Project Documentation

1. Objective

The primary objective of this project is to conduct advanced infrastructure projects data analysis using a knowledge graph and Large Language Models (LLMs). Specifically, the project aims to leverage techniques such as ensemble models and entity-relationship mapping to interpret World Bank infrastructure project data. The key goals include inferring relationships between governments and companies, identifying frequent collaborators, and gaining insights into project networks.

2. Overview

2.1 Data Source

The project utilizes data from World Bank Projects, accessible through Projects.Worldbank.org. This data provides detailed information about the World Bank's lending projects globally, including project status, financing details, and sectors involved.

The data used here is a cleaned version of the raw project data provided by the worldbank.org. The data cleansing is done primarily using Pandas.The data contains 22307 records with 27 columns.The cleaned data contains only 22137 entries with 15 columns;

The cleaned data can be downloaded at this link :
https://drive.google.com/file/d/1a6_XemPLYi1xUSRFSTQhDjKkckZ8Gyl8/view?usp=sharing

2.2 Tools Used

- **Neo4j Graph Database:** The CSV data from World Bank Projects has been converted into a Neo4j graph database. The database schema includes three node types: Country, Region, and Projects, connected by relationships DONE_IN and BELONGS_TO.
- **Large Language Model (LLM):** The project employs the Lang chains GraphCypherQAChain module and Google's Palm 2 model as the Large Language Model. This combination allows the conversion of natural language queries into Cypher queries for information retrieval.
- **Gradio for Chatbot UI:** Gradio is utilized to create a user-friendly chatbot interface. This enables users to interact with the project and obtain information using natural language queries.

3. Workflow

3.1 Data Conversion and Graph Creation

1. **Data Extraction:** The World Bank Projects data is sourced from Projects.Worldbank.org.The data is cleaned and processed by using pandas and numpy libraries and the n exported to a CSV file as per the requirements Of neo4j

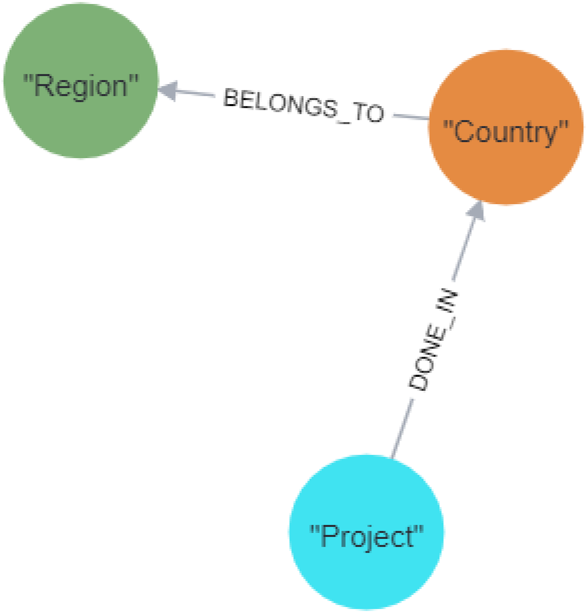
Data Transformation: The CSV data is transformed into a format suitable of graph and relations for Neo4j graph database representation.

2. **Graph Creation:** Using Neo4j, a graph database is created with three node types (Country, Region, and Projects) and two relationships (DONE_IN and BELONGS_TO).Below given are the properties of each node and their relationships

