Faculdade de Ciências e Tecnologia

Data Modelling

Knowledge graphs for social network analysis

Anna Biausque
71935

Agnieszka Jedrzejak
71754

December 15, 2024

Contents

1 Abstract

In a world where social networks play a central role in information dissemination and the structuring of human interactions, analyzing these networks is essential to understanding social dynamics and collective behavior. Reddit, as a major platform for discussion and exchange, offers a treasure trove of data to explore these interactions through an analytical lens.

This project is situated in the field of Social Network Analysis (SNA), with the primary objective of modeling a Reddit dataset using various knowledge graph approaches. These models enable the structuring of data into relational graphs, thereby facilitating the extraction of complex information about relationships between users, subreddits, and discussed topics.

By constructing multiple graph-based representations of Reddit data, this project aims to compare these models through a series of relevant metrics to highlight their specificities, strengths, and limitations.

2 Research

2.1 What is a Knowledge Graph?

A Knowledge Graph is a structured representation of information in the form of nodes (entities) and edges (relationships), enriched with metadata, semantic attributes, and sometimes ontologies. It allows modeling complex knowledge in a way that is computationally exploitable.

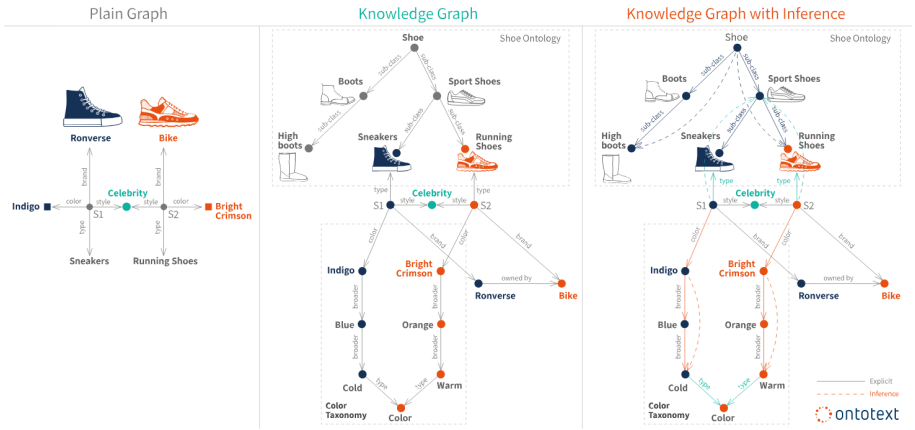


Figure 1: Difference between a plain graph and a knowledge graph [3]

2.2 Why Use Knowledge Graphs in SNA?

2.2.1 Flexible Data Modeling

Social networks are naturally represented as graphs, with nodes representing entities (users, posts, hashtags) and edges representing relationships (interactions, subscriptions). KGs add a semantic layer to this structure, such as contextual attributes (location, topic, temporal information)

2.2.2 Integration of External Knowledge

KGs allow the integration of external knowledge to contextualize social interactions: Ontologies (e.g., DBpedia, Wikidata) to associate entities with broader concepts. Links to external databases to enrich metadata. Relevance for SNA: Identify implicit relationships between users or content by leveraging external information, such as mapping hashtags to knowledge domains.

2.2.3 Multidimensional Analysis

Unlike simple graphs, KGs enable the modeling of multidimensional networks:

- Social dimensions: Relationships between users.

- Thematic dimensions: Topics discussed (tags, categories).
- Temporal dimensions: Evolution of interactions over time.

2.3 Examples and models

The paper titled "From Social Networks to Knowledge Graphs: A Plea for Interdisciplinary Approaches", authored by Jens Dörpinghaus, Sonja Klante, Martin Christian, Christof Meigen, and Carsten Düing, explores the intersection of data science, graph theory, and social network analysis. Among its contributions, the paper proposes a knowledge graph-based model for representing multi-layered social networks.

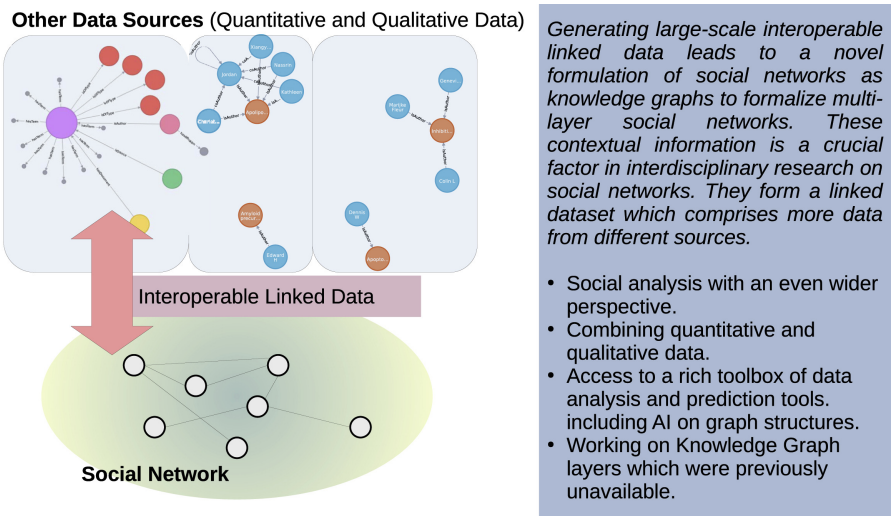


Figure 2: Combining Social Network Analysis with interoperable data from other data sources and fields and building a Knowledge Graph. [2]

The Politoscope project, led by the french researcher David Chavalarias, analyzes political discourse and interactions on social networks to uncover patterns of influence, manipulation, and polarization. It leverages large-scale data analysis and visualization tools to track the dynamics of online communities, focusing on how narratives and alliances evolve over time.

