Submitted by:

### Debonil Ghosh (M21AIE225)

### Ravi Shankar Kumar (M21AIE247)

### Saurav Chowdhury (M21AIE256)

Executive MTech

Artificial Intelligence

Indian Institute of Technology, Jodhpur

Natural Language Understanding

Assignment 2

### Problem Statement:

## Relation Classifier:

Dataset:- KnowledgeNet (data description and dataset can be found here:- link), use train.json inside the dataset as your source of data.

Tasks:-

### **Questions 1.**

Building a relation classifier that can detect a predefined class of relations, as specified in the dataset.

### **Questions 2.**

Create a subset of the KnowledgeNet data using sentences which contain any of the following relations: (make a subset of train.json with these relations only)

a. DATE\_OF\_BIRTH (PER–DATE)

b. RESIDENCE (PER–LOC)

c. BIRTHPLACE (PER–LOC)

d. NATIONALITY (PER–LOC)

e. EMPLOYEE\_OF (PER–ORG)

f. EDUCATED\_AT (PER–ORG)

### **Questions 3.**

Create a Knowledge Graph that can store the information contained in these sentences. You can use any open-source graph database for this purpose.

### **Questions 4.**

Connect the Knowledge Graph to a front end that can take in Natural Language Queries and give the answers back. You can use any open-source chatbot for this purpose. That way, the system will also have the power to

continue a conversation rather than only Question-Answering.

NOTE

1. Save your relation classification models. We will test with arbitrary inputs while evaluating the system.
2. Your KG will be tested with random questions during the evaluation. You have to justify the answers generated by the system.
3. During evaluation - a conversation with the system will be tried.
4. Detailed report about the implementation and evaluation of each component should be submitted using train-test splits.
5. Evaluation will be in-person - new inputs will be used for testing

# **Solution Github Link:**

# <https://github.com/debonil/nluassignments/tree/main/Assignment2>

# **Question 1: Building a relation classifier that can detect a predefined class of relations, as specified in the dataset.**

**Solution:**

Following steps were followed to build relation Classifier:

* + 1. Pre-processed the data by tokenizing them, creating vocabulary and

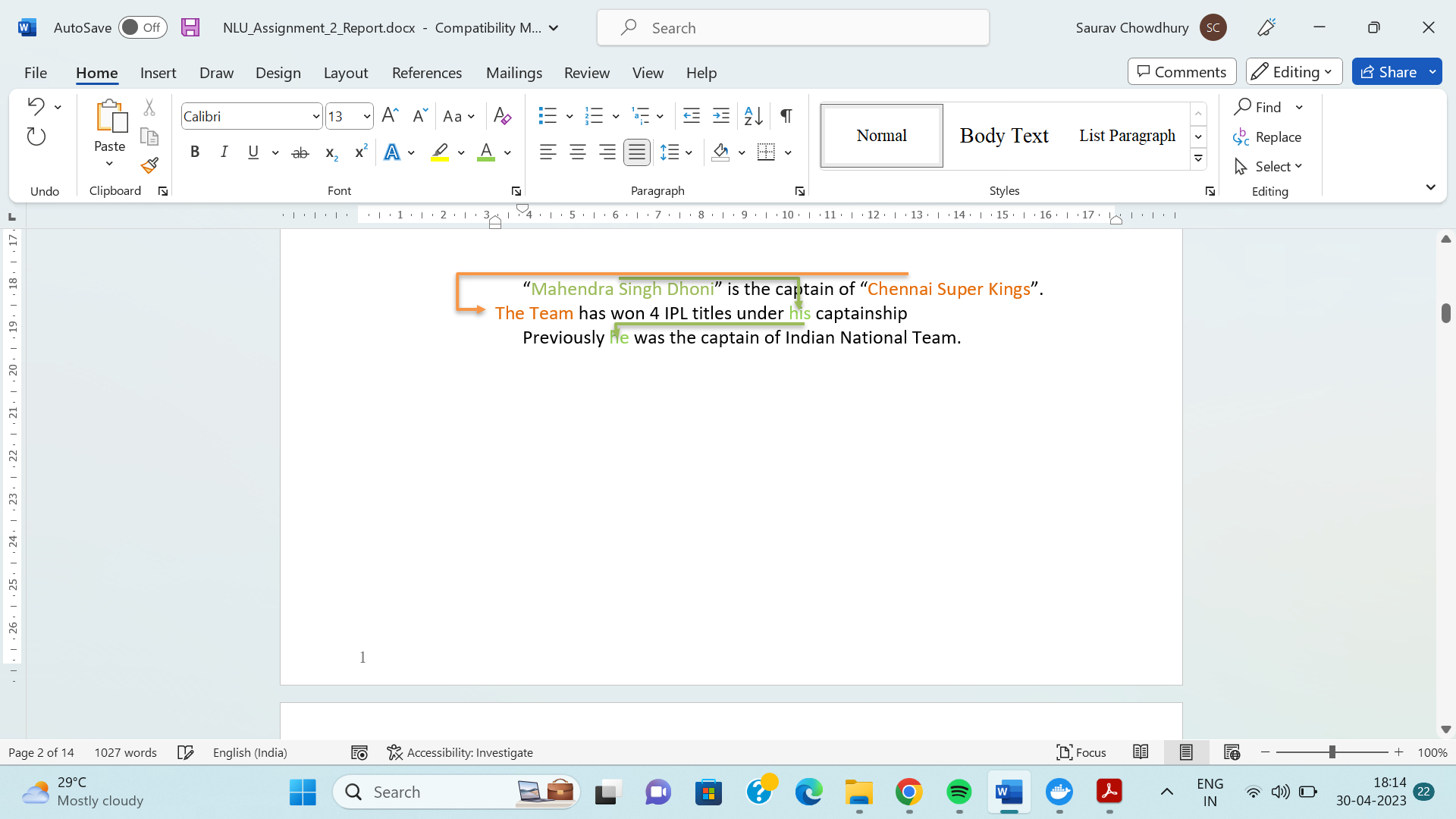
Converting them into numerical values.

* + 1. Trained the data in RNN and obtained Accuracy, Precision , f1-score and support.
    2. Deployed the model in form of [relation\_classifier.joblib](https://github.com/debonil/nlu-assignments/tree/main/Assignment2/models).

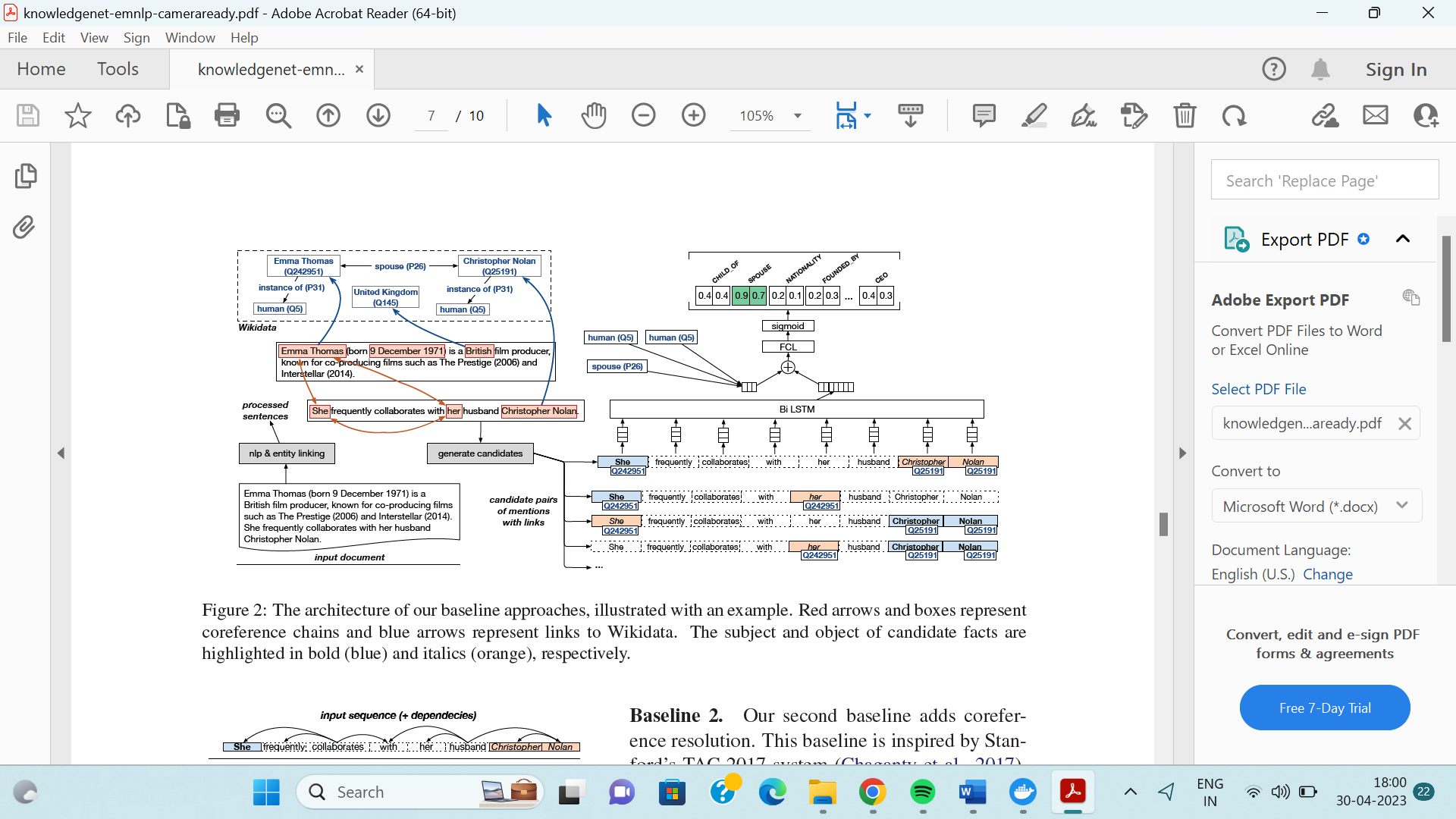
Relationship Classifier:

