liang et al. (2023) [56].

The pipeline leverages RDF triples generated in Pipeline 1 structured according to the MCRO ontology, which are converted into a property graph schema compatible with Neo4j. These triples are transformed into nodes and relationships through a preprocessing stage that maps RDF subjects and objects to graph nodes, assigns literals as node properties, and preserves predicate semantics as relationship types. The converted graph is ingested into a Neo4j database using Cypher scripts, forming the core knowledge representation. As shown in Figure 3.6, not all clusters in the resulting graph are connected. This is due to the nature of the model card RDF data: some model components (such as metrics, datasets, licenses, or hardware specifications) are described independently and are not explicitly linked via ontology-defined relations. These disconnected components reflect the modularity of model documentation, where some metadata segments (e.g., evaluation metrics or licensing details) are optional or specified in isolation, hence forming isolated subgraphs. A version of this graph with a starting node “UseCaseInformationSection”, providing a more detailed view of the clusters and their interconnections, is shown in Figure 3.10

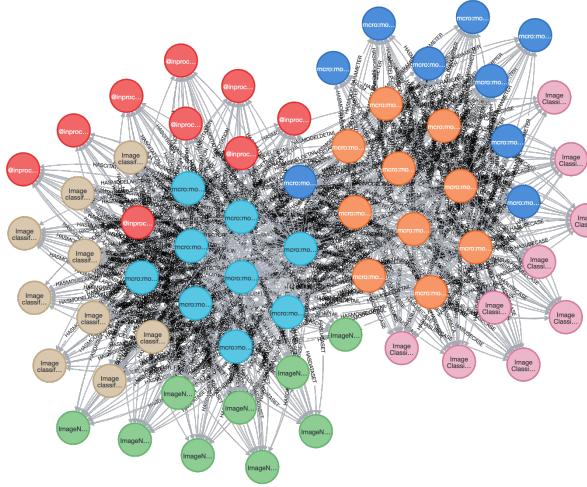


Figure 3.10: Neo4j Graph with Starting Node “UseCaseInformationSection”

The full pipeline, which details each of these processes from RDF triple generation to query execution and response synthesis, is illustrated in Figure 3.11.

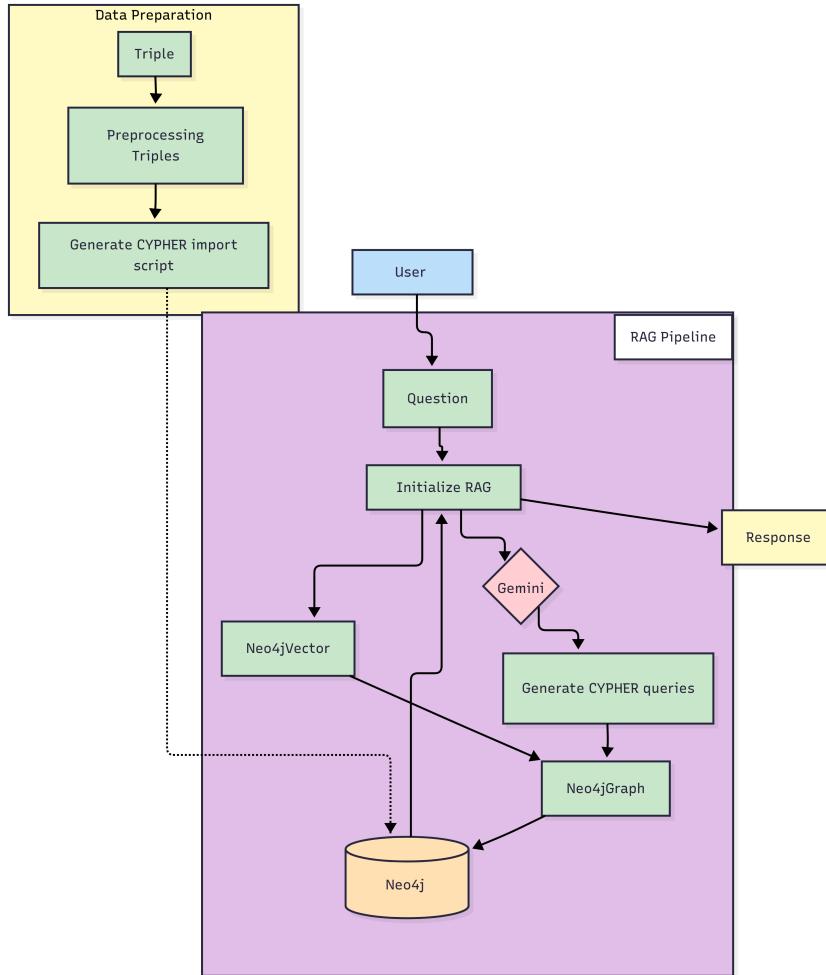


Figure 3.11: RAG-Neo4j Pipeline

To facilitate natural language querying, the pipeline integrates vector-based semantic retrieval with symbolic Cypher query generation. Node-level properties, such as descriptions or model names, are embedded using Gemini 2.0 Flash generated embeddings. These embeddings are indexed via **Neo4jVector**, enabling k-nearest neighbor search based on query similarity. This allows users to retrieve semantically similar nodes even if exact matches are unavailable. In parallel, the pipeline employs an LLM driven Listing 3.7 tailored for Cypher generation. This prompt instructs the model to extract the semantic intent of a question, identify relevant graph labels, and compose Cypher queries that leverage graph traversal patterns. It avoids hardcoded identifiers, encourages generality through label-based reasoning, and targets entities relevant to the MCRO schema (e.g., Model, Dataset, Metric, UseCase, etc.). These queries are executed over the Neo4j graph to return structured answers based on the knowledge graph topology.

Both retrieval paths vector and Cypher are integrated into a single retrieval-augmented agent, which interprets user queries, selects the appropriate querying method(s), and returns a synthesized result. This dual-query architecture ensures comprehensive coverage: Cypher queries capture explicit logical relations, while vector retrieval addresses implicit or fuzzy matches in user queries.

The construction of the G-RAG pipeline was achieved using Langchain, which offers modules for tool management, prompt templating, agent orchestration, and graph integration. Langchains support for Neo4j and vector stores was instrumental in implementing the dual-query architecture. However, several technical challenges were encountered during development. These included incompatibilities with Langchain versions, issues with Cypher syntax in multi-hop queries, and errors related to APOC (Awesome Procedures on Cypher)<sup>14</sup> procedures required for graph construction and querying. These were

<sup>14</sup>APOC is a Neo4j add on library, offering numerous procedures and functions to enhance functionality [75].

resolved through the installation of the APOC plugin, updates to Cypher queries to conform to Neo4js requirements, and aligning Langchain modules with compatible library versions.

Additionally, an early oversight in RDF triple handling led to the misinterpretation of literal values such as string descriptions or numerical scores as relationships instead of node properties. This was corrected by modifying the preprocessing logic to ensure that literals were attached as properties to their corresponding nodes, preserving the semantic integrity of the graph and enabling accurate information retrieval.

### 3.7.2 Preprocessing Knowledge Graph

The RDF triples extracted from Pipeline 1, structured as JSON serve as the foundational data for the knowledge graph population process. Prior to generating Cypher queries for Neo4j, these triples undergo a preprocessing phase to ensure syntactic and semantic consistency. All entity labels and values are cleaned to remove LaTeX formatting shown in Listing 3.6 , special characters, and inconsistencies that may interfere with Cypher compatibility. This step ensures that the resulting graph structure is both readable and machine executable.

---

```
text = re.sub(r'\\url\{(.*)\}', r'\1', text)
text = re.sub(r'\\', '', text)
text = re.sub(r'[\t]', '', text)
text = text.replace('\"', '\"')
```

---

Listing 3.6: Latex Formatting removal

During transformation, triples with a predicate denoting a type association (i.e., rdf:type) are interpreted as instructions to assign labels to corresponding subject nodes. In contrast, triples where the object is a resource URI are mapped as directed relationships between two nodes. Literal values, such as descriptions, names, or metrics, are instead incorporated as properties of the associated nodes. The result is a structured set of Cypher queries that creates and links nodes based on the underlying RDF semantics. For instance, a model entity becomes a node with type labels and metadata, while its training dataset is linked through a named relationship (e.g., TRAINED\_ON).

To enable natural language access to this structured graph, Multiple prompt was used where the Final prompt as depicted in Listing 3.7. This prompt informs a Gemini 2.0 Flash that it is operating as a Cypher expert interfacing with a Neo4j database. It explicitly outlines the internal graph schema, specifying that all nodes share the label Node, and are distinguished by a label property containing semicolon-separated type indicators (e.g., NamedIndividual;mcro\_Model). It highlights that model nodes are identified by the presence of 'mcro\_Model' in their label and emphasizes the importance of querying by this property rather than by fixed graph labels (e.g., avoiding direct use of :Model) [117].

---

You are a Cypher expert working with a Neo4j knowledge graph.

All entities are represented as nodes with the label 'Node'. Each node has:

- id: a unique model identifier (e.g., "mcro\_resnet50ain1k")
- label: a semicolon-separated string of type tags (e.g., "NamedIndividual;mcro\_Model")
- embedding: a vector for similarity search

**Important:**

- The `label` property is crucial for identifying the type of node.
- Model nodes are identified when the `label` contains the string 'mcro\_Model'.
- Do not use a fixed label like ':Model' in your queries.

To retrieve all models, use the following structure:

```
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: {query}

Cypher Query:

---

Listing 3.7: Pipeline 3 - Final Prompt

The prompt provides an example Cypher query and instructs the model to return queries that match its structure, aligning with the format used in the graph. This design enables the seamless translation of natural language questions, such as Which models are trained on dataset X? into Cypher queries that can extract relevant nodes and relationships. In doing so, the system abstracts away the complexity of query formulation, allowing users to interact with the knowledge graph through intuitive language, while maintaining precise and type-aware query execution underneath.

### 3.7.3 Evaluation Metrics and Process

To rigorously assess the performance of the RAG pipeline, we utilized the same twelve evaluation questions employed in Pipeline 2, as outlined in Table 3.1). These questions were strategically categorized into three semantic types: Description, Comparison, and Reason, and then further evaluated with the ground truth. This typology ensured comprehensive coverage of metadata-related information needs relevant to Hugging Face model cards, including functional capabilities, architectural traits, licensing details, and citation information.

The evaluation framework was built around three key metrics: Schema Compliance Accuracy (SCA), Syntax Validity Rate (SVR), and Correct Output Rate (COR). These metrics were designed to assess not only the factual correctness of responses but also their structural and semantic alignment with the MCRO ontology [95] [52].

For **Schema Compliance Accuracy (SCA)**, RAG achieved a score of 75%, with 9 out of 12 responses adhering either fully or partially to the MCRO schema. Within this, 8 responses (66.7%) were entirely correct and correctly mapped to the ontology-aligned metadata roles, while 1 response was partially correct due to schema drift, such as answering a use case question using architectural data. 3 responses (25%) were schema-incompliant due to retrieval inaccuracies or mismatched semantic grounding.

The **Syntax Validity Rate (SVR)** was 100%, as all responses generated by the RAG pipeline were grammatically well-formed and linguistically fluent. This suggests strong underlying language modeling performance, even in cases where factual accuracy was compromised.

The **Correct Output Rate (COR)**, RAG achieved 66.7%, where 8 of the 12 responses were factually and semantically accurate. 2 responses were partially correct, for instance, when the model retrieved information about a similarly named model, leading to contextual ambiguity. The remaining 2 responses were incorrect due to either hallucination or retrieval from irrelevant sections of the knowledge base.

A breakdown by question type revealed that RAG performed best on Description-type questions, which typically involved direct retrieval of clearly defined metadata such as model use cases, dataset names, or license types. Performance was moderate on comparison-type questions, where the system occasionally failed to effectively align or contrast metadata between two distinct models. The most challenging were Reason-type questions, which required justification or rationale (e.g., architectural design decisions or dataset suitability). These necessitate deeper inferencing and multi-source contextual synthesis, where RAG currently shows limitations due to the absence of long-context reasoning capabilities and ontology-guided retrieval strategies.

# Chapter 4

## Results

This chapter presents the empirical findings from the evaluation of an LLM-based pipeline designed to extract and structure model metadata from Hugging Face model cards. The primary focus is on the systems ability to generate RDF triples that adhere to the MCRO ontology and the accuracy of SPARQL queries generated based on these triples. A dataset comprising 10 model cards was utilized to assess the effectiveness of the pipeline across various evaluation metrics, including precision, recall, F1-score, schema adherence, syntax correctness, and completeness.

The chapter begins by detailing the results of the precision and recall evaluations, providing insight into the systems ability to correctly extract relevant metadata and its tendency towards over- or under-generation. The analysis then moves to the examination of the pipelines schema compliance accuracy, evaluating how well the generated RDF triples align with the MCRO ontology. Syntax validity is also explored, focusing on the technical correctness of the RDF serialization.

In addition to these core metrics, the chapter also covers the completeness of the metadata extraction, highlighting the specific metadata categories that were either successfully or unsuccessfully retrieved. The impact of key system adjustments, such as the incorporation of the MCRO ontology file into the extraction process, is also discussed. Through these evaluations, this chapter provides a comprehensive understanding of the pipelines performance and identifies areas for improvement in future iterations.

### 4.1 RQ1: Impact of LLM-Based Ontology Extraction on Model Card Metadata Quality

This section presents findings addressing RQ1: **To what extent can LLMs for ontology-driven metadata extraction improve the consistency, completeness, and machine-readability of model card information?** The results, derived from Pipeline 1, were obtained by analysing two datasets: a 10-model dataset with 315 expert-annotated ground truth triples and a 100-model dataset for structural estimation. Quantitative metrics (**precision, recall, F1-score, schema adherence, syntax correctness, triple completeness**), defined in Equations 4.1 to 4.3 and summarized in Table 4.2, quantify improvements in consistency, completeness, and machine-readability. Qualitative insights from expert feedback highlight factors limiting completeness. The findings demonstrate significant advancements in metadata structuring, with challenges in completeness due to linguistic variability, which directly addresses RQ1.

#### 4.1.1 Quantitative Evaluation on 10-Model Dataset

The 10-model dataset (315 ground truth triples) provides evidence of the pipelines impact on RQ1s objectives. The metrics are defined as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (4.1)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (4.2)$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4.3)$$

where TP is true positives (correct triples), FP is false positives, and FN is false negatives. The pipeline

achieved a **precision** of 89.84% (283/315, 32 false positives from over-inferred architectures/datasets, e.g., Vision Transformer for `Falconsai/nsfw_image_detection`), supporting high consistency. The **recall** of 89.84% (283/315, missing non-standard licensing/use case triples) indicates robust but incomplete extraction. The **F1-score** (89.84%) balances these aspects, as shown in Table 4.1.

**Schema adherence** (95%, 299/315 triples) ensures consistent, machine-readable MCRO compliance, with 16 violations from domain/range errors. **Syntax correctness** (95%) confirms machine-readability, with errors from malformed URIs or repeated predicates. **Triple completeness** (85%) reflects omissions in `mcro:hasLimitation` and `mcro:hasHyperparameter` due to free-form text, limiting full completeness. These results demonstrate strong consistency and machine-readability, while highlighting challenges to completeness, which directly address RQ1.

Table 4.1: Quantitative Results Addressing RQ1

Metric	10-Model Dataset	100-Model Dataset	RQ1 Implication
Precision	89.84% (283/315)		High consistency; over-inference errors
Recall	89.94% (283/315)		Robust but incomplete extraction
F1-score	89.84%		Balanced consistency/completeness
Schema Adherence	95% (299/315)	88%	Strong consistency, machine-readability
Syntax Correctness	95%	93%	Ensures machine-readability
Triple Completeness	85%	83%	Limited by free-form text

#### 4.1.2 Expert Review and Qualitative Assessment

Qualitative feedback from Expert 1 reveals patterns affecting RQ1's objectives, as summarized in Table 4.2. A prevalent issue is the omission of licensing information in LLM-generated triples, observed in seven models (e.g., `Falconsai/nsfw_image_detection`, `google-bert/bert-base-uncased`), which reduces completeness by excluding critical metadata. In the case of `dima806/fairface_age_image_detection`, the dataset section is not found on Hugging Face but on Kaggle. The training data was likely inferred from the dataset section on Hugging Face.

The pipeline enhanced completeness by inferring details absent in the ground truth. For `Falconsai/nsfw_image_detection`, it correctly identified the Vision Transformer architecture, and for `timm/mobilenetv3_small_100.lamb_in1k`, it provided detailed training data. However, such inferences risk inconsistency if inaccurate. Structured information extraction was a strength, with the pipeline accurately capturing performance metrics for `openai/clip-vit-base-patch32` (e.g., accuracy greater than 96%) and YOLO models under `Bingsu/adetailer` (mAP 50 = 0.669–0.810), supporting both completeness and machine-readability.

In contrast, for `openai/clip-vit-large-patch14`, the ground truth was more verbose in training data and use cases, suggesting expert annotations include external knowledge, which complicates completeness comparisons. Experts 2 recommended replacing hyphens with underscores in URIs to improve machine-readability, a minor adjustment for better ontology compliance.

#### 4.1.3 Generalizability on 100-Model Dataset

The 100-model dataset assessed the pipelines ability to generalize RQ1's objectives across diverse model cards. As per Table 4.1, **schema adherence** dropped to 88% from 95%, indicating reduced consistency due to heterogeneous documentation. **Syntax correctness** remained high at 93%, ensuring machine-readability. **Triple completeness** was 83%, with omissions in free-form fields like `mcro:hasLimitation`, reflecting completeness challenges. The pipeline consistently extracted structured metrics (e.g., mAP for YOLO models), supporting completeness where documentation was standardized. These results highlight

Table 4.2: Expert 1 Feedback on Model Cards

Model Name	Feedback
Falconsai/nsfw.image_detection	Empty architecture in ground truth, but LLM has “Model architecture: Vision Transformer (ViT)”. Ground truth has license information, but LLM-generated content has no license.
all-MiniLM-L6-v2	License not available in LLM-generated version. Limitation also missing in LLM.
dima806/fairface.age.image_detection	License missing in LLM-generated. Dataset found through Kaggle: <a href="https://huggingface.co/datasets/nateraw/fairface">https://huggingface.co/datasets/nateraw/fairface</a> .
amazon/chronos-t5-small	Correct.
google-bert/bert-base-uncased	LLM drops sources and links for training dataset and citation. License missing as well.
timm/mobilenetv3_small_100.lamb_in1k	Model architecture missing in ground truth, but included in LLM. Dataset’s detailed information only in LLM.
openai/clip-vit-large-patch14	Ground truth is more verbose in training data, intended use case, and reference. License missing in LLM.
openai/clip-vit-base-patch32	Performance metrics more verbose in LLM-generated triples, available at <a href="https://huggingface.co/openai/clip-vit-base-patch32">https://huggingface.co/openai/clip-vit-base-patch32</a> .
Bingsu/adetailer	Models missing in ground truth: face_yolov8n_v2.pt (mAP 50 = 0.669), face_yolov8s.pt (mAP 50 = 0.713), face_yolov8m.pt (mAP 50 = 0.737), face_yolov9c.pt (mAP 50 = 0.748), hand_yolov8s.pt (mAP 50 = 0.794), hand_yolov9c.pt (mAP 50 = 0.810), person_yolov8s-seg.pt (bbox + mask mAP 50 and mAP 50-95), person_yolov8m-seg.pt (bbox + mask mAP 50 and mAP 50-95).
sentence-transformers/all-mpnet-base-v2	Model architecture not mentioned in ground truth, but LLM has “Model architecture: We used the pretrained [microsoft/mpnet-base]( <a href="https://huggingface.co/microsoft/mpnet-base">https://huggingface.co/microsoft/mpnet-base</a> ) model and finetuned it on a 1B sentence pairs dataset.” License missing in LLM.

the pipelines robustness in machine readability and consistency, with completeness being sensitive to documentation variability.

#### 4.1.4 Summary of Findings

The findings directly address RQ1, demonstrating that LLMs for ontology-driven metadata extraction significantly improve *consistency* and *machine-readability*, as evidenced by high **schema adherence** (88–95%) and **syntax correctness** (93–95%). The **F1-score** of 89.84% on the 10-model dataset reflects balanced performance. However, *completeness* (83–85%) is limited by omissions in non-standard fields (e.g., licensing), as shown in Table 4.2. Expert feedback underscores strengths in inferring implicit details and extracting structured data, but notes challenges with URI styling. The results of the 100-model dataset confirm generalizability in consistency and machine readability, with completeness constrained by documentation heterogeneity. These insights affirm the pipelines contributions to RQ1 while identifying linguistic variability as a key area for enhancing completeness.

## 4.2 RQ2: Effectiveness of LLMs in Natural Language Querying of Semantic Catalogues

This section presents the results for RQ2: **To what extent can LLMs support NL queries over a semantic model catalogue?** The evaluation assesses the LLM’s capability to translate NL questions into SPARQL queries that align with the MCRO, are syntactically valid, and retrieve the correct information from the knowledge graph. The analysis is based on a set of 12 representative NL questions, categorized into descriptive, comparative, and reasoning-based types, designed to test the LLM’s ability to handle diverse query scenarios. Three primary metrics were used: Schema Compliance Accuracy (SCA), Syntax Validity Rate (SVR), and Correct Output Rate (COR). These quantitative results are complemented by qualitative feedback from ontology experts, providing a comprehensive view of the LLM’s performance.

### 4.2.1 Quantitative Results

The quantitative evaluation employed three metrics to measure different facets of the LLM-generated SPARQL queries:

1. **Schema Compliance Accuracy (SCA):** This metric evaluates whether the generated queries conform to the MCRO ontology, including the correct use of predicates, classes, and instances.

$$\text{SCA} = \frac{\text{Number of schema-compliant queries}}{\text{Total number of queries}} \times 100\%$$

Of the 12 evaluated queries, 9 adhered to the ontology, yielding an **SCA of 75%**. Failures occurred due to references to non-existent entities or incorrect predicate mappings.

2. **Syntax Validity Rate (SVR):** This metric assesses the syntactic correctness of the SPARQL queries, checking for proper structure and clause usage.

$$\text{SVR} = \frac{\text{Number of syntactically valid queries}}{\text{Total number of queries}} \times 100\%$$

11 out of 12 queries were syntactically valid, resulting in an **SVR of 91.7%**. The single invalid query had a missing closing bracket in a FILTER clause.

3. **Correct Output Rate (COR):** This metric determines if the executed queries return the expected results as per the ground truth provided by experts.

$$\text{COR} = \frac{\text{Number of queries returning correct output}}{\text{Total number of queries}} \times 100\%$$

8 out of 12 queries produced correct outputs, giving a **COR of 66.7%**. Errors stemmed from vague filters or missing triples critical to the query logic.

These results are summarized in Table 4.3.

Table 4.3: Quantitative Results for RQ2

Metric	Value	Key Observations
Schema Compliance Accuracy (SCA)	75% (9/12)	Issues with missing entities
Syntax Validity Rate (SVR)	91.7% (11/12)	Strong syntactic performance
Correct Output Rate (COR)	66.7% (8/12)	Semantic errors in output

#### 4.2.2 Detailed Analysis of Generated Queries

The detailed comparison of the LLM-generated SPARQL queries and their outputs against the expert-provided ground truth for each of the 12 questions. The ground truth SPARQL queries include comments from experts, and the comparisons explain the differences in performance and accuracy.

##### Q1. What is the use case of DeBERTaDecodingenhancedBERTwithDisentangledAttention?

LLM-Generated SPARQL:

