# Relation Classification using LSTM and BERT Models and leveraging Knowledge Graph


- **Supervisor: Prof. Alfio Ferrara**
- **Author: Shojaat Joodi Bigdilo**


- **23 July 2024**

# Overview

- Our study investigates the effectiveness of BERT, LSTM, and bi-directional LSTM models for relation classification tasks, focusing on identifying relationships between people, locations, dates, and educational degrees mentioned in the text. Our analysis aims to provide insights into the capabilities and limitations of BERT-based and LSTM-based models for **relation Classification.**

- Additionally, we employ the 'networkx' library to construct and display **Knowledge Graphs**, representing the relationships between entities (subjects and objects) using provided predicates that indicate their relations.

## Relation Classification Steps:

- Step 1: Data Loading and Data Exploration
- Step 2: Data Preprocessing
- Step 3: Models' Architecture (LSTM, BiLSTM, BERT)
- Step 4: Defining Important Functions (Trian and validation func, Visualization func)
- Step 5: Training and Validation of all Models
- Step 6: Result and Comparison

## Knowledge Graph Steps:

- Data Preparation ('sub', 'obj', 'relation')
- Creating Nodes and Edges
- Static Graph Visualization
- Dynamic Graph Visualization
- Display some Result

# Relation Classification

- The task of relation classification involves predicting semantic relations between pairs of nominals.
- Given a text sequence (usually a sentence) S and a pair of nominals e1 and e2, the objective is to identify the relation between e1 and e2 (Hendrickx et al., 2010).
- For example, consider the sentence:

"The [kitchen]e1 is the last renovated part of the [house]e2."

Here, the relation between "kitchen" and "house" is classified as Component-Whole. (Suchanek and Yifan, 2019)

# Goal of Relation Classification

- The main goal is to extract valuable insights from text that enrich our understanding of the relationships that bind people, places, organizations, concepts, etc.

# Dataset

- The provided datasets has 5 different JSON files:

`place_of_death.json`,

`place_of_birth.json`

`date_of_birth.json`,

`education-degree.json`,

`institution.json`.

| | pred | sub | obj | evidences |
|---|---|---|---|---|
| 0 | date_of_birth | James Cunningham | 1973 | [{'url': 'http://en.wikipedia.org/wiki/James_C... |
| 1 | date_of_birth | Kepookalani | 1760 | [{'url': 'http://en.wikipedia.org/wiki/Kepooka... |
| 2 | date_of_birth | Shamsher M. Chowdhury | 1950 | [{'url': 'http://en.wikipedia.org/wiki/Shamshe... |
| 3 | date_of_birth | Gary Sykes | 1984-02-13 | [{'url': 'http://en.wikipedia.org/wiki/Gary_Sy... |
| 4 | date_of_birth | Carolus Hacquart | 1640 | [{'url': 'http://en.wikipedia.org/wiki/Carolus... |

- Each file represents a relation (predicate) between the subject and object, and their corresponding snippet text.
- These files are combined for further analysis.

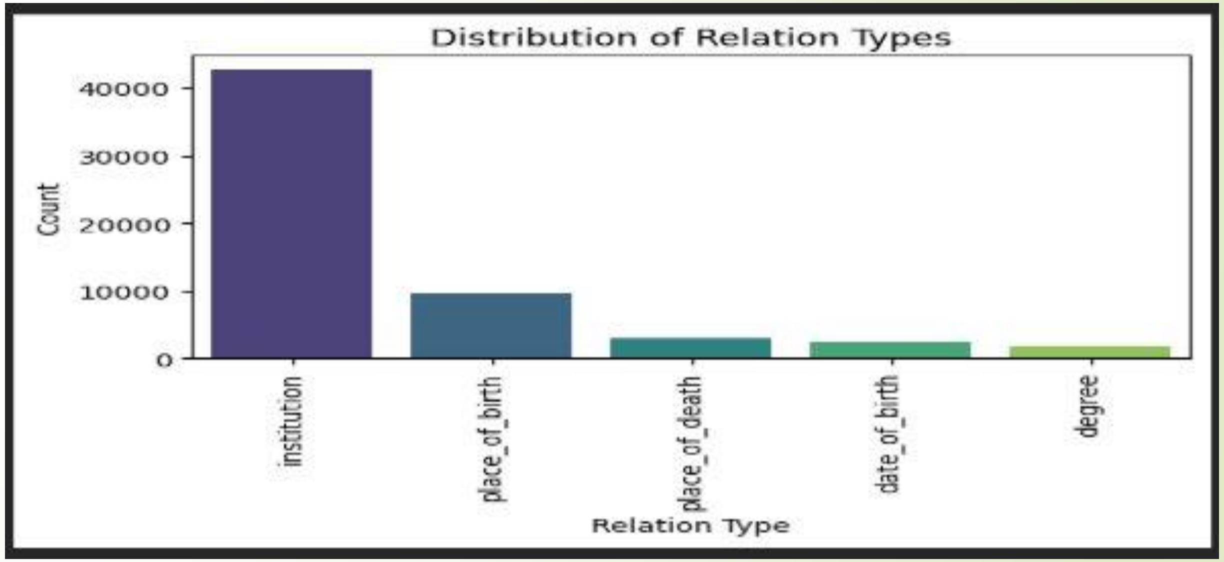
```
{'pred': 'date_of_birth',
 'sub': 'James Cunningham',
 'obj': '1973',
 'evidences': [{'url': 'http://en.wikipedia.org/wiki/James_Cunningham_(comedian)',
   'snippet': 'James Cunningham (born 1973 or 1974) is a Canadian stand-up comedian and TV ho
st.'}]}
```

# Data Cleaning

- Dataset shows significant class imbalance, with the 'institution' class representing approximately 70 percent of the data.
- Applying **under-sampling techniques.**
- balanced dataset with 9250 texts,