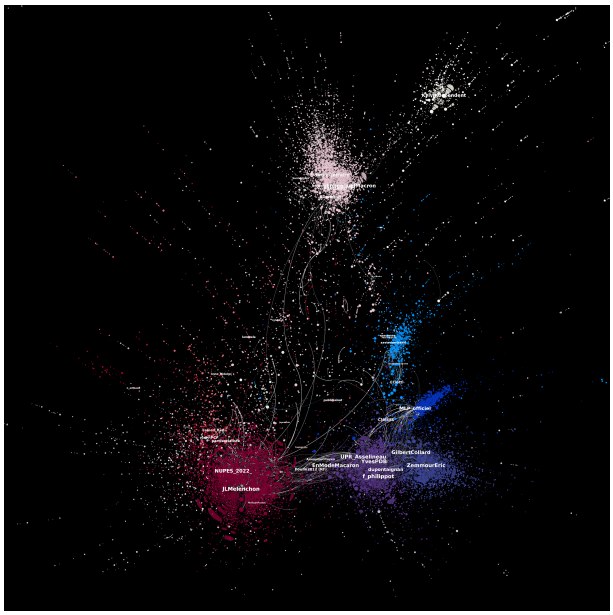


Figure 3: Political Twittersphere from June 10 to June 16, 2022 (France)

2.4 Knowledge graphs with Neo4j

Building a knowledge graph with Neo4j involves several key steps to effectively model, integrate, and utilize interconnected data. According to Neo4j's Building Knowledge Graphs: A Practitioner's Guide

[1], the process includes:

- **Understanding Knowledge Graph Fundamentals**: Grasp the core principles, components, and significance of knowledge graphs in the modern data landscape.
- **Data Modeling and Integration**: Develop a schema that accurately represents entities and their relationships within your domain. Then, import data from your sources, ensuring proper alignment with the established schema.
- **Data Enrichment and Semantics**: Incorporate ontologies to provide semantic context and enhance data interoperability.
- **Querying and Analysis**: Utilize Cypher to retrieve and manipulate data within the graph and apply graph algorithms to uncover patterns, relationships, and insights.
- **Visualization and Interaction**: Employ visualization tools like Neo4j Bloom to explore and present the knowledge graph interactively.

By following these steps, practitioners can construct robust knowledge graphs that facilitate advanced analytics and informed decision-making.

3 Theoretical work on the graph representations

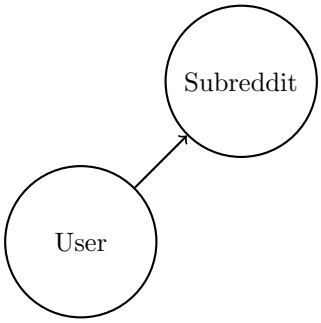
3.1 Which models ?

We thought about different models of knowledge graphs to perform an analysis on Reddit posts, that would allow us to satisfy the following conditions :

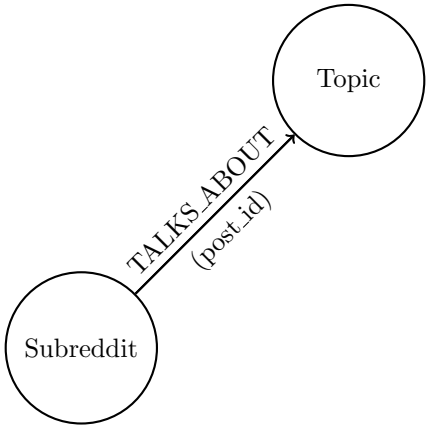
- We want to be able to perform a quick visual analysis of our graph representations: unique characteristics should appear depending on the chosen representation (communities, importance of certain topics, particular links, etc.).
- Aside from the visualization, we want to be able to evaluate the quality of our representation based on certain conditions. For instance, we want to represent the evolution of the data
- We finally want to assess the quality of our representation with concrete metrics (centrality for instance, but also metrics that would fit best our goals and that we would implement ourselves).

3.1.1 The basic graph for SNA

In the initial phase, we wanted to create graph representation of users connected to subreddits, which reflects that they posted in particular subreddit. Our idea was to create users and subreddits as nodes and posts written by users as relationships between the user (author of the post) and the subreddit in which the user posted in. We decided to start with a smaller dataset of about 40,000 rows to be able to display it as a graph database in Neo4j. In the table below we can see ranking of subreddit names ordered by the number of postes in subreddit to see how accurate the names of subreddits are.



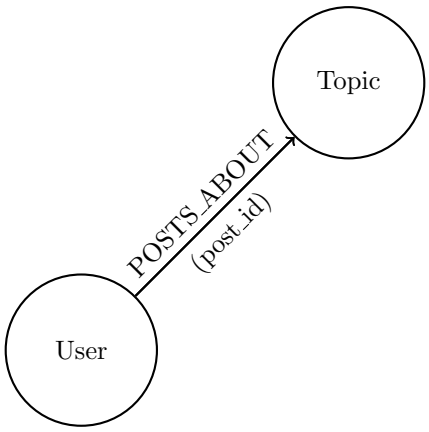
3.1.2 The model Subreddit→Topic



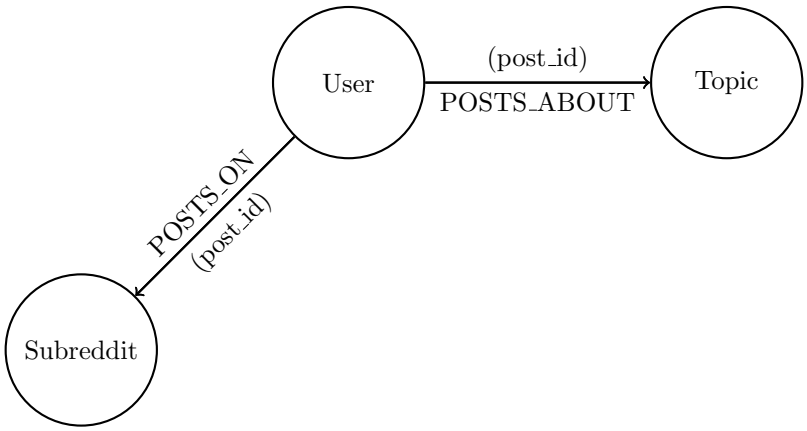
This model allows us to visualize and compute how diverse are the topics evocated in the subreddit (and how relevant are the subreddit titles as a consequence).