```
PREFIX mcrō: <http://purl.obolibrary.org/obo/MCRO_>
PREFIX prov1: <https://www.w3.org/ns/prov#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?value
WHERE {
  ?model a mcrō:Model ;
    mcrō:hasUseCase ?section .
  ?section prov1:hasTextValue ?value .
  FILTER regex(str(?model), "DeBERTaDecodingenhancedBERTwithDisentangledAttention", "i")
}
```

LLM-Generated Output: "Fine-tuning on NLU tasks"

Ground Truth SPARQL (Expert-Provided):

```
PREFIX mcrō: <http://purl.obolibrary.org/obo/MCRO_>
PREFIX prov1: <https://www.w3.org/ns/prov#>
```

```

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?value
WHERE {
  ?model a mcro:Model ;
  mcro:hasUseCase ?section .
  ?section provi:hasTextValue ?value .
  FILTER(CONTAINS(LCASE(STR(?model)), "debertadecodingenhancedbertwithdisentangledattention"))
}

Ground Truth Output: "Fine-tuning on NLU tasks"

```

This is a case of functional equivalence with syntactic variation. Both queries correctly extract the use case for DeBERTa, returning “Fine-tuning on NLU tasks”. However, the expert-provided query is more efficient, maintainable, and readable, making it preferable for production-grade SPARQL systems. While the LLM performs correctly in this case, the ground truth version reflects best practices in SPARQL filtering. This shows a positive case where the LLMs SPARQL generation is accurate and aligns closely with expert expectations.

## Q2. What datasets were used to train RoBERTaLargeModel?

---

LLM-Generated SPARQL:

```

PREFIX mcro: <http://purl.obolibrary.org/obo/MCRO_>
PREFIX provi: <https://www.w3.org/ns/prov#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?trainingDataText
WHERE {
  ?model a mcro:Model ;
  mcro:hasTrainingData ?trainingData .
  ?trainingData provi:hasTextValue ?trainingDataText .
  FILTER(CONTAINS(STR(?model), "RoBERTaLargeModel"))
}

```

LLM-Generated Output: 1806.02847) a dataset containing a subset of CommonCrawl data filtered to match the

Ground Truth SPARQL (Expert-Provided):

```

PREFIX mcro: <http://purl.obolibrary.org/obo/MCRO_>
PREFIX provi: <https://www.w3.org/ns/prov#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT DISTINCT ?model ?trainingDataText
WHERE {
  ?model a mcro:Model ;
  mcro:hasTrainingData ?trainingData .
  ?trainingData provi:hasTextValue ?trainingDataText .
  FILTER(CONTAINS(LCASE(STR(?model)), "roberta"))
}

```

Ground Truth Output: (Abridged for brevity) 21 results, including: mcro:RoBERTaLargeModel {\ttfamily }\The  
RoBERTa model was pretrained on the reunion of five datasets: - [BookCorpus], - [English Wikipedia],  
- [CC-News], - [OpenWebText], - [Stories].

---

Full output of Ground Truth is in the Appendix .2.1.

The LLM-generated output lacks precision, providing incomplete and vague references to training datasets with unclear model URIs, resulting in a broader and less reliable dataset. In contrast, the Ground Truth output is well-structured, providing distinct model URIs linked to specific, detailed training datasets, ensuring clear relationships between models and their data. It utilizes optimized querying techniques, such as DISTINCT and case-insensitive matching, to deliver accurate, high-quality results. The

Ground Truth output adheres to best practices for semantic data querying, ensuring higher data integrity, efficiency, and clarity in the knowledge graph.

### Q3. What tasks does CLIP support?

---

LLM-Generated SPARQL:

```
PREFIX mcr0: <http://purl.obolibrary.org/obo/MCR0_>
PREFIX prov1: <https://www.w3.org/ns/prov#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
```

```
SELECT ?value
WHERE {
    ?model mcr0:hasUseCase ?section .
    ?section prov1:hasTextValue ?value .
    FILTER(CONTAINS(?value, "CLIP"))
}
```

LLM-Generated Output: Found 4 result(s):

The model is intended as a research output for research communities. We hope that this model will enable researchers to better understand and explore zero-shot, arbitrary image classification. We also hope it can be used for interdisciplinary studies of the potential impact of such models - the CLIP paper includes a discussion of potential downstream impacts to provide an example for this sort of analysis

The model is intended as a research output for research communities. We hope that this model will enable researchers to better understand and explore zero-shot, arbitrary image classification. We also hope it can be used for interdisciplinary studies of the potential impact of such models - the CLIP paper includes a discussion of potential downstream impacts to provide an example for this sort of analysis

) and upcoming paper include additional discussion as it relates specifically to the training dataset. FashionCLIP is a CLIP-based model developed to produce general product representations for fashion concepts.

Ground Truth SPARQL (Expert-Provided):

```
PREFIX mcr0: <http://purl.obolibrary.org/obo/MCR0_>
PREFIX prov1: <https://www.w3.org/ns/prov#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

```
SELECT DISTINCT ?value
WHERE {
    ?model a mcr0:Model ;
        mcr0:hasUseCase ?section .
    ?section prov1:hasTextValue ?value .
    FILTER(CONTAINS(STR(?value), "CLIP"))
}
```

Ground Truth Output: 1. "The model is intended as a research output for research communities. We hope that this model will enable researchers to better understand and explore zero-shot, arbitrary image classification. We also hope it can be used for interdisciplinary studies of the potential impact of such models - the CLIP paper includes a discussion of potential downstream impacts to provide an example for this sort of analysis."

2. "As per the original [OpenAI CLIP model card] (<https://github.com/openai/CLIP/blob/d50d76daa670286dd6cacf3bcd80b5e4823fc8e1/model-card.md>), this model is intended as a research output for research communities. We hope that this model will enable researchers to better understand and explore zero-shot, arbitrary image classification. We also hope it can be used for interdisciplinary studies of the potential impact of such model. The OpenAI CLIP paper includes a discussion of potential downstream impacts to provide an example for this sort of analysis. Additionally, the LAION -5B blog (<https://laion.ai/blog/laion-5b/>) and upcoming paper include additional discussion as it relates specifically to the training dataset."

3. "FashionCLIP is a CLIP-based model developed to produce general product representations for fashion concepts."

---

The LLM-generated SPARQL query demonstrates close alignment with the expert-crafted version but diverges in its structural rigor and semantic precision. Notably, the LLM omits the RDF type triple (?model a mcr0:Model) present in the ground truth, which explicitly constrains the query to valid model instances in the ontology. This omission may lead to the retrieval of semantically irrelevant or redundant entries. Furthermore, while both queries employ a FILTER(CONTAINS(...)) clause to match substrings containing CLIP, the LLMs version lacks the STR() cast used in the ground truth,

potentially impacting compatibility across SPARQL engines that enforce stricter type semantics. Output-wise, the LLM retrieved four responses, including duplications and a fragmentary sentence, whereas the ground truth returns three well-formed and contextually rich entries, indicating better schema compliance and content fidelity. Overall, the LLM demonstrates strong baseline capability in query generation but benefits significantly from structural refinements and stricter adherence to ontology constraints for accurate and deduplicated output.

#### Q4. what is the architecture of deepseekr10528?

---

LLM-Generated SPARQL:

```
PREFIX mcr0: <http://purl.obolibrary.org/obo/MCR0_>
PREFIX prov1: <https://www.w3.org/ns/prov#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?architectureText
WHERE {
    ?model a mcr0:Model ;
        mcr0:hasModelArchitecture ?architectureSection .
    ?architectureSection prov1:hasTextValue ?architectureText .
    FILTER regex(str(?model), "deepseekr10528", "i")
}
```

LLM-Generated Output: Found 2 result(s):  
The model architecture of DeepSeek-R1-0528-Qwen3-8B is identical to that of Qwen3-8B, but it shares the same tokenizer configuration as DeepSeek-R1-0528.  
identical to that of Qwen3-8B, but it shares the same tokenizer configuration as DeepSeek-R1-0528

Ground Truth SPARQL (Expert-Provided):

```
PREFIX mcr0: <http://purl.obolibrary.org/obo/MCR0_>
PREFIX prov1: <https://www.w3.org/ns/prov#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?architectureText
WHERE {
    ?model a mcr0:Model ;
        mcr0:hasModelArchitecture ?architectureSection .
    ?architectureSection prov1:hasTextValue ?architectureText .
    FILTER(CONTAINS(LCASE(STR(?model)), "deepseekr10528"))
}
```

Ground Truth Output: 1. "The model architecture of DeepSeek-R1-0528-Qwen3-8B is identical to that of Qwen3-8B, but it shares the same tokenizer configuration as DeepSeek-R1-0528."  
2. "identical to that of Qwen3-8B, but it shares the same tokenizer configuration as DeepSeek-R1-0528"

---

The LLM-generated SPARQL query achieves functional parity with the expert-crafted version but reveals key differences in technical precision and robustness. Structurally, both queries correctly specify the RDF type of the subject as mcr0:Model and traverse the mcr0:hasModelArchitecture and prov1:hasTextValue predicates to extract architectural details. However, the LLM uses FILTER regex(str(?model), "deepseekr10528", "i"), whereas the expert version employs FILTER(CONTAINS(LCASE(STR(?model)), "deepseekr10528")). While both achieve case-insensitive matching, the expert approach offers greater consistency and performance predictability across SPARQL engines by avoiding the use of regular expressions.

In terms of output, both queries retrieve the same two results, but one of them is a partial substring, likely due to improper segmentation during RDF preprocessing. This highlights a broader limitation in the data pipeline rather than the query itself, yet the lack of post-processing in the LLM output to remove redundancy or merge fragments reduces its final quality. Overall, while the LLM-generated query is syntactically valid and semantically close, it lacks the fine-tuned best practices of expert-crafted SPARQL, particularly in string matching and result post-processing.

**Q5.:** How does the use case of DeBERTaDecodingenhancedBERTwithDisentangledAttention compare to that of ESMFold?

---

LLM-Generated SPARQL:

```
PREFIX mcro: <http://purl.obolibrary.org/obo/MCRO_>
PREFIX prov1: <https://www.w3.org/ns/prov#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?useCaseText1 ?useCaseText2
WHERE {
    ?model1 mcro:hasUseCase ?useCase1 .
    ?useCase1 prov1:hasTextValue ?useCaseText1 .
    FILTER(REGEX(str(?model1), "DeBERTaDecodingenhancedBERTwithDisentangledAttention")) .

    ?model2 mcro:hasUseCase ?useCase2 .
    ?useCase2 prov1:hasTextValue ?useCaseText2 .
    FILTER(REGEX(str(?model2), "ESMFold")) .
}
```

LLM-Generated Output: Found 1 result(s):

```
Fine-tuning on NLU tasks (useCaseText2: If you're interested in using ESMFold in practice, please check
out the associated [tutorial notebook](https://colab.research.google.com/github/huggingface/notebooks
/blob/main/examples/protein_folding.ipynb).)
```

Ground Truth SPARQL (Expert-Provided):

```
PREFIX mcro: <http://purl.obolibrary.org/obo/MCRO_>
PREFIX prov1: <https://www.w3.org/ns/prov#>

SELECT ?model ?useCaseText
WHERE {
    ?model mcro:hasUseCase ?useCase .
    ?useCase prov1:hasTextValue ?useCaseText .
    FILTER(
        CONTAINS(LCASE(STR(?model)), "debertadecodingenhancedbertwithdisentangledattention") ||
        CONTAINS(LCASE(STR(?model)), "esmfold")
    )
}
```

```
Ground Truth Output: 1. mcro:ESMFold - "If you're interested in using ESMFold in practice, please check
out the associated [tutorial notebook](https://colab.research.google.com/github/huggingface/notebooks
/blob/main/examples/protein_folding.ipynb)."
2. mcro:esmFold - "If you're interested in using ESMFold in practice, please check out the associated [
    tutorial notebook](https://colab.research.google.com/github/huggingface/notebooks/blob/main/examples/
    protein_folding.ipynb)."
3. mcro:DeBERTaDecodingenhancedBERTwithDisentangledAttention - "Fine-tuning on NLU tasks"
```

---

The LLM-generated SPARQL query attempts to retrieve use cases for two specific models DeBERTaDecodingenhancedBERTwithDisentangledAttention and ESMFold using two separate triple patterns with distinct variables (?model1, ?model2, etc.). While the logic is semantically aligned with the intent, the structure is suboptimal for SPARQL execution. Specifically, the use of two independent triple patterns in the same WHERE block without a unifying UNION clause effectively performs a Cartesian product join, which can result in incomplete or unintended outputs if any branch returns no matches. The output supports this: only a single result is returned, showing a mismatched pairing of one model's output with the others placeholder, suggesting that the structure failed to isolate and bind results independently. Moreover, the strict use of REGEX on full IRI strings (str(?model1)) is less robust than the expert-crafted version's use of CONTAINS on LCASE(STR(?model)), which better handles case variance and is more portable across RDF stores. The expert SPARQL, by contrast, uses a disjunctive FILTER within a single pattern to match either model by IRI content. This approach is both cleaner and more efficient. It produces all three ground truth results: two from ESMFold (due to IRI variants like mcro:esmFold and mcro:ESMFold) and one from DeBERTa. The broader use of LCASE and CONTAINS ensures higher recall and better alignment with RDF naming variability. The LLM-generated query lacks query normalization, pattern optimization, and robust IRI matching, leading to incomplete

results. The expert query demonstrates best practices in SPARQL for multi-entity retrieval and defensive string filtering.