Entity relation classification is one of the tasks performed by NLU that involves identifying the relationship between two or more entities mentioned in text. For example, if a sentence mentions "John works for Microsoft," the entity relation classifier would recognize that John is an employee of Microsoft.

Goal of entity relation classification is to enable machines to understand the relationships between entities mentioned in text, which can be useful for tasks such as information extraction, question answering, and text summarization



Here the Sentences are connected to each other through relations. The entities are Mahendra Singh Dhoni and Chennai Super Kings



Relation Classifier Architecture where it is classifying a sequence of tokens from KnowledgeNet

Source: Paper : KnowledgeNet: A Benchmark Dataset for Knowledge Base Population

## Technic Used in Assignment:

## Stemmed Count Vectorizer (with snowball stemmer)

## For converting passage texts to vector tokens

## Random Forest Classifier

## For classification task

## Results of Classification:

## Accuracy: 79.670%

## F1 Score: 79.298%

**Classification Report:**

precision recall f1-score support

1 0.94 0.49 0.65 67

10 0.92 0.80 0.86 85

11 0.26 0.83 0.40 139

12 0.75 0.46 0.57 132

14 0.70 0.51 0.59 63

15 0.86 0.76 0.80 90

2 0.47 0.27 0.34 81

25 0.68 0.44 0.53 125

3 0.79 0.51 0.62 181

34 0.59 0.46 0.52 102

4 0.53 0.48 0.50 71

45 0.91 0.76 0.83 51

5 0.72 0.37 0.49 71

6 0.60 0.75 0.66 102

9 0.73 0.81 0.77 109

accuracy 0.58 1469

macro avg 0.70 0.58 0.61 1469

weighted avg 0.68 0.58 0.60 1469

**Class wise Accuracy Score:**

**| SUBSIDIARY\_OF | NATIONALITY | PLACE\_OF\_RESIDENCE | PLACE\_OF\_BIRTH | DATE\_OF\_DEATH | DATE\_OF\_BIRTH |**

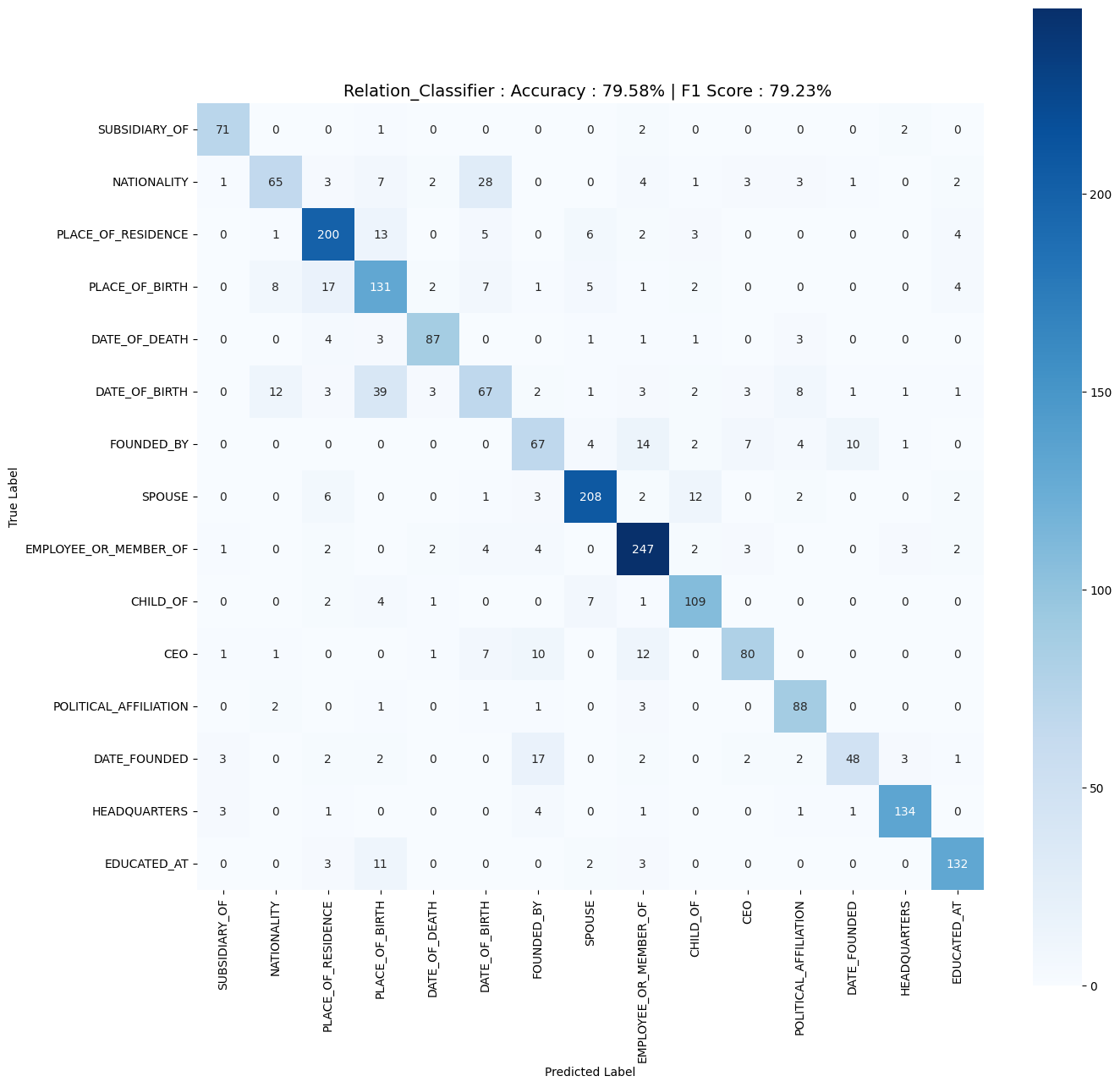
| 0.921053 | 0.558333 | 0.858974 | 0.747191 | 0.86 | 0.417808 |

**FOUNDED\_BY | SPOUSE | EMPLOYEE\_OR\_MEMBER\_OF | CHILD\_OF | CEO | POLITICAL\_AFFILIATION | DATE\_FOUNDED** | 0.651376 | 0.889831 | 0.9 | 0.870968 | 0.678571 | 0.927083 | 0.658537 |

**HEADQUARTERS | EDUCATED\_AT |**

0.924138 | 0.880795 |

**Confusion Matrix of the Relation Classifier Model**



# **Question 2: Create a subset of the KnowledgeNet data using sentences which contain any of the following relations: (make a subset of train.json with these relations only*)***

Solution:

The KnowledgeNet dataset used in this Assignment is a collection of pairs, annotated with multiple semantic relations. The dataset is in the form of a json file where there are 4 entities:

* + 1. Date
    2. Location.
    3. Organisation
    4. Person

There are 6 different relationships between the entities:

DATE\_OF\_BIRTH: 15

EDUCATED\_AT: 9

EMPLOYEE\_OR\_MEMBER\_OF:3

NATIONALITY:10

PLACE\_OF\_BIRTH:12

PLACE\_OF\_RESIDENCE: 11

The data set was parsed and the entities and their relationship were established.

