1. Dataset

The KnowledgeNet dataset is used for the complete exercise. The dataset can be accessed from here. Only the train.json of the dataset is used for all the tasks in the assignment as instructed.

Issue noticed with dataset:

We have noticed a few flaws during data exploration. The indices of subject and object are not matching for many documents. In the complete dataset there are total 10895 facts but out of which 4179 facts are not matching as per index.

We have tried to find the reason for this and also tried to fix by ourselves but could not find out any consistent pattern which can be countered to get data as expected.

We have proceeded with those facts for which the index is consistent and matching with the subject and object text mentioned in the json file.

2. Building Relation Classifier

2.a Relations in the dataset

In the data there are 15 relations.

Sr. #	Relation
1	EDUCATED_AT
2	SUBSIDIARY_OF
3	DATE_OF_DEATH
4	NATIONALITY
5	CEO
6	HEADQUARTERS
7	FOUNDED_BY
8	SPOUSE
9	EMPLOYEE_OR_MEMBER_OF
10	DATE_FOUNDED
11	PLACE_OF_RESIDENCE
12	POLITICAL_AFFILIATION
13	PLACE_OF_BIRTH
14	DATE_OF_BIRTH
15	CHILD_OF

2.b Training the Relation Classifier

We have used LUKE from Hugging Face. We have used LUKE-base version and fine-tuned the model to classify the 15 relations from our dataset.

Model Architecture – The LUKE base model is downloaded using Hugging Face transformers library. We have added a linear projection layer of 15 outputs to classify in one of class. The last 5 layers are considered for fine-tuning. So 4 layers of LUKE base model and the projection layer is trained.

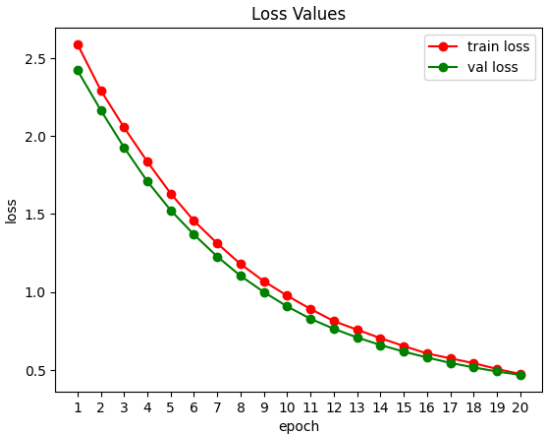
Data Preparation – The Luke based model tokenization required texts as well as indexes of entity pair. Each pair is in format *([<subject_start> , <subject_end>], [<object_start>, <object_end>]).*

Hyper-Parameters - The model is trained based on below parameters. The based model is stored based on best accuracy over validation dataset.

Hyper-Parameter	Values
Learning Rate	10^{-5}
Epochs	20
Fine-tune Layers	4
Batch Size	16
Optimizer	AdamW

Further training was stopped when training and validation accuracy starts to diverge.

