PROJECT MILESTONE 1

Introduction:
This report aims to provide an overview of the progress made in Milestone 1 of the project "Ontology Validation using Language Models and Knowledge Graphs." This milestone primarily focused on completing Tasks 1 and 2 from the proposal, initially setting up the Gemini-pro API and developing code to extract ontologies from DBpedia.

Database Setup and Ontologies Extraction from DBpedia:
The preprocessing program fetches information from DBpedia, a database of structured information extracted from Wikipedia. It uses the SPARQL query language to retrieve data about distinct classes (types of entities) from DBpedia. Then, for each class, it fetches instances of that class along with their properties and values and saves this information in RDF format (a way to represent data in a graph format) in separate files for each class.

The step-by-step flow for the setup of the Ontology structure is described as follows:-

1. Set up the SPARQL endpoint: Create an instance of SPARQLWrapper for the DBpedia SPARQL endpoint.

2. Fetch distinct classes: Use a SPARQL query to retrieve distinct classes from DBpedia. This query looks for entities that are instances of some class.

3. Post-processing: Filter out only those classes that belong to the DBpedia ontology (classes whose URIs start with "http://dbpedia.org/ontology/"). For each ontology class, run a SPARQL query to fetch instances of that class along with their properties and values.

4. Create an RDF graph: Create a new RDF graph using rdflib.Graph().Add triples to the graph: For each result of the SPARQL query, add a triple (subject, predicate, object) to the RDF graph.

5. Create a subfolder for the ontology class: Extract the ontology class name from its URI and create a subfolder to store the RDF data for that class. Save the graph to a TTL file: Serialize the RDF graph in Turtle format (TTL) and save it to a file in the subfolder created for the ontology class.

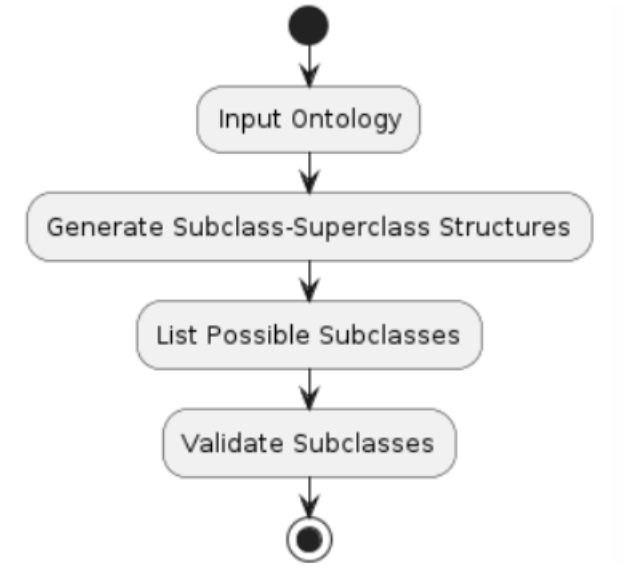
Setting up Gemini-pro API:

1. As part of the milestone, we successfully set up the Gemini-pro API. This API provides access to advanced language models, which will be instrumental in our ontology validation process. By integrating Gemini-pro, we gain access to state-of-the-art language understanding capabilities, enabling more accurate analysis of ontology data.
2. To utilize this API, we only need to download some libraries and set up an API key.

3. We needed to configure a project on Google AI Studio per our project's specific requirements.

Tasks Completed:

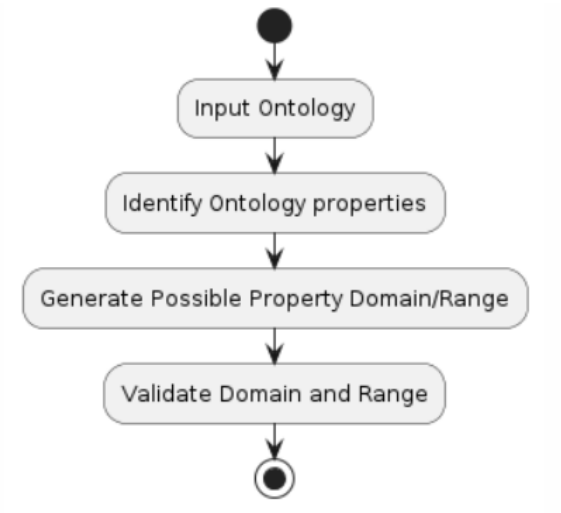
Task 1: Ontology Hierarchy Validation
The first task involved validating the hierarchy within the ontology. We experimented with a few prompts to generate proper hierarchies from ontologies for this particular LLM and followed the following framework to validate the hierarchy:



We use a language model to validate an ontology by generating subclass-superclass structures, identifying possible subclasses for each superclass, and validating the identified subclasses. We first create a query based on the project's introduction and ontology file, then generate responses for ontology hierarchy validation tasks. The responses are displayed, and the process is repeated for identifying subclasses and validating them against the superclass.

Task 2: Property Domain/Range Validation

In Task 2, we focused on validating the domain and range of properties within the ontology. This involved examining the properties defined in the ontology and verifying that their domains and ranges were correctly specified.



We first extract the properties in the ontology using LLM. We then use the LLM to generate predictions for the domain and range of each property in your ontology. Compare the predicted domains and ranges with the actual domains and ranges in your ontology. If there are discrepancies, it indicates issues with the validity of the property domain/range.

Individual Contribution:

Sarthak Maini (2020576):
Database Setup and Ontology Extraction, Task 1: Class Hierarchy Validation

Aaditya Gupta (2020552):
LLM API Setup and Properties extraction, Task 2: Property Domain/Range Validation