Original document count = 3977

Document with given relations count = 1117

Total fact count = 2483

MERGE (p:Person {name: 'بد الله بن محمد بن سعود آل ثاني'}) ON CREATE SET p.name = 'بد الله بن محمد بن سعود آل ثاني'

MATCH (d:Person {name: 'بد الله بن محمد بن سعود آل ثاني'}), (p:Location {name: 'Qatari'}) MERGE (d)-[:NATIONALITY]->(p)

MERGE (p:Person {name: 'Abdullah bin Mohammed bin Saud Al Thani'}) ON CREATE SET p.name = 'Abdullah bin Mohammed bin Saud Al Thani'

MATCH (d:Person {name: 'Abdullah bin Mohammed bin Saud Al Thani'}), (p:Location {name: 'Qatari'}) MERGE (d)-[:NATIONALITY]->(p)

MERGE (p:Person {name: 'Jim Harris'}) ON CREATE SET p.name = 'Jim Harris'

MATCH (d:Person {name: 'Jim Harris'}), (p:Location {name: 'American'}) MERGE (d)-[:NATIONALITY]->(p)

MERGE (p:Person {name: 'James Patrick Harris'}) ON CREATE SET p.name = 'James Patrick Harris'

...

MERGE (p:Date {name: 'August 3, 1980'}) ON CREATE SET p.name = 'August 3, 1980'

MATCH (d:Person {name: 'Eric N. "E. J." Henderson'}), (p:Date {name: 'August 3, 1980'}) MERGE (d)-[:DATE\_OF\_BIRTH]->(p)

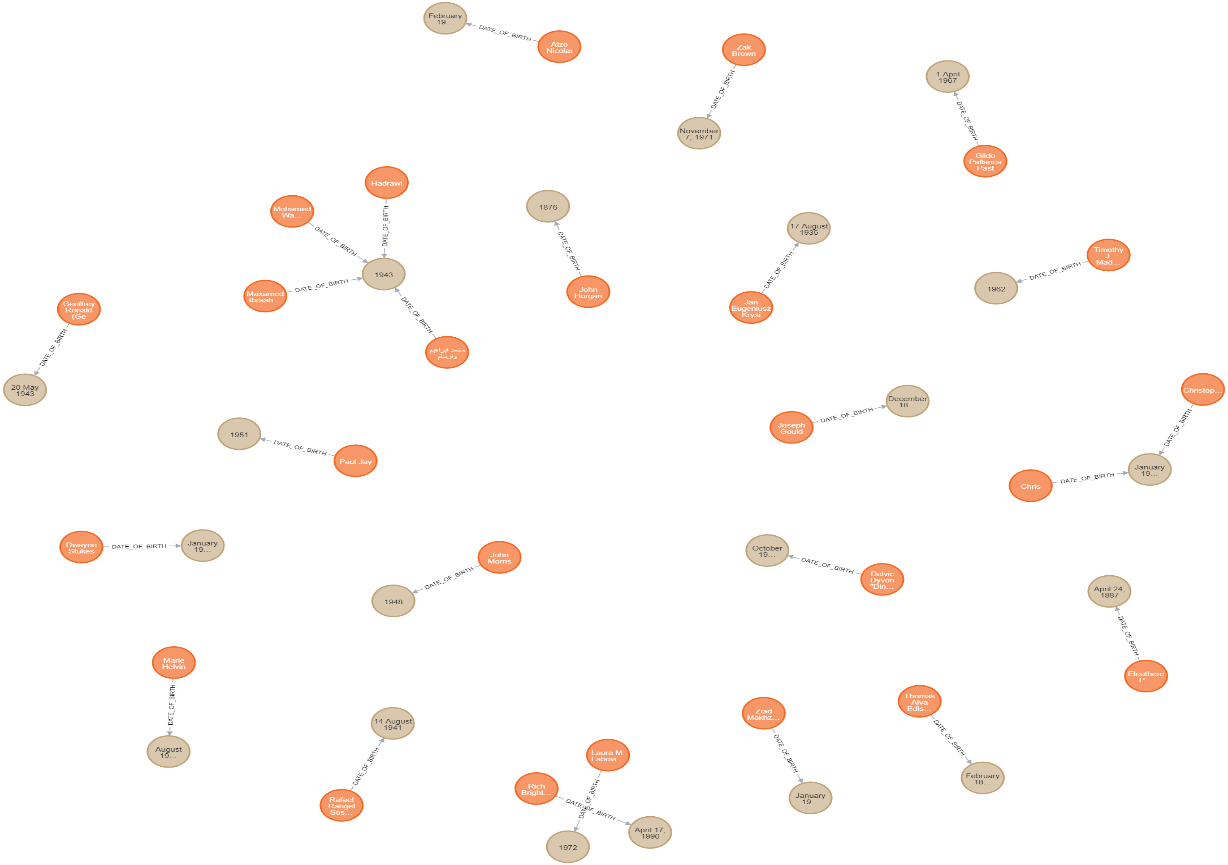
MERGE (p:Person {name: 'Eric N. "E. J." Henderson'}) ON CREATE SET p.name = 'Eric N. "E. J." Henderson'

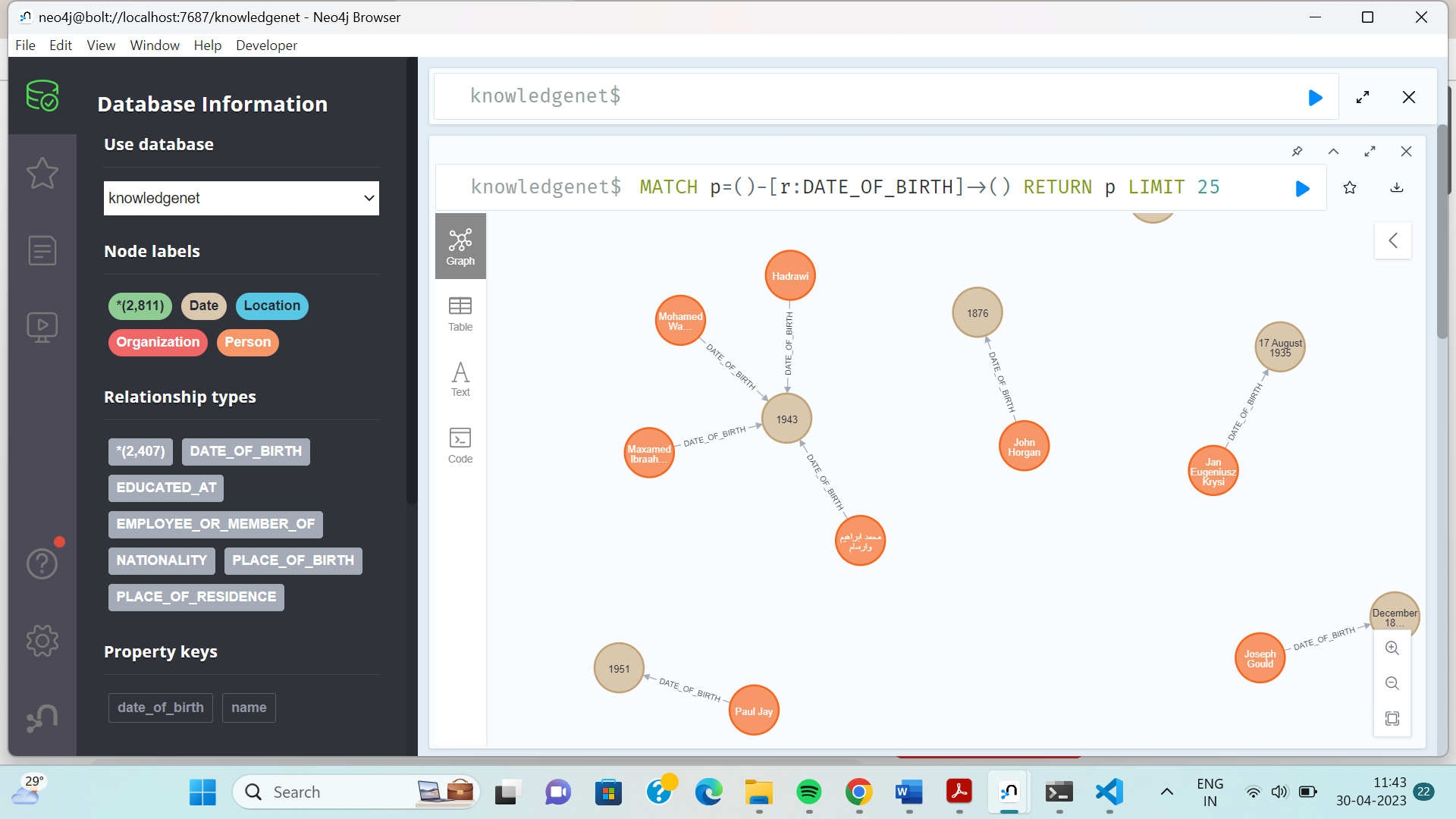
MATCH (d:Person {name: 'Eric N. "E. J." Henderson'}), (p:Location {name: 'American'}) MERGE (d)-[:NATIONALITY]->(p)