- Subreddit attributes: subreddit_name, subreddit_id, degree_score
- Topic attributes: topic_name, topic_id

3.1.3 The model User→Topic

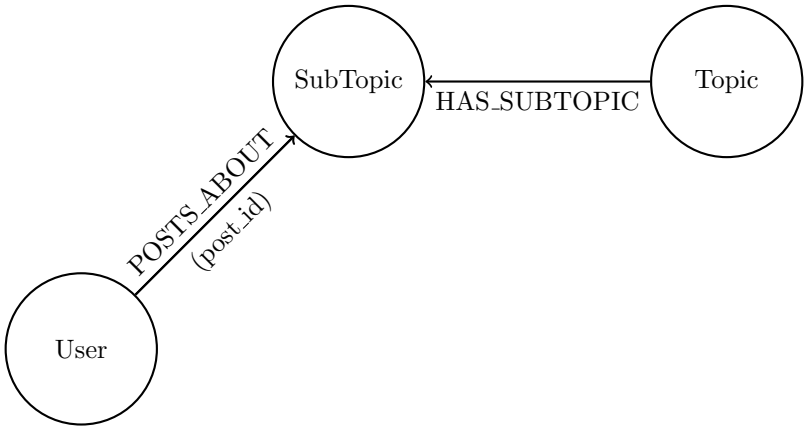


3.1.4 A more advanced knowledge graph : Subreddit←User→Topic



- User attributes: username, user_id
- Topic attributes: topic_name, topic_id, degree_score
- Subreddit attributes: subreddit_name, subreddit_id, degree_score

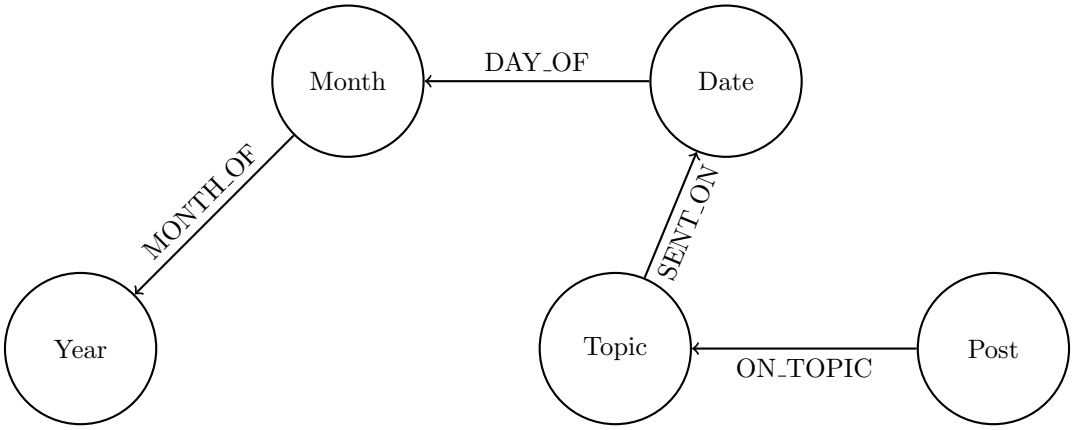
3.1.5 A different ontology : User→SubTopic←Topic



- User attributes: username, user_id
- Topic attributes: topic_name, topic_id, degree_score
- SubTopic attributes: subtopic_name, subtopic_id, parent_topic.

This model represents a knowledge graph as it captures structured data as entities (nodes) and relationships (edges) within a defined domain, enriched granularity on the topics. That will allow us to pursue a more precise and meaningful analysis, particularly for communities detections and underlining intersections between communities.

3.1.6 Time representation: Year ← Month ← Date ← Topic ← Post



This model represents the topics that were the most popular in a specific month. Because we were using dataset with only one type of Subreddit, as topics we mean words that were the most commonly used - for each post we chose up to three the most frequent words after removing words like 'the','and' and commas and then connected them to a specific post.

3.2 Which metrics ?

The analysis of graphs constructed from a dataset of Reddit posts leverages various metrics rooted in Social Network Analysis (SNA). These metrics provide critical insights into the structure, function, and interaction patterns within the network. By focusing on the following metrics (we won't compute all of them but the most relevant ones), we aim to capture both global and local characteristics of the network, reflecting its connectivity, influence distribution, and community structure.

3.2.1 Density

Density measures the proportion of potential connections in the graph that are actual connections. This metric helps understand how tightly connected the network is. A dense graph suggests frequent interactions between nodes (e.g., users or topics), which can indicate a cohesive community. Conversely, a

sparse graph may highlight fragmented discussions or isolated topics.

Why Density?
Density is essential for understanding the overall connectivity of the network and for identifying whether the interactions are evenly distributed or concentrated. Studies such as Wasserman and Faust (1994) highlight that density can serve as a baseline measure of cohesion in social networks, offering a starting point for more detailed structural analyses.

3.2.2 Diameter

The diameter represents the longest shortest-path distance between any two nodes in the graph. It reflects the graph's overall size in terms of reachability and highlights how far information or influence must travel to connect the most distant nodes.

Why Diameter?
The diameter provides insights into the efficiency of communication within the network. Networks with a smaller diameter allow for rapid dissemination of information, whereas larger diameters might indicate disconnected or loosely connected communities.

3.2.3 Centrality Metrics

Centrality metrics assess the importance of nodes within the network. We use the following centrality measures:

Degree Centrality : Measures the number of direct connections (incoming or outgoing) a node has. It identifies highly connected nodes that act as hubs within the network.

Betweenness Centrality : Identifies nodes that frequently occur on the shortest paths between other nodes, highlighting key influencers or gatekeepers in the network.

Eigenvector Centrality : Measures a node's influence based not just on its connections but on the connections of its neighbors, providing a transitive measure of influence.

Why Centrality? Centrality metrics capture different dimensions of influence and connectivity. For instance, degree centrality identifies local hubs, while eigenvector centrality highlights nodes with global influence. Borgatti and Everett (2006) emphasized the relevance of these metrics in understanding node roles within a network.

3.2.4 Community Detection

Identifying clusters or communities within the network helps reveal groups of nodes that interact more frequently with each other than with the rest of the network. We utilize:

Louvain Algorithm: Detects modular structures by optimizing modularity scores to find dense clusters.

Label Propagation: Identifies communities through iterative propagation of labels, allowing partial supervision for tailored clustering.

Weakly Connected Components (WCC): Identifies subgraphs where every node is reachable from any other node within the same component.