**Q6.** Which model has fewer than 1 million parameters?

---

LLM-Generated SPARQL:

```
PREFIX mcro: <http://purl.obolibrary.org/obo/MCRO_>
PREFIX prov1: <https://www.w3.org/ns/prov#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
```

```
SELECT ?model
WHERE {
  ?model mcro:hasModelArchitecture ?architecture .
  ?architecture prov1:hasTextValue ?value .
  FILTER (xsd:integer(?value) <1000000)
}
```

LLM-Generated Output: Found 1 result(s): MCRO\_Qwen306B

Ground Truth SPARQL (Expert-Provided):

```
PREFIX mcro: <http://purl.obolibrary.org/obo/MCRO_>
PREFIX prov1: <https://www.w3.org/ns/prov#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?model ?value
WHERE {
  ?model mcro:hasModelArchitecture ?architecture .
  ?architecture prov1:hasTextValue ?value .
  FILTER(xsd:integer(?value) <1000000)
}
```

Ground Truth Output: Multiple results, including mcro:Qwen306B with values like "16 for Q and 8 for KV".

---

Full output of Ground Truth in the Appendix .2.2

The LLM-generated SPARQL query reveals a critical flaw in interpreting and handling the prov1:hasTextValue predicate. This predicate stores natural language text, such as descriptive phrases about model architectures (e.g., Vision Transformer (ViT) model pre-trained on ImageNet-21k), rather than strict numeric values. However, the LLM attempts to cast these text values into integers using the xsd:integer() function and applies a numeric filter (< 1000000). This leads to two major issues: first, most hasTextValue entries are not purely numeric, so they are silently excluded or throw coercion errors, depending on the SPARQL engine; second, even when partial numbers are embedded in the text (e.g., 6 layers or 3 Transformer layers), they may not be reliably parsed without preprocessing. As a result, the LLM query fails to retrieve several valid entries that the expert ground truth query successfully identifies. Furthermore, the LLM query only selects the ?model variable, omitting the corresponding ?value, which strips away the context necessary to validate or interpret the results. In contrast, the expert-provided query correctly projects both the model and its associated architecture description, enabling transparent validation of the filter condition. The ground truth output includes models such as mcro:Qwen306B with the value 28 and mcro:debertaV3Small with 6 layers, showing that although the hasTextValue field contains mixed content, some entries are indeed parseable for numeric comparison. The LLMs incorrect parsing logic and incomplete projection resulted in a false positive (e.g., returning MCRO\_Qwen306 B without proof of a matching numeric value) and missed multiple correct entries. Therefore, the discrepancy arises from a fundamental misunderstanding of data types and projection requirements in SPARQL by the LLM, emphasizing the need for more accurate literal parsing strategies in natural language-to-SPARQL systems.

**Q7:** How does the citation of HuggingFaceM4idefics28b compare to that of IDEAResearchgrounding-dinobase?

---

LLM-Generated SPARQL:

```
PREFIX mcro: <http://purl.obolibrary.org/obo/MCRO_>
```

```
PREFIX prov1: <https://www.w3.org/ns/prov#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?huggingfaceCitation ?ideaResearchCitation
WHERE {
    # Get the citation detail for HuggingFaceM4idefics28b
    OPTIONAL {
        mcr: HuggingFaceM4idefics28b mcr:hasUseCase ?huggingfaceUseCase .
        ?huggingfaceUseCase prov1:hasTextValue ?huggingfaceCitation .
    }

    # Get the citation detail for IDEAResearchgroundingdinobase
    OPTIONAL {
        mcr: IDEAResearchgroundingdinobase mcr:hasUseCase ?ideaResearchUseCase .
        ?ideaResearchUseCase prov1:hasTextValue ?ideaResearchCitation .
    }

    #FILTER (BOUND(?huggingfaceCitation) || BOUND(?ideaResearchCitation)) #Optional: only return a result if
    # at least one citation exists
}
```

LLM-Generated Output: Found 1 result(s):

```
`idefics2-8b-base` and `idefics2-8b` can be used to perform inference on multimodal (image + text) tasks
in which the input is composed of a text query along with one (or multiple) image(s). Text and images
can be arbitrarily interleaved. That includes image captioning, visual question answering, etc. (
ideaResearchCitation: zero-shot object detection (the task of detecting things in an image out-of-the
-box without labeled data))
```

Ground Truth SPARQL (Expert-Provided):

```
PREFIX mcr: <http://purl.obolibrary.org/obo/MCRO_>
PREFIX prov1: <https://www.w3.org/ns/prov#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?huggingfaceCitation ?ideaResearchCitation
WHERE {
    OPTIONAL {
        mcr: HuggingFaceM4idefics28b mcr:hasCitation ?huggingfaceCitationSection .
        ?huggingfaceCitationSection prov1:hasTextValue ?huggingfaceCitation .
    }

    OPTIONAL {
        mcr: IDEAResearchgroundingdinobase mcr:hasCitation ?ideaResearchCitationSection .
        ?ideaResearchCitationSection prov1:hasTextValue ?ideaResearchCitation .
    }
}
```

Ground Truth Output: "@misc{laurencon2023obelics, title={OBELICS: An Open Web-Scale Filtered Dataset of Interleaved Image-Text Documents}, author={Hugo Laurencon and Lucile Saulnier and Lo Tronchon and Stas Bekman and Amanpreet Singh and Anton Lozhkov and Thomas Wang and Siddharth Karamchetti and Alexander M. Rush and Douwe Kiela and Matthieu Cord and Victor Sanh}, year={2023}, eprint={2306.16527}, archivePrefix={arXiv}, primaryClass={cs.IR} } @misc{laurencon2024matters, title={What matters when building vision-language models?}, author={Hugo Laurencon and Lo Tronchon and Matthieu Cord and Victor Sanh}, year={2024}, eprint={2405.02246}, archivePrefix={arXiv}, primaryClass={cs.CV} }""  
"Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection by Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, Lei Zhang"

---

The LLM-generated SPARQL query attempts to retrieve citation information for two models HuggingFaceM4idefics28b and IDEAResearchgroundingdinobase, but does so by incorrectly referencing the predicate mcr:has UseCase rather than the accurate mcr:hasCitation used in the ontology. Consequently, instead of extracting formal publication metadata, the query retrieves descriptive use-case text such as zero-shot object detection, which is not a citation but a functional application of the model. This semantic misalignment yields a partial and misleading output: while the model descriptions may contain useful contextual information, they fail to meet the precision required for citation retrieval. Additionally, the LLM-generated result conflates information by including a general description of idefics2-8b-base with a citation meant for IDEAResearchgroundingdinobase, demonstrating a mismatch in model-to-text

association. In contrast, the expert-provided SPARQL correctly queries the `mcro:hasCitation` predicate, followed by accessing the `prov1:hasTextValue` property, which contains well-formatted bibliographic entries including author names, titles, publication years, and arXiv identifiers. This yields a reliable and verifiable citation output for both models. The LLMs failure to recognize the proper ontology structure and its substitution of use-case text for formal citations results in an incomplete and inaccurate retrieval, illustrating a semantic parsing error that undermines the integrity of the query output.

**Q8.** What are the differences in the use cases of vitmatte-model and llavahflavav16mistral7bfh?

---

LLM-Generated SPARQL:

```
PREFIX mcro: <http://purl.obolibrary.org/obo/MCRO_>
PREFIX prov1: <https://www.w3.org/ns/prov#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?vitmatteUseCase ?llavahflavav16mistral7bfhUseCase WHERE {
  OPTIONAL {
    mcro:vitmatte-model mcro:hasUseCase ?vitmatteUseCaseSection .
    ?vitmatteUseCaseSection prov1:hasTextValue ?vitmatteUseCase .
  }
  OPTIONAL {
    mcro:llavahflavav16mistral7bfh mcro:hasUseCase ?llavahflavav16mistral7bfhUseCaseSection .
    ?llavahflavav16mistral7bfhUseCaseSection prov1:hasTextValue ?llavahflavav16mistral7bfhUseCase .
  }
  FILTER ( BOUND(?vitmatteUseCase) || BOUND(?llavahflavav16mistral7bfhUseCase) )
}
```

LLM-Generated Output: Found 1 result(s):  
 models?search=vitmatte) to look for other  
 fine-tuned versions that may interest you. (llavahflavav16mistral7bfhUseCase: You can use the raw model  
 for tasks like image captioning, visual question answering, multimodal chatbot use cases. See the [  
 model hub](https://huggingface.co/models?search=llava-hf) to look for  
 other versions on a task that interests you.)

Ground Truth SPARQL (Expert-Provided):

```
PREFIX mcro: <http://purl.obolibrary.org/obo/MCRO_>
PREFIX prov1: <https://www.w3.org/ns/prov#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?vitmatteUseCase ?llavahflavav16mistral7bfhUseCase WHERE {
  OPTIONAL {
    mcro:vitmatte-model mcro:hasUseCase ?vitmatteUseCaseSection .
    ?vitmatteUseCaseSection prov1:hasTextValue ?vitmatteUseCase .
  }
  OPTIONAL {
    mcro:llavahflavav16mistral7bfh mcro:hasUseCase ?llavahflavav16mistral7bfhUseCaseSection .
    ?llavahflavav16mistral7bfhUseCaseSection prov1:hasTextValue ?llavahflavav16mistral7bfhUseCase .
  }
  FILTER ( BOUND(?vitmatteUseCase) || BOUND(?llavahflavav16mistral7bfhUseCase) )
}
```

Ground Truth Output: "You can use the raw model for image matting. See the [model hub](https://huggingface.co/models?search=vitmatte) to look for other fine-tuned versions that may interest you."  
 "You can use the raw model for tasks like image captioning, visual question answering, multimodal chatbot use cases. See the [model hub](https://huggingface.co/models?search=llava-hf) to look for other versions on a task that interests you."

---

The LLM-generated SPARQL query is syntactically and semantically aligned with the expert-provided version, correctly using the `mcro:hasUseCase` predicate followed by the `prov1:hasTextValue` property to retrieve the intended use-case descriptions for the models `vitmatte-model` and `llavahflavav16mistral7bfh`. The structure of the query is accurate, with both `OPTIONAL` clauses scoped properly and a `FILTER` condition to ensure that at least one result is returned. The LLM-generated output also closely mirrors the ground truth output, particularly for `llavahflavav16mistral7bfh`, where the use-case description

related to image captioning, visual question answering, and multimodal chatbot usage is retrieved in full. However, for vitmatte-model, the LLM output omits the beginning part of the expected use-case sentence (You can use the raw model for image matting.) and starts mid-sentence from the link reference. This partial truncation introduces minor incompleteness in the returned value but does not reflect a fundamental flaw in query logic. Overall, this example demonstrates a largely successful LLM-to-SPARQL generation, with correct predicate usage and coherent structure, although it still suffers from slight content truncation, likely caused by post-processing or response parsing issues rather than semantic misunderstanding.

**Q9.** Which model has a more permissive license: FlagEmbedding or HuggingFaceH4zephyr7bbeta?

LLM-Generated SPARQL:

```
PREFIX mcro: <http://purl.obolibrary.org/obo/MCRO_>
PREFIX prov1: <https://www.w3.org/ns/prov#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?model ?licenseText
WHERE {
  ?model a mcro:Model .
  ?model mcro:hasUseCase ?useCase .
  ?useCase prov1:hasTextValue ?licenseText .
  FILTER (?model = mcro:FlagEmbedding || ?model = mcro:HuggingFaceH4zephyr7bbeta)
}
```

LLM-Generated Output: Found 2 result(s):

```
MCRO_FlagEmbedding (licenseText: Retrieval-augmented LLMs)
MCRO_FlagEmbedding (licenseText: FlagEmbedding focuses on retrieval-augmented LLMs, consisting of the
following projects currently:
```

- \*\*Long-Context LLM\*\*: [Activation Beacon] ([https://github.com/FlagOpen/FlagEmbedding/tree/master/Long\\_LLM/activation\\_beacon](https://github.com/FlagOpen/FlagEmbedding/tree/master/Long_LLM/activation_beacon))
- \*\*Fine-tuning of LM\*\* : [LM-Cocktail] ([https://github.com/FlagOpen/FlagEmbedding/tree/master/LM\\_Cocktail](https://github.com/FlagOpen/FlagEmbedding/tree/master/LM_Cocktail))
- \*\*Dense Retrieval\*\*: [BGE-M3] ([https://github.com/FlagOpen/FlagEmbedding/tree/master/FlagEmbedding/BGE\\_M3](https://github.com/FlagOpen/FlagEmbedding/tree/master/FlagEmbedding/BGE_M3)), [LLM Embedder] ([https://github.com/FlagOpen/FlagEmbedding/tree/master/FlagEmbedding/llm\\_embedder](https://github.com/FlagOpen/FlagEmbedding/tree/master/FlagEmbedding/llm_embedder)), [BGE Embedding] ([https://github.com/FlagOpen/FlagEmbedding/tree/master/FlagEmbedding/baai\\_general\\_embedding](https://github.com/FlagOpen/FlagEmbedding/tree/master/FlagEmbedding/baai_general_embedding))
- \*\*Reranker Model\*\*: [BGE Reranker] (<https://github.com/FlagOpen/FlagEmbedding/tree/master/FlagEmbedding/reranker>)
- \*\*Benchmark\*\*: [C-MTEB] ([https://github.com/FlagOpen/FlagEmbedding/tree/master/C\\_MTEB](https://github.com/FlagOpen/FlagEmbedding/tree/master/C_MTEB))

Ground Truth SPARQL (Expert-Provided):

```
PREFIX mcro: <http://purl.obolibrary.org/obo/MCRO_>
PREFIX prov1: <https://www.w3.org/ns/prov#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?model ?licenseText
WHERE {
  ?model a mcro:Model .
  ?model mcro:hasLicense ?licenseNode .
  ?licenseNode prov1:hasTextValue ?licenseText .
  FILTER (?model = mcro:FlagEmbedding || ?model = mcro:HuggingFaceH4zephyr7bbeta)
}
```

Ground Truth Output: mcro:FlagEmbedding  
 "FlagEmbedding is licensed under the [MIT License] (<https://github.com/FlagOpen/FlagEmbedding/blob/master/LICENSE>). The released models can be used for commercial purposes free of charge."  
 mcro:FlagEmbedding  
 "MIT License"  
 mcro:FlagEmbedding  
 "FlagEmbedding is licensed under the [MIT License] (<https://github.com/FlagOpen/FlagEmbedding/blob/master/LICENSE>). The released models can be used for commercial purposes free of charge."  
 mcro:FlagEmbedding  
 "MIT License"  
 mcro:HuggingFaceH4zephyr7bbeta  
 "MIT"

The LLM-generated SPARQL query exhibits a critical semantic error by mistakenly targeting the `mcro:hasUseCase` property to retrieve license information, instead of the correct predicate `mcro:hasLicense` as used in the expert-provided query. While the query is syntactically valid and structurally sound, its use of the incorrect predicate results in the retrieval of content related to model use cases rather than license details. As a result, the LLM-generated output includes text describing the FlagEmbedding models technical capabilities and repository projects but fails to return any actual licensing information, such as MIT License or statements about commercial usage rights. In contrast, the ground truth SPARQL correctly accesses the `mcro:hasLicense` property and returns multiple relevant license statements for both `mcro:FlagEmbedding` and `mcro:HuggingFaceH4zephyr7bbeta`. This case highlights a significant semantic mismatch: despite surface-level coherence and a correctly filtered model scope, the LLM misinterprets the target predicate, ultimately leading to a complete deviation from the intended query objective. Such an error undermines the utility of the output and demonstrates the need for more robust predicate alignment in LLM-to-SPARQL generation tasks.

#### Q10. Why was the model architecture chosen for ESMFold?

---

LLM-Generated SPARQL:

```
PREFIX mcrō: <http://purl.obolibrary.org/obo/MCRO_>
PREFIX prov1: <https://www.w3.org/ns/prov#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?text
WHERE {
    ?model a mcrō:Model ;
        mcrō:hasModelArchitecture ?section .
    FILTER regex(str(?model), "ESMFold", "i")
    ?section prov1:hasTextValue ?text .
}
```

LLM-Generated Output: Found 3 result(s):

```
ESMFold is a state-of-the-art end-to-end protein folding model based on an ESM-2 backbone.
ESMFold is a state-of-the-art end-to-end protein folding model based on an ESM-2 backbone. It does not
    require any lookup or MSA step, and therefore does not require any external databases to be present
    in order to make predictions. As a result, inference time is very significantly faster than
        AlphaFold2.
ESMFold is a state-of-the-art end-to-end protein folding model based on an ESM-2 backbone. It does not
    require any lookup or MSA step, and therefore does not require any external databases to be present
    in order to make predictions. As a result, inference time is very significantly faster than
        AlphaFold2.
```

Ground Truth SPARQL (Expert-Provided):

```
PREFIX mcrō: <http://purl.obolibrary.org/obo/MCRO_>
PREFIX prov1: <https://www.w3.org/ns/prov#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?text
WHERE {
    ?model a mcrō:Model ;
        mcrō:hasModelArchitecture ?section .
    ?section prov1:hasTextValue ?text .
    FILTER(REGEX(STR(?model), "ESMFold", "i"))
}
```

```
Ground Truth Output: "ESMFold is a state-of-the-art end-to-end protein folding model based on an ESM-2
    backbone."
"ESMFold is a state-of-the-art end-to-end protein folding model based on an ESM-2 backbone. It does not
    require any lookup or MSA step, and therefore does not require any external databases to be present
    in order to make predictions. As a result, inference time is very significantly faster than
        AlphaFold2."
"ESMFold is a state-of-the-art end-to-end protein folding model based on an ESM-2 backbone. It does not
    require any lookup or MSA step, and therefore does not require any external databases to be present
    in order to make predictions. As a result, inference time is very significantly faster than
        AlphaFold2."
```

---

The SPARQL query generated by the LLM is functionally equivalent to the expert-provided ground truth and successfully retrieves the intended information. Both queries correctly target models whose identifiers contain ESMFold and extract the textual content associated with the `mcro:hasModelArchitecture` property via the `prov1:hasTextValue` predicate. While the LLM query includes additional prefixes that are not used in the actual query (e.g., `rdfs`, `xsd`), these do not impact execution or results. The main structural difference lies in the order of the `FILTER` clause, which is placed earlier in the LLM query but is logically equivalent. The output from both queries includes three architecture descriptions, with the LLM-generated output containing a redundant duplicate, which also appears in the ground truth. This confirms that the LLM-generated SPARQL accurately captures the intended semantics and retrieves the correct data, making this a valid and correct result.