**The entities and their relationships were established and the knowledge graph were visualised using neo4j queries.**

*DATE\_OF\_BIRTH (PER–DATE)*

MATCH p=()-[r:DATE\_OF\_BIRTH]->() RETURN p LIMIT 25

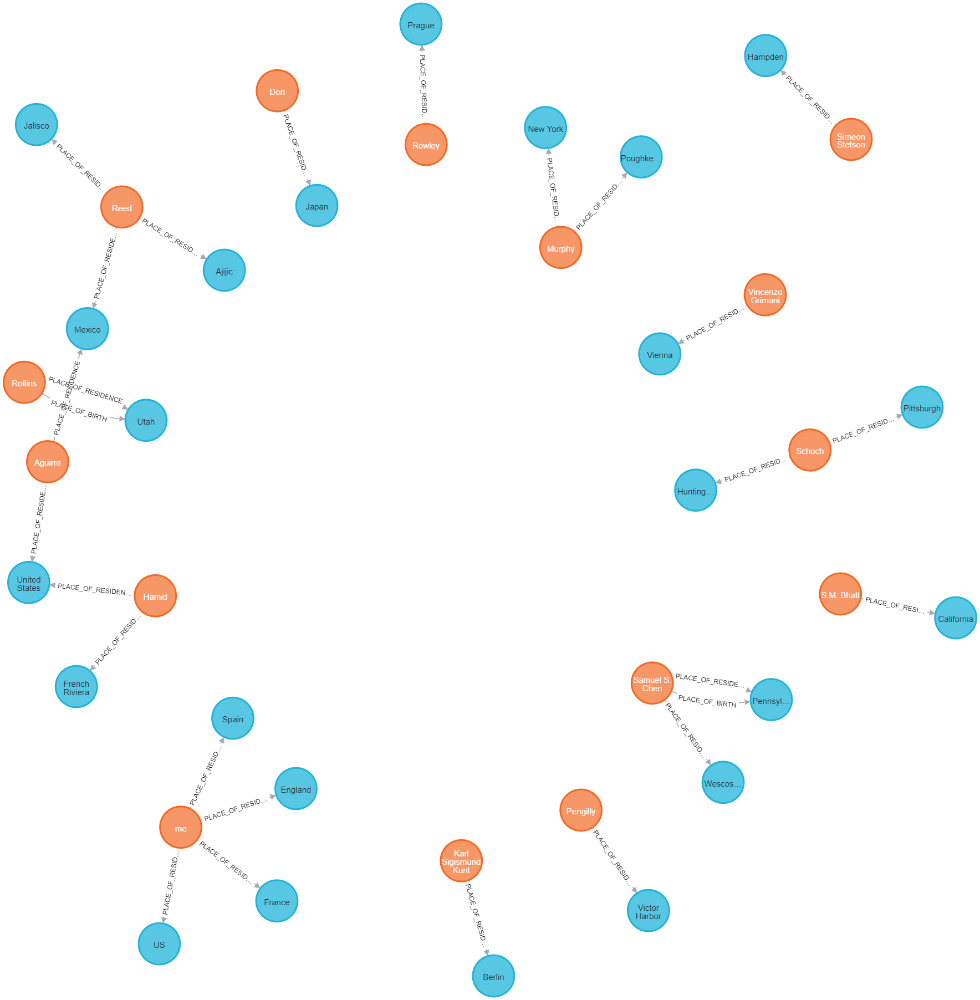


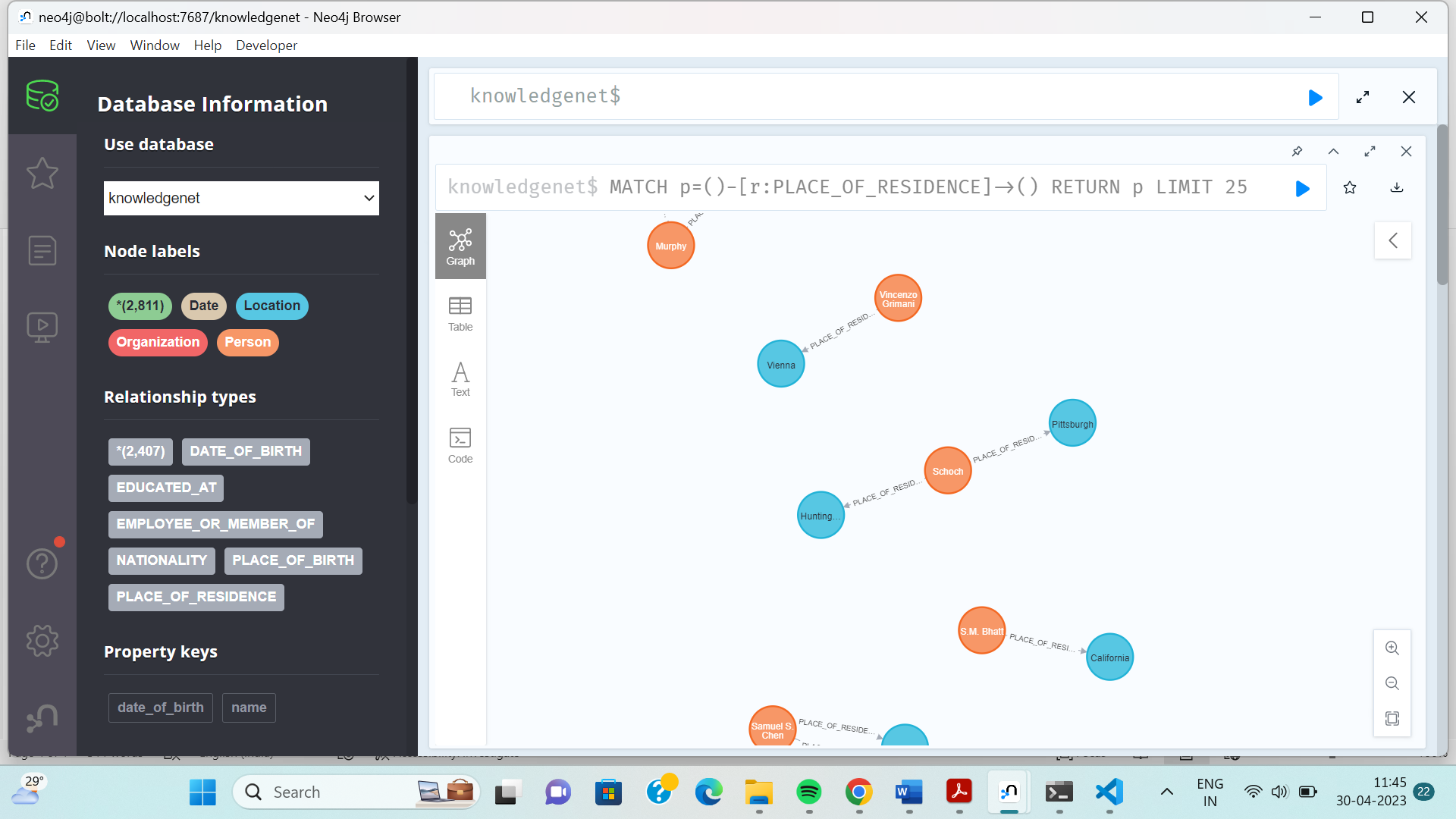


Relationship between Entities Name and Date through Date of Birth

*RESIDENCE (PER–LOC)*

MATCH p=()-[r:PLACE\_OF\_RESIDENCE]->() RETURN p LIMIT 25



