European Research Studies Journal
Volume XXVII, Special Issue 2, 2024

pp. 94-102

Exploring and Analyzing YouTube Communities through Data Mining and Knowledge Graphs

Submitted 18/02/24, 1st revision 16/03/24, 2nd revision 20/04/24, accepted 16/05/24

Bartosz Przysucha[1], Magdalena Hałas[2], Cezary Figura[3], Natalia Rak[4], Paweł Barwiak[5], Adam Hernas[6]

Abstract:

Purpose: *The paper explores using knowledge graphs to analyze and model social interactions on the YouTube platform. The study uses advanced data structures to uncover more profound insights into community dynamics and user engagement in the digital space.*
Design/Methodology/Approach: *The study uses a mixed-methods approach, combining real-time data extraction from YouTube's live chat feature with knowledge graph construction to map complex relationships between users, content, and interactions. The data is managed using a Neo4j graph database and processed using Redis queuing mechanisms and Kubernetes for distributed computing, providing scalability and flexibility in data handling.*
Findings: *The study shows that knowledge graphs provide a solid framework for capturing and analyzing the complex network of social interactions on YouTube. By integrating natural language processing (NLP) techniques, the designed framework effectively processes and interprets queries and shows user interactions.*
Practical Implications: *The study's results offer significant implications for developing more sophisticated recommendation systems and analytics tools that dynamically adapt to new data and user behavior. Implementing knowledge graphs can help platform designers and marketers better understand user engagement and content popularity, leading to more targeted and effective strategies.*
Originality/Value: *The article contributes to the field of digital analytics by presenting a new application of knowledge graphs in social media analysis. Emphasizes the enhanced capabilities of graph-based data structures in combination with real-time data processing and NLP.*

Keywords: *Knowledge graphs, YouTube, social media analysis, natural language processing (NLP), Neo4j, social interaction.*

JEL codes: *C45, C61, M31, L8, D83.*

Paper type: *Research article.*

[1]*Corresponding Author: Management Faculty, Lublin University of Technology, Lublin, Poland, e-mail: b.przysucha@pollub.pl;*
[2]*WSEI University, Lublin, Poland, e-mail: Magdalena.Halas@wsei.lublin.pl;*
[3]*WSEI University, Lublin, Poland, e-mail: Cezary.Figura@wsei.lublin.pl;*
[4]*WSEI University, Lublin, Poland, e-mail: Natalia.Rak@wsei.lublin.pl;*
[5]*WSEI University, Lublin, Poland, e-mail: Paweł.Barwiak@wsei.lublin.pl;*
[6]*Wyższa Szkoła Biznesu - National Louis University, e-mail: ahernas@wsb-nlu.edu.pl;*

1. Introduction

In the era of the evolving digital space, in which we are experiencing information and data growth at a highly rapid pace, the use of data and information to understand and model social interactions on social platforms is not only valuable but essential (Grover and Kar, 2017).

YouTube, being one of the largest aggregators of video content, provides a rich source of data that can be analyzed to obtain modeling and analysis of community dynamics. Such studies focus on surveying users (Rotman *et al.,* 2009; Rotman and Preece, 2010) or using metadata (Spathis and Gorcitz, 2011; Che *et al.,* 2015).

The flow of YouTube data, with a particular focus on comments related to live streaming, creates unique opportunities for real-time community research. This data can be transformed into knowledge graphs that effectively represent the complex relationships between users and enable graph analysis techniques to identify critical nodes, such as influencers or dominant topics (Shanmukhaa *et al.,* 2020).

Knowledge graphs, a key component of modern information processing systems, provide an effective tool for modeling complex relationships between diverse entities (Gupta *et al.,* 2022). These data structures differ from traditional relational databases in treating the relationships between entities as equally crucial as the entities themselves (Gutierrez and Sequeda, 2021).

In the context of YouTube, knowledge graphs enable the representation of relationships between users, their comments, and video content and support advanced semantic search. Semantic search, using natural language processing (NLP) technology, allows queries to be interpreted in a way that more closely resembles a human's natural understanding of language, enabling more relevant and contextual responses (Heist *et al.,* 2020).

For example, in a knowledge graph-based system, a query about “the most popular AI videos in the last month” can be efficiently processed by analyzing nodes representing videos, related comments, and ratings. This makes it possible to identify the most viewed content and the content that generates the most social interaction.