Why Community Detection? Community detection metrics highlight the underlying structure of the graph, uncovering patterns in user interactions and topic associations. These insights are valuable for understanding topic-specific engagement and identifying tightly knit subcommunities. Studies such as Blondel et al. (2008) emphasize the utility of Louvain for modularity-based clustering in large-scale networks.

3.2.5 Diversity and Clusters

Diversity of Network Members: Analyzes the presence of various types of actors (e.g., different subreddits or user categories).

Number of Clusters: Quantifies isolated groups where nodes interact frequently within the cluster but sparingly with others.

Why Diversity and Clusters? Diversity and clustering metrics provide insights into the heterogeneity of actors and the segmentation of the network. These aspects are critical for understanding the roles of niche communities versus generalist contributors.

3.3 Datasets

For our base analysis, we decided to go with a 4 million posts dataset. In order to test freely different models without computation issues, we worked with a sub-dataset of 40 000 posts along the project and tested our methods at the end with the whole dataset.

The chosen data set gives us the following information :

"author"	"post_id"	"subreddit"	"subreddit_id"	"content"

You can visualize the dataset following this link.

For the time-based analysis, we chose a dataset that focuses on a particular topic, in order to spot the evolutions of the models along time knowing the irl evolutions concerning the topic.

Our second dataset contains 40.3 thousand rows and has timestamps. We added a new column with a date yyyy-mm-dd based on the "created utc" column.

"type"	"id"	"subreddit.id"	"created utc"	"selftext"	"date"	"top 3 words

3.4 Method

To be able to achieve these goals, we realized that our "plain data set" would not be sufficient.

The subreddits are an interesting and useful feature to represent our data : they are "pre-made" communities and should represent the topics discussed within. However, some of them are really general: "AskReddit", which is one of the most famous, is a place where Reddit users can ask any question they want. Thus, representing the interactions only used the untransformed data lacks interest, and reduces the window of opportunities to design different knowledge graphs.

3.4.1 First method

The first idea we had to extract topics from the posts was to extract to semantically relevant words from each post, group them as meaningful clusters and link each post to the clusters.

3.4.1.1 First step : Preprocessing

The `preprocess()` function is designed to prepare textual data for subsequent natural language processing (NLP) tasks by cleaning and standardizing it.

```
from nltk.stem import SnowballStemmer
from gensim.utils import simple_preprocess

def is_meaningful_word(word):
    return wordnet.synsets(word)

def preprocess(text, stop_words, stemmer_language="english"):

    if not text:
        raise ValueError("Input text is empty or invalid.")

    # Initialize the stemmer
    stemmer = SnowballStemmer(stemmer_language)

    # Tokenize, filter, and stem tokens
    result = []
    for token in simple_preprocess(text, deacc=True):
        if token not in stop_words
        and is_meaningful_word(token)
        and len(token) > 3:
            stemmed_token = stemmer.stem(token)
            result.append(stemmed_token)
    return result
```

Function preprocess(text, stop_words, stemmer_language="english") : Tokenizes and preprocesses the input text, removing stopwords, short tokens, and applying stemming.

Parameters:

- text (str): The input text to extract topics from.
- stop_words (set): A set of stopwords to be removed from the text.
- stemmer_language (str): The language to use for stemming (default is English).
- text (str): The input text to preprocess.

Returns:

- list: A list of preprocessed tokens.

3.4.1.2 Second step : Topic extraction

```
def extract_topics(text, num_topics=2, num_words=2, passes=15):

    if not text:
        raise ValueError("Input text is empty or invalid.")

    stop_words = set(stopwords.words('english'))

    tokens = preprocess(text, stop_words)
    if not tokens:
        raise ValueError("Preprocessing resulted in no
--------valid tokens.")

    dictionary = corpora.Dictionary([tokens])
    corpus = [dictionary.doc2bow(tokens)]

    lda_model = LdaModel(corpus, num_topics=num_topics,
    id2word=dictionary, passes=passes)

    topics = lda_model.print_topics(num_topics=num_topics,
    num_words=num_words)

    return topics
```

Function extract_topics(text, num_topics=2, num_words=2, passes=15): Extracts topics from the input text using Latent Dirichlet Allocation (LDA).

Parameters:

- text (str): The input text to extract topics from.
- num_topics (int): The number of topics to extract.
- num_words (int): The number of words to display for each topic.
- passes (int): The number of passes for LDA training.

Returns:

- list: A list of tuples containing the topic keywords and their weights.

3.4.1.3 Third step : clustering with Bert and K-means

```
from transformers import BertTokenizer, BertModel
import torch
from sklearn.cluster import KMeans
```

```
import numpy as np
import json

tokenizer = BertTokenizer.from_pretrained("bert-base-multilingual-cased")
model = BertModel.from_pretrained("bert-base-multilingual-cased")

with open("distinct_topics.txt", "r", encoding="utf-8") as file:
    words = [line.strip() for line in file]

embeddings = []
for word in words:
    inputs = tokenizer(word, return_tensors="pt", truncation=True)
    outputs = model(**inputs)
    # Using the last layer
    embedding = outputs.last_hidden_state.mean(dim=1).detach().numpy()
    embeddings.append(embedding)

embedding_matrix = np.vstack(embeddings)

num_clusters = 100
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
kmeans.fit(embedding_matrix)

clusters = kmeans.labels_

# Grouping words by cluster
grouped_words = {i: [] for i in range(num_clusters)}
for word, cluster_id in zip(words, clusters):
    grouped_words[cluster_id].append(word)

# Saving clusters in a json file
output_file_json = "clusters_100.json"
with open(output_file_json, mode="w", encoding="utf-8") as json_file:
    json.dump(grouped_words, json_file, ensure_ascii=False, indent=4)
```