evenly distributed across 5 classes,

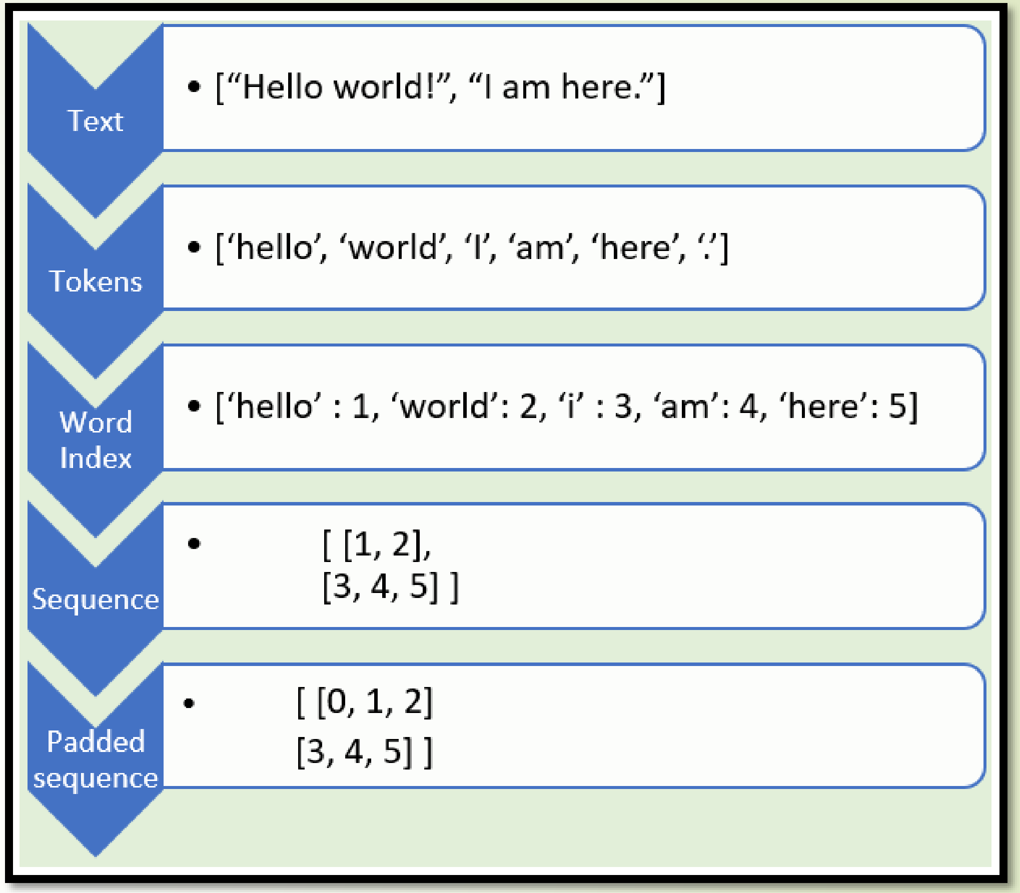
Each class containing 1850 texts.



# Data Preprocessing

**1.Data preprocessing stage:**

- Tokenization
- Converting Text Data to Numerical Matrices
- Creating Matrices (Sequence)
- Padding Sequences




**2. Splitting Data into Training and Validation Sets (80 %, 20 %)**

# Objectives

- 1. Investigate the performance of LSTM and bi-directional LSTM models in relation classification.
- 2. Evaluate the effectiveness of BERT models with different layers in the same task.
- 3. Compare and analyze the results to understand the potential and limitations of both Models.
- 4. Using Knowledge Graph (KG) techniques to represent the relation between subjects and objects by using provided predicates.

# LSTM Models

LSTMs can remember information for **longer periods**, allowing them to capture complex relationships between words even when they are far apart in a sentence.

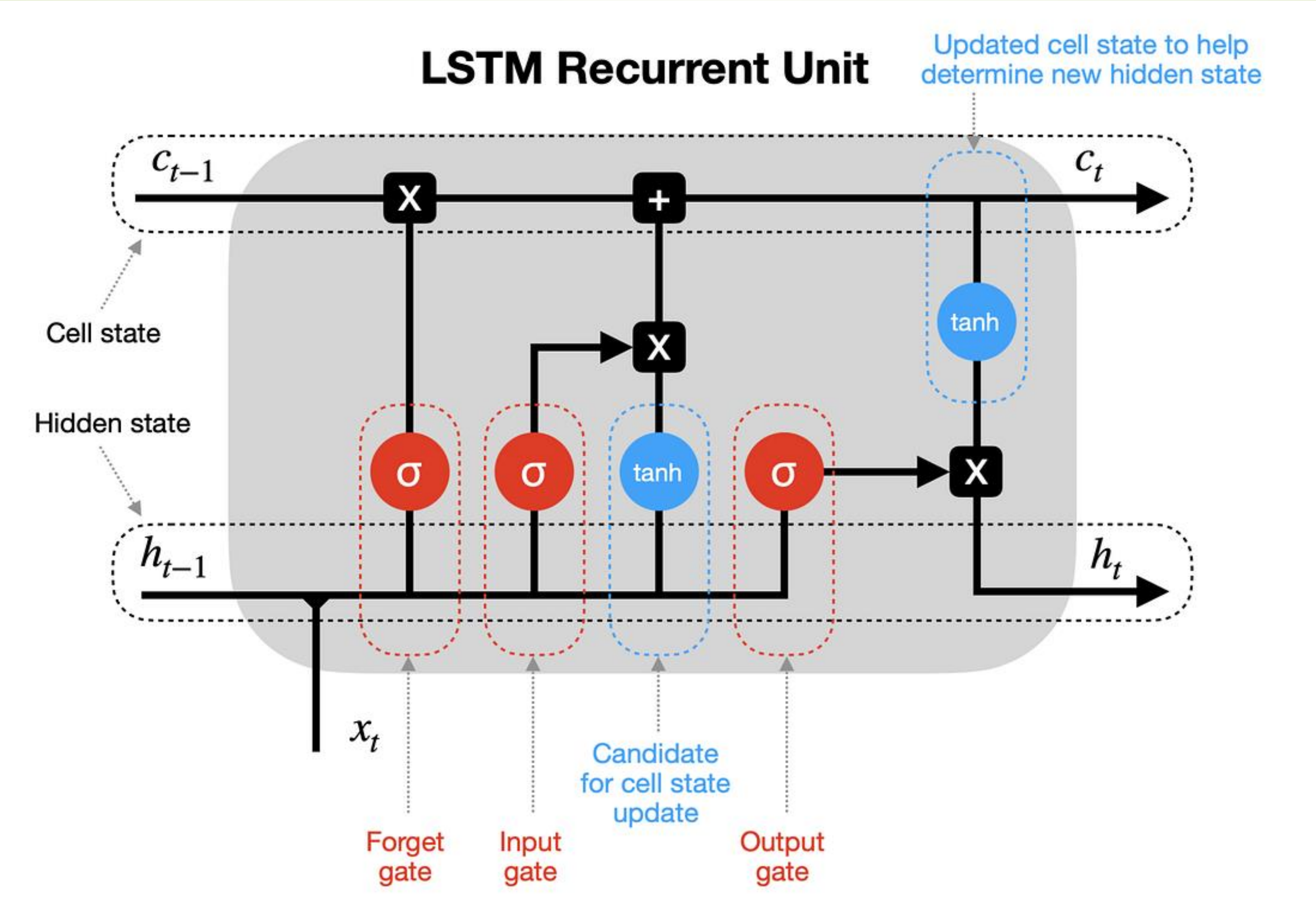
- **Input Layer**: processes sentences with a fixed maximum length.
- **Embedding Layer:** This layer converts raw text into numerical representations that capture word meaning and context.
- **LSTM Layers:** The core of the model, 2 LSTM layers with 128 units capture long-range dependencies within the sequence.
- **Dropout Layer:** 0.30 percent
- **Output Layer**: The final layer uses a linear transformation (`nn.Linear`) to map the hidden state (ht) from the LSTM layer to the output dimension (5).

```
class LSTMClassifier(nn.Module):
    def __init__(self, vocab_size, embedding_dim, hidden_dim, output_dim, n_layers=2, dropout=0.3):
        super(LSTMClassifier, self).__init__()
        self.embedding = nn.Embedding(vocab_size, embedding_dim, padding_idx=vocab['<pad>'])
        self.lstm = nn.LSTM(embedding_dim, hidden_dim, num_layers=n_layers, dropout=dropout, batch_first=True)
        self.dropout = nn.Dropout(dropout)
        self.fc = nn.Linear(hidden_dim, output_dim)

    def forward(self, x, lengths):
        x = self.embedding(x)
        packed_x = nn.utils.rnn.pack_padded_sequence(x, lengths.cpu(), batch_first=True, enforce_sorted=False)
        packed_output, (hidden, _) = self.lstm(packed_x)
        output = self.dropout(hidden[-1])
        output = self.fc(output)
        return output
```