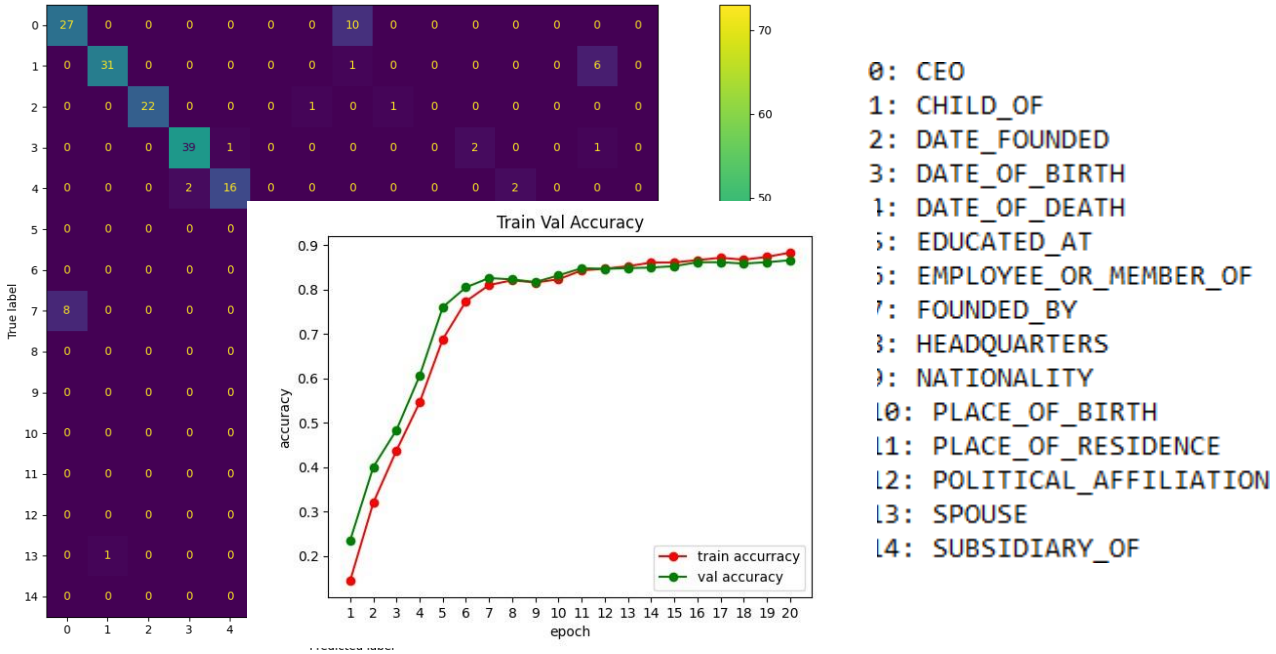
The training loss and validation loss accuracy are plotted below.



The training and validation accuracy is shown below epoch-wise.

2.c Results

The overall accuracy is **87.80% on test set**. The confusion matrix on test set is shown below which shows some interesting patterns which are discussed in findings.



Findings – If we notice the misclassifications, the three most common misclassifications are

1> CEO classified as FOUNDED_BY
2> PLACE_OF_BIRTH as PLACE_OF_RESIDENCE
3> CHILD_OF as SPOUSE

CEO and FOUNDED_BY both are relation between an organization and person, similarly PLACE_OF_BIRTH and PLACE_OF_RESIDENCE both are relation between person and place. So although the classifier is not 100% accurate but it is getting confused between similar relations. For example, SPOUSE and HEADQUARTERS are not getting mixed.

Improvements – The model currently requires entity spans to predict the output. We have tried to create a NER model which will produce the span of subject and object spans. It did not produce good enough results, so we have not used it in the pipeline.

3. Subset of Knowledgenet Dataset

As we have seen earlier the knowledge graph contains 15 relations. Out of which 6 are to be selected for further tasks. We have used simple filtering from each document to get the desired subset. Few samples shown below.

```
1 relationship.head(20)
```