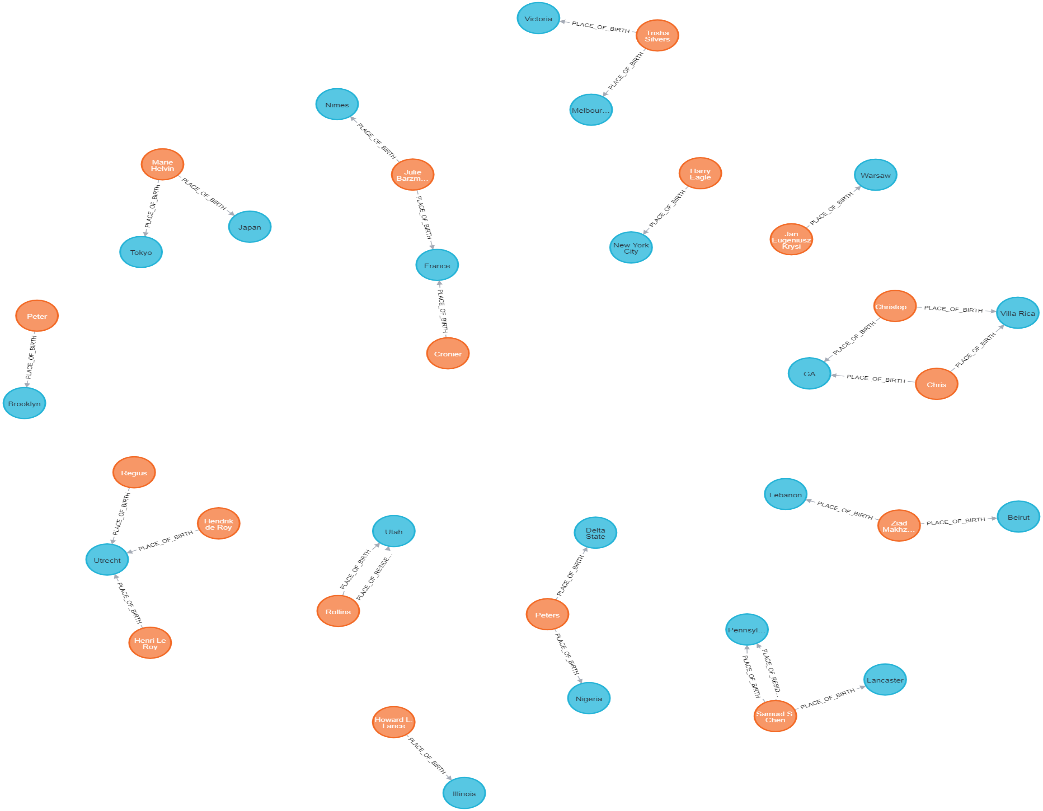


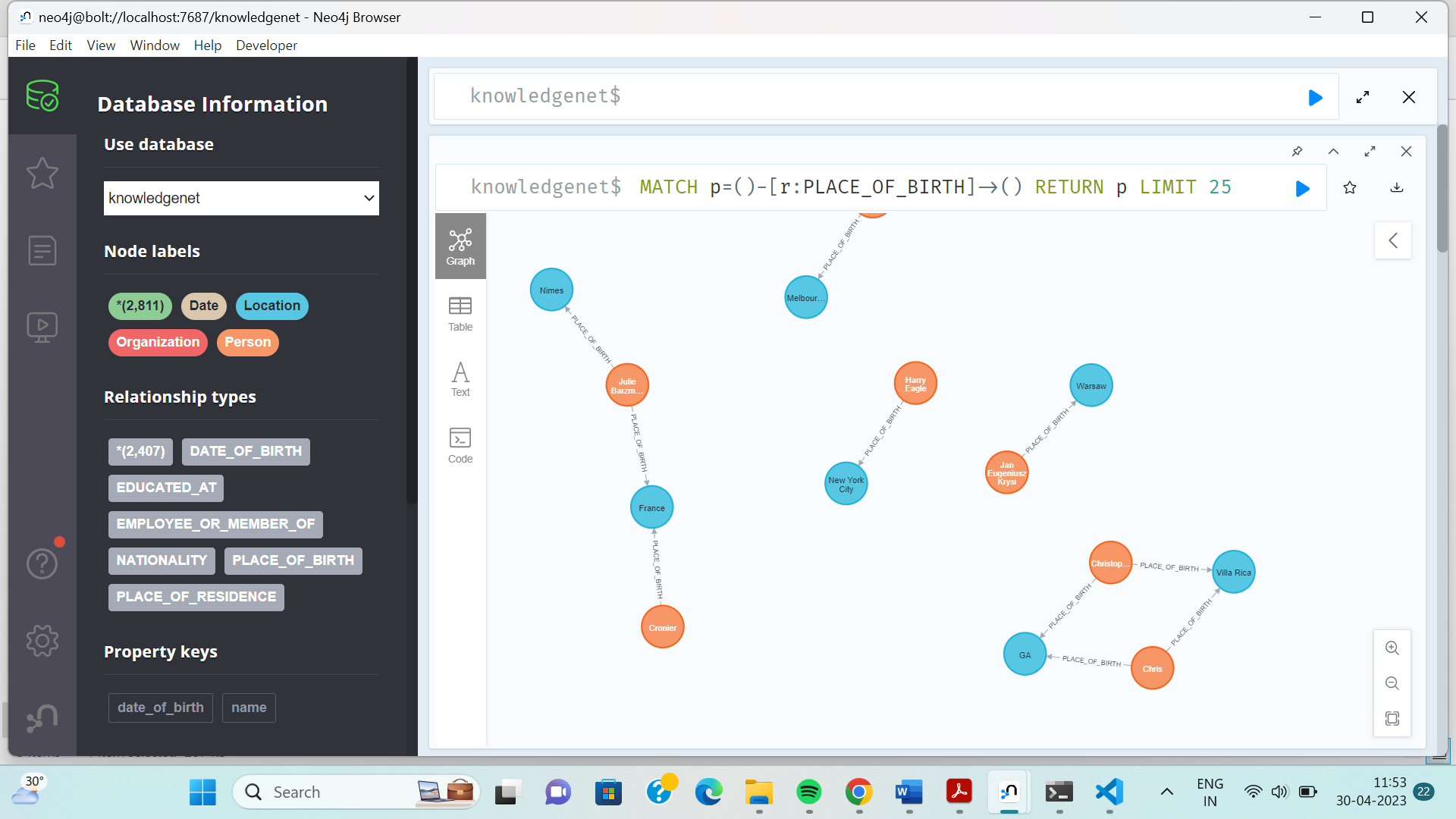
Relation Ship between entities Name and Location through relationship

Place of Residence

*BIRTHPLACE (PER–LOC)*

MATCH p=()-[r:PLACE\_OF\_BIRTH]->() RETURN p LIMIT 25



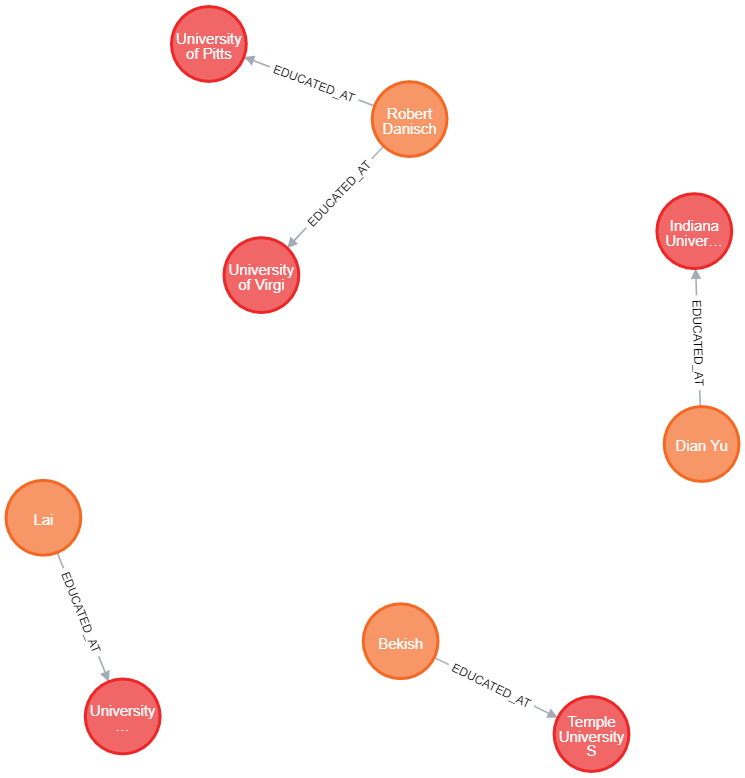


Relation Ship between entities Name and Location through relationship

Place of Birth.