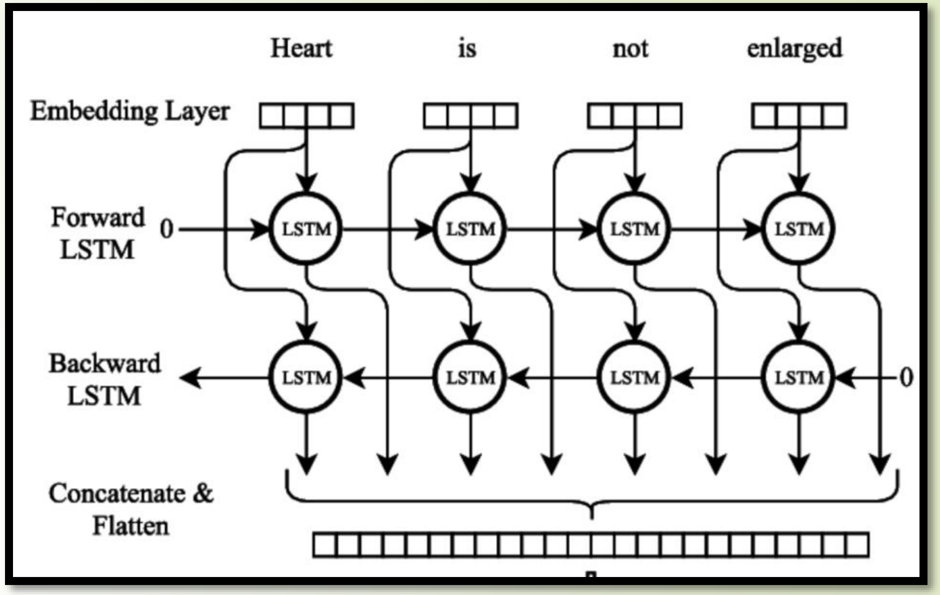
# LSTM Architecture





# Bidirectional LSTM Models

BiLSTM is a term used for a sequence model that contains **two LSTM layers**, one for processing input in the **forward direction** and the other for processing in the **backward direction.** Their hidden states are concatenated along the feature dimension (dim=1) to create a single vector.




- **Input Layer (**max length and padded**)**
- **Embedding Layer**
- **Bi-Directional LSTM Layer (**`bidirectional=True`**)**
- **Dropout Layer**
- **Output Layer (**`hidden_dim*2, Concate hidden State (h) of forward and backword`**)**

```
class BiLSTMClassifier(nn.Module):
    def __init__(self, vocab_size, embedding_dim, hidden_dim, output_dim, n_layers=2, dropout=0.3):
        super(BiLSTMClassifier, self).__init__()
        self.embedding = nn.Embedding(vocab_size, embedding_dim, padding_idx=vocab['<pad>'])
        # bidirectional=True , and in FC: hidden_dim * 2
        self.lstm = nn.LSTM(embedding_dim, hidden_dim, num_layers=n_layers, dropout=dropout, batch_first=True, bidirectional=True)
        self.dropout = nn.Dropout(dropout)
        self.fc = nn.Linear(hidden_dim * 2, output_dim)

    def forward(self, x, lengths):
        x = self.embedding(x)
        packed_x = nn.utils.rnn.pack_padded_sequence(x, lengths.cpu(), batch_first=True, enforce_sorted=False)
        packed_output, (hidden, _) = self.lstm(packed_x)
        hidden_cat = torch.cat((hidden[-2,:,:], hidden[-1,:,:]), dim=1)
        output = self.dropout(hidden_cat)
        output = self.fc(output)
        return output
```

# BERT Models

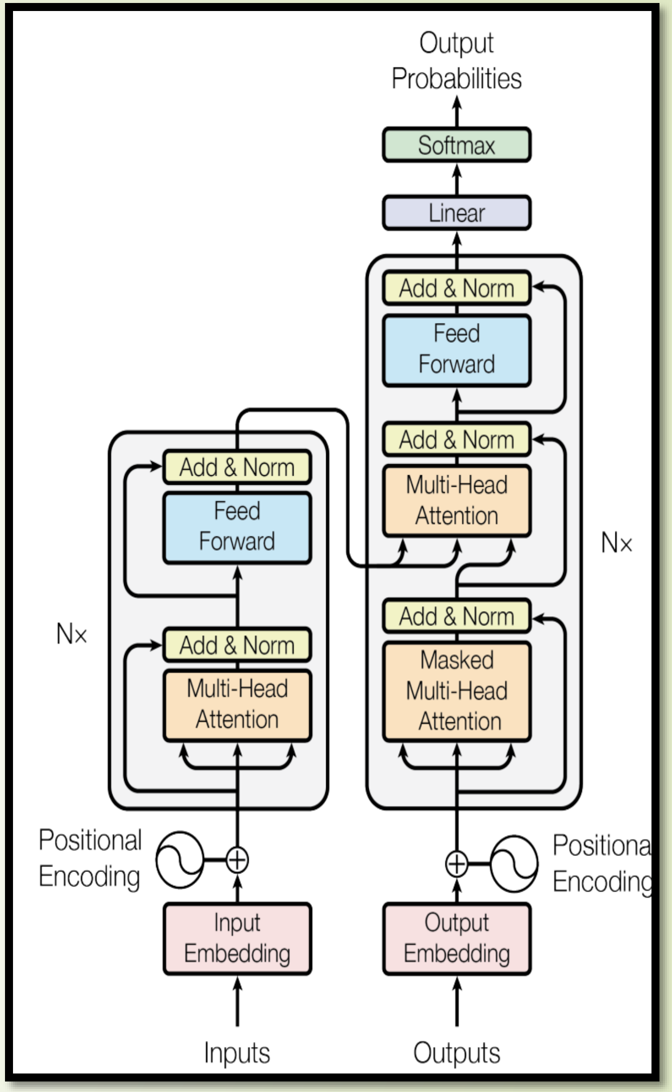


BERT (Bidirectional Encoder Representations from Transformers) employs a transformer architecture, allowing it to capture contextual information from **both the left and right sides of a word** in a sentence.