Graph databases such as Neo4j used in this work offer APIs built around graphs that enable efficient management and analysis of large data sets (Allen *et al.,* 2019; Jordan, 2014; Zampeta and Chondrokoukis, 2023). Knowledge graphs and NLP technologies lay the foundation for next-generation recommendation and analytics systems that passively present data and actively add value through deeper understanding and interpretation of user needs (Hoobyar *et al.,* 2013).

The research presented in the article focuses on creating dynamic knowledge graphs that can adapt and evolve in response to the constant influx of new data. The information processing system uses the Pytchat library to capture live chat data and the Neo4j platform to store and analyze knowledge graphs (Polyakova *et al.*, 2019).

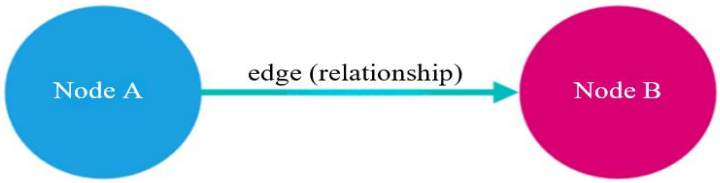
The resulting solution is scalable and designed for high availability and flexibility by using queuing mechanisms in Redis (Tkachenko and Lukianiuk, 2021) and distributed processing in the Kubernetes environment (Thurgood and Lennon, 2019).

This paper uses advanced data mining techniques and knowledge graph methodology to analyze and model user interactions on YouTube. The paper aims to present the method of knowledge graph construction and demonstrate its practical application in community analysis. By studying the structure and dynamics of graphs, one can better understand how information is exchanged in digital communities and the key factors that influence the formation of opinions and trends.

2. Creating Knowledge Graphs to Model Social Interactions on the YouTube Platform

The research focused on applying knowledge graphs to analyze and manage data extracted from the YouTube platform. Knowledge graphs are advanced data structures representing information collection in a graph, where nodes symbolize entities and edges reflect their relationships. Above all, a knowledge graph prioritizes the connection between data points as much as the data point itself.

Figure 1. *General form of a knowledge graph*



Source: *Own creation.*

Node A and Node B are two separate entities. These nodes are connected by an edge representing their relationship. This is the smallest knowledge graph, also known as a triplet, that can be created. In the context of knowledge graphs, an entity can represent various objects, such as people, places, or concepts relevant to a domain.

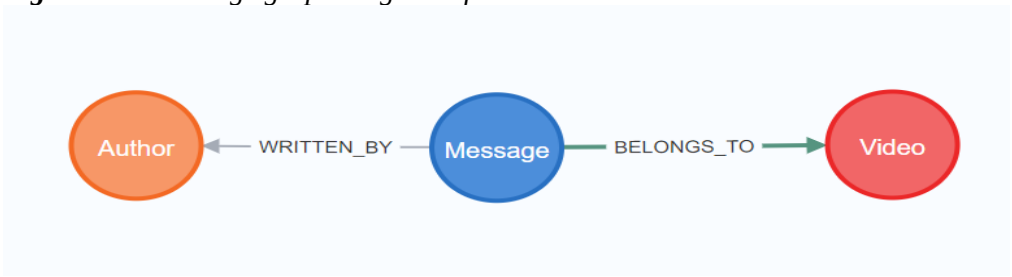
The relationships between these entities describe their interrelationships. With this structure, knowledge graphs offer a natural and intuitive way to represent knowledge that is understandable to humans and adaptable to machine processing. Also crucial to the whole process is the use of natural language processing (NLP) technologies,

which enable the analysis and extraction of textual data, transforming it into graph structures. Thanks to NLP, it is possible to understand the context and semantics of the text, which significantly contributes to the accurate modeling of relationships and entities.

Semantic search in knowledge graphs allows for much more advanced data search than traditional keyword-based methods. It takes advantage of context and understanding of query meaning, making it highly effective in extracting relevant information from rich databases such as those collected in knowledge graphs.

Figure 2 shows a knowledge graph diagram for comments on YouTube videos.

Figure 2: *Knowledge graph diagram of YouTube video comments*



Source: *Own creation.*

The node shown can have multiple relationships. The designed knowledge graph contains three different nodes and two types of relationships, each containing comments related to live chat in the context of a specific video. The base diagram shown in Figure 2 distinguishes two trinities. The first is Message->Belongs_to->Video, while the second is Message->Written_by->Author.