EDUCATED\_AT

MATCH p=()-[r:EDUCATED\_AT]->() RETURN p LIMIT 5

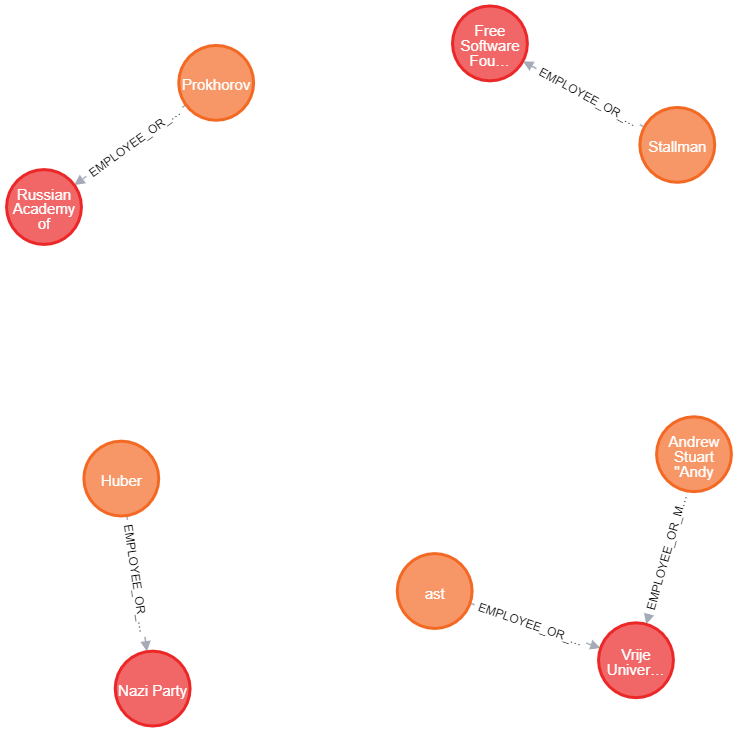


Relation Ship between entities Name and University through relationship

EDUCATED\_AT.

EMPLOYEE\_OR\_MEMBER\_OF

MATCH p=()-[r:EMPLOYEE\_OR\_MEMBER\_OF]->() RETURN p LIMIT 5

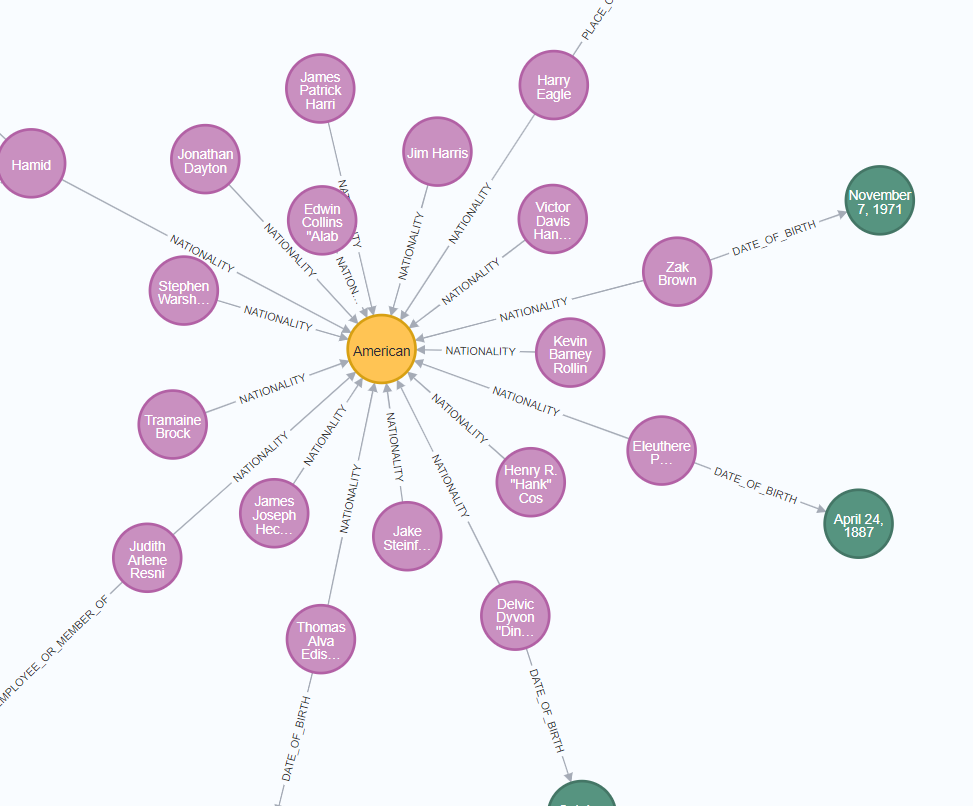


Relation Ship between entities Name and Employeer through relationship

EMPLOYEE\_OR\_MEMBER\_OF

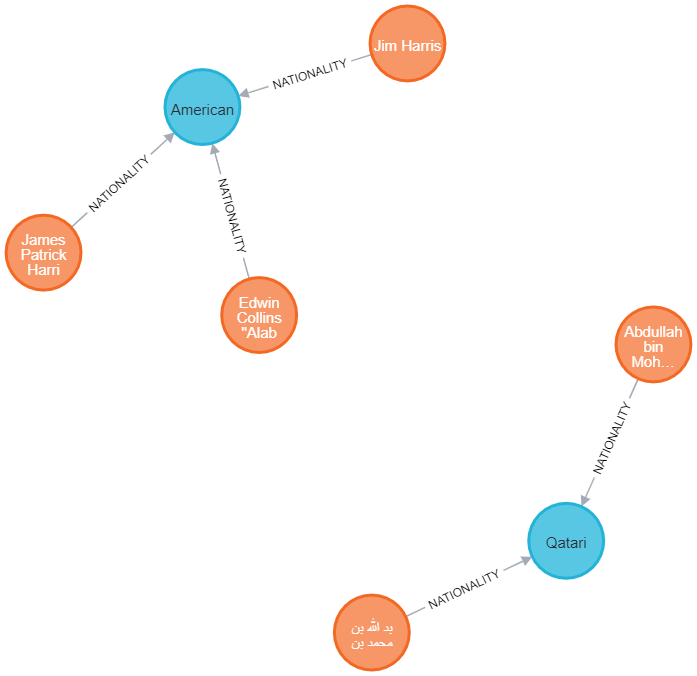
NATIONALITY

MATCH p=()-[r:NATIONALITY]->() RETURN p LIMIT 5



Relation Ship between entities Name and Location through relationship

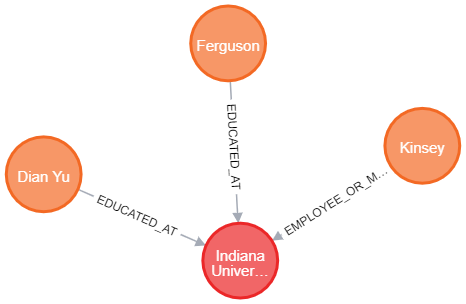
Nationality



Now we produced few more complex Knowledge Graphs using neo4j on the dataset:

* + - 1. A KG showing all the people who were educated\_at “Indiana University”