3.4.1.4 Cluster visualisation

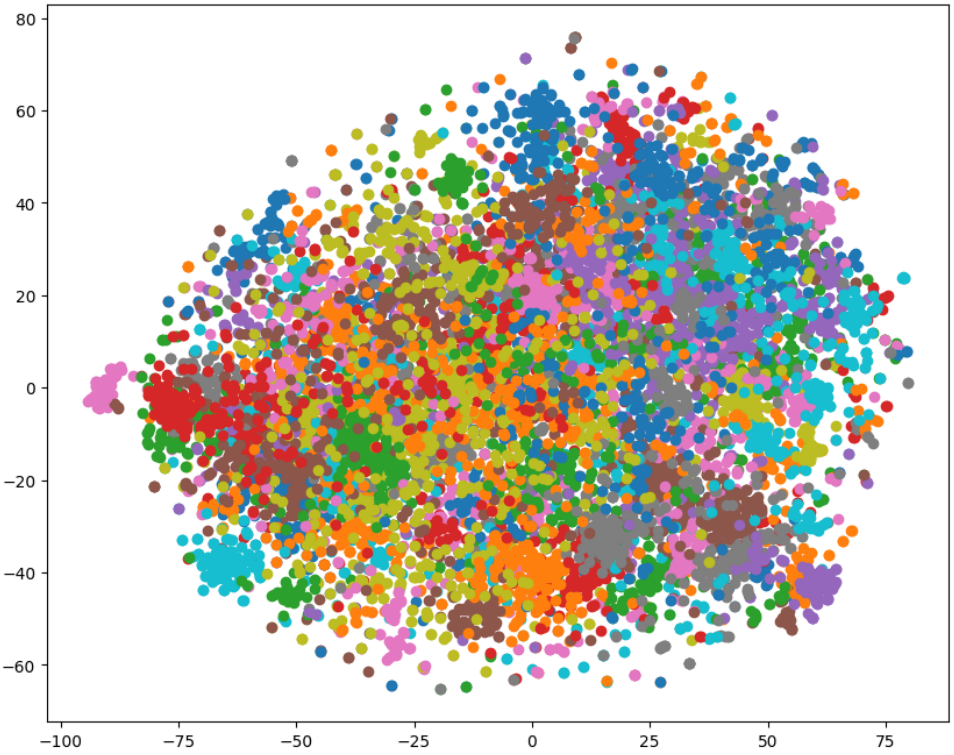


Figure 4: Repartition of the topics associated with their clusters

3.4.1.5 Limits of the method

The goals was to extract from each cluster a "main theme", which could be used for graph visualization and allow a much more understandable graph visualization. However some words in the clusters were too diverse, and we struggled to extract coherent themes from them.

We decided to go for another method with predefined topics.

3.4.2 Second method

3.4.2.1 Use of a predefined list of topics

We searched for a list of predefined topics to avoid the clustering problem we had with the first method. The one we chose is inspired by Wikipedia's taxonomy for articles

```
topics = [
    "Science", "Technology", "Environment", "Politics", "Religion",
    "Philosophy", "Art", "Literature", "Music", "Education", "History",
    "Health", "Psychology", "Economics", "Democracy", "Freedom", "Gender",
    "Family", "Relationship", "Sexuality", "Emotion", "Nature",
    "Animals", "Sports", "Travel", "Humor", "Violence", "Game", "War",
    "Fashion", "Industry", "Food", "Country", "Geography", "Body" , "Justice",
    "Society"
]
```

```
import pandas as pd
import json
from sentence_transformers import SentenceTransformer
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

def classify_documents(texts, predefined_topics,
model_name="all-MiniLM-L12-v2", top_n=2, batch_size=512):

    if not texts or not predefined_topics:
```

```
        raise ValueError("Texts and predefined topics must not be empty.")
    if top_n <= 0:
        raise ValueError("top_n must be greater than 0.")

    # Load the embedding model
    model = SentenceTransformer(model_name)

    # Compute embeddings for predefined topics once
    topic_embeddings = model.encode(predefined_topics, convert_to_tensor=True)

    results = []
    // batch_size  # Calculate total number of batches
    num_batches = (len(texts) + batch_size - 1)

    for batch_idx in range(num_batches):
        print(batch_idx)
        batch_texts = texts[batch_idx * batch_size : (batch_idx + 1) *
        batch_size]
        text_embeddings = model.encode(batch_texts, convert_to_tensor=True)

        # Compute cosine similarity for the batch
        similarity_matrix = cosine_similarity(text_embeddings.cpu().numpy(),
        topic_embeddings.cpu().numpy())

        for i, text in enumerate(batch_texts):
            topic_scores = {predefined_topics[j]: similarity_matrix[i][j]
            for j in range(len(predefined_topics))}
            sorted_topics = sorted(topic_scores.items(), key=lambda x: x[1],
            reverse=True)[:top_n]
            results.append(sorted_topics)

        print(f"Processed batch {batch_idx + 1}/{num_batches}")

    return results
```

The function **classify_documents**(texts, predefined_topics, model_name="all-MiniLM-L12-v2", top_n=2, batch_size=512) classifies a list of documents in batches by their similarity to predefined topics.

Parameters:

- texts (list of str): The input documents to classify.
- predefined_topics (list of str): A list of predefined topics.
- model_name (str): The name of the SentenceTransformer model to use.
- top_n (int): The number of top similar topics to return.
- batch_size (int): The size of each batch.

Returns: list of list: Each element contains the top N topics for the corresponding document.

3.4.3 Timestamps representation

The challenge for the timestamps represantation was to find alternative to subreddit names because of the fact that the dataset with timestamps contains only one type of subreddits, so we decided to extract 3 most common used words for each post as a "key words" to be able to create a representation of words most frequently used in specific time, e.g. in March 2022.

```
import pandas as pd
import string
from collections import Counter
from nltk.corpus import stopwords
```

```
from nltk.tokenize import word_tokenize
import nltk

# Download necessary NLTK data
nltk.download('punkt')
nltk.download('stopwords')

# Load the CSV file
file_path = 'reddit_posts_with_date.csv'
df = pd.read_csv(file_path)

# Prepare stop words and punctuation
stop_words = set(stopwords.words('english'))
punctuation_table = str.maketrans('', '', string.punctuation)

def process_text(text):
    """
    Tokenizes the text, removes stop words and punctuation, and returns a list of cleane
    """
    if not isinstance(text, str):
        return []
    # Tokenize and clean text
    words = word_tokenize(text.lower())
    cleaned_words = [
        word.translate(punctuation_table) for word in words
        if word not in stop_words and word.isalpha()
    ]
    return cleaned_words

def get_top_words(text_list):
    """
    Given a list of words, returns the top 3 most common words.
    """
    word_counter = Counter(text_list)
    return [word for word, _ in word_counter.most_common(3)]

# Apply the text processing and extract top words
df['selftext_cleaned'] = df['selftext'].apply(process_text)
df['top_3_words'] = df['selftext_cleaned'].apply(get_top_words)

# Drop intermediate cleaned text column if not needed
df.drop(columns=['selftext_cleaned'], inplace=True)

# Save the modified DataFrame to a new CSV
output_file = 'reddit_posts_with_top_words.csv'
df.to_csv(output_file, index=False)

print(f"Processed data saved to {output_file}")
```

4 Modelling with knowledge graphs

4.1 Tools for graph visualization

Visual representation plays a crucial role in Social Network Analysis (SNA) as it enables the exploration of patterns, relationships, and structures within networks.