- **Input Layer (**input_id and attention_mask**)**
- **BERT Layer:** We use the pre-trained "bert-base-uncased" model to obtain contextual embeddings of the input text.
- **Classification Layer:** A final linear layer maps the transformed features to the number of labels (5 in this case).

```
class BERTClassifier(nn.Module):
    def __init__(self, num_labels):
        super(BERTClassifier, self).__init__()
        self.bert = BertModel.from_pretrained('bert-base-uncased')
        self.classifier = nn.Linear(self.bert.config.hidden_size, num_labels)

    def forward(self, input_ids, attention_mask):
        outputs = self.bert(input_ids=input_ids, attention_mask=attention_mask)
        cls_output = outputs.pooler_output
        logits = self.classifier(cls_output)
        return logits
```



# BERT_Larg Models



Additional layers for improved feature extraction and classification. The structure includes:

- **Input Layer (**input_id and attention_mask**)**
- **BERT Layer:** We use the pre-trained “bert-base-uncased" model
- **Dropout Layer:** dropout rate of 20% , prevent overfitting.
- **Fully Connected Layer:** A linear layer reduces the dimensionality from 768 to 64.
- **ReLU Activation:** A ReLU activation function introduces non-linearity.
- **Classification Layer:** A final linear layer maps the transformed features to the desired number of labels (5 in this case).

```
class BERTClassifier_Larg(nn.Module):
    def __init__(self, num_labels):
        super(BERTClassifier_Larg, self).__init__()
        self.bert = BertModel.from_pretrained('bert-base-uncased')
        self.dropout = nn.Dropout(p=0.2)
        self.linear1 = nn.Linear(768,64)
        self.ReLu = nn.ReLU()
        self.classifier = nn.Linear(64,5)

    def forward(self, input_ids, attention_mask):
        outputs = self.bert(input_ids=input_ids, attention_mask=attention_mask)
        cls_output = outputs.pooler_output
        out = self.dropout(cls_output)
        out = self.linear1(out)
        out = self.ReLu(out)
        logits = self.classifier(out)
        return logits
```

# Result

- **Hyperparameter tuning:**

- **LSTM classifier:**

Different learning rates

2 different hidden dimensions (128 and 256),

with 20 epochs.

- **BERT classifier:**
- Different learning rates
- 2 different epochs (4 and 5)

# Hyperparameter Tuning Report for LSTM and Bidirectional LSTM Model

**Average F1**-Score for different learning rate

| Model | Hidden Dimension | **Avg** F1-Score | | | | | |
|---|---|---|---|---|---|---|---|
| | | Lr = 1e-05 | Lr = 1e-04 | Lr = 0.001 | Lr = 0.005 | Lr = 0.01 | Lr = 0.05 |
| LSTM | **128** | 0.56 | 0.68 | 0.69 | 0.72 | 0.72 | 0.07 |
| | 256 | 0.52 | 0.69 | 0.75 | 0.74 | 0.71 | 0.07 |
| BiLSTM | **128** | 0.69 | 0.80 | 0.79 | 0.75 | 0.61 | 0.07 |
| | 256 | 0.70 | 0.82 | 0.81 | 0.72 | 0.07 | 0.07 |

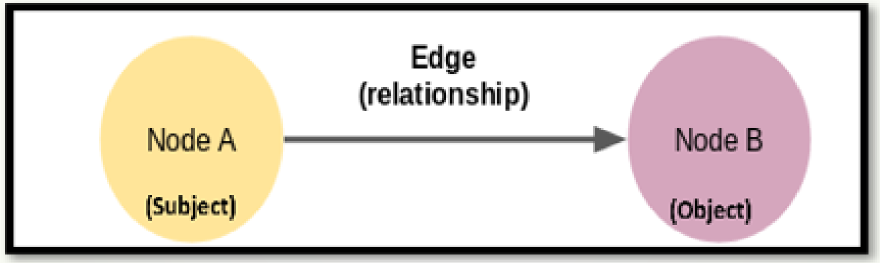
# Hyperparameter Tuning Report for BERT and BERT_Large Model

**Average F1**-Score for different learning rate and two Epochs

| Model | Epoch number | **Avg** F1-Score | | | | | |
|---|---|---|---|---|---|---|---|
| | | Lr = 1e-05 | Lr = 1e-04 | Lr = 0.001 | Lr = 0.005 | Lr = 0.01 | Lr = 0.05 |
| BERT | **4** | 0.87 | 0.86 | 0.07 | 0.07 | 0.07 | 0.06 |
| | 5 | 0.88 | 0.85 | 0.06 | 0.06 | 0.07 | 0.06 |
| BERT_Large | **4** | 0.88 | 0.85 | 0.07 | 0.07 | 0.07 | 0.06 |
| | 5 | 0.89 | 0.77 | 0.06 | 0.07 | 0.07 | 0.06 |

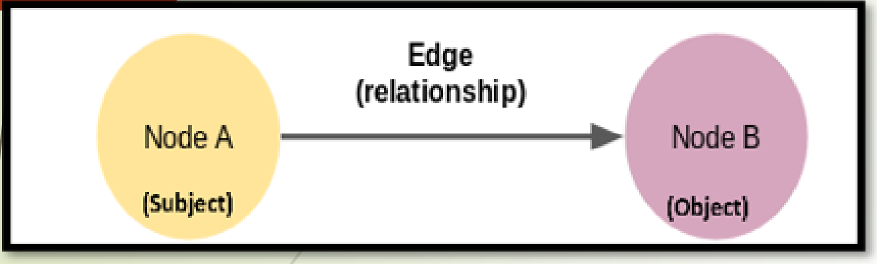
# Knowledge Graph

- A knowledge graph is a way of storing data that resulted from an information extraction task.
- Many basic implementations of knowledge graphs make use of a concept we call **triple.**
- **Triple** is a set of three items(a **subject**, a **predicate** and an **object**)