**Q11.** What makes the dataset suitable for FinetunedT5SmallTextSummarizations use case?

---

LLM-Generated SPARQL:

```
PREFIX mcro: <http://purl.obolibrary.org/obo/MCRO_>
PREFIX prov1: <https://www.w3.org/ns/prov#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?value
WHERE {
    ?model mcro:hasUseCase ?useCase .
    ?useCase prov1:hasTextValue ?value .
    FILTER (?model = mcro:FinetunedT5SmallTextSummarization)
}
```

LLM-Generated Output: Found 1 result(s):  
Text Summarization

Ground Truth SPARQL (Expert-Provided):

```
PREFIX mcro: <http://purl.obolibrary.org/obo/MCRO_>
PREFIX prov1: <https://www.w3.org/ns/prov#>

SELECT ?value
WHERE {
    mcro:FinetunedT5SmallTextSummarization mcro:hasUseCase ?useCase .
    ?useCase prov1:hasTextValue ?value .
}
```

Ground Truth Output: "Text Summarization"

---

The LLM-generated SPARQL and the expert-provided (ground truth) SPARQL queries are functionally equivalent, as both retrieve the same result, Text Summarization. The primary difference between them lies in the structure of the queries. The LLM-generated query uses a `FILTER` clause to specifically restrict the model to `mcro:FinetunedT5SmallTextSummarization`, while the ground truth query directly references this model in the triple pattern. This direct approach in the ground truth query is more concise and simpler, making it slightly more efficient. However, both queries are correct and yield the same output. While the `FILTER` clause in the LLM query introduces a slight overhead, this is generally negligible unless the dataset is significantly large. Therefore, both queries are valid, but the ground truth query is preferred for its simplicity and clarity.

**Q12.** What is the primary tradeoff of using LTXVideo?

---

LLM-Generated SPARQL:

```
PREFIX mcro: <http://purl.obolibrary.org/obo/MCRO_>
PREFIX prov1: <https://www.w3.org/ns/prov#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?text
WHERE {
    ?model mcro:hasLimitation ?section .
```

```
?section prov1:hasTextValue ?text .
FILTER(CONTAINS(STR(?model), "LTXVideo"))
}
```

**LLM-Generated Output:** Prompts should be in English. The more elaborate the better. Good prompt looks like  
`The turquoise waves crash against the dark, jagged rocks of the shore, sending white foam spraying into the air. The scene is dominated by the stark contrast between the bright blue water and the dark , almost black rocks. The water is a clear, turquoise color, and the waves are capped with white foam . The rocks are dark and jagged, and they are covered in patches of green moss. The shore is lined with lush green vegetation, including trees and bushes. In the background, there are rolling hills covered in dense forest. The sky is cloudy, and the light is dim.`

The model works best on resolutions under 720 x 1280 and number of frames below 257.

The model works on resolutions that are divisible by 32 and number of frames that are divisible by 8 + 1 ( e.g. 257). In case the resolution or number of frames are not divisible by 32 or 8 + 1, the input will be padded with -1 and then cropped to the desired resolution and number of frames.

**Ground Truth SPARQL (Expert-Provided):**

```
PREFIX mcro: <http://purl.obolibrary.org/obo/MCRO_>
PREFIX prov1: <https://www.w3.org/ns/prov#>

SELECT ?text
WHERE {
  ?model mcro:hasLimitation/prov1:hasTextValue ?text .
  FILTER(CONTAINS(LCASE(STR(?model)), "ltxvideo"))
}
```

**Ground Truth Output:** "Prompts should be in English. The more elaborate the better. Good prompt looks like  
`The turquoise waves crash against the dark, jagged rocks of the shore, sending white foam spraying into the air. The scene is dominated by the stark contrast between the bright blue water and the dark , almost black rocks. The water is a clear, turquoise color, and the waves are capped with white foam . The rocks are dark and jagged, and they are covered in patches of green moss. The shore is lined with lush green vegetation, including trees and bushes. In the background, there are rolling hills covered in dense forest. The sky is cloudy, and the light is dim.`"

"The model works best on resolutions under 720 x 1280 and number of frames below 257."

"The model works on resolutions that are divisible by 32 and number of frames that are divisible by 8 + 1 ( e.g. 257). In case the resolution or number of frames are not divisible by 32 or 8 + 1, the input will be padded with -1 and then cropped to the desired resolution and number of frames."

The LLM-generated SPARQL query and the ground truth SPARQL query both aim to retrieve limitations related to the LTXVideo model, but they differ in structure and handling of case sensitivity. The LLM-generated query checks for the models limitations using FILTER(CONTAINS(STR(?model), "LTXVideo")), which is case-sensitive. This means that if the models URI is not an exact match in the exact case, the query might fail to retrieve the correct result. On the other hand, the ground truth query employs LCASE(STR(?model)) within the FILTER(CONTAINS(LCASE(STR(?model)), "ltxvideo")) clause, making it case-insensitive. This ensures that the query will retrieve the correct results even if the models URI has a different case. Additionally, while the LLM-generated query uses two separate triple patterns?model mcro:hasLimitation ?section and ?section prov1:hasTextValue ?textthe ground truth query combines these into a more compact form: ?model mcro:hasLimitation/prov1:hasTextValue ?text. This makes the ground truth query more efficient and concise. Despite these differences in structure and flexibility, both queries ultimately produce the same output, which includes limitations related to LTXVideo, such as prompt requirements and frame resolution details. The ground truth query is therefore more optimal due to its case insensitivity and streamlined structure.

#### 4.2.3 Qualitative Feedback from Expert Review

Two ontology experts reviewed the generated queries, providing detailed insights into their strengths and weaknesses, which are critical for understanding the LLMs limitations and potential improvements in NL-to-SPARQL translation. The experts praised the high syntactic validity but identified recurring issues:

- **Incorrect Property Targeting:** In Questions 7 and 9, the LLM used `mcro:hasUseCase` instead of `mcro:hasCitation` or `mcro:hasLicense`, respectively, leading to schema non-compliance and incorrect outputs. This suggests a lack of precise understanding of ontology, particularly for specialized properties.

- **Ambiguous Filtering:** In Question 3, the LLMs `FILTER(CONTAINS(?value, "CLIP"))` matched irrelevant substrings, producing redundant results. The experts noted that filtering on output values rather than model identifiers compromises specificity.
- **Entity Variability and Case Sensitivity:** In Question 2, the LLMs case-sensitive `CONTAINS` filter missed variant model names (e.g., `robertalargemodel`), unlike the ground truths case-insensitive approach. This indicates challenges in handling inconsistent naming conventions.
- **Unstructured Data Handling:** In Question 6, the LLMs assumption that parameter counts are integers failed for textual descriptions, as noted by the experts comment on the need for data cleaning to standardize numerical data.

To address these issues, the experts proposed enhancing the LLMs prompt with modular, ontology-specific rules to improve robustness and interpretability:

---

```
main_rules_prompt = "rules_for_prompt" + some_more_rules
some_more_rules = "input_for_complicated_cases"
```

---

For example, rules could specify correct predicates for citations or licenses, guide case-insensitive matching, or provide patterns for extracting numerical data. The experts also recommended pre-processing steps, such as ontology-guided entity normalization, to handle variations in model names, and post-processing to filter out redundant results. These suggestions align with RQ2s aim of improving LLM robustness by ensuring better alignment with the MCRO ontology and reducing ambiguity in query generation.

The qualitative feedback underscores the need for enhanced ontology awareness and data preprocessing to bridge the gap between NL queries and formal SPARQL syntax. By implementing these recommendations, the LLM could better support non-expert users, making semantic model catalogues more accessible and advancing RQ2s goal of democratizing access.

#### 4.2.4 Summary of Evaluation Findings

The findings directly address RQ2 by demonstrating that LLMs can support NL queries over semantic model catalogues with notable strengths and addressable limitations. The high SVR (91.7%) confirms the LLMs ability to generate syntactically valid SPARQL queries, while the SCA (75%) indicates reasonable ontology compliance for most queries (e.g., Questions 1, 4, 8, 10, 12). However, the COR (66.7%) reveals challenges in achieving semantic accuracy, particularly for complex queries (e.g., Questions 2, 5, 6, 7, 9, 11), where issues like ontology misalignment, entity variability, and unstructured data handling (e.g., Question 6s parameter counts) reduce effectiveness. The LLM excels in simple descriptive queries (e.g., Questions 1, 4, 10, 12) but struggles with comparative and reasoning-based queries that require a nuanced understanding of ontology (e.g., Questions 5, 7, 9). The expanded qualitative feedback highlights specific failure patterns, incorrect property targeting, ambiguous filtering, and entity variability, and proposes actionable strategies, such as modular prompt engineering and ontology-guided preprocessing. These insights contribute to RQ2 by confirming the potential of LLMs to democratize access to semantic model catalogues for non-expert users, thereby enabling broader participation in model selection and reuse.

### 4.3 RQ3: Effectiveness of G-RAG Compared to SPARQL for Model QA

This section presents the findings addressing RQ3: To what extent does G-RAG compare to SPARQL in answering questions about models? The results are derived from the analysis conducted in Pipeline 2 and Pipeline 3, where a comparison was made between the outputs of Graph RAG (from Pipeline 3) and SPARQL (from Pipeline 2). Qualitative insights from the comparison of SPARQL and G-RAG are provided, based on evaluations against both the Hugging Face model cards and the Ground Truth dataset. A detailed summary of these findings is presented in Table 4.5.

#### 4.3.1 Quantitative Analysis

To evaluate the performance of SPARQL-based Q/A and G-RAG-based Q/A in answering knowledge graph-based questions, we reused the 12 representative queries formulated for RQ2, covering diverse

categories such as descriptive, comparative, and reasoning-based questions. Each response was assessed along the schema compliance, accuracy, syntax validity rate, and correct output rate.

Table 4.4: Evaluation Metrics Comparison

Metric	Pipeline 2 (SPARQL)	Pipeline 3 (RAG)
Schema Compliance Accuracy (SCA)	75% (9 out of 12)	75% (9 out of 12)
Syntax Validity Rate (SVR)	91.7% (11 out of 12)	100% (12 out of 12)
Correct Output Rate (COR)	66.7% (8 out of 12)	66.7% (8 out of 12)

### Schema Compliance Accuracy (SCA)

#### SPARQL

9 out of 12 generated queries fully conformed to the MCRO ontology schema.

The 3 non-compliant cases involved:

1. Use of undefined predicates (e.g., `hasDescriptionText` instead of `hasDescription`)
2. Incorrect triple structure (e.g., missing object or misused datatype property)
3. Entity mismatches due to casing or tokenization errors (e.g., GPT-2 vs gpt2)

#### G-RAG:

Also scored 9 out of 12. Failures were due to:

1. Mentioning attributes that are not modeled in the KG (e.g., hallucinated training datasets)
2. Implicit references instead of schema-aligned terms (e.g., licensing terms without specifying `hasLicense`)
3. Generalization of model names or replacing them with pronouns that caused ambiguity

Interpretation: Both pipelines achieved equal compliance, but Pipeline 2's errors were syntactic and structural, while Pipeline 3's errors were semantic and content-based.

### Syntax Validity Rate (SVR)

#### SPARQL

11 out of 12 queries were executable without modification.

The single invalid query failed due to:

1. Unclosed parentheses and incorrect use of FILTER expression (e.g., `FILTER(?x = "GPT-2" with a missing )`)

#### G-RAG:

Achieved a perfect score of 100%. All 12 responses were grammatically correct, coherent, and free from structural language errors.

Interpretation: SPARQL queries are highly sensitive to minor formatting errors; even a single misplaced bracket causes a query to fail execution. RAGs' use of natural language makes it more robust in syntactic terms, although correctness still depends on content grounding. **Correct Output Rate (COR)**

#### SPARQL

8 out of 12 queries returned the correct and expected result from the knowledge graph.

The 4 incorrect responses resulted from:

1. Incomplete WHERE clause logic leading to empty query results
2. Use of partial filters that overgeneralize (e.g., using only `?x hasLicense ?y` without restricting the subject)
3. Retrieval of older or default metadata in place of updated attributes (versioning ambiguity)

#### G-RAG

Also produced 8 correct answers. Failure cases involved:

1. Hallucination (e.g., inserting BERT was trained on SQuAD when not stated in the model card)
2. Incorrect disambiguation between similar models (e.g., mixing distilBERT and BERT-base)
3. Over-reliance on irrelevant context windows during retrieval

Interpretation: Both pipelines suffer from a one-third error rate. SPARQL-based errors are more deterministic (logic-based), while RAG errors tend to be contextual or due to fuzzy matching and LLM hallucination.

### 4.3.2 Qualitative Analysis

To complement the quantitative findings, a qualitative comparison sheds light on the practical behavior of SPARQL and G-RAG, summarised in Figure 4.5.

#### Q1. What is the use case of DeBERTaDecodingEnhancedBERTwithDisentangledAttention?

---

SPARQL Answer: Fine-tuning on NLU tasks

RAG Answer: DeBERTa is used to improve BERT and RoBERTa models using disentangled attention and an enhanced mask decoder.

Comparison:

SPARQL: States fine-tuning on natural language understanding (NLU) tasks, aligning with the model card for microsoft/deberta-v3-base, which highlights NLU tasks like GLUE benchmarks.

RAG: Describes DeBERTa as improving BERT and RoBERTa using disentangled attention, but lacks specificity on use cases, focusing more on technical enhancements.

Analysis: SPARQL is more precise, directly addressing the use case, while RAG provides context but is less focused. The paper "[DEBERTA: DECODING-ENHANCED BERT WITH DISENTANGLED ATTENTION](#)" confirms NLU tasks, supporting SPARQL's accuracy.

---

#### Q2. What datasets were used to train RoBERTaLargeModel?

---

SPARQL Answer: 1806.02847) a dataset containing a subset of CommonCrawl data filtered to match the

RAG Answer: The RoBERTa model was pretrained on the reunion of five datasets: BookCorpus, English Wikipedia, CC-News, OpenWebText, and Stories.

Comparison:

SPARQL: Mentions a subset of CommonCrawl, likely referring to the Stories dataset, but is incomplete, as per the model card for FacebookAI/roberta-large.

RAG: Lists BookCorpus, English Wikipedia, CC-News, OpenWebText, and Stories, matching the comprehensive dataset details from the model card, totaling 160GB of text.

Analysis: RAG is more comprehensive, aligning with the paper "[RoBERTa: A Robustly Optimized BERT Pretraining Approach](#)", while SPARQL is partially correct but lacks detail.

---

#### Q3. What tasks does CLIP support?

---

SPARQL Answer: The model is intended as a research output for research communities. We hope that this model will enable researchers to better understand and explore zero-shot, arbitrary image classification. We also hope it can be used for interdisciplinary studies of the potential impact of such models - the CLIP paper includes a discussion of potential downstream impacts to provide an example for this sort of analysis.

The model is intended as a research output for research communities. We hope that this model will enable researchers to better understand and explore zero-shot, arbitrary image classification. We also hope it can be used for interdisciplinary studies of the potential impact of such models - the CLIP paper includes a discussion of potential downstream impacts to provide an example for this sort of analysis

) and upcoming paper include additional discussion as it relates specifically to the training dataset.

FashionCLIP is a CLIP-based model developed to produce general product representations for fashion concepts.

RAG Answer: Mentions CLIP supports image classification and zero-shot learning but struggles with fine-grained classification and object counting.

Comparison:

SPARQL: Highlights zero-shot image classification and research applications, supported by the model card for openai/clip-vit-base-patch32, which emphasizes zero-shot transfer.

RAG: Mentions image classification and zero-shot learning, noting limitations like fine-grained classification, consistent with the paper "[Learning Transferable Visual Models From Natural Language Supervision](#)".

Analysis: Both are accurate, with SPARQL providing more research context and RAG noting practical limitations, offering complementary insights.

---

#### Q4. What is the architecture of deepseekr10528?

---

SPARQL Answer: The model architecture of DeepSeek-R1-0528-Qwen3-8B is identical to that of Qwen3-8B, but it shares the same tokenizer configuration as DeepSeek-R1-0528.

identical to that of Qwen3-8B, but it shares the same tokenizer configuration as DeepSeek-R1-0528

RAG Answer: The model architecture of DeepSeek-R1-0528-Qwen3-8B is identical to that of Qwen3-8B, but the specific architecture of Qwen3-8B is not available in the current knowledge base.

**Comparison:**

SPARQL: States DeepSeek-R1-0528-Qwen3-8B is identical to Qwen3-8B, sharing tokenizer configuration with DeepSeek-R1-0528, confirmed by the model card for deepseek-ai/DeepSeek-R1-0528-Qwen3-8B.

RAG: Repeats the identity with Qwen3-8B but claims specific architecture details are unavailable, which is incorrect as the model card for Qwen/Qwen3-8B lists 36 layers and GQA.

Analysis: SPARQL is accurate, while RAG's claim of unavailability is a misstep, reducing its reliability here.

---

**Q5.** How does the use case of DeBERTaDecodingEnhancedBERTwithDisentangledAttention compare to that of ESMFold?

---

SPARQL Answer: Fine-tuning on NLU tasks (useCaseText2: If you're interested in using ESMFold in practice, please check out the associated [tutorial notebook] ([https://colab.research.google.com/github/huggingface/notebooks/blob/main/examples/protein\\_folding.ipynb](https://colab.research.google.com/github/huggingface/notebooks/blob/main/examples/protein_folding.ipynb))).

RAG Answer: DeBERTa improves BERT and RoBERTa models using disentangled attention, while ESMFold is a protein folding model.

**Comparison:**

SPARQL: Notes DeBERTa for NLU tasks and mentions ESMFold's tutorial, suggesting protein folding, supported by the model card for facebook/esmfold\_v1.

RAG: Clearly distinguishes DeBERTa for language model improvement and ESMFold for protein folding

Analysis: RAG provides a clearer comparison, while SPARQL is less focused, making RAG more useful for this query.

---

**Q6.** Which model has fewer than 1 million parameters?

---

SPARQL Answer: MCRO\_Qwen306B

RAG Answer: Based on the search results, Chronos-T5 models have fewer parameters than the original T5 models. Smaller BERT models are also mentioned as being designed for environments with restricted computational resources, suggesting they might have fewer parameters. However, I cannot definitively state a model with fewer than 1 million parameters.