Neo4j is designed to efficiently store and query large graphs, often containing millions of nodes and relationships. However, visualizing such large datasets in a meaningful way is difficult, as traditional graph visualization tools struggle to handle the sheer volume of data. Even if visualization tools can render the data, the human mind finds it hard to interpret overly dense or cluttered graphs. Large

graphs often result in "hairball" visualizations, where connections overlap excessively, making patterns and insights almost impossible to discern.

As a consequence, we have been searching for efficient tools that would allow us to visually represent the graphs.

- **UCINet**: A comprehensive software package designed for analyzing social network data as well as other one-mode and two-mode datasets. It offers robust analytical capabilities and is often complemented by NetDraw, a companion tool specifically used for visualizing networks.
- **Pajek**: A software solution optimized for analyzing and visualizing large networks. Pajek is well-suited for handling extensive datasets and offers advanced features for exploring the structure of massive networks.
- **Gephi**: An open-source and highly extensible software package written in Java, Gephi is one of the most popular tools for network visualization and analysis. It provides an intuitive interface and supports a wide range of network layouts and metrics, making it suitable for both beginners and advanced users.
- **SocNetV (Social Networks Visualizer)**: A user-friendly, free, and open-source tool that simplifies network analysis and visualization. Its ease of use makes it an excellent choice for newcomers to SNA.

These tools not only enhance the visualization of social networks but also provide advanced features for statistical analysis and interactive exploration. The choice of tool often depends on the size of the dataset, the analytical requirements, and the user's level of expertise. By leveraging these platforms, researchers and practitioners can gain deeper insights into the dynamics of social systems and relationships within networks.

4.2 Modelling with Neo4j

For practical purposes (it was easier to manipulate and test with a smaller amount of data because of computational complexity), the following modelling was done with a 40 000 posts extract from the initial 4 million posts database.

The final modelling and analysis with the 4 million database will be presented in the last part.

4.2.1 Basic graph for SNA

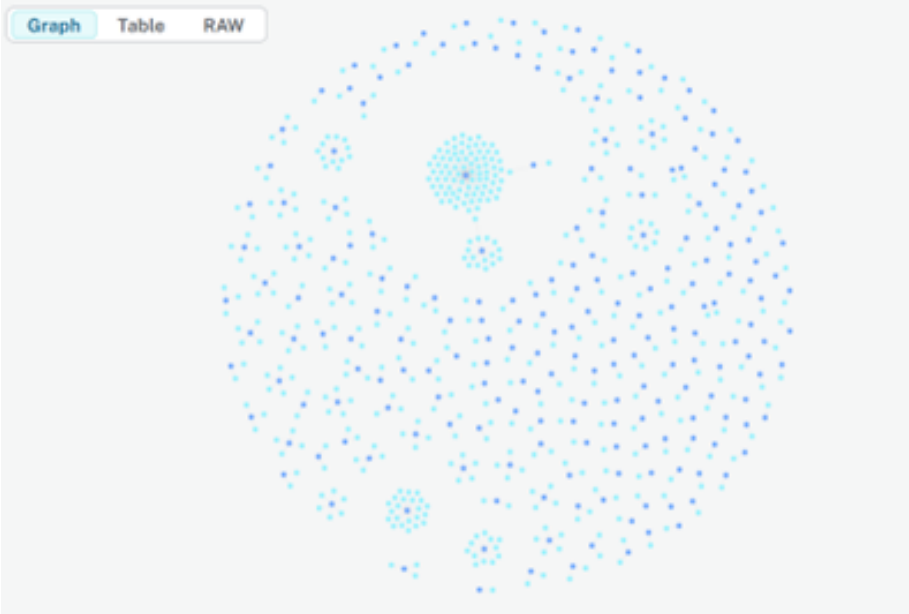


Figure 5: Visualization of user-subreddit relationships on smaller part of dataset

As a result, we observed that the subreddit with the biggest number of posts is AskReddit, which is very general name, it can include posts on completely different topics like traveling or animals. It could be more related to another post based on its main idea, and that prompted us to create our own topics, that would be more relevant to the idea of the posts.

Subreddit name	Number of posts (degree centrality)
AskReddit	9003
leagueoflegends	974
AdviceAnimals	717
funny	695
...	...
billards	4
itookapicture	4
Detroit	4

Table 1: Number of posts per Subreddit

Mean number of posts (degree centrality)
12.1887

Table 2: N

4.2.2 Model Subreddit→Topic

We decided to represent the data two different ways.

The first is created with simple and non-redundant relationships. A Subreddit is linked to a Topic with the relationship "TALKS_ABOUT". Every Subreddit is linked to a Topic at most one time.

```
LOAD CSV WITH HEADERS
FROM 'http://localhost:11001/project-ff609657-a9c9-4d79-888c-d886b7508728
    /final_topics.csv' AS row
MERGE (t:Topic {name: row.topic2})
MERGE (s:Subreddit {title: row.subreddit, id: row.subreddit_id})
MERGE (s)-[:TALKS_ABOUT]->(t);
```

This allows us to visualize the different topics linked to each Subreddit with the following query :

```
MATCH (s:Subreddit)-[r]-(t:Topic)
RETURN s.title AS subreddit, COUNT(DISTINCT(t)) AS topics_count
ORDER BY topics_count DESC
```

Subreddit	Topics Count
AskReddit	30
Minecraft	30
explainlikeimfive	30
politics	30
...	...
relationship_tips	2
askTO	2
PCOS	2
cuboulder	2
PrettyGirls	2