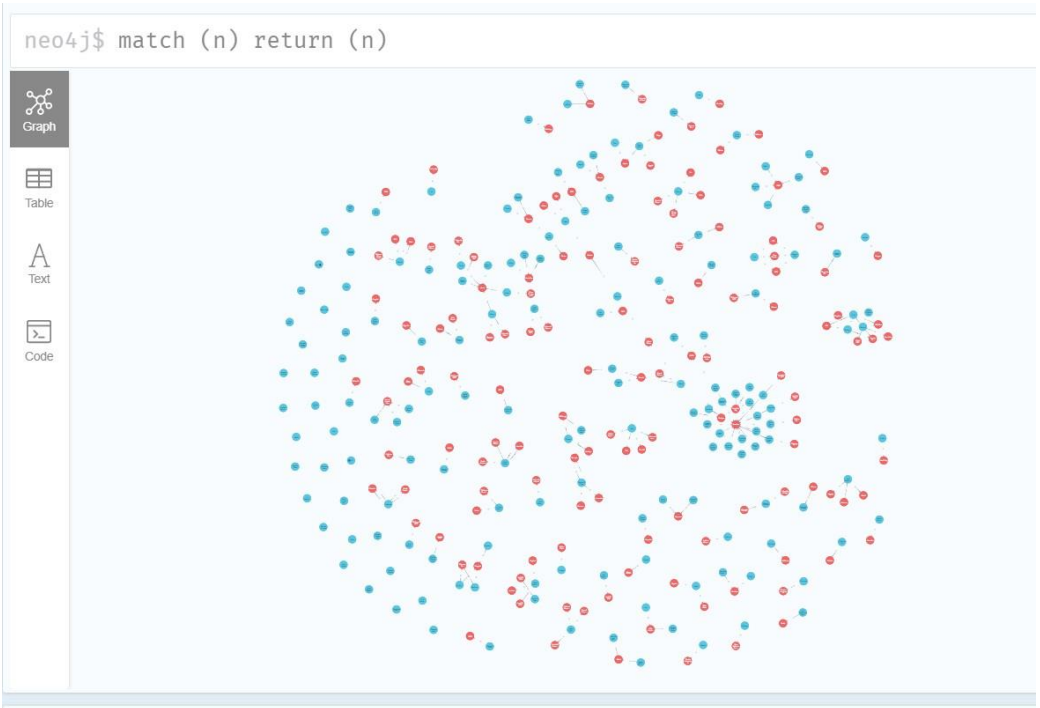
	subject	object	relation
0	╪и╪п ╪з┘Д╪п╪з ╪и┘Ж ┘Е╪н┘Е╪п ╪и┘Ж ╪╢┘И╪п ╪в┘Д...	Qatari	NATIONALITY
1	Abdullah bin Mohammed bin Saud Al Thani	Qatari	NATIONALITY
2	Jim Harris	American	NATIONALITY
3	James Patrick Harris	American	NATIONALITY

4. Create Knowledge Graph

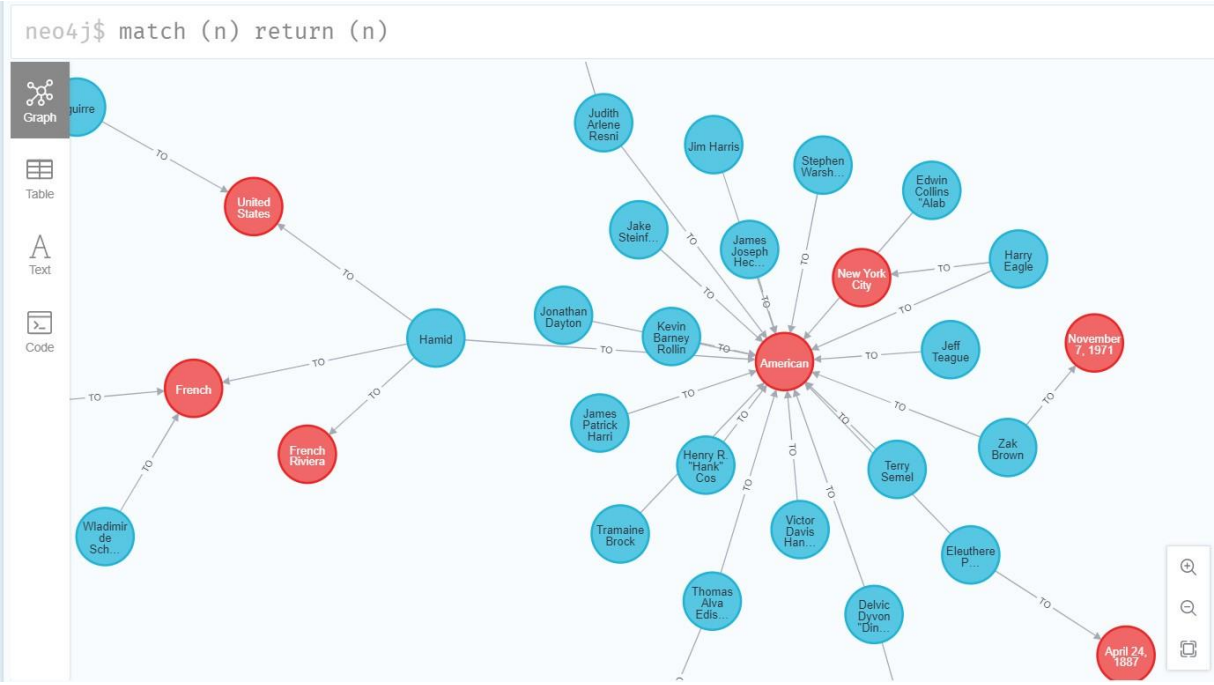
We have used Neo4j for this purpose. NLU_DB is created for this assignment. Default neo4j database is used for this purpose. Relationship data created in the previous step was loaded to the neo4j with the help of the below query:

```
LOAD CSV WITH HEADERS from 'file:///relationship.csv' as row with row where row.subject is not NULL
MERGE (n:subject {Name: row.subject})
MERGE (m:object {Name: row.object})
MERGE (n) -[:TO {rel:row.relation}]-> (m)
```