**Comparison:**

SPARQL: Mentions MCRO\_Qwen306B, likely a typo for Qwen3-0.6B (600M parameters), far exceeding 1M, as per Qwen/Qwen3-0.6B.

RAG: Notes Chronos-T5 and smaller BERT models (e.g., prajjwal1/bert-tiny at ~4.5M parameters) but cannot definitively state a model under 1M, consistent with findings that pre-trained transformer models typically exceed 1M.

Analysis: RAG is more accurate, acknowledging the lack of such models, while SPARQL's answer is likely erroneous.

---

**Q7.** How does the citation of HuggingFaceM4idefics28b compare to that of IDEAREsearchgrounding-dinobase?

---

SPARQL Answer: `idefics2-8b-base` and `idefics2-8b` can be used to perform inference on multimodal (image + text) tasks in which the input is composed of a text query along with one (or multiple) image(s). Text and images can be arbitrarily interleaved. That includes image captioning, visual question answering, etc. (ideaResearchCitation: zero-shot object detection (the task of detecting things in an image out-of-the-box without labeled data))

RAG Answer: HuggingFaceM4idefics28b cites the Hugging Face Model Hub, a ViT paper, and the ImageNet-21k dataset. IDEAREsearchgroundingdinobase cites an IEEE paper. The citation information is different.

**Comparison:**

SPARQL: Mixes use cases and citations, mentioning zero-shot object detection for grounding-dino

RAG: States idefics2-8b cites Hugging Face Model Hub, ViT paper, and ImageNet-21k, while grounding-dino cites an IEEE paper, noting differences, as per model cards for HuggingFaceM4/idefics2-8b and IDEAREsearch/grounding-dino-base.

Analysis: RAG correctly identifies citation differences, while SPARQL is less clear, making RAG more reliable.

---

**Q8.** What are the differences in the use cases of vitmatte-model and llavahflavav16mistral7bhf?

---

SPARQL Answer: models?search=vitmatte) to look for other

fine-tuned versions that may interest you. ([llavahflavav16mistral7bfhUseCase](https://huggingface.co/models?search=llava-hf): You can use the raw model for tasks like image captioning, visual question answering, multimodal chatbot use cases. See the [model hub](<https://huggingface.co/models?search=llava-hf>) to look for other versions on a task that interests you.)

RAG Answer: vitmatte-model can be used for content creation, communication (text generation, chatbots, text summarization, image data extraction), research and education (NLP/VLM research, language learning tools, knowledge exploration). llavahflavav16mistral7bfh can be used for image captioning, visual question answering, and multimodal chatbot use cases. The main differences are that vitmatte-model has use cases more geared towards general text and language tasks, whereas llavahflavav16mistral7bfh is focused on multimodal tasks, particularly those involving images.

Comparison:

SPARQL: Implies vitmatte for image matting and llava for image captioning, VQA, and multimodal chat, supported by model cards for hustvl/vitmatte-base-composition-1k and llava-hf/llava-v1.6-mistral-7b-hf.

RAG: Incorrectly describes vitmatte for text and language tasks, while correctly noting llava for multimodal tasks, misaligning with vitmatte's image matting focus.

Analysis: SPARQL is accurate, while RAG's error reduces its reliability for this question.

---

#### Q9. Which model has a more permissive license: FlagEmbedding or HuggingFaceH4zephyr7bbeta?

---

SPARQL Answer: MCRO\_FlagEmbedding (licenseText: Retrieval-augmented LLMs)

MCRO\_FlagEmbedding (licenseText: FlagEmbedding focuses on retrieval-augmented LLMs, consisting of the following projects currently:

- \*\*Long-Context LLM\*\*: [Activation Beacon] ([https://github.com/FlagOpen/FlagEmbedding/tree/master/Long\\_LLM/activation\\_beacon](https://github.com/FlagOpen/FlagEmbedding/tree/master/Long_LLM/activation_beacon))
- \*\*Fine-tuning of LM\*\* : [LM-Cocktail] ([https://github.com/FlagOpen/FlagEmbedding/tree/master/LM\\_Cocktail](https://github.com/FlagOpen/FlagEmbedding/tree/master/LM_Cocktail))
- \*\*Dense Retrieval\*\*: [BGE-M3] ([https://github.com/FlagOpen/FlagEmbedding/tree/master/FlagEmbedding/BGE\\_M3](https://github.com/FlagOpen/FlagEmbedding/tree/master/FlagEmbedding/BGE_M3)), [LLM Embedder] ([https://github.com/FlagOpen/FlagEmbedding/tree/master/FlagEmbedding/llm\\_embedder](https://github.com/FlagOpen/FlagEmbedding/tree/master/FlagEmbedding/llm_embedder)), [BGE Embedding] ([https://github.com/FlagOpen/FlagEmbedding/tree/master/FlagEmbedding/baai\\_general\\_embedding](https://github.com/FlagOpen/FlagEmbedding/tree/master/FlagEmbedding/baai_general_embedding))
- \*\*Reranker Model\*\*: [BGE Reranker] (<https://github.com/FlagOpen/FlagEmbedding/tree/master/FlagEmbedding/reranker>)
- \*\*Benchmark\*\*: [C-MTEB] ([https://github.com/FlagOpen/FlagEmbedding/tree/master/C\\_MTEB](https://github.com/FlagOpen/FlagEmbedding/tree/master/C_MTEB))

RAG Answer: Both models have the MIT license, which is a permissive license.

Comparison:

SPARQL: Provides use case details, not license, for FlagEmbedding, while RAG states both have MIT licenses, confirmed by model cards for BAAI/bge-small-en and HuggingFaceH4/zephyr-7b-beta.

RAG: Clearly identifies the permissive license for both models.

Analysis: RAG is correct, noting both models share the permissive MIT license, while SPARQL is off-topic.

---

#### Q10. Why was the model architecture chosen for ESMFold?

---

SPARQL Answer: ESMFold is a state-of-the-art end-to-end protein folding model based on an ESM-2 backbone. ESMFold is a state-of-the-art end-to-end protein folding model based on an ESM-2 backbone. It does not require any lookup or MSA step, and therefore does not require any external databases to be present in order to make predictions. As a result, inference time is very significantly faster than AlphaFold2.

ESMFold is a state-of-the-art end-to-end protein folding model based on an ESM-2 backbone. It does not require any lookup or MSA step, and therefore does not require any external databases to be present in order to make predictions. As a result, inference time is very significantly faster than AlphaFold2.

RAG Answer: ESMFold uses an ESM-2 backbone, transformers with RoPE, SwiGLU, RMSNorm, Attention QKV bias and tied word embeddings and features like GEGLU activation in feed-forward hidden layer. These choices were made to improve the model's performance in protein folding tasks.

Comparison:

SPARQL: Highlights ESMFold's speed and lack of MSA requirement, supported by the model card for facebook/esmfold-v1, aligning with the paper's focus on efficiency.

RAG: Lists architectural features (ESM-2 backbone, RoPE, etc.) without explaining why chosen, less informative for the "why" query.

Analysis: SPARQL provides a better explanation, focusing on practical advantages.

---

#### Q11. What makes the dataset suitable for FinetunedT5SmallTextSummarizations use case?

---

---

SPARQL Answer: Text Summarization

RAG Answer: The datasets used are suitable for FinetunedT5SmallTextSummarization because they provide text suitable for summarization tasks, such as question-answering, document summarization, and conversational summarization. The model is pre-trained on C4 and fine-tuned on datasets like MS MARCO, NQ, Trivia QA, and ELI5, which provide different kinds of text suitable for summarization.

Comparison:

SPARQL: Simply states "Text Summarization," vague but correct, as per the model card for google-t5/t5-small.

RAG: Mentions datasets like MS MARCO (question-answering), not specifically for summarization, misaligning with the use case, while noting C4 pre-training.

Analysis: SPARQL is concise and correct, while RAG includes inaccuracies, reducing its reliability.

---

## Q12. What is the primary tradeoff of using LTXVideo?

---

SPARQL Answer: Prompts should be in English. The more elaborate the better. Good prompt looks like `The turquoise waves crash against the dark, jagged rocks of the shore, sending white foam spraying into the air. The scene is dominated by the stark contrast between the bright blue water and the dark, almost black rocks. The water is a clear, turquoise color, and the waves are capped with white foam. The rocks are dark and jagged, and they are covered in patches of green moss. The shore is lined with lush green vegetation, including trees and bushes. In the background, there are rolling hills covered in dense forest. The sky is cloudy, and the light is dim.'

The model works best on resolutions under 720 x 1280 and number of frames below 257.

The model works on resolutions that are divisible by 32 and number of frames that are divisible by 8 + 1 (e.g. 257). In case the resolution or number of frames are not divisible by 32 or 8 + 1, the input will be padded with -1 and then cropped to the desired resolution and number of frames.

RAG Answer: A primary tradeoff of using LTXVideo is likely its computational cost, due to its large model size and the computational demands of video processing.

Comparison:

SPARQL: Describes input constraints (resolution, frames), suggesting computational limits, supported by the model card for Lightricks/LTX-Video.

RAG: Identifies computational cost as the primary tradeoff, aligning with the model's video processing demands.

Both are relevant, but RAG directly addresses the tradeoff, making it more focused.

---

Question	SPARQL Accuracy	RAG Accuracy	Notes
1	High	Medium	SPARQL-specific, RAG contextual.
2	Medium	High	RAG comprehensive, SPARQL incomplete.
3	High	High	Both accurate, complementary.
4	High	Medium	SPARQL correct, RAG claims unavailability.
5	Medium	High	RAG clearer comparison, SPARQL less focused.
6	Low	High	RAG acknowledges no model under 1M, SPARQL likely typo.
7	Medium	High	RAG notes citation differences, SPARQL mixed.
8	High	Low	SPARQL correct, RAG misidentifies vitmatte.
9	Low	High	RAG correct on licenses, SPARQL off-topic.
10	High	Medium	SPARQL explains choice, RAG lists features.
11	High	Medium	SPARQL concise, RAG includes inaccuracies.
12	High	High	Both relevant, RAG more direct on tradeoff.

Table 4.5: SPARQL vs. RAG Accuracy Comparison

### 4.3.3 Summary of Findings

In addressing RQ3, the comparison between G-RAG and SPARQL for querying model cards reveals distinct strengths and limitations. Quantitatively, both methods perform comparably, with each achieving a Schema Compliance Accuracy of 75% and a Correct Output Rate of 66.7%. However, G-RAG demonstrates a notable advantage in Syntax Validity Rate, achieving 100% compared to SPARQLs 91.7%, suggesting higher reliability in generating syntactically correct queries. It is important to note that these percentages are based on the current dataset and may vary with larger or more diverse data samples. Qualitatively, SPARQL provides high precision but is highly sensitive to syntax errors, making it well-suited for tasks that demand exact and structured data retrieval. In contrast, G-RAG excels at delivering

context-rich responses, leveraging its generative capabilities, though it may introduce inaccuracies due to its probabilistic nature. Ultimately, SPARQL is preferable for tasks where precision and control are paramount, whereas G-RAG is better suited for applications that benefit from broader contextual understanding. The choice between the two depends on the specific requirements and tolerance for error in the target task.

Metric	G-RAG	SPARQL
Schema Compliance Accuracy	75%	75%
Correct Output Rate	66.7%	66.7%
Syntax Validity Rate	100%	91.7%
Strengths	Context-rich responses, generative capabilities	High precision, structured retrieval
Limitations	Potential inaccuracies due to probabilistic nature	Highly sensitive to syntax errors
Suitability	Best for tasks needing broader understanding	Best for tasks requiring precise, structured data

Table 4.6: Summary of findings comparing G-RAG and SPARQL for querying model cards

# Chapter 5

## Discussions

This chapter synthesizes and interprets the findings presented in Chapter 4. The discussion is structured around the core experimental themes of metadata extraction, natural language to SPARQL translation, and querying knowledge graphs. Each theme is reflected upon in terms of its motivations, implications, and the unexpected observations that arose during experimentation. The discussion also compares these results with existing literature and examines how they contribute to advancing the field of automated model documentation.

### 5.1 Ontology-Guided Metadata Structuring with LLMs

The core motivation behind this experimental theme was to explore how LLMs could be used to automatically extract metadata from model cards, making it more consistent and machine-readable. In this context, model cards are crucial documents that describe machine learning models, including metadata about their functionality, limitations, and ethical considerations. The idea was to guide the LLMs extraction process by using a predefined ontology MCRO to improve the consistency and format of the output. The findings showed that LLMs, when directed by an ontology like MCRO, are highly effective at extracting RDF triples (a standardized way of storing structured data) [4]. This capability allows for metadata to be more easily interpreted by machines, facilitating easier querying and analysis.

However, the results were not without limitations. The ontology-driven approach worked well for standardized sections of model cards, such as model architecture and performance metrics, where the structure is clear and consistent [98]. The extraction process, in this case, was mostly successful. But, when the LLM encountered less standardized sections, such as licensing information or hyperparameters, it struggled to produce accurate or complete outputs [89]. This is a significant finding because it highlights a limitation in the LLMs ability to deal with free-text sections, which often contain irregularities in structure or content. It was an unexpected observation that sections of model cards that seemed straightforward were quite complex when considered in the context of an ontology-based extraction process. This finding suggests that ontology-guided extraction systems require fine-tuning to handle less structured and free-form sections more effectively. This can be achieved through improved prompt engineering, validation mechanisms, and possibly incorporating more flexible rules to account for variability in source documents [89].

### 5.2 LLMs for Natural Language to SPARQL Translation

The second experimental theme was motivated by the need to enable natural language users (who may not be familiar with complex query languages, such as SPARQL), to query a knowledge graph easily. The idea was that by leveraging the capabilities of LLMs, one could convert natural language queries into SPARQL queries, allowing users to interact with knowledge graphs without needing specialized technical knowledge. The findings indicated that LLMs could generate syntactically valid SPARQL queries from simple, descriptive natural language inputs. For instance, if a user asked a question like What are the hyperparameters of model X?, the LLM was able to generate a valid SPARQL query that could retrieve that information from the knowledge graph [24, 19]

However, the system struggled with more complex queries, especially those that required multi-step reasoning or comparative analysis. When the queries demanded higher-order logical processing, such as determining the relationship between two models or resolving ambiguity between predicates, the LLM was prone to errors [57]. These errors included semantic drift, where the meaning of the query was lost during translation, incorrect predicate mappings (where the LLM chose the wrong relationship or entity), and missing entities (where the LLM failed to identify all relevant parts of the query). An unexpected finding here was that the limitations were not necessarily due to the LLMs language processing capabilities, but rather the misalignment between the users query and the ontology itself [57]. This means that, even though the LLM understood the natural language input, it struggled to apply that understanding to the structured format of the knowledge graph.

The implication is clear: while LLMs can be a powerful tool for translating natural language into structured queries, their performance is heavily dependent on the alignment between the users input and the underlying ontology. Improving this alignment, perhaps through better pre-filtering of the ontology or more nuanced prompt engineering, could help address these challenges. This experiment also highlights the potential benefits of incorporating a two-stage semantic parsing process, where the user query is first analyzed and grounded in the correct ontology elements before being converted into a query language, such as SPARQL [24].

### 5.3 Retrieval Augmentation and SPARQL Approaches

The third theme explored the use of Retrieval-Augmented Generation (RAG) as an alternative to traditional SPARQL queries for interacting with knowledge graphs. RAG, which combines the strengths of retrieval-based systems (retrieving relevant data from a knowledge base) and generative models (producing new text based on retrieved information), was evaluated to see if it could offer a more flexible and user-friendly querying experience [53]. The comparison between RAG and SPARQL revealed that RAG offered an advantage in terms of conversational flexibility, particularly for queries that are loosely structured or vague. For example, if a user asked an open-ended question, RAG could generate a more fluid and natural response compared to the rigid structure of a SPARQL query [23].

However, the RAG-based system also faced significant challenges. Despite its natural conversational abilities, RAG occasionally generated hallucinations, incorrect or fabricated facts, or retrieved semantically irrelevant information [103]. This undermined the accuracy of the answers. The more rigid SPARQL, although less flexible, was able to provide more deterministic and precise answers, especially for well-structured and unambiguous queries. An unexpected observation was that RAGs generative flexibility came with a trade-off in factual correctness, especially when the queries required multi-step reasoning or complex relationships between entities [107]. SPARQL, on the other hand, was able to handle these structured queries more reliably, albeit at the cost of flexibility.

This highlights an important implication: while RAG has the potential to enhance user experience by offering more natural interactions, it needs to be paired with mechanisms to improve the precision and correctness of its responses. Additionally, RAG was found to struggle with multi-hop inferencesqueries that require the system to connect multiple pieces of information in the graph. This was likely due to the lack of a long-context memory or explicit ontology-guided fusion strategies, which could be addressed in future research by embedding more sophisticated semantic grounding techniques into the retrieval process [107].

### 5.4 Cross-Cutting Insights

Across all experiments, several cross-cutting insights emerged that address broader themes and challenges in utilizing LLMs for model card metadata extraction and knowledge graph querying. One of the most notable themes was the impact of documentation heterogeneity on system performance. The system performed best when the model cards adhered to standardized formats, enabling the LLMs to more easily extract and structure the metadata [70, 58]. However, when the model cards varied in structure, with incomplete or missing sections, the LLMs were unable to extract the relevant data consistently. This highlights the importance of standardizing documentation practices within the AI community to facilitate improved automation and consistency in metadata extraction.

Another key insight was the role of ontology quality in the performance of the LLM-based systems. The

MCRO ontology provided a solid backbone for metadata extraction; however, there were gaps in its ability to handle edge-case metadata, such as environmental impacts or responsible AI metrics [6]. This suggests that the ontology needs to evolve to capture more nuanced information, which would, in turn, enhance the overall completeness of the system. Expert feedback also emerged as a critical factor in refining the system. Experts helped identify misalignments between the LLM outputs and the ontology, demonstrating the value of a human-in-the-loop approach in fine-tuning the system and ensuring its outputs align with expert knowledge [113].

## 5.5 Threats to Validity

This section critically examines the potential threats to validity that may influence the interpretation and generalizability of the results presented in this thesis. These threats are divided into internal and external validity, which help assess both the accuracy of the findings within the scope of this study and their applicability to broader contexts.

### 5.5.1 Internal Validity

Internal validity concerns whether the observed effects in this study can be attributed to the experimental interventions (e.g., specific metadata extraction methods or query generation strategies), rather than to confounding factors. In this work, several potential threats to internal validity are acknowledged:

#### Expert Review Dependency

Expert reviews played a critical role in assessing the systems performance. However, expert assessments can introduce subjectivity into the evaluation process. Different experts may interpret the results in diverse ways, potentially leading to inconsistencies in evaluation. This subjectivity introduces a threat to internal validity, as the findings may be more reflective of the reviewers' interpretations rather than the system's inherent capabilities.