Table 3: Subreddits ordered by the number of topics they talk about

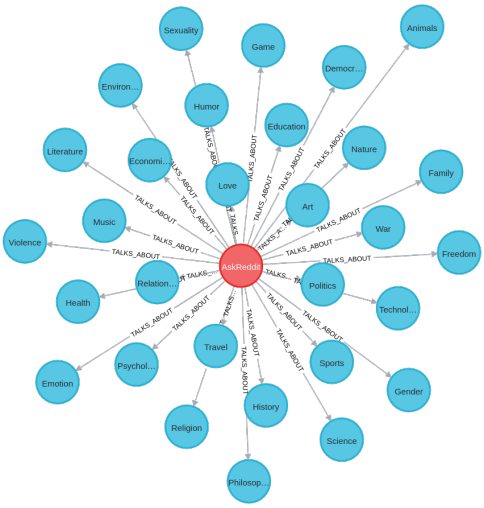


Figure 6: Topics linked to the Subreddit "AskReddit"

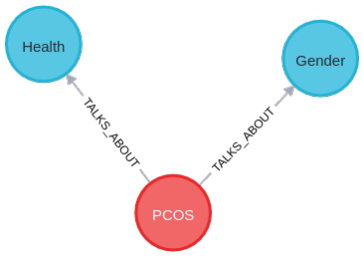


Figure 7: Topics linked to the Subreddit "PCOS"

4.2.3 Model User→Topic

```
LOAD CSV WITH HEADERS FROM 'http://localhost:11001/project-ff609657-a9c9-4d79-888c-
    d886b7508728/final_topics.csv' AS row
MERGE (t:Topic {topic_name: row.topic1})
MERGE (u:User{username:row.author})
MERGE (u)-[:POSTS_ABOUT{post_id: row.id}]->(t);

MATCH (t:Topic)-[r]-(u:User)
RETURN t.topic_name, COUNT(u) as user_nb ORDER BY user_nb DESC
```

Topic name	Number of posts (degree centrality)
Violence	5435
Game	4913
Relationship	4078
Health	4049
...	...
Freedom	1447
Science	1373
Love	1331

Table 4: Number of posts per topic

Some quick metrics that could be computed to compare with the model User→Subreddit.

- Mean: 2565.6 posts talk about the same topic on average
- Median: 2393 posts
- Standard Deviation: 1123.91 posts

We can We should also consider that these results are not really representative in terms of "posts repartition" as they are computed on 40 000 posts only. Final results should be more relevant.

4.2.4 Model Subreddit←User→Topic

```
LOAD CSV WITH HEADERS FROM 'http://localhost:11001/project-ff609657-a9c9-4d79-888c-
    d886b7508728/final_topics.csv' AS row
MERGE (t:Topic {topic_name: row.topic1})
MERGE (u:User{username:row.author})
MERGE (s:Subreddit {title: row.subreddit, id: row.subreddit_id})
MERGE (u)-[:POSTS_ABOUT{post_id: row.id}]->(t);
MERGE (u)-[:POSTS_ON{post_id: row.id}]->(s);

MATCH (t:Topic)-[r]-(u:User)
RETURN t.topic_name, COUNT(u) as user_nb ORDER BY user_nb DESC
```

Topic name	Number of posts (degree centrality)
Violence	5435
Game	4913
Relationship	4078
Health	4049
...	...
Freedom	1447
Science	1373
Love	1331

Table 5: Number of posts per topic

5 Analysis

5.1 Metrics computation

To compute the metrics with Neo4j, we decided to use a library called GDS.

For instance, with the model User→Topic :

Before running algorithms or metrics, the graph needs to be projected into memory. This step creates a virtual representation of the graph for fast in-memory processing, without modifying the underlying database.

Next step is identifying the nodes (e.g., User, Topic) and relationships (e.g., POSTS_ABOUT) in the dataset, and specify any properties required for the analysis. It is not possible to project properties that are strings, so that required we created (e.g., numeric properties like topic_name_numeric).

```
CALL gds.graph.project(
  'userTopicGraph_4',
  {
    User: {
      properties: []
    },
    Topic: {
      properties: {
        topic_name_numeric: {
          defaultValue: 0,
          type: 'INTEGER'
        },
        type_numeric: {
          defaultValue: 0,
          type: 'INTEGER'
        }
      }
    }
  },
  {
    POSTS_ABOUT: {
      type: 'POSTS_ABOUT',
      orientation: 'NATURAL'
    }
  }
)
```

Once the graph is projected, we can ensure it is successfully loaded into memory with

Degree Distribution	Graph Name	...	nodeCount	relationshipCount
min: 1, ..., mean: 2.199211383507629	userTopicGraph_1	...	34998	76968

Table 6: Visualization of the projected graph

We can now compute the metrics available with the GDS library.

5.1.1 Computation of the betweenness degree

First, let's compute the betwenness centrality for the topic nodes.

```
CALL gds.betweenness.stream('userTopicGraph')
YIELD nodeId, score
WHERE gds.util.nodeProperty('userTopicGraph', nodeId, 'type_numeric') = 1
RETURN nodeId, score
ORDER BY score DESC;
```

Then, retrieve the score and write them in the database.

```
CALL gds.betweenness.write('userTopicGraph', {
    writeProperty: 'betweennessScore'
})
YIELD nodePropertiesWritten;
```

Finally, visualize the scores along with the topic names.

```
MATCH (n:Topic)
WHERE (n.betweennessScore IS NOT NULL)
RETURN n.topic_name AS topicName, n.betwennessScore AS betweennessScore
ORDER BY betweennessScore DESC
```

Topic name	Betweenness Score
Game	95869589
Violence	86551124
Relationship	76388612
Health	75895692
...	...
Science	16531527
Philosophy	14121254
Love	13734436

Table 7: Topics ordered by their betweenness centrality score

The betweenness centrality is slightly different from the degree centrality but the overall classification remains the same.

5.1.2 Community detection with Louvain

```
CALL gds.louvain.stream('userTopicGraph')
YIELD nodeId, communityId
WITH communityId, count(nodeId) AS nodeCount
RETURN communityId, nodeCount
ORDER BY nodeCount DESC;
```

Louvain's algorithm retrieved 30 communities : the same number as the number of communities we preselected.

It's logical because the model we chose was already highly segmented into communities. The results should be more relevant with the models Subreddit←User→Topic and User→SubTopic←Topic.