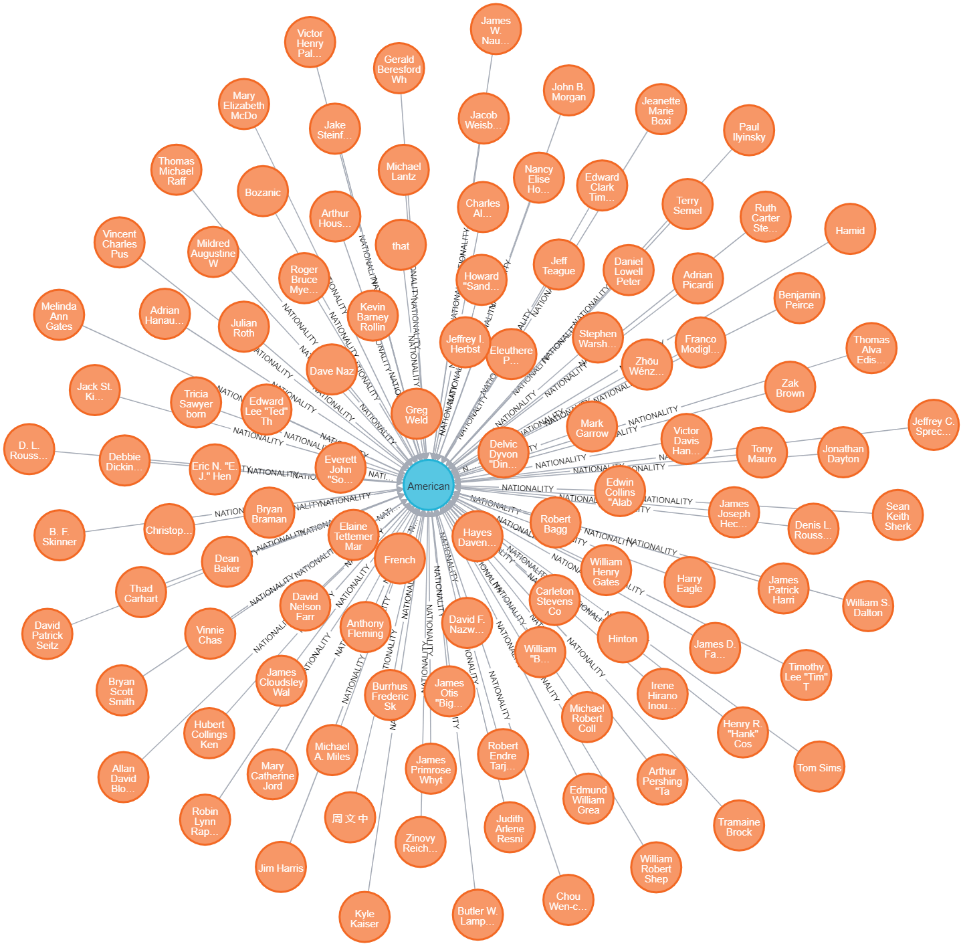
match (n:Organization{name: 'Indiana University'})<-[\*]-(r) return n,r



Relationship Between Educated\_at and people

* + - 1. A KG showing all people who are American:

match (n:Location{name: 'American'})<-[\*]-(r) return n,r



A Relationship Knowledge Graph of All person who are American

# **Question 4: Connect the Knowledge Graph to a front end that can take in Natural Language Queries and give the answers back. You can use any open-source chatbot for this purpose. That way, the system will also have the power to continue a conversation rather than only Question-Answering.**

**Solution:**

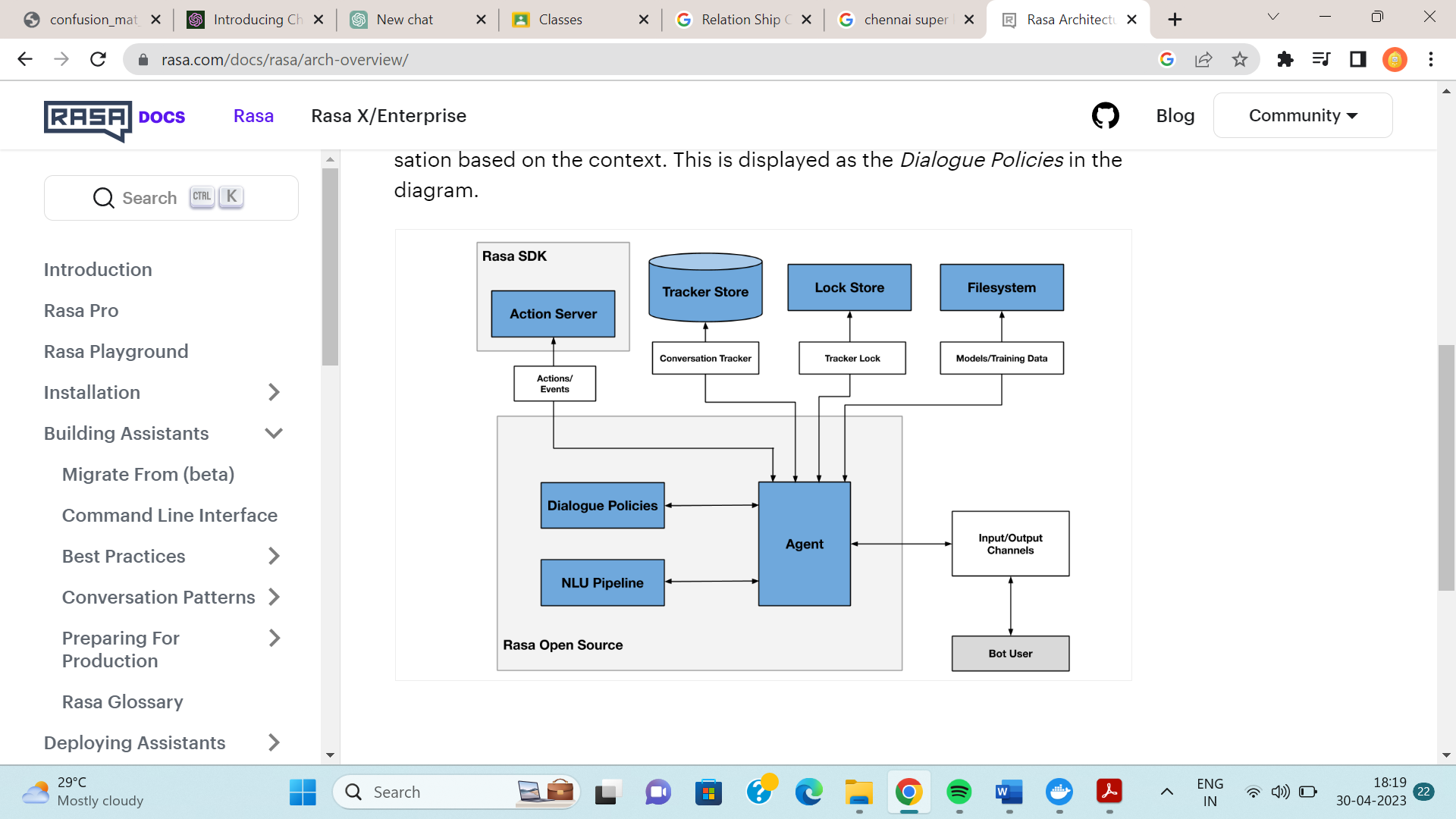
The knowledge graphs are connected using Open-Source framework “**RASA**” which is a framework for building AI chatbots. It is designed to support entire lifecycle of a chatbot. The framework is built on top of machine learning libraries such as TensorFlow and allows developers to create custom dialogue management policies, integrate with external APIs, and deploy their chatbots on various platforms.

Rasa consists of two main components: Rasa NLU and Rasa Core.

1. Rasa NLU (Natural Language Understanding) is responsible for understanding user input by extracting entities and intents from the user's messages. It is built on top of popular natural language processing (NLP) libraries such as spaCy and scikit-learn, and allows developers to train custom machine learning models to recognize user intents and entities.
2. Rasa Core is responsible for managing the conversation flow by selecting the appropriate response based on the current user input and the chatbot's dialogue management policy. It uses machine learning algorithms such as deep reinforcement learning to learn from user interactions and improve the conversation experience over time.

**Some of the key features of Rasa include**:

* Customizable dialogue management policies: Rasa allows developers to create custom dialogue management policies based on their specific use case and business requirements.
* Support for multiple channels: Rasa supports various channels such as Facebook Messenger, Slack, and custom web interfaces, allowing developers to deploy their chatbots on multiple platforms.
* Integration with external APIs: Rasa allows developers to integrate with external APIs and databases to retrieve and manipulate data during the conversation.
* Open-source and community-driven: Rasa is an open-source framework with an active community of developers contributing to its development and maintenance.



Neo4j Knowledge

Net

RASA Architecture

The knowledge graph created out of the train.json using Neo4j database is connected to RASA actions so that front end application can fetch information from knowledgebase whenever required.

### **Development using RASA:**

1. **Intent identification in problem statement**

In Rasa, an intent represents the goal or intention behind a user's message. It is a specific action or task that the user wants the chatbot to perform, such as making a reservation or checking an account balance. Intent recognition is a critical component of the chatbot's natural language understanding (NLU) and is used to route the user's message to the appropriate part of the chatbot's dialogue management system. In Rasa, intents are defined using natural language examples provided by the developer and are learned by the chatbot's machine learning models during training.

In our problem, following domain specific and some general purpose intents are used.