1 id : every world bank project has has a unique 7 character id starting with P

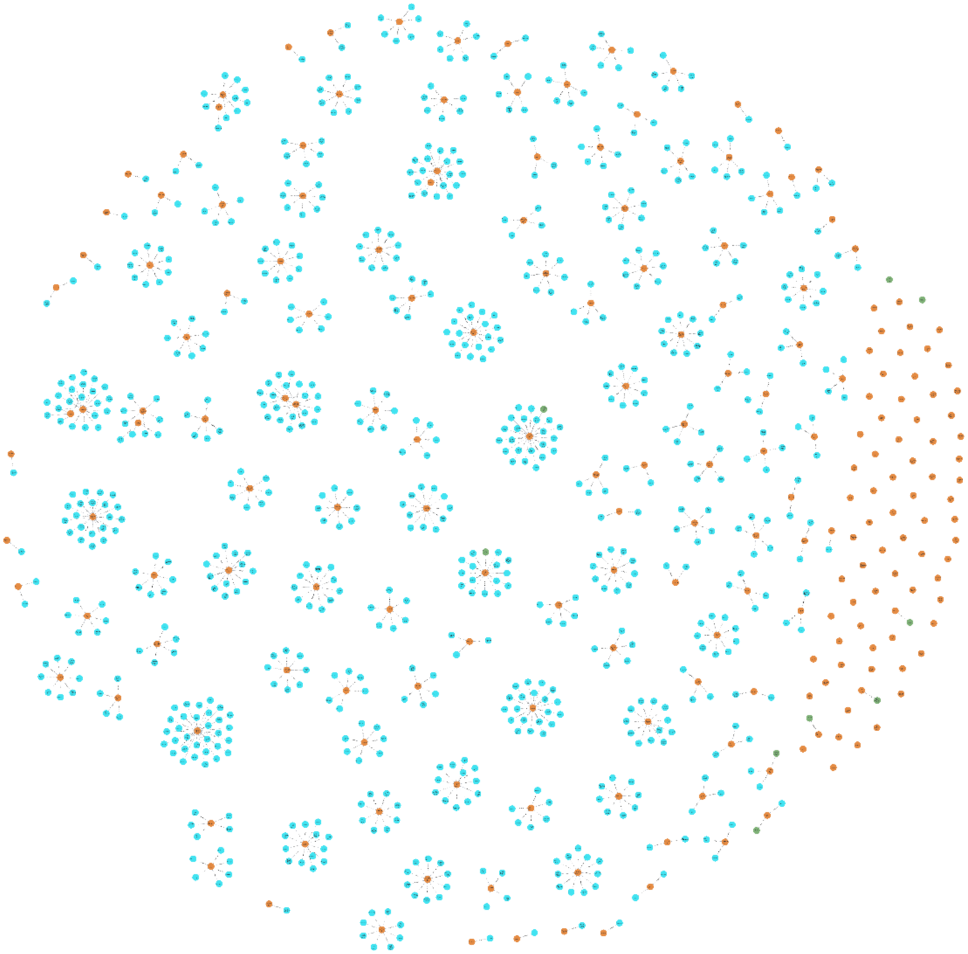
1 URL : a string value having the url of each project
2 agency : showing the agency which implements the project
3 cname : country in which the project is being done
4 current_cost : float value that is current cost the project has caused
5 grant : the amount the particular project has recieved in the form of grant in float value
6 l : lending instrument used by the worldbank ,it is a string value
7 name : name of the project
8 objective : the objective the company or the bank wishes to achieve by implementing the project
9 status : current status of the project , it may have values Pipeline,Dropped,Completed etc
10 total_commitment : float value which says the amount in total required fro the project completion

2.1 Schema:



Graph contains 3 types of nodes with labels Country,Project and Region with relationship DONE_IN between Project and Country ,and BELONGS_TO between Region and Country

2.2 Graph



The cypher query required to convert the CSV to Neo4j Graph is give in the document : https://docs.google.com/document/d/1WcyFLOLBSSDwUJndrElvHzqqwS9dXWYxuvj6KEutJsY/edit?usp=sharing

After this step the graph is hosted on AuraDB cloud instance

3.2 Large Language Model (LLM) Integration

1. **Lang Chains GraphCypherQAChain Module:** This module is integrated to facilitate the conversion of natural language queries into Cypher queries for querying the Neo4j graph.

   GraphCypherQAChain is a LangChain chain for question-answering against a graph by generating Cypher statements. It is a powerful tool that allows you to query your graph database using natural language, even if you are not familiar with Cypher.

   To use GraphCypherQAChain, you simply need to provide it with a question in natural language and it will generate a Cypher statement that you can then execute against your graph database. The chain is able to generate Cypher statements for a wide range of question types, including:

Entity search: "Find all movies that Tom Hanks has starred in."

Relationship search: "Find all friends of friends of John Doe."