Community id	Number of posts
3	2684
16	2409
9	2142
12	2137
...	...
25	530
8	451
22	440

Table 8: Louvain's communities ordered by the number of posts they gather

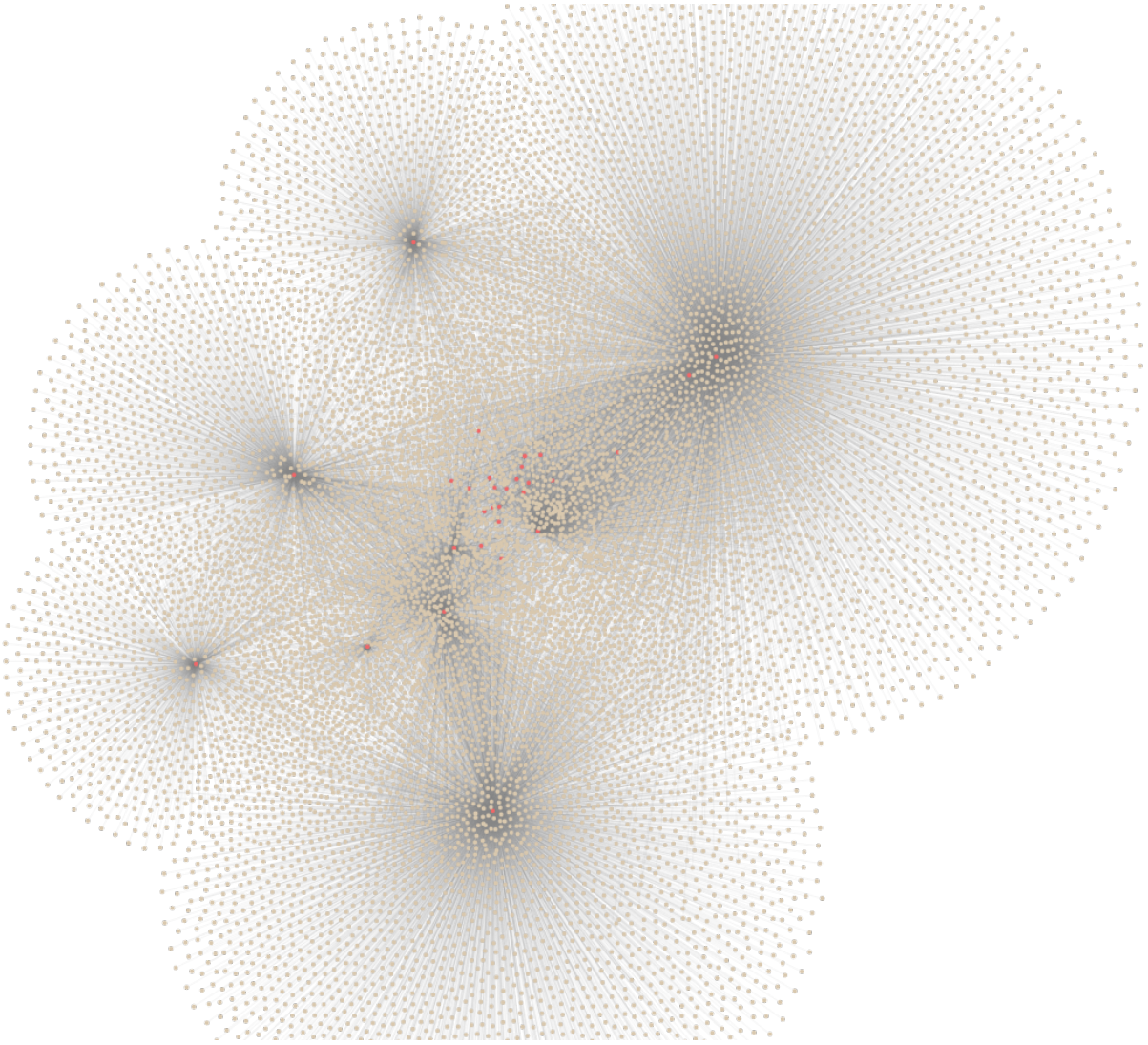


Figure 8: Partial visualization with Neo4j Bloom for the model User→Topic

5.2 Final results

Metrics	Basic	SR→T	U→T	SR←U→T	U→ST←T
Density					
Diameter					
Degree centrality					
Betweenness centrality					
Louvain's scores					
Graph visualization					

Table 9: Final results with the 4 million posts dataset

Due to the complexity of computation, we couldn't present you the final results in time. They will be presented during the presentation on tuesday.

6 Conclusion

We tried, with this project, to highlight the potential of knowledge graphs in the field of Social Network Analysis (SNA). By structuring Reddit data into various graph representations, we were able to explore the intricate relationships between the users and the content they post on Reddit. This process showcased the value of integrating semantic layers into network data to facilitate multidimensional analyses.

Knowledge graphs provide a flexible framework to represent complex, interconnected data, enabling insights into community dynamics and user interactions.

Metrics such as centrality, combined with clustering algorithms like Louvain, revealed patterns of influence, community structures, and thematic concentrations within the network.

Despite challenges in handling large datasets and ensuring consistency across models, tools like Neo4j and its Graph Data Science library proved to be efficient in processing and analyzing massive social graphs.

Future work could expand on the findings presented here, leveraging more sophisticated models and larger datasets (especially on time representation) to gain deeper insights into the evolution of online communities. The techniques and methodologies outlined in this study demonstrate the significant role of knowledge graphs in making social network data comprehensible and actionable, paving the way for innovative applications in social science.

References

[1] *Building Knowledge Graphs: A Practitioner's Guide*. en. URL: https://neo4j.com/knowledge-graphs-practitioners-guide/ (visited on 12/14/2024).

[2] Jens Dörpinghaus et al. "From social networks to knowledge graphs: A plea for interdisciplinary approaches". In: *Social Sciences & Humanities Open* 6.1 (Jan. 2022), p. 100337. ISSN: 2590-2911. DOI: 10.1016/j.ssaho.2022.100337. URL: https://www.sciencedirect.com/science/article/pii/S2590291122000912 (visited on 12/14/2024).

[3] *What Is a Knowledge Graph?* en-US. URL: https://www.ontotext.com/knowledgehub/fundamentals/what-is-a-knowledge-graph/ (visited on 12/14/2024).