General Intents:

  - greet

  - goodbye

  - affirm

  - deny

  - mood\_great

  - mood\_unhappy

  - bot\_challenge

  - asking\_how\_are\_u

  - enquire\_bot

Domain Specific Intents:

- query\_person\_DATE\_OF\_BIRTH

- query\_person\_PLACE\_OF\_RESIDENCE

- query\_person\_PLACE\_OF\_BIRTH

- query\_person\_NATIONALITY

- query\_person\_EMPLOYEE\_OR\_MEMBER\_OF

- query\_person\_EDUCATED\_AT

- query\_person\_all

- query\_person\_known\_or\_not

- get\_all\_person\_EDUCATED\_AT

- get\_all\_person\_PLACE\_OF\_RESIDENCE

1. **Domain design**

Index of the whole application stored in domain file. It contains following details:

* 1. Dynamic responses
  2. List of entities
  3. List of variables (slots)
  4. List of Intents
  5. Actions
  6. Session config

1. **Actions planned**

Probable actions that can be taken and so backend operation like database data fetch etc. Following actions are used in our application:

  - utter\_greet

  - utter\_happy

  - utter\_unclear

  - utter\_happy\_state

  - action\_hello\_world

  - action\_desc\_person

  - action\_person\_known\_or\_not

  - action\_person\_all\_details

  - action\_get\_all\_person\_frm\_releation

1. **Stories**

Here rasa is given probable paths of intent and action series with some examples.

For our case one example story is:

- story: happy story block buster

  steps:

  - intent: greet

  - action: utter\_greet

  - intent: asking\_how\_are\_u

  - action: utter\_happy\_state

  - intent: mood\_great

  - action: utter\_happy

  - intent: query\_person\_known\_or\_not

  - action: action\_person\_known\_or\_not

  - intent: query\_person\_PLACE\_OF\_RESIDENCE

  - action: action\_desc\_person

  - intent: query\_person\_DATE\_OF\_BIRTH

  - action: action\_desc\_person

  - intent: query\_person\_PLACE\_OF\_BIRTH

  - action: action\_desc\_person

  - intent: query\_person\_EDUCATED\_AT

  - action: action\_desc\_person

  - intent: query\_person\_EMPLOYEE\_OR\_MEMBER\_OF

  - action: action\_desc\_person

  - intent: get\_all\_person\_EDUCATED\_AT

  - action: action\_get\_all\_person\_frm\_releation

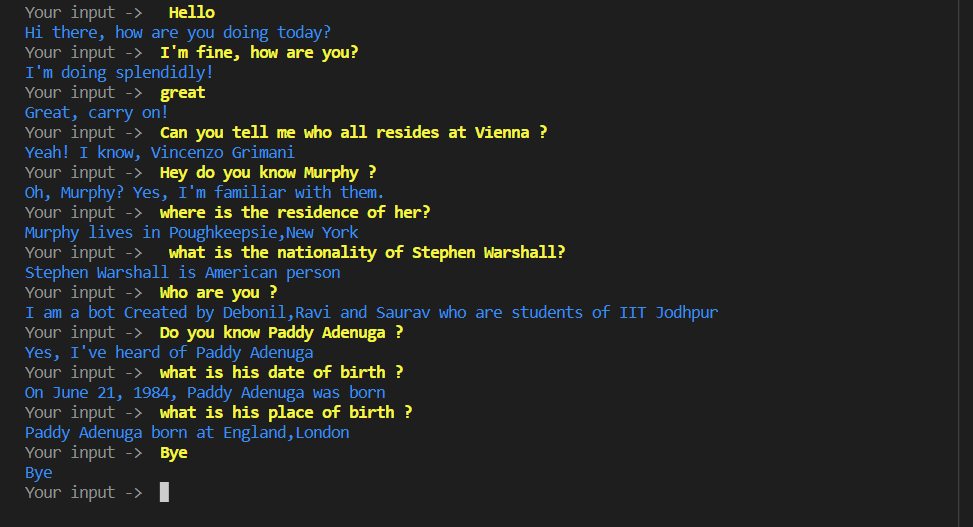
  - intent: get\_all\_person\_PLACE\_OF\_RESIDENCE

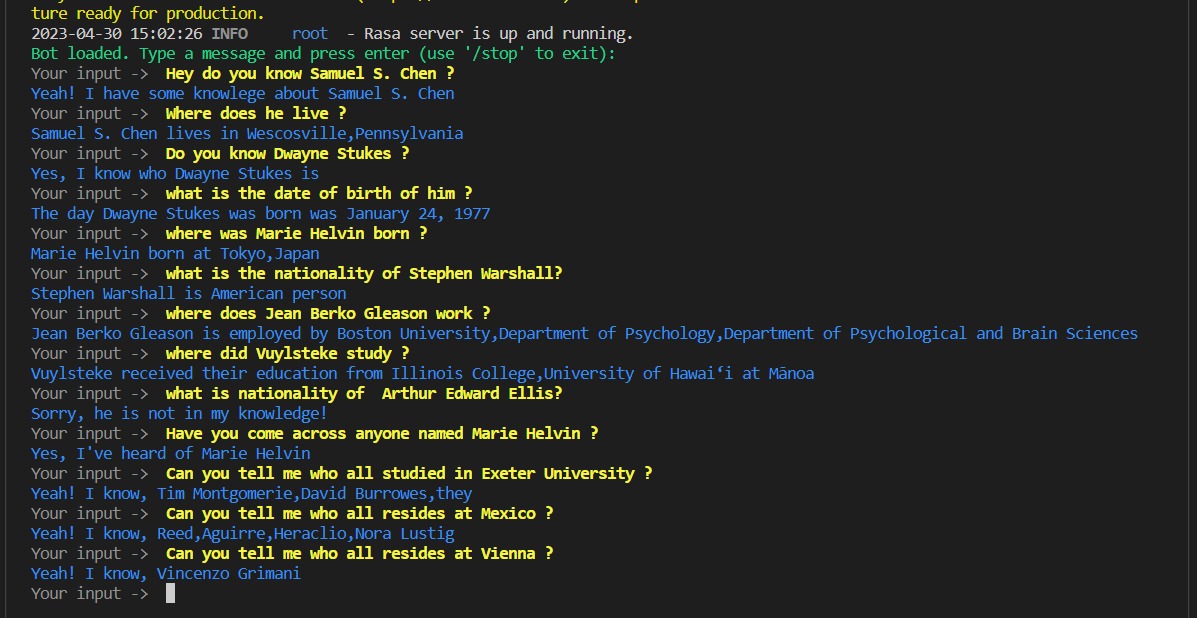
  - action: action\_get\_all\_person\_frm\_releation

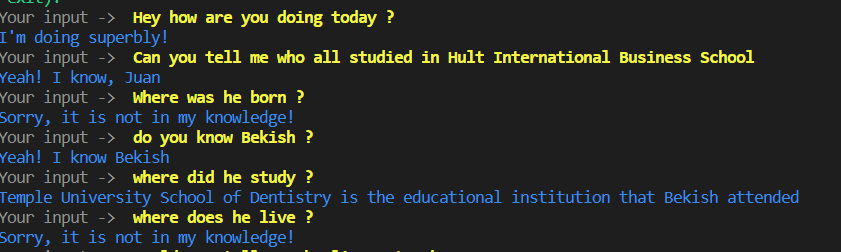
1. **Changes in config** to enable Person, Geo Political Entity (Locations ), Organization Named Entity Recognition in the natural language query

### **Conversation with our chat bot:**

**Conversation 1:**



**Conversation 2:****Conversation 3**:



**Capabilities:**

1. Chat bot can greet, start conversation
2. Can identify entities that exists in the natural language query
3. Can identify our domain specific intents hidden in the natural language query and generate proper reply, triggers actions whenever required.
4. It can continue conversation by answering questions regarding entities that exists in the backend knowledge graph
5. Can identify entity from previous chat, referred by pronouns (like he/she)
6. Can reply questions related to the Person, Location, Organization entities.

**Limitations:**

1. Knowledge of this chat-bot limited to the backend knowledgenet.
2. Sometime struggles to identify intent if query is not clear or related to domain.

**Contribution of Each Member:**

1. All the member discussed the theory and scope of the problem through meeting virtually and literature of the problem was shared.
2. The models were than chosen after discussion.
3. Coding, Theory and Reports were prepared through constant discussion between the members.

**References:**

1. Lectures given by Respected Professors
2. <https://neo4j.com/>
3. KnowledgeNet: A Benchmark Dataset for Knowledge Base Population by Filipe Mesquita, Matteo Cannaviccio,Jordan Schmidek, Paramita Mirza.
4. <https://rasa.com/docs/rasa/>
5. NLU Class Slides.