Once the query was executed, the required knowledge graph was created as shown below:

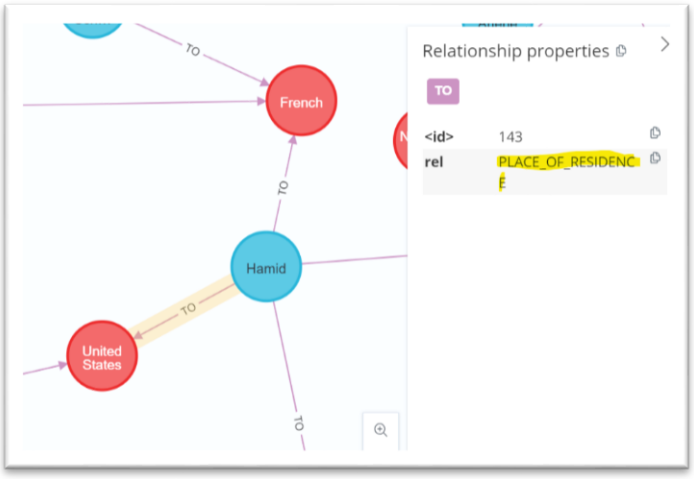


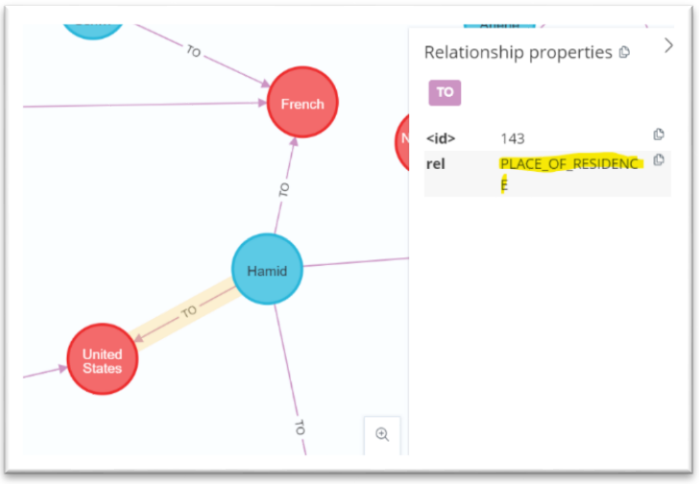
Taking a closer look at few nodes



This knowledge graph contains all the required relations as mentioned in the questions.

If we further dive deep and look into relations:





5. Creation of Chatbot

The Chatbot interface is made using Streamlit. A connection was made with neo4j.