- **Graph nodes** represented entities (e.g., person name, places name, type of degrees, date of birth),
- **Edges** represented the classified relations (e.g., place of birth, date of birth, place of death, degree, institution).
- Using tools like **"Networkx"** library and "**plotly.graph_objects"** to visualize

# K_G



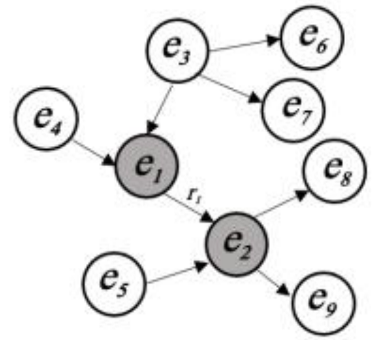




**Fig. 1** An example of a knowledge graph. In this knowledge graph, $(e_1, r_1, e_2)$ is a triplet that indicates $e_1$ and $e_2$ are connected by relation $r_1$.

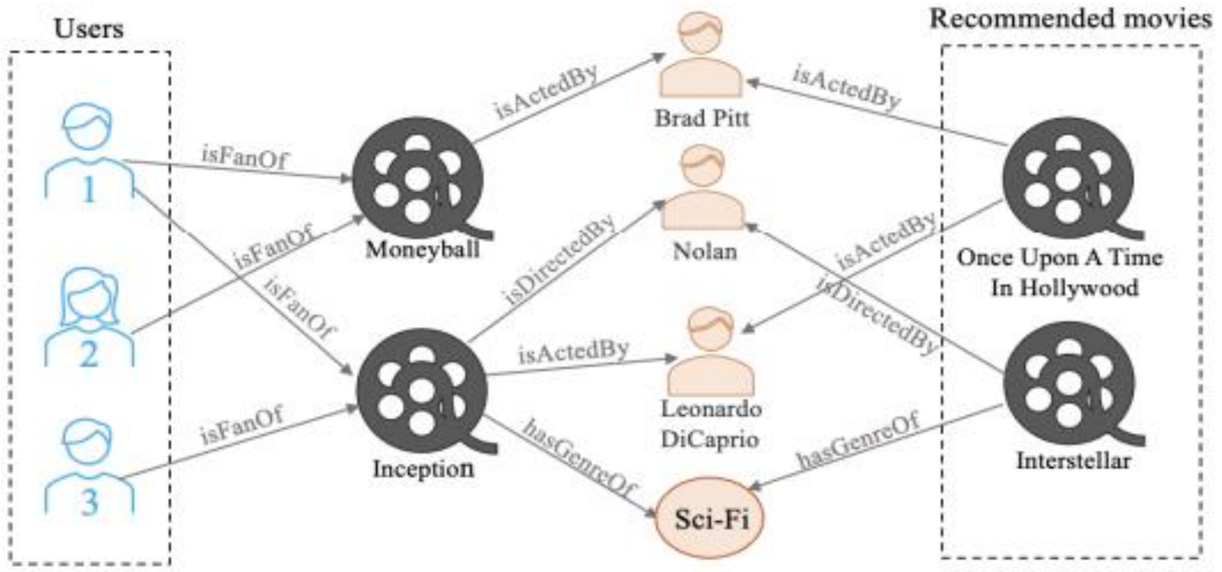
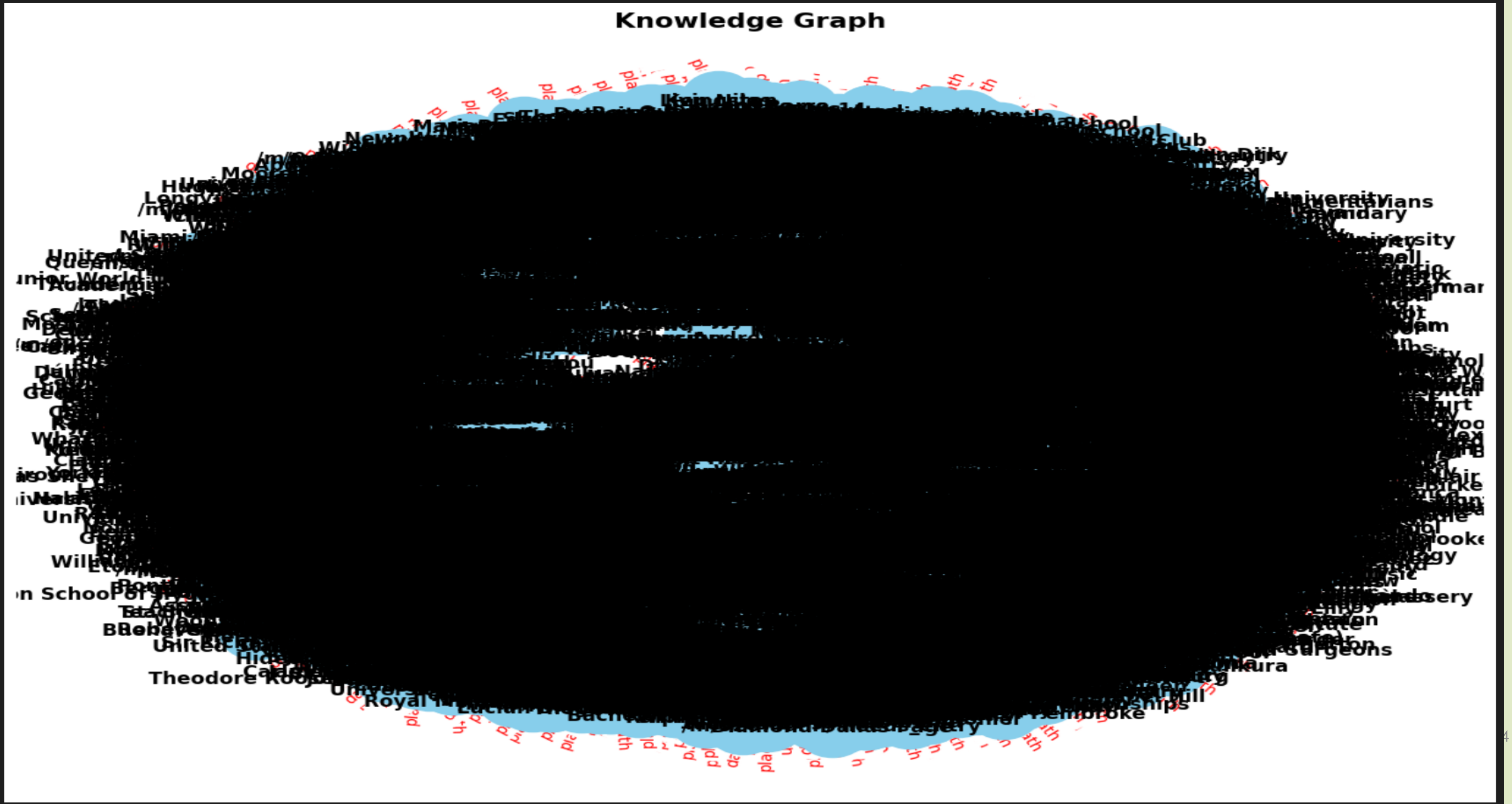
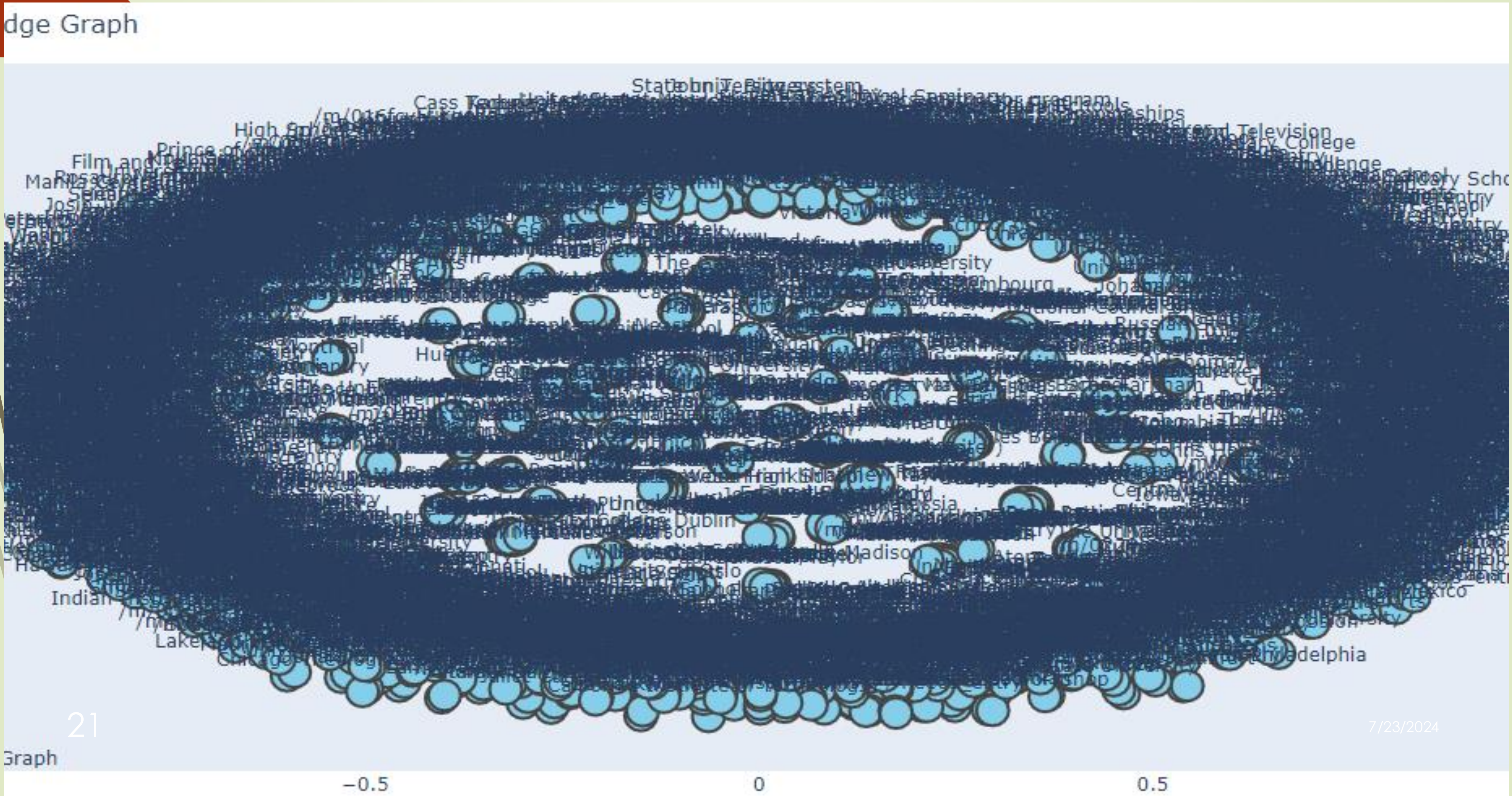