#### Query Complexity and Ambiguity

The handling of complex, comparative, or reasoning-based queries introduces another threat to internal validity. Natural language queries are often ambiguous and can be phrased in various ways by different users [122]. This variability introduces the potential for inconsistent results. Additionally, the systems difficulty in handling multi-hop reasoning or advanced logical constructs may lead to errors in processing specific queries, which could affect the accuracy of the conclusions drawn from the systems output [60]. The observed performance of the system may therefore be influenced more by the inherent complexity of user queries than by the interventions themselves [122, 60].

#### Prompting

The reliance on static prompts for RDF triple extraction poses a significant threat to internal validity. The design of these prompts is tailored to the current metadata structure present in model cards. As model card formats evolve or as new metadata fields are introduced, the static nature of these prompts may result in reduced accuracy, and the observed effects may only apply to the existing metadata structure. This limitation undermines the generalizability of the systems effectiveness over time, making it difficult to attribute improvements solely to the interventions implemented in this study.

### 5.5.2 External Validity

External validity refers to the extent to which the findings can be generalized beyond the specific conditions of this study. Several key limitations to external validity are noted:

#### Ontology and Schema Quality

The MCRO ontology, which served as the backbone for metadata extraction and query generation, presented both strengths and weaknesses. While it was effective for most structured metadata fields, the ontology struggled to handle edge cases, such as environmental impact or responsible AI metrics, which are increasingly crucial for model cards. This limitation poses a threat to external validity, as the findings

of this study may not apply to other ontologies that better handle these edge cases or to future model card metadata categories that the MCRO ontology does not fully support. To enhance generalizability, future systems should integrate more comprehensive ontologies that can represent a broader range of metadata.

### Documentation Heterogeneity

A major threat to external validity is the heterogeneity of model card documentation across repositories. While model cards that adhere to standardized formats allowed the system to perform effectively, those that deviated from this structure led to reduced accuracy in metadata extraction. This suggests that the systems findings may not generalize to other repositories that do not follow the same formatting conventions. As model card structures continue to evolve, the system's ability to maintain its performance across different datasets or repositories may be compromised, limiting the applicability of the results [58].

### Multilingual and Noisy Inputs

While the dataset includes model cards in multiple languages, the system was primarily tested and evaluated on content written in or partially resembling English. Multilingual text introduces inconsistencies in metadata formatting, which can impair both prompt-based extraction and RDF mapping. Additionally, cards with mixed-language content or non-ASCII characters may pose parsing challenges. Hence, although multilingual data is present, the studys conclusions are most robustly supported in scenarios where model cards are predominantly linguistically close to the training and prompting distribution.

# Chapter 6

## Conclusions

This chapter distils the main insights and implications emerging from the empirical and theoretical investigations of this thesis. It systematically addresses each research question by synthesizing quantitative results and analytical perspectives on metadata extraction, ontology alignment, SPARQL query generation, and comparative retrieval mechanisms. The chapter further articulates the core research contributions and concludes with actionable recommendations for both system designers and academic researchers. In doing so, it offers a comprehensive perspective on the strengths, challenges, and future directions of LLM-assisted, ontology-grounded model cataloging systems.

### 6.1 Answers to Research Questions

This section summarizes the findings of each research question posed at the outset of this thesis, integrating the empirical results and analytical insights derived from the architectural components and evaluation trials.

**RQ1: To what extent can leveraging LLMs for ontology-driven metadata extraction improve the consistency, completeness, and machine-readability of model card information?**

The analysis conducted in this thesis demonstrates that LLMs, when guided by a structured ontology and schema-constrained prompts, significantly improve the consistency and machine-readability of model card metadata. By using an RDF-based schema aligned with the MCRO ontology, the study demonstrates that LLMs are capable of extracting structured metadata with high syntactic accuracy across a range of model cards. The consistency was particularly noticeable in high-frequency fields like `model.name`, `language`, and `license`, though challenges arose in less standardized sections, such as `hyperparameters` and `training.details`, where inconsistencies and under-extraction were more common. Completeness of the extracted metadata varied depending on the quality of the source documents. In cases where the source card included vague or incomplete information, the LLM was able to make educated inferences, sometimes generating triples that were not explicitly stated but logically implied. However, this ability also highlighted an unexpected risk: overgeneration, where the LLM produced triples that might not have been grounded in the original text. Overall, this thesis shows that ontology-driven LLMs can significantly improve the machine-readability and consistency of model cards, although limitations persist when the source material lacks clarity or standardization.

**RQ2: To what extent can LLMs support natural language (NL) queries over a semantic model catalogue?**

The analysis conducted in this thesis indicates that LLMs are capable of translating a broad range of natural language queries into valid SPARQL queries when accessing a semantic model catalogue. The results show that LLMs excel at answering straightforward fact-based queries, such as retrieving specific model properties or details from the knowledge graph, when given clear and descriptive NL

queries. However, the system faced challenges with more complex queries, such as those requiring multi-hop reasoning, disambiguation, or advanced logical operations. The ability of the LLM to generate syntactically correct queries was often hindered by the complexity of the question and the need to map nuanced NL structures to specific ontology terms. This observation highlights the importance of prompt engineering and ontology exposure: when provided with adequate schema context, LLMs performed significantly better, with fewer errors and more accurate results. The analysis also revealed that while LLMs offer a user-friendly interface for querying, the performance still depends heavily on how well the system internalizes the ontology and the clarity of the natural language input. This confirms that LLMs can indeed support natural language queries, but their effectiveness is influenced by the depth of their reasoning capabilities and the precision of the ontological mappings.

**RQ3: To what extent does G-RAG compare to SPARQL in querying a semantic model catalogue?**

The analysis conducted in this thesis compares the use of G-RAG and SPARQL querying for accessing a semantic model catalogue, revealing complementary strengths and weaknesses. G-RAG, which integrates a graph retrieval layer with a generative model, proved to be more accessible to non-technical users, as it effectively handled conversational and imprecise queries. It was robust in scenarios where the user had vague or incomplete knowledge about the query subject. However, this approach had limitations in terms of factual accuracy. Some answers generated by the system were not fully aligned with the ground truth, as the model sometimes retrieved semantically irrelevant triples or hallucinated data. On the other hand, SPARQL querying was precise, deterministic, and ensured exact alignment with the ontology schema, making it suitable for technical users or cases where accuracy was paramount. The analysis revealed that while SPARQL excels at strict, schema-aligned querying, G-RAG provides greater flexibility and fluency in answering exploratory or less-structured questions. These findings suggest that G-RAG and SPARQL should be seen as complementary tools: G-RAG for ease of use and flexibility, and SPARQL for precision and reliability. Future systems could integrate both methods, allowing users to select the most suitable query mechanism based on the task's complexity and their level of expertise.

## 6.2 Research Contributions

This thesis makes several key contributions to the field of ontology-driven metadata extraction and querying within the context of semantic model cataloging:

First, it introduces a novel methodology for LLMs to automate ontology-driven metadata extraction from model cards. This approach utilizes the MCRO ontology to improve the consistency, completeness, and machine-readability of model card information. By applying this ontology-based framework, the thesis demonstrates how LLMs can be used to transform unstructured model card data into structured, semantically rich metadata that enhances model transparency and accessibility.

Second, the thesis presents an in-depth evaluation of the use of LLMs in supporting NL queries over a semantic model catalogue. The study quantifies the effectiveness of LLMs in translating user-friendly NL queries into formal SPARQL queries, thereby facilitating more intuitive and accessible interactions with semantic knowledge graphs. This contribution significantly expands the potential applications of LLMs in querying and navigating complex, large-scale semantic data models, offering both theoretical insights and practical implications for researchers and practitioners.

Third, this research conducts a comparative analysis of G-RAG versus SPARQL in querying a semantic model catalogue. The study highlights the strengths and limitations of both approaches in a multi-database setting, where G-RAG excels in flexibility and ease of use. At the same time, SPARQL ensures greater schema compliance and precision. This comparison lays the foundation for future research into hybrid approaches that combine generative models with formal querying systems, potentially offering a more robust solution for querying large and complex model catalogs.

In addition, this thesis empirically evaluates the performance of LLM-based systems for metadata extraction and querying, presenting quantitative analyses of system accuracy, completeness, and the ability to handle NL queries. These evaluations offer valuable insights into the strengths and limitations of current LLM techniques and provide a benchmark for further improvements in automating metadata extraction and querying in real-world applications.

## 6.3 Industrial Recommendations

This thesis offers several key insights and recommendations for industries seeking to enhance the consistency, completeness, and machine-readability of model card information through ontology-driven metadata extraction, support natural language queries over semantic model catalogs, and compare the effectiveness of G-RAG and SPARQL in querying semantic model catalogs.

### 6.3.1 Implement Ontology-Driven Metadata Extraction for Model Cards

Implement widely recognized ontologies, such as MCRO, to consistently represent model card metadata. Develop or integrate tools that can automatically extract relevant information from model documentation and populate the ontology-based model cards. Design systems that can consume and produce model cards in standardized formats, enabling easy sharing and integration across platforms.

### 6.3.2 Enhance Natural Language Querying Over Semantic Model Catalogues

Combine NLP models with semantic query engines to interpret and process natural language queries effectively. Create intuitive interfaces that allow users to input queries in natural language and receive understandable responses. Implement feedback mechanisms to refine the systems understanding of user queries and improve accuracy over time.

### 6.3.3 Invest in Performance

Streamline algorithms and workflows to reduce the time and resources required for metadata extraction. Implement indexing, caching, and other optimization techniques to improve the speed and efficiency of query processing. Design systems that can scale horizontally to accommodate increasing data volumes and user loads. By implementing these recommendations, industries can significantly improve the management, querying, and usability of model card information.

## 6.4 Academic Recommendations

This thesis identifies several avenues for advancing the state of research in the automation of model metadata extraction and the construction of knowledge graphs from model cards.

A key recommendation is to develop and rigorously evaluate hybrid extraction strategies that combine rule-based methods, large language models, and domain-specific ontologies. Hybrid systems can flexibly adapt to different types of model metadata, leveraging rule-based systems for structured elements like model architecture and training data, LLMs for unstructured content such as performance details and limitations, and ontologies for domain-specific linking. Orchestrating these mechanisms within a unified extraction pipeline has the potential to improve both the accuracy and completeness of model metadata, while enabling empirical analysis of trade-offs between precision, recall, and contextual understanding in real-world model card tasks [5].

Additionally, future research should explore iterative and modular ontology refinement frameworks. Recent advances in ontology-based metadata extraction suggest the benefits of continuously refining the ontology used for model card metadata. By adopting an iterative approach, involving feedback loops from both human experts and machine learning models, systems can dynamically evolve to accommodate new model types and metadata structures. Modular architectures would facilitate the integration of components like query rewriting, response filtering, and self-correction, enabling dynamic adaptation to evolving model card formats and user needs. By generalizing iterative refinement and pipeline modularity, researchers can develop more robust and adaptable systems for large-scale model metadata extraction [48].

Furthermore, integrating knowledge graphs and external context into metadata extraction pipelines remains a promising area of study. KGs can be leveraged to resolve entity ambiguity, disambiguate schema references, and enable more complex reasoning over multi-faceted model data. Incorporating KGs into the pipeline could significantly enhance the accuracy of metadata extraction, support sophisticated queries over model cards, and facilitate reasoning across multiple levels of metadata, such as relationships between model performance, training data, and deployment scenarios [99].

#### *6.4. Academic Recommendations*

---

Importantly, there is a growing need to train and evaluate model metadata extraction systems under adversarial conditions. Datasets introducing controlled noise, incomplete metadata, and schema variations could challenge models to generalize under more realistic, difficult, or intentionally misleading conditions. Incorporating adversarial training and robust benchmarking against such datasets would reveal vulnerabilities, strengthen the systems ability to handle noisy or evolving data, and ensure that the model card metadata extraction system performs reliably in real-world scenarios.[?]

By pursuing these directions, academic research can establish new standards of rigor and generalizability for automated model card metadata extraction, knowledge graph construction, and querying, thereby advancing the state of AI model management and transparency.

# Bibliography

- [1] Mohiuddin Ahmed and Jinpeng Wei. Prompting llm to enforce and validate cis critical security control. *University of North Carolina at Charlotte, Software and Information Systems*, 2025. unpublished. 31
- [2] Dean Allemang and Juan Sequeda. Increasing the accuracy of llm question-answering systems with ontologies. In Stefan Decker et al., editors, *The Semantic Web ISWC 2024*, volume 15233 of *Lecture Notes in Computer Science*, pages 324–339. Springer, 2024. 1, 3
- [3] Dean Allemang and Juan Sequeda. Increasing the llm accuracy for question answering: Ontologies to the rescue! Technical report, data.world AI Lab, 2024. 4
- [4] Zaid Alyafeai, Maged S. Al-Shaibani, and Bernard Ghanem. Mole: Metadata extraction and validation in scientific papers using llms, May 2025. Preprint available at <https://arxiv.org/abs/2505.19800>. 70
- [5] Zaid Alyafeai, Maged S. AlShaibani, and Bernard Ghanem. Mole: Metadata extraction and validation in scientific papers using LLMs, 2025. 76
- [6] Muhammad Tuan Amith, Licong Cui, Degui Zhi, Kirk Roberts, Xiaoqian Jiang, Fang Li, Evan Yu, and Cui Tao. Toward a standard formal semantic representation of the model card report. *BMC Bioinformatics*, 23(Suppl 6):281, July 2022. 7, 8, 9, 72
- [7] Marco Arazzi, Davide Ligari, Serena Nicolazzo, and Antonino Nocera. Augmented knowledge graph querying leveraging LLMs. *arXiv preprint arXiv:2502.01298*, February 2025. 2, 17
- [8] Basic Formal Ontology Project. Basic Formal Ontology (BFO). <https://basic-formal-ontology.org/>, 2025. Accessed: 2025-07-01. 33
- [9] Tim Berners-Lee and Dan Connolly. Rdf: A vocabulary description language for the web. W3C Recommendation, 2004. 8
- [10] Mark Birbeck and Shane McCarron. Curie syntax 1.0. World Wide Web Consortium (W3C), December 2010. W3C Working Group Note. 33
- [11] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. 11, 14
- [12] Marc Bruni, Fabio Gabrielli, Mohammad Ghafari, and Martin Kropp. Benchmarking prompt engineering techniques for secure code generation with gpt models. *arXiv preprint arXiv:2502.06039*, 2025. Accepted at Forge 2025 (IEEE/ACM 2nd International Conference on AI Foundation Models and Software Engineering). 31
- [13] J. Harry Caufield, Harshad Hegde, Vincent Emonet, Nomi L. Harris, Marcin P. Joachimiak, Nicholas Matentzoglu, HyeongSik Kim, Sierra A. T. Moxon, Justin T. Reese, Melissa A. Haendel, Peter N. Robinson, and Christopher J. Mungall. Structured prompt interrogation and recursive extraction of semantics (spires): A method for populating knowledge bases using zero-shot learning. *Bioinformatics*, 40(3):btae104, March 2024. 11

- [14] Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. Unleashing the potential of prompt engineering in large language models: A comprehensive review. *arXiv preprint arXiv:2310.14735*, 2023. 13, 14
- [15] Ruixi Chen. Retrieval-augmented generation with knowledge graphs: A survey, March 2025. 15, 16
- [16] Gustavo M. C. Coelho, Eduardo R. S. Nascimento, Yenier T. Izquierdo, Grettel M. Garca, Lucas Feij, Melissa Lemos, Robinson L. S. Garcia, Aiko R. de Oliveira, Joo P. Pinheiro, and Marco A. Casanova. Improving the accuracy of text-to-sql tools based on large language models for real-world relational databases. In *Proceedings of the 35th International Conference on Database and Expert Systems Applications (DEXA 2024)*, pages 93–107. Springer, 2024. 4
- [17] Google DeepMind. Gemini, 2023. Accessed: 2025-07-01. 22
- [18] Information Artifact Ontology Developers. Information artifact ontology (iao). <http://purl.obolibrary.org/obo/iao.owl>, 2021. 9, 33
- [19] Papa Abdou Karim Karou Diallo, Samuel Reyd, and Amal Zouaq. A comprehensive evaluation of neural sparql query generation from natural language questions. *arXiv preprint arXiv:2304.07772*, 2023. Version 3, last revised 11 January 2024. 70
- [20] Haomin Dong, Wenbin Wang, Zhenjiang Sun, Ziyi Kang, Xiaojun Ge, Fei Gao, and Jixin Wang. Knowledge graph construction for intelligent cockpits based on large language models. *Scientific Reports*, 15, 2025. Open Access. 18
- [21] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024. 15
- [22] Lisa Ehrlinger and Werner W. Towards a definition of knowledge graphs. In *Proceedings of SEMANTiCS 2016: Posters, Demos & SuCESS Track*, volume 1695 of *CEUR Workshop Proceedings*, 2016. 10
- [23] Vincent Emonet, Jerven Bolleman, Severine Duvaud, Tarçisio Mendes de Farias, and Ana Claudia Sima. Llm-based sparql query generation from natural language over federated knowledge graphs. *arXiv preprint arXiv:2410.06062*, 2024. Accessed: 2025-07-01. 71
- [24] Vincent Emonet, Jerven Bolleman, Severine Duvaud, Tarçisio Mendes de Farias, and Ana Claudia Sima. Llm-based sparql query generation from natural language over federated knowledge graphs. *arXiv preprint arXiv:2410.06062*, 2025. Version 4, last revised 10 February 2025. 70, 71
- [25] Vincent Emonet, Jerven Bolleman, Severine Duvaud, Tarçisio Mendes de Farias, and Ana Claudia Sima. Llm-based sparql query generation from natural language over federated knowledge graphs. *arXiv preprint arXiv:2410.06062*, 2024. 2
- [26] Vincent Emonet, Jerven Bolleman, Severine Duvaud, Tarçisio Mendes de Farias, and Ana Claudia Sima. Llmbased sparql query generation from natural language over federated knowledge graphs, February 2025. 17
- [27] John S. Erickson, Henrique Santos, Vládia Pinheiro, Jamie P. McCusker, and Deborah L. McGuinness. Llm experimentation through knowledge graphs: Towards improved management, repeatability, and verification. *Journal of Web Semantics*, 85:100853, 2024. Open Access under CC BY-NC-ND 4.0 license. 17
- [28] Apache Software Foundation. Apache jena, 2025. Accessed: 2025-06-28. 8
- [29] Chengcheng Fu, Xueli Pan, Jieyu Wu, Junkai Cai, Zhisheng Huang, Frank van Harmelen, Weizhong Zhao, Xingpeng Jiang, and Tingting He. Kg4nh: A comprehensive knowledge graph for question answering in dietary nutrition and human health. *IEEE Journal of Biomedical and Health Informatics*, 29(3):1793–1804, March 2025. 41
- [30] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023. 3