5.a. Important Logical steps

- **Identifying subject and object**
- **Relationship identification between subject and object**
- **List of subjects related to a particular object**

5.b. Solution Steps

1. To identify subject and object we have used spacy model

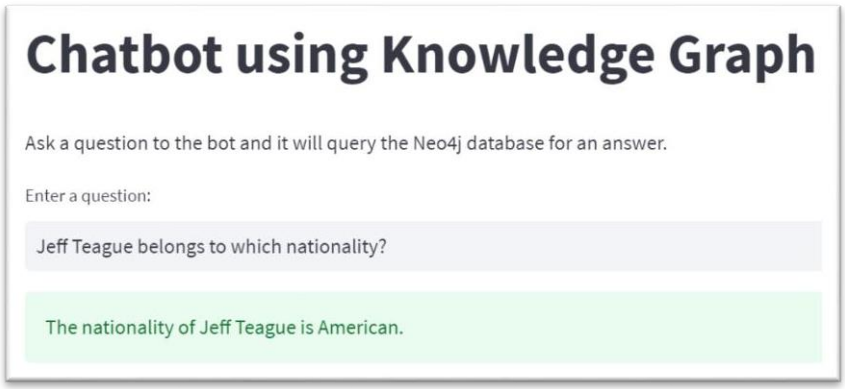
```
nlp = spacy.load("en_core_web_sm")
```

2. Use the information extracted from the question to query the neo4j database and send the result to the streamlit frontend

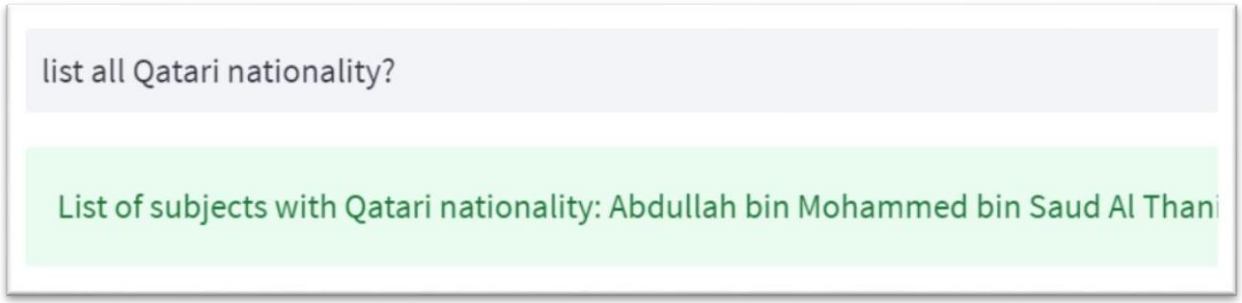
5.c. Sample Conversations

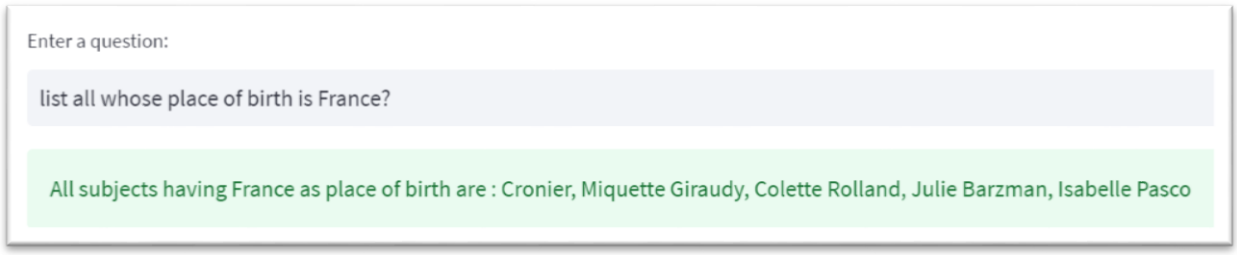
Below are the excerpts from our chat with the chatbot (Keeping an example of all scenarios which are implemented)

Question based on single Node



Questions to get list of Subjects





5.d. Area of Improvements

We can implement some other capabilities for the chatbot like:

How is subject and object related?

e.g. ***How is Messi related to Argentina***

List all objects related to a particular subject?

e.g. ***Provide me all the information about Cristiano Ronald***