**Fig. 3** An example of knowledge graph-based recommender system.

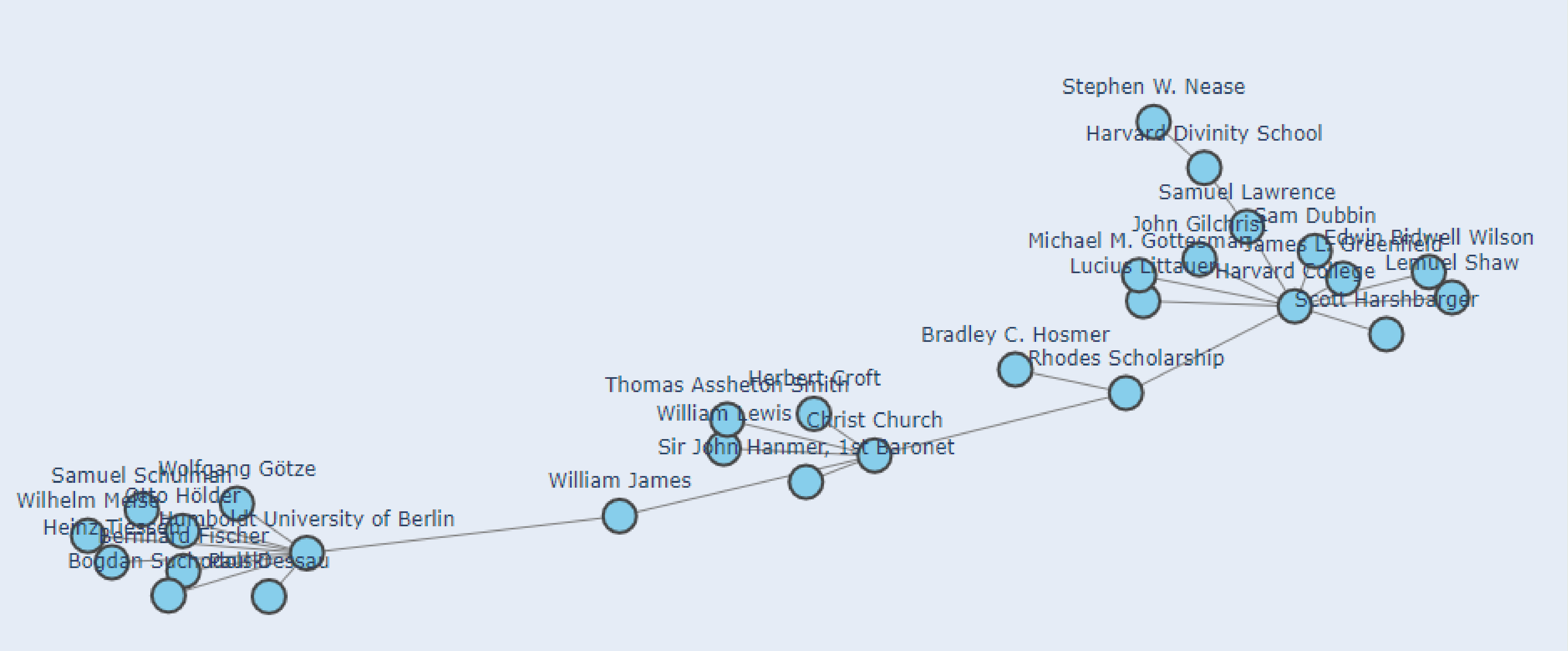
# Static K_G



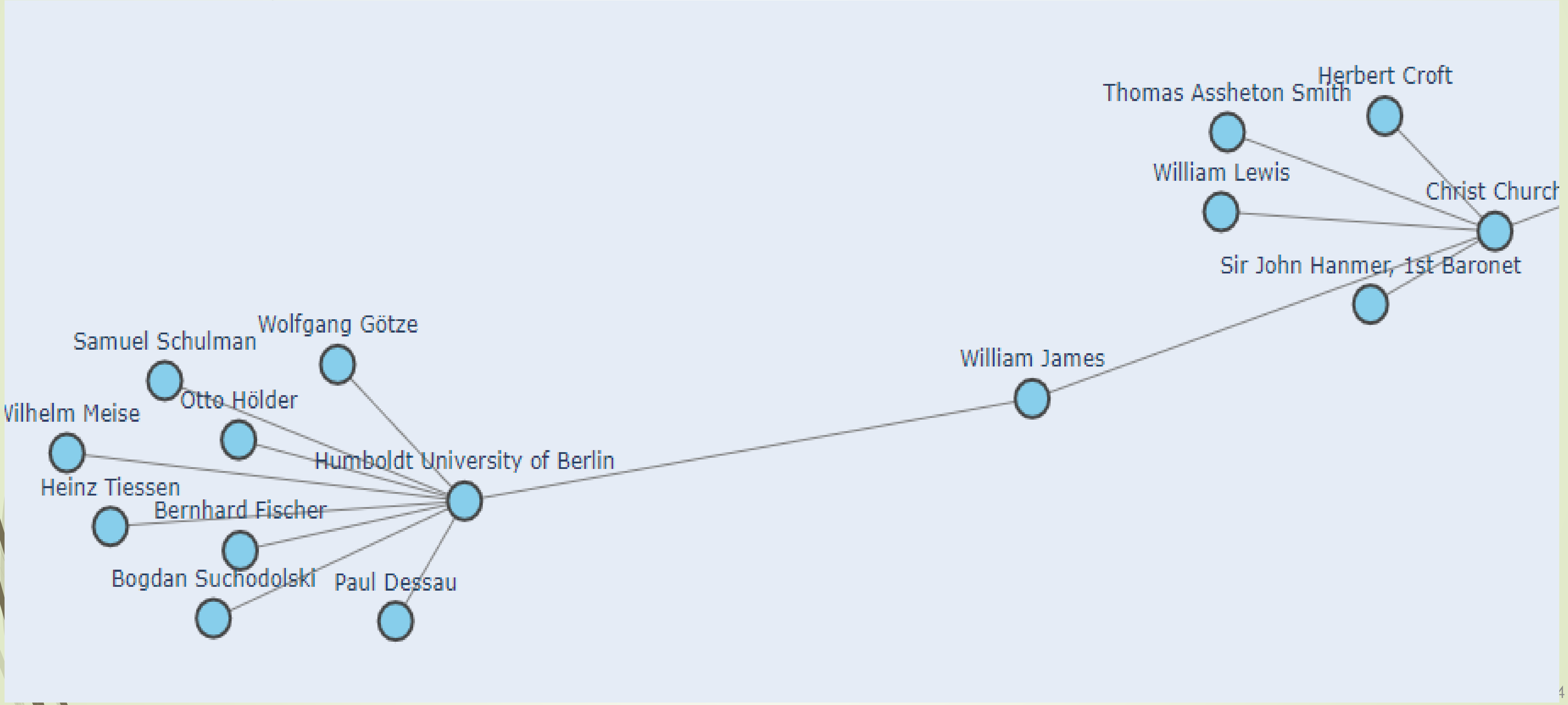
# Dynamic K_G



# Dynamic K_G, by Zoom

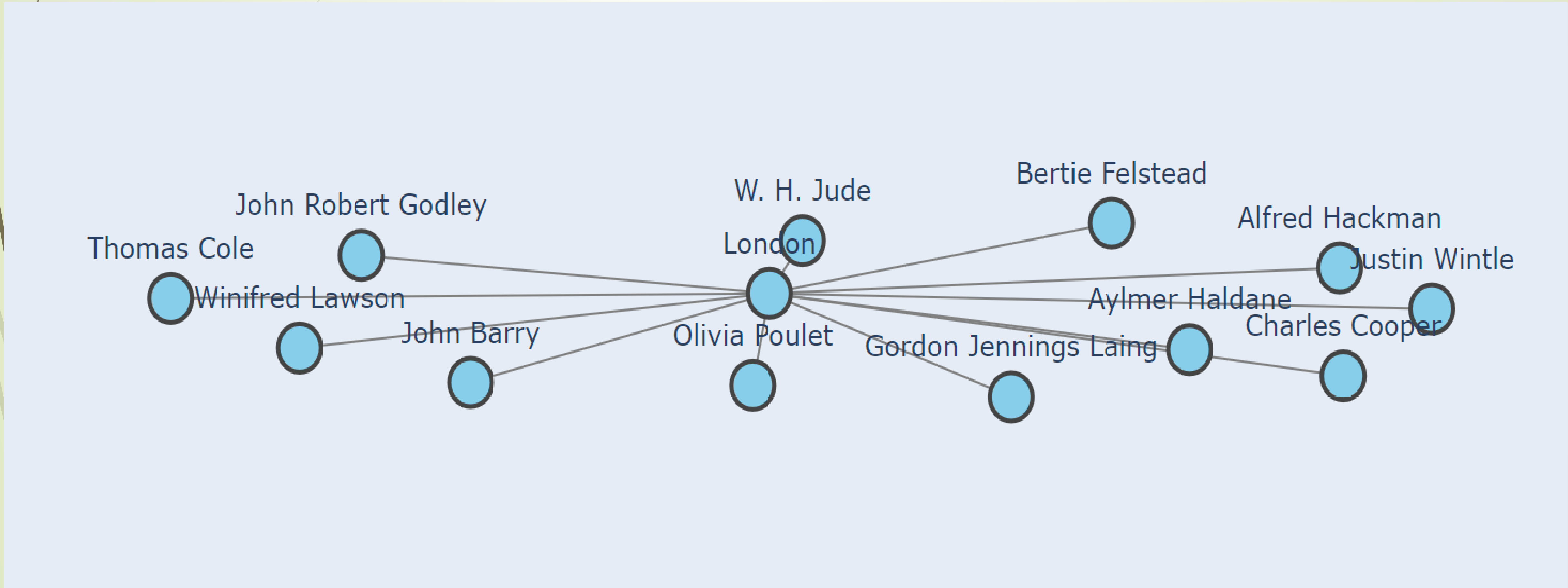


# K_G


Herbert Croft
Thomas Assheton Smith
William Lewis
Christ Church
Sir John Hanmer, 1st Baronet
Wolfgang Götze
Samuel Schulman
William James
Otto Hölder
Vilhelm Meise
Humboldt University of Berlin
Heinz Tiessen
Bernhard Fischer
Bogdan Suchodolski
Paul Dessau

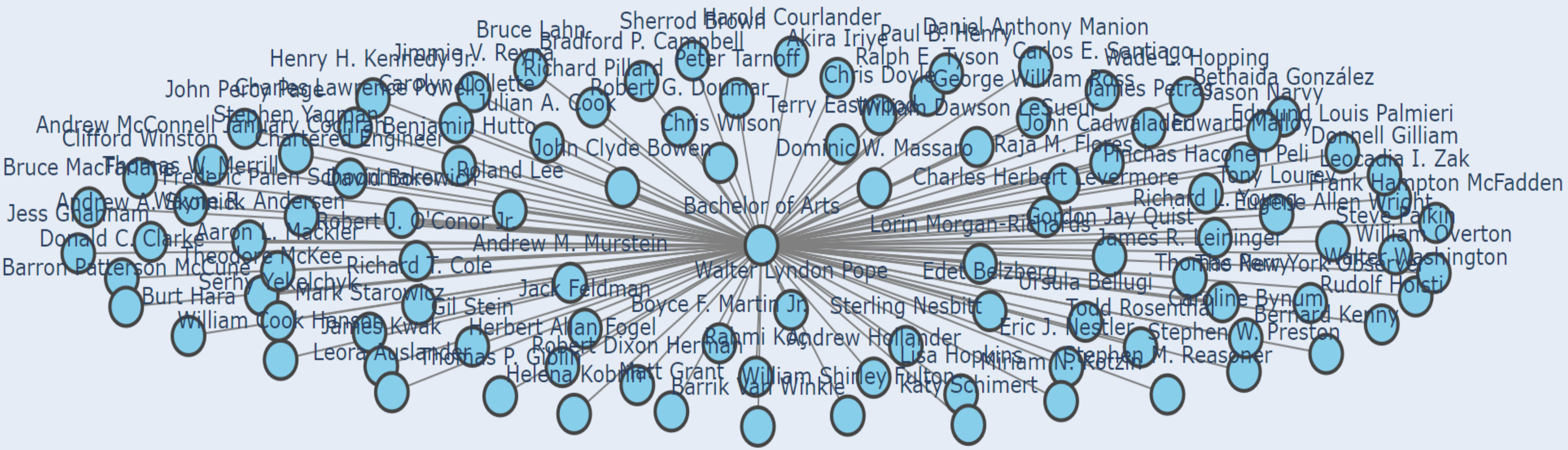

Theresa McDermott
1967
Mary Ann Kennedy
Glasgow
John Baird

# K_G



John Robert Godley
Thomas Cole
Winifred Lawson
John Barry
W. H. Jude
London
Olivia Poulet
Bertie Felstead
Alfred Hackman
Justin Wintle
Aylmer Haldane
Charles Cooper
Gordon Jennings Laing

# K_G


Bachelor of Arts
Harold Courlander
Sherrod Brown
Bruce Lahn
Bradford P. Campbell
Akira Iriye
Paul B. Henry
Daniel Anthony Manion
Jimmie V. Reyna
Henry H. Kennedy Jr.
Peter Tarnoff
Ralph E. Tyson
Carlos E. Santiago
Wade L. Hopping
Richard Pillard
Chris Doyle
George William Ross
Bethaida González
Carolyn Collette
Robert G. Doumar
James Petras
Jason Narvy
John Percy Page
Charles Lawrence Powell
Julian A. Cook
Terry Eastwood