- [31] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. <https://arxiv.org/abs/2312.10997>, March 2024. arXiv:2312.10997 [cs]. 15
- [32] Gartner, Inc. Gartner predicts 70% of enterprises adopting genai will cite sustainability and digital sovereignty as top criteria for selecting between different public cloud genai services by 2027. Press release, February 2024. 1
- [33] GeeksforGeeks. Relationship extraction in nlp. <https://www.geeksforgeeks.org/relationship-extraction-in-nlp/>, Apr 2025. Last updated on April 28, 2025; accessed 20250701. 16
- [34] Joan GinerMiguel, Abel Gmez, and Jordi Cabot. Using large language models to enrich the documentation of datasets for machine learning, April 2024. 17
- [35] Louie Giray. Prompt engineering with chatgpt: A guide for academic writers. *Annals of Biomedical Engineering*, 51(12):2629–2633, December 2023. 13
- [36] GitHub. Github. <https://github.com/>, 2025. 22
- [37] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5–6):907–928, 1995. 7
- [38] Ian Horrocks, Peter F. Patel-Schneider, et al. Owl web ontology language overview. W3C Recommendation, 2004. 8
- [39] Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. Grag: Graph retrieval-augmented generation. <https://arxiv.org/abs/2405.16506>, October 2024. arXiv:2405.16506. 16
- [40] Nam Huynh and Beiyu Lin. Large language models for code generation: A comprehensive survey of challenges, techniques, evaluation, and applications. *arXiv preprint arXiv:2503.01245*, 2025. 12
- [41] IBM. Named entity recognition (ner). <https://www.ibm.com/think/topics/named-entity-recognition>, n.d. Accessed: 2025-07-01. 16
- [42] Nourhan Ibrahim, Samar Aboulela, Ahmed Ibrahim, and Rasha Kashef. A survey on augmenting knowledge graphs (kgs) with large language models (llms): models, evaluation metrics, benchmarks, and challenges. *Discover Artificial Intelligence*, 4, 2024. 18
- [43] Neo4j Inc. Neo4j graph database, 2025. 19
- [44] Runsong Jia, Bowen Zhang, Sergio J. Rodríguez Méndez, and Pouya G. Omran. Leveraging large language models for semantic query processing in a scholarly knowledge graph, May 2024. 17
- [45] Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*, 2024. 11
- [46] Kaggle. Kaggle. <https://www.kaggle.com/>, 2025. 22
- [47] Vamsi Krishna Kommineni, Birgitta Knig-Ries, and Sheeba Samuel. Docs2kg: A human-llm collaborative approach to unified knowledge graph construction from documents. In *Proceedings of the ACM Conference on Knowledge, Data and Discovery (KDD)*. ACM, May 2025. 1
- [48] Vamsi Krishna Kommineni, Birgitta KnigRies, and Sheeba Samuel. From human experts to machines: An llm supported approach to ontology and knowledge graph construction. *arXiv preprint arXiv:2403.08345*, 2024. 1, 76
- [49] Ana Kostovska, Jasmin Bogatinovski, Sašo Džeroski, Dragi Kočev, and Panče Panov. A catalogue with semantic annotations makes multilabel datasets fair. *Scientific Reports*, 12, 2022. 1, 20
- [50] Humam Kourani, Alberto Berti, Dorothea Schuster, and Wil M. P. van der Aalst. Process modeling with large language models. In Han van der Aa, Dominik Bork, Rainer Schmidt, and Arnon Sturm, editors, *Enterprise, BusinessProcess and Information Systems Modeling*, volume 511 of *Lecture Notes in Business Information Processing*, pages 229–244. Springer, Cham, 2024. 27, 28, 34
- [51] LangChain. Langchain documentation, 2025. Accessed: 2025-07-01. 19

- [52] Jens Lehmann, Preetam Gattogi, Dhananjay Bhandiwad, Sbastien Ferr, and Sahar Vahdat. Language models as controlled natural language semantic parsers for knowledge graph question answering. 2023. 4, 13, 20, 23, 35, 42, 47
- [53] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kttler, Mike Lewis, Wen tau Yih, Tim Rocktschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020. 71
- [54] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kttler, Mike Lewis, Wen-tau Yih, Tim Rocktschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474, 2020. 15
- [55] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kttler, Mike Lewis, Wen-tau Yih, Tim Rocktschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*, 2020. 2
- [56] Lei Liang, Mengshu Sun, Zhengke Gui, Zhongshu Zhu, Zhouyu Jiang, Ling Zhong, Yuan Qu, Peilong Zhao, Zhongpu Bo, Jin Yang, Huaidong Xiong, Lin Yuan, Jun Xu, Zaoyang Wang, Zhiqiang Zhang, Wen Zhang, Huajun Chen, Wenguang Chen, and Jun Zhou. Kag: Boosting llms in professional domains via knowledge augmented generation, 2024. 4, 20, 44
- [57] Shiqi Liang, Kurt Stockinger, Tarcisio Mendes de Farias, Maria Anisimova, and Manuel Gil. Querying knowledge graphs in natural language. *Journal of Big Data*, 8(3), 2021. 71
- [58] Weixin Liang, Nazneen Rajani, Xinyu Yang, Ezinwanne Ozoani, Eric Wu, Yiqun Chen, Daniel Scott Smith, and James Zou. What’s documented in ai? systematic analysis of 32k ai model cards. *arXiv preprint arXiv:2402.05160*, 2024. 1, 2, 4, 7, 20, 23, 71, 73, 86
- [59] Q. Vera Liao and Jennifer Wortman Vaughan. Ai transparency in the age of llms: A human-centered research roadmap. *Harvard Data Science Review*, Special Issue 5, 2024. 3
- [60] Xi Victoria Lin, Richard Socher, and Caiming Xiong. Multi-hop knowledge graph reasoning with reward shaping. *arXiv preprint arXiv:1808.10568*, 2018. 72
- [61] Yifan Liu, Zhenyu Zhang, Xinyu Li, Yujie Zhang, Zhiwei Luo, and Qiang Yang. Empowering large language models by hybrid retrieval-augmented generation for industrial applications. *Computers in Industry*, 143:103758, 2025. Open Access under CC BY-NC-ND 4.0 license. 18
- [62] Andy Lo, Albert Qiaochu Jiang, Wenda Li, and Mateja Jamnik. End-to-end ontology learning with large language models. *CoRR*, abs/2410.23584, 2024. arXiv preprint. 7
- [63] Chuangtao Ma, Yongrui Chen, Tianxing Wu, Arijit Khan, and Haofen Wang. Large language models meet knowledge graphs for question answering: Synthesis and opportunities. *arXiv preprint arXiv:2505.20099v1*, 2025. Under review. 18
- [64] Yury A. Malkov and Dmitry A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2020. 16
- [65] James Malone, Andy Brown, Allyson L. Lister, Jon Ison, Duncan Hull, Helen Parkinson, and Robert Stevens. The software ontology (swo): a resource for reproducibility in biomedical data analysis, curation and digital preservation. *Journal of Biomedical Semantics*, 5:25, 2014. 9
- [66] Adnan Masood. The knowledge graph advantage: How smart companies are using knowledge graphs to power ai and drive real-world results, 2025. 18
- [67] LarsPeter Meyer, Johannes Frey, Felix Brei, and Natanael Arndt. Assessing sparql capabilities of large language models, April 2025. 17
- [68] Alistair Miles and Sean Bechhofer. Skos simple knowledge organization system - home page. <https://www.w3.org/2004/02/skos/intro>, 2009. W3C Working Group Note, 18 August 2009.

- [69] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 2018. 6, 7, 10
- [70] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT\*)*. ACM, January 2019. 1, 71
- [71] Luc Moreau, Timothy Lebo, Satya Sahoo, Khalid Belhajjame, Deborah McGuinness, James Cheney, and Simon Miles. Prov-o: The prov ontology. <https://www.w3.org/TR/prov-o/>, 2013. W3C Recommendation, 30 April 2013. 9
- [72] Michalis Mountantonakis et al. Generating sparql queries over cidoc-crm using a two-stage approach. *Journal on Computing and Cultural Heritage (JOCCH)*, 18(1):1–22, 2025. 38
- [73] National Academies of Sciences, Engineering, and Medicine. *Understanding Ontologies*, chapter 3. National Academies Press (US), Washington, DC, May 2022. Division of Behavioral and Social Sciences and Education; Board on Behavioral, Cognitive, and Sensory Sciences; Committee on Accelerating Behavioral Science through Ontology Development and Use. 7
- [74] Neo4j. Cypher graph query language, 2025. Accessed: 2025-07-01. 19
- [75] Neo4j Labs. Apoc documentation, 2025. Accessed: 2025-07-01. 45
- [76] Bo Ni, Zheyuan Liu, Leyao Wang, Yongjia Lei, Yuying Zhao, Xueqi Cheng, Qingkai Zeng, Luna Dong, Yinglong Xia, Krishnaram Kenthapadi, Ryan Rossi, Franck Dernoncourt, Md Mehrab Tanjim, Nesreen Ahmed, Xiaorui Liu, Wenqi Fan, Erik Blasch, Yu Wang, Meng Jiang, and Tyler Derr. Towards trustworthy retrieval augmented generation for large language models: A survey. *arXiv preprint arXiv:2502.06872*, 2025. 3
- [77] Bo Ni, Zheyuan Liu, Leyao Wang, Yongjia Lei, Yuying Zhao, Xueqi Cheng, Qingkai Zeng, Luna Dong, Yinglong Xia, Krishnaram Kenthapadi, Ryan Rossi, Franck Dernoncourt, Md Mehrab Tanjim, Nesreen Ahmed, Xiaorui Liu, Wenqi Fan, Erik Blasch, Yu Wang, Meng Jiang, and Tyler Derr. Towards trustworthy retrieval augmented generation for large language models: A survey. <https://arxiv.org/abs/2502.06872>, February 2025. arXiv:2502.06872 [cs]. 15
- [78] Sanaz Saki Norouzi, Adrita Barua, Antrea Christou, Nikita Gautam, Andrew Eells, Pascal Hitzler, and Cogan Shimizu. Ontology population using llms. *arXiv preprint arXiv:2411.01612*, 2024. 1
- [79] José Luiz Nunes, G. D. J. Barbosa, C. S. D. Souza, and Simone D. J. Barbosa. Using model cards for ethical reflection on machine learning models: an interview-based study. *J. Interact. Syst.*, 15:1–19, 2024. 6
- [80] Jos Luiz Nunes, Gabriel D. J. Barbosa, Clárisse Sieckenius de Souza, and Simone D. J. Barbosa. Using model cards for ethical reflection on machine learning models: an interview-based study. *Journal on Interactive Systems*, 15(1):1–19, Jan. 2024. 2
- [81] OBO Foundry / IAO. Information artifact ontology (iao). <http://purl.obolibrary.org/obo/iao.owl>, 2022. VersionIRI: 20221107; Licensed under CC BY 4.0; Contact: JieZheng. 9, 33
- [82] Ontotext. Graphdb, 2025. Accessed: 2025-06-28. 8
- [83] Ontotext. Graphdb, 2025. 19
- [84] Ontotext. Graphdb 11.0 documentation, 2025. Accessed: 2025-07-01. 19
- [85] Ontotext. GraphDB: Enterpriseready semantic graph database. <https://graphdb.ontotext.com/>, 2025. Accessed: 2025-07-01. 31
- [86] Ontotext. What is sparql? <https://www.ontotext.com/knowledgehub/fundamentals/what-is-sparql/>, 2025. Accessed: 2025-07-01. 7
- [87] Shuyin Ouyang, Jie M. Zhang, Zeyu Sun, and Albert Merono Penuela. Knowledge-enhanced program repair for data science code. *arXiv preprint arXiv:2502.09771*, 2025. 4, 20

- [88] Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. Graph retrieval-augmented generation: A survey. <https://arxiv.org/abs/2408.08921>, September 2024. arXiv:2408.08921 [cs]. 15, 16
- [89] Maciej P. Polak and Dane Morgan. Extracting accurate materials data from research papers with conversational language models and prompt engineering. *Nature Communications*, 15:1569, 2024. 70
- [90] Eric Prud'hommeaux and Andy Seaborne. Sparql query language for rdf. In *W3C Recommendation*, 2008. 9
- [91] Xiao Pu, Mingqi Gao, and Xiaojun Wan. Summarization is (almost) dead. *arXiv preprint arXiv:2309.09558*, 2023. 11
- [92] André Gomes Regino, Rodrigo Oliveira Caus, Victor Hochgreb, and Julio Cesar dos Reis. From natural language texts to rdf triples: A novel approach to generating ecommerce knowledge graphs. In *Knowledge Discovery, Knowledge Engineering and Knowledge Management*, volume 149174 of *Communications in Computer and Information Science*, pages 149–174. Springer, 2023. 17
- [93] Chanatip Saetia, Jiratha Phruethiset, Tawunrat Chalothorn, Monchai Lertsutthiwong, Supawat Taerungruang, and Pakpoom Buabthong. Financial product ontology population with large language models. *Unpublished*, 2025. unpublished. 31
- [94] Aya Sahbi, Cline Alec, and Pierre Beust. Semantic vs. llm-based approach: A case study of konpote vs. claude for ontology population from french advertisements. *Data & Knowledge Engineering*, 156:102392, 2025. 23
- [95] Aya Sahbi, Cline Alec, and Pierre Beust. Semantic vs. llm-based approach: A case study of konpote vs. claude for ontology population from french advertisements. *Data & Knowledge Engineering*, 156:102392, 2025. 31, 35, 47
- [96] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*, 2024. 14
- [97] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*, 2024. 14
- [98] Mara Schilling-Wilhelmi, Martíno Ríos-García, Sherjeel Shabih, María Victoria Gil, Santiago Miret, Christoph T. Koch, José A. Márquez, and Kevin Maik Jablonka. From text to insight: Large language models for materials science data extraction. *arXiv preprint*, arXiv:2407.16867, July 2024. Condensed Matter Materials Science (cond-mat.mtrlsci), revised version v2 Dec22024. 70
- [99] Mara SchillingWilhelmi, Marti no RíosGarcía, Sherjeel Shabih, María Victoria Gil, Santiago Miret, Christoph T. Koch, José A. Mrquez, and Kevin Maik Jablonka. From text to insight: Large language models for materials science data extraction, 2024. Submitted 23 Jul 2024; version 2 updated 2 Dec 2024. 76
- [100] Amazon Science. Language models as controlled natural language semantic parsers for knowledge graph question answering. <https://www.amazon.science/publications/language-models-as-controlled-natural-language-semantic-parsers-for-knowledge-graph-question-answering> 2025. 12
- [101] Jiawei Shen, Chengcheng Wan, Ruoyi Qiao, Jiazen Zou, Hang Xu, Yuchen Shao, Yueming Zhang, Weikai Miao, and Geguang Pu. A study of in-context-learning-based text-to-sql errors. *arXiv preprint arXiv:2501.09310*, 2025. 14
- [102] Cogan Shimizu and Pascal Hitzler. Accelerating knowledge graph and ontology engineering with large language models. *arXiv preprint arXiv:2411.09601*, 2024. 10
- [103] Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*, 2021. Accessed: 2025-07-01. 71

- [104] Tommaso Soru and Maxime Marx. Sparql as a foreign language. In *The Semantic Web ISWC 2021*, pages 232–247. Springer International Publishing, 2021. 12
- [105] SPDX Workgroup. *SPDX 3.0.1 Specification*. SPDX Project, Linux Foundation, December 2024. Version 3.0.11. 31
- [106] Suhas Jayaram Subramanya, Fnu Devvrit, Harsha Vardhan Simhadri, Ravishankar Krishnawamy, and Rohan Kadekodi. Diskann: Fast accurate billionpoint nearest neighbor search on a single node. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, page , 2019. 16
- [107] Yixuan Tang and Yi Yang. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv preprint arXiv:2401.15391*, 2024. Accessed: 2025-07-01. 71
- [108] The Hermit Team. Hermit reasoner, 2023. Accessed: 2025-07-01. 25
- [109] Shengwei Tian, Lifeng Han, and Goran Nenadic. Autollm-card: Towards a description and landscape of large language models. *arXiv preprint arXiv:2409.17011*, 2024. 3
- [110] Shengwei Tian, Lifeng Han, and Goran Nenadic. Autollm-card: Towards a description and landscape of large language models. *arXiv preprint arXiv:2409.17011*, 2024. Accepted for publication in the International Journal on Digital Libraries. 10, 16, 17
- [111] Catherine Tony, Nicols E. Daz Ferreyra, Markus Mutas, Salem Dhiff, and Riccardo Scandariato. Prompting techniques for secure code generation: A systematic investigation. *arXiv preprint arXiv:2407.07064*, 2024. Accepted at ACM Transactions on Software Engineering and Methodology (Feb 2025). 31
- [112] Milena Trajanoska, Riste Stojanov, and Dimitar Trajanov. Enhancing knowledge graph construction using large language models, 2023. Preprint on arXiv. 8, 10
- [113] Stefani Tsaneva, Danilo Dess, Francesco Osborne, and Marta Sabou. Knowledge graph validation by integrating llms and human-in-the-loop. *Information Processing & Management*, 62(5):104145, 2025. 72
- [114] D. Varagnolo, D. Pimenta Melo, and I. Rodrigues. Generating sparql queries over cidoc-crm using a two-stage approach. *J. Comput. Cult. Herit.*, 18(1):1–20, 2025. 20
- [115] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint*, arXiv:1706.03762, 2017. 11
- [116] Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. *Proceedings of the VLDB Endowment*, 14(1):1–14, 2021. arXiv:2101.12631v2. 16
- [117] Ahmed M. Wei and Ehab Al-Shaer. Prompting llm to enforce and validate cis critical security control. In *Proceedings of the 29th ACM Symposium on Access Control Models and Technologies (SACMAT 2024)*, pages 93–104. Association for Computing Machinery, 2024. 4, 46
- [118] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierrick Cistac, Tim Rault, Rmi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. <https://huggingface.co>, 2020. 19
- [119] Qinyuan Ye, Maxamed Axmed, Reid Pryzant, and Fereshte Khani. Prompt engineering a prompt engineer. <https://arxiv.org/abs/2311.05661>, July 2024. arXiv:2311.05661 [cs]. 13
- [120] Qinggang Zhang, Shengyuan Chen, Yuanchen Bei, Zheng Yuan, Huachi Zhou, Zijin Hong, Junnan Dong, Hao Chen, Yi Chang, and Xiao Huang. A survey of graph retrievalaugmented generation for customized large language models, January 2025. 17
- [121] Yang Zhang, Caiqi Liu, Mujiexin Liu, Tianyuan Liu, Hao Lin, Cheng-Bing Huang, and Lin Ning. Attention is all you need: Utilizing attention in ai-enabled drug discovery. *Briefings in Bioinformatics*, 25(1):bbad467, January 2024. 11