Aggregation: "What is the average rating of movies directed by Steven Spielberg?"

Pathfinding: "Find the shortest path between two nodes in a graph."

GraphCypherQAChain is still under development, but it has already been used to create a variety of applications, including:

A natural language interface to a knowledge graph

A chatbot that can answer questions about a company's internal data

A search engine for graph databases

To see more information on the GraphQAChain check out the official documentation of Langchain : https://python.langchain.com/docs/use_cases/graph/graph_cypher_qa

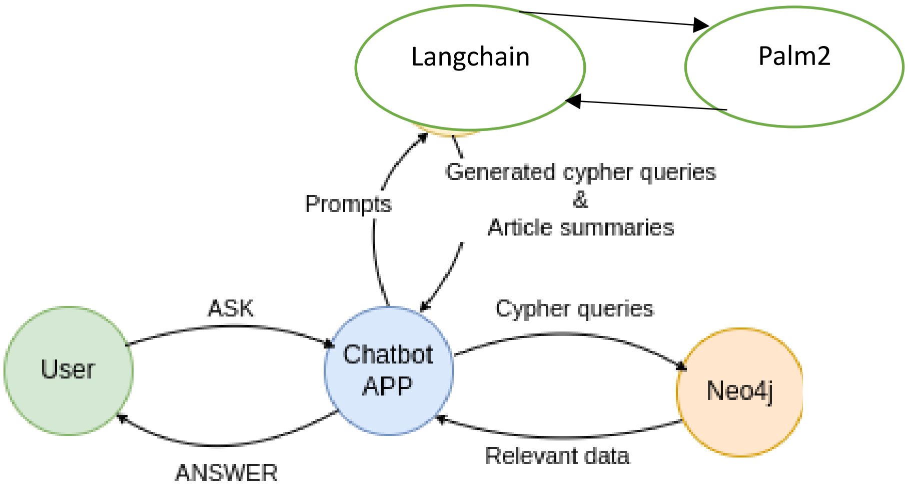
2. **Google's Palm 2 Model:** Google's Palm 2 model is used as the underlying Large Language Model for advanced language understanding and query generation. PaLM 2 is google's next generation large language model that builds on Google's legacy of breakthrough research in machine learning and responsible AI.It excels at advanced reasoning tasks, including code and math, classification and question answering, translation and multilingual proficiency, and natural language generation better than our previous state-of-the-art LLMs, including PaLM. It can accomplish these tasks because of the way it was built – bringing together compute-optimal scaling, an improved dataset mixture, and model architecture improvements.
   to see more information on the PaLM model check out the official link:
   https://ai.google/discover/palm2/

3. **Gradio :** Gradio is an open-source Python library for building, documenting and sharing interactive machine learning models. It allows users to quickly and easily deploy models to the web, with no need for any knowledge of web development. Gradio also provides a number of features that make it easy to document and share models, such as the ability to generate code snippets and interactive demos.One of the key benefits of Gradio is that it makes it easy to share machine learning models with non-technical users. By deploying models to the web, Gradio allows users to interact with models without having to install any software or have any knowledge of programming. This makes it a powerful tool for democratizing access to machine learning
   for more information check out the gradio official documentation : https://www.gradio.app/

3.3 User Interaction with Chatbot UI

1. **Gradio Integration:** Gradio is employed to create a chatbot UI that allows users to interact with the graph database using natural language.

2. **Natural Language Query Processing:** Users can input natural language queries, and the system processes these queries using LLM to generate corresponding Cypher queries for Neo4j.
3. **Graph Database Query and Response:** The Cypher queries are executed on the Neo4j graph database, and the results are presented to the user through the chatbot UI.



4. Conclusion

In conclusion, this project successfully integrates advanced infrastructure project data analysis with knowledge graph technology and Large Language Models. By converting natural language queries into actionable Cypher queries, users can interact with the World Bank Projects data graphically. The chatbot UI provides an intuitive and user-friendly interface for obtaining insights into project networks, relationships between entities, and other relevant information. This project demonstrates the synergy between graph databases, language models, and user interfaces for effective data exploration and analysis.