William Dawson LeSueur
Edmund Louis Palmieri
Andrew McConnell
Stephen Yagman
January Cochran
Benjamin Hutto
Chris Wilson
John Cadwalader
Edward Malloy
Donnell Gilliam
Clifford Winston
Chartered Engineer
John Clyde Bowen
Dominic W. Massaro
Raja M. Flores
Pinchas Hacohen Peli
Leocadia I. Zak
Bruce MacFarlane
Thomas W. Merrill
Frederic Palen Schoonmaker
David Borowich
Roland Lee
Charles Herbert Levermore
Tony Lourey
Frank Hampton McFadden
Wayne R. Andersen
Richard L. Young
Eugene Allen Wright
Steve Paikin
Robert J. O'Conor Jr.
Lorin Morgan-Richards
Gordon Jay Quist
William Overton
Donald C. Clarke
Aaron L. Mackler
James R. Leininger
Andrew M. Murstein
Theodore McKee
Richard T. Cole
Thomas Perry
The New York Observer
Walter Washington
Barron Patterson McCune
Walter Lyndon Pope
Edet Belzberg
Ursula Bellugi
Serhiy Yekelchyk
Jack Feldman
Rudolf Holsti
Mark Starowicz
Caroline Bynum
Burt Hara
Gil Stein
Boyce F. Martin Jr.
Sterling Nesbitt
Todd Rosenthal
Bernard Kenny
William Cook Hanson
James Kwak
Herbert Allan Fogel
Eric J. Nestler
Stephen W. Preston
Andrew Hollander
Robert Dixon Herman
Leora Auslander
Thomas P. Giblin
Lisa Hopkins
Miriam N. Kotzin
Stephen M. Reasoner
Matt Grant
William Shirley Fulton
Barrik Van Winkle
Katy Schimert

# Thanks