- [122] Yifan Zhang, Zhiqiang Wang, Xueqi Zhang, and Lei Zhang. Knowledge graph question answering with ambiguous query. In *Proceedings of the ACM Web Conference 2023*, pages 2477–2486. ACM, 2023. 72
- [123] Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. A comprehensive survey on automatic knowledge graph construction. *ACM Computing Surveys*, 56(4):1–62, 2024. 10
- [124] Xishi Zhu, Xiaoming Guo, Shengting Cao, Shenglin Li, et al. Structugraphrag: Structured document-informed knowledge graphs for retrieval-augmented generation. In *Proceedings of the AAAI Symposium Series*, volume 4, pages 242–251, 2024. 17
- [125] Yuqi Zhu, Xiaohan Wang, Jing Chen, Shuofei Qiao, Yixin Ou, Yunzhi Yao, Shumin Deng, Huajun Chen, and Ningyu Zhang. Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *arXiv preprint arXiv:2305.13168*, 2023. 8

# List of Figures

1.1	Figure 1: Overview of Hugging Face Model Repositories and Model Card Adoption.(a) Exponential Growth of Model Repositories: The number of models hosted on Hugging Face has grown rapidly, with a weekly growth rate of 3.16% and a doubling time of 22 weeks. As of October 1st, 2022, there are 74,970 models available. (b) Temporal Trends in Model Popularity: Analysis of the top-5 trending model repositories reveals shifts in focus from language models to image generation models over time. (c) Model Card Adoption and Download Traffic: While only 42.8% of models (32,111 out of 74,970) feature model cards, these models account for 90.5% of the total download traffic, emphasizing the critical role of documentation in model usage. (d) Model Card Example: An illustrative example of a typical model card, showing key metadata elements including model purpose, performance, and limitations [58] . . . . .	2
2.1	Model Card metadata of Model StarCoder from HuggingFace . . . . .	7
2.2	A key component of large language models is the Transformer architecture. Every input token is embedded and sent through multi-head attention stacking encoder-decoder blocks. . . . .	12
2.3	Process of Converting a Natural Language Query to a SPARQL Query . . . . .	13
2.4	Zero-shot learning: The LLM is given only an instruction and input without examples. . . . .	14
2.5	Few-shot learning: The LLM receives two examples and an instruction to perform the task. . . . .	15
2.6	Typical RAG workflow . . . . .	15
3.1	Overall System Architecture . . . . .	21
3.2	Semantic augmentation pipeline applied to LLM-extracted metadata . . . . .	22
3.3	Generating normalized RDF triples . . . . .	22
3.4	Comparing Gemini (Left) and LLama (Right) Output . . . . .	24
3.5	BGE-M3 Structure Validate using property Assertion . . . . .	25
3.6	Sample of Extracted_triples.ttl in Neo4j . . . . .	26
3.7	Preprocessing Pipeline of Ontology Mapper . . . . .	27
3.8	Ontology Mapper - ModelCards to KG . . . . .	30
3.9	Knowledge Graph Question Answer Pipeline . . . . .	39
3.10	Neo4j Graph with Starting Node “UseCaseInformationSection” . . . . .	44
3.11	RAG-Neo4j Pipleine . . . . .	45
1	Three types of questions and Examples . . . . .	92

# List of Tables

3.1 Examples of Questions Used in KGQA . . . . .	43
4.1 Quantitative Results Addressing RQ1 . . . . .	49
4.2 Expert 1 Feedback on Model Cards . . . . .	50
4.3 Quantitative Results for RQ2 . . . . .	51
4.4 Evaluation Metrics Comparison . . . . .	64
4.5 SPARQL vs. RAG Accuracy Comparison . . . . .	68
4.6 Summary of findings comparing G-RAG and SPARQL for querying model cards . . . . .	69

# Listings

2.1	Sample of Model Card Ontology . . . . .	7
2.2	Prefix Sample of MCRO . . . . .	9
2.3	Sample SPARQL query to retrieves all models with their creators . . . . .	9
2.4	Basic example of KG Components . . . . .	10
3.1	Final Prompt - RDF Triple Extraction with MCRO Ontology . . . . .	28
3.2	Prompt - Schema-Based Extraction with SPDX Validation . . . . .	31
3.3	Prompt - Validation-Heavy Structured Extraction . . . . .	32
3.4	Prompt - Ontology-Based Extraction with RDF Triples . . . . .	33
3.5	Pipeline 2 : Final Prompt . . . . .	40
3.6	Latex Formatting removal . . . . .	46
3.7	Pipeline 3 - Final Prompt . . . . .	46

## .1 Triples

File path json format: Ontology\_mapper/Output/1/triples.json  
File path turtle format: Ontology\_mapper/Output/1/triples.ttl

## .2 Result

### **.2.1 Research Question2 - Q2**

```

\begin{itemize}
    \item \texttt{macro:RoBERTa\_base\_model} \\
    "The RoBERTa model was pretrained on the reunion of five datasets:
\begin{itemize}
    \item \texttt{macro:BookCorpus}, a dataset consisting of 11,038 unpublished books;
    \item \texttt{macro:English Wikipedia} (excluding lists, tables and headers);
    \item \texttt{macro:CC-News}, a dataset containing 63 million English news articles crawled between September 2016 and February 2019;
    \item \texttt{macro:OpenWebText}, an open-source recreation of the WebText dataset used to train GPT-2;
    \item \texttt{macro:Stories}, a dataset containing a subset of CommonCrawl data filtered to match the story-like style of Winograd schemas.
\end{itemize}
\end{itemize}
Together these datasets weigh 160GB of text.

\texttt{macro:robertalargemodel} "BookCorpus"
\texttt{macro:robertalargemodel} "English Wikipedia"
\texttt{macro:robertalargemode} "CC-News"
\texttt{macro:robertalargemode} "OpenWebText"
\texttt{macro:robertalargemode} "Stories"
\texttt{macro:RoBERTaLargemode} "BookCorpus"
\texttt{macro:RoBERTaLargemode} "English Wikipedia"
\texttt{macro:RoBERTaLargemode} "CC-News"
\texttt{macro:RoBERTaLargemode} "OpenWebText"
\texttt{macro:RoBERTaLargemode} "Stories"
\texttt{macro:RoBERTaBaseModel} "BookCorpus"
\texttt{macro:RoBERTaBaseModel} "English Wikipedia"
\texttt{macro:RoBERTaBaseModel} "CC-News"
\texttt{macro:RoBERTaBaseModel} "OpenWebText"
\texttt{macro:RoBERTaBaseModel} "Stories"
\texttt{macro:RoBERTaLargeModel} "The RoBERTa model was pretrained on the reunion of five datasets:
\begin{itemize}
    \item \texttt{macro:BookCorpus}, a dataset consisting of 11,038 unpublished books;
    \item \texttt{macro:English Wikipedia} (excluding lists, tables and headers);
    \item \texttt{macro:CC-News}, a dataset containing 63 million English news articles crawled between September 2016 and February 2019;
    \item \texttt{macro:OpenWebText}, an open-source recreation of the WebText dataset used to train GPT-2;
    \item \texttt{macro:Stories}, a dataset containing a subset of CommonCrawl data filtered to match the story-like style of Winograd schemas.
\end{itemize}
\end{itemize}
\end{itemize}
\texttt{macro:all-distilroberta-v1} "We use the concatenation from multiple datasets to fine-tune our model. The total number of sentence pairs is above 1 billion sentences. We sampled each dataset given a weighted probability which configuration is detailed in the \texttt{data\_config.json} file.
\begin{tabular}{|c|c|c|c|}
\hline
Dataset & Paper & Number of training tuples \\
\hline
\texttt{https://github.com/PolyAI-LDN/conversational-datasets/tree/master/reddit} & \texttt{Reddit comments (2015-2018)} & \texttt{https://arxiv.org/abs/1904.06472} & 726,484,430 \\
\texttt{https://github.com/allenai/s2orc} & \texttt{S2ORC (Citation pairs - Abstracts)} & \texttt{https://aclanthology.org/2020.acl-main.447} & 116,288,806 \\
\texttt{https://github.com/afader/oqa#wikianswers-corpus} & \texttt{WikiAnswers (Duplicate question pairs)} & \texttt{https://doi.org/10.1145/2623330.2623677} & 77,427,422
\end{tabular}

```

---

```

\href{https://github.com/facebookresearch/PAQ}{PAQ (Question, Answer pairs)} & \href{https://arxiv.org/abs/2102.07033}{paper} & 64,371,441 \\
\href{https://github.com/allenai/s2orc}{S2ORC (Citation pairs - Titles)} & \href{https://aclanthology.org/2020.acl-main.447/}{paper} & 52,603,982 \\
\href{https://github.com/allenai/s2orc}{S2ORC (Title, Abstract)} & \href{https://aclanthology.org/2020.acl-main.447/}{paper} & 41,769,185 \\
\href{https://huggingface.co/datasets/flax-sentence-embeddings/stackexchange_xml}{Stack Exchange (Title, Body pairs)} & - & 25,316,456 \\
\href{https://microsoft.github.io/msmarco/}{MS MARCO (triplets)} & \href{https://doi.org/10.1145/3404835.3462804}{paper} & 9,144,553 \\
\href{https://github.com/allenai/gooaq}{GOOAQ: Open Question Answering with Diverse Answer Types} & \href{https://arxiv.org/pdf/2104.08727.pdf}{paper} & 3,012,496 \\
\hline
\textbf{Total} & & \textbf{1,124,818,467} \\
\end{tabular}

\item \texttt{\textbf{mcro:allrobertalargev1}} "We use the concatenation from multiple datasets to fine-tune our model. The total number of sentence pairs is above 1 billion sentences. We sampled each dataset given a weighted probability which configuration is detailed in the \texttt{\textbf{data}\_config.json} file."
\item \texttt{\textbf{mcro:obideidrobertai2b2}} "* Steps on how this model was trained can be found here: \href{https://github.com/obi-ml-public/ehr_deidentification/tree/master/steps/train}{Training}. The ``model\_name\_or\_path'' was set to: ``roberta-large''."
\item \texttt{\textbf{mcro:xlmrobertabaselanguagedetection}} "Language Identification dataset, which consists of text sequences in 20 languages"
\end{itemize}

```

---

## .2.2 Reasearch Question2 - Q6

```

\texttt{\textbf{mcro:Qwen306B OR mcro:fashionclip-ModelDetail}}
"The model uses a ViT-B/32 Transformer architecture as an image encoder and uses a masked self-attention Transformer as a text encoder. These encoders are trained, starting from a pre-trained checkpoint, to maximize the similarity of (image, text) pairs via a contrastive loss on a fashion dataset containing 800K products."

\texttt{\textbf{mcro:flux1dev}}
"12 billion parameter rectified flow transformer"

\texttt{\textbf{mcro:jonatasgrosmanwav2vec2largexlsr53chinesezhcn}}
"Fine-tuned [facebook/wav2vec2-large-xlsr-53](https://huggingface.co/facebook/wav2vec2-large-xlsr-53) on Chinese using the train and validation splits of [Common Voice 6.1](https://huggingface.co/datasets/common_voice), [CSS10](https://github.com/Kyubyong/css10) and [ST-CMDS](http://www.openslr.org/38/). When using this model, make sure that your speech input is sampled at 16kHz."

\texttt{\textbf{mcro:visiontransformerbase}}
"Vision Transformer (ViT) model pre-trained on ImageNet-21k (14 million images, 21,843 classes) at resolution 224x224"

\texttt{\textbf{mcro:Qwen306B}}
"16 for Q and 8 for KV"

\texttt{\textbf{mcro:Qwen306B}}
"28"

\texttt{\textbf{mcro:debertaV3Small}}
"6 layers"

\texttt{\textbf{mcro:debertaV3Small}}
"vocabulary containing 128K tokens"

\texttt{\textbf{mcro:googlegemma31bit}}
"Gemma is a family of lightweight, state-of-the-art open models from Google, built from the same research and technology used to create the Gemini models. Gemma 3 models are multimodal, handling text and image input and generating text output, with open weights for both pre-trained variants and instruction-tuned variants. Gemma 3 has a large, 128K context window, multilingual support in over 140 languages, and is available in more sizes than previous versions. Gemma 3 models are well-suited for a variety of text generation and image understanding tasks, including question answering, summarization, and reasoning. Their relatively small size makes it possible to deploy them in environments with limited resources such as laptops, desktops or your own cloud infrastructure, democratizing access to state of the art AI models and helping foster innovation for everyone."

\texttt{\textbf{mcro:hubertbase}}

```

### .3. Methodology

---

"The base model pretrained on 16kHz sampled speech audio. When using the model make sure that your speech input is also sampled at 16Khz."

mcro:microsoftPhi35visioninstruct  
"\*\*Architecture:\*\* Phi-3.5-vision has 4.2B parameters and contains image encoder, connector, projector, and Phi-3 Mini language model.<br> \*\*Inputs:\*\* Text and Image. Its best suited for prompts using the chat format.<br> \*\*Context length:\*\* 128K tokens<br> \*\*GPUs:\*\* 256 A100-80G<br> \*\*Training time:\*\* 6 days<br> \*\*Training data:\*\* 500B tokens (vision tokens + text tokens)<br> \*\*Outputs:\*\* Generated text in response to the input<br> \*\*Dates:\*\* Trained between July and August 2024<br> \*\*Status:\*\* This is a static model trained on an offline text dataset with cutoff date March 15, 2024. Future versions of the tuned models may be released as we improve models.<br> \*\*Release date:\*\* August 2024<br>"

mcro:microsoftPhi3mini128kinstruct-ModelDetail  
"Phi-3 Mini-128K-Instruct has 3.8B parameters and is a dense decoder-only Transformer model. The model is fine-tuned with Supervised fine-tuning (SFT) and Direct Preference Optimization (DPO) to ensure alignment with human preferences and safety guidelines."

mcro:microsoftdebertav3base  
"The DeBERTa V3 base model comes with 12 layers and a hidden size of 768. It has only 86M backbone parameters with a vocabulary containing 128K tokens which introduces 98M parameters in the Embedding layer. This model was trained using the 160GB data as DeBERTa V2."

mcro:microsoftmdebertav3base  
"mDeBERTa is multilingual version of DeBERTa which use the same structure as DeBERTa and was trained with CC100 multilingual data. The mDeBERTa V3 base model comes with 12 layers and a hidden size of 768. It has 86M backbone parameters with a vocabulary containing 250K tokens which introduces 190M parameters in the Embedding layer. This model was trained using the 2.5T CC100 data as XLM-R."

mcro:microsoftwavlmbaseplussv  
"The model was pretrained on 16kHz sampled speech audio with utterance and speaker contrastive loss."

mcro:sat-3l-sm  
"3 Transfomer layers"

mcro:segformerb0finetunedade20k  
"SegFormer consists of a hierarchical Transformer encoder and a lightweight all-MLP decode head to achieve great results on semantic segmentation benchmarks such as ADE20K and Cityscapes. The hierarchical Transformer is first pre-trained on ImageNet-1k, after which a decode head is added and fine-tuned altogether on a downstream dataset."

mcro:visiontransformerbase-sizedmodel  
"Vision Transformer (ViT) model pre-trained on ImageNet-21k (14 million images, 21,843 classes) at resolution 224x224, and fine-tuned on ImageNet 2012 (1 million images, 1,000 classes) at resolution 224x224."

mcro:visiontransformerbasesizedmodel  
"Vision Transformer (ViT) model pre-trained on ImageNet-21k (14 million images, 21,843 classes) at resolution 224x224."

mcro:wav2vec2-baseforemotionrecognition  
"This is a ported version of [S3PRL's Wav2Vec2 for the SUPERB Emotion Recognition task](<https://github.com/s3prl/s3prl/tree/master/s3prl/downstream/emotion>). The base model is [wav2vec2-base](<https://huggingface.co/facebook/wav2vec2-base>), which is pretrained on 16kHz sampled speech audio. When using the model make sure that your speech input is also sampled at 16Khz. For more information refer to [SUPERB: Speech processing Universal PERformance Benchmark](<https://arxiv.org/abs/2105.01051>)"

---

## .3 Methodology

### .3.1 3.6.3 Evaluation Metrics and Process

**TABLE III**  
THREE TYPES OF QUESTIONS AND EXAMPLES

Type	Description	Template <sup>1</sup>	Example
Description	You want to understand the meaning/introduction/ attribute of a concept/event.	How much {N} is in {F}? What is the biological hierarchy of {N}? What are the {DC} related to nutrition?	How much vitamin B is in apples? What is the biological hierarchy of glutamate? What are the congenital diseases related to nutrition?
Comparison	You want to compare/calculate/contrast two or more things.	Which food has higher {N} between {F <sub>1</sub> } and {F <sub>2</sub> }? How much {N} is in a dietary pattern that includes {F <sub>1</sub> } and {F <sub>2</sub> }? Are there shared nutrient associations between {DC <sub>1</sub> } and {DC <sub>2</sub> }?	Which food has higher cholesterol between beef and beans? How much vitamin B is in a dietary pattern that includes egg and milk? Are there shared nutrient associations between colon cancer and diabetes?
Reason	You want to find out the reasons of an event/effect.	How does {N} affect the gut microbiota? Why do {FC} affect human health? Why does food affect {D}?	How does vitamin A affect the gut microbiota? Why do soft drinks affect human health? Why does food affect anemia?

<sup>1</sup>: {N}:{Nutrients}, {F}:{Food}, {DC}:{Diseasecategory}, {D}:{Disease}, {FC}:{Food category}

Figure 1: Three types of questions and Examples