2.1 Filling in the Knowledge Graph from YouTube Data

As part of the project, data is extracted from data streams, which are the communication channels of YouTube users. The system focuses on capturing interesting videos, primarily those livestreamed. A key element is the comments collected from the so-called TopChat in real-time. These comments are treated as events in the system.

To ensure adequate flexibility and throughput, these events are queued with the Redis system using a first-in, first-out (FIFO) queue. The processed message contents are then written to the Neo4j database.

The solution has been implemented on the Kubernetes platform, which enables horizontal scaling of message processing services using the Jobs mechanism. This allows efficient resource management and processing in response to changing system loads.

The service for processing YouTube comments is based on the pitch (Python) library, which allows for downloading live chat from YouTube, working without needing tools such as Selenium or BeautifulSoup. The library offers features to configure chat data processors compatible with the YouTube API and work in an asyncio context. This makes it possible to quickly retrieve the initial chat data by generating the appropriate parameters, a key element of efficient data collection for the knowledge graph.

The following is the Python code for the above task. This code integrates several technologies: real-time data retrieval, asynchronous processing, queue handling, and working with graph databases. It illustrates a modern approach to processing and analyzing large data sets in real-time.

```
import time import rediswq

from py2neo import Graph, Node, Subgraph, Relationship import pytchat

host="redis" q = rediswq.RedisWQ(name="job2", host=host)

graph = Graph("neo4j://91.232.58.144:7687", auth=("neo4j", "N…"))
writen_by = Relationship.type("WRITTEN_BY")
belongs_to = Relationship.type("BELONGS_TO")

print("Worker with sessionID: " +  q.sessionID())
print("Initial queue state: empty=" + str(q.empty()))
while True:    item = q.lease(lease_secs=100, block=True, timeout=2)
  if item is not None:
    itemstr = item.decode("utf-8")

    video_id = itemstr     chat = pytchat.create(video_id=video_id)

    v = Node("Video", video_id=video_id)
    v.__primarylabel__ = "Video"     v.__primarykey__ = "video_id"     graph.merge(v)

    print("Working on " + itemstr)
    while chat.is_alive():
      for c in chat.get().sync_items():
        print(f"{c.datetime} [{c.author.name} {c.author.isVerified} {c.author.channelId}] -
{c.messageEx} - {c.type}")
        a = Node("Author", name=c.author.name, author_id=c.author.channelId,
verified=c.author.isVerified)
        a.__primarylabel__ = "Author"         a.__primarykey__ = "author_id"         m =
Node("Message", message_id=c.id, message_date=c.datetime, message_text=c.message)
        m.__primarylabel__ = "Message"         m.__primarykey__ = "message_id"
graph.merge(writen_by(m, a))
        graph.merge(belongs_to(m, v))
    q.complete(item)
  else:
    print("waiting for a task")
```

The functionality of writing data to the Neo4j database is realized by using the py2neo library. Py2neo is a robust client library designed to interact with the Neo4j database from Python and command-line applications. It supports the Bolt and HTTP protocols, offering a high-level API, object graph mapping, administrative tools, an interactive console, and the Cypher lexer for Pygments.

Another essential tool used in the system is Redis, which acts as a message broker, a system used to transfer and queue messages. Redis comes with data structures such as Redis Lists and Redis Sorted Sets, which are the foundation for implementing queuing systems. As a result, Redis can be used directly to create dedicated solutions or through frameworks, allowing for more natural and efficient message processing in the context of the programming language used.

2.2 Message Queuing

Message queues are based on dynamic lists and are often used by tools to implement standard design patterns. The main difference between message queues and event streams is how they communicate: message queues work on a push basis, where a service actively sends a new message to the receiving queue of another service when a new event requires a response.

In contrast, event streams operate on a continuous push basis, where the consumer retrieves the needed information himself. Messages in message queues may contain status information, such as the number of retries, and are removed from the system after they have been successfully processed. In contrast, stream events are immutable, and their history, even after processing, is often stored.

Redis lists and sorted Redis sets are two data types used to implement this behavior. Both data structures can be used to create individual solutions and as the basis for more complex ecosystem-specific structures.

2.3 Implementation of Message Processing Services

The management of message processing services is implemented using the Jobs mechanism in the Kubernetes platform. The Jobs mechanism can be described as a supervisor of pods starting with a specific purpose, such as performing calculations or processing data from a message queue. Jobs are responsible for creating the appropriate number of containers, supervising their operation, and shutting them down when the task is completed.

In the solution under discussion, services are initially created, usually two in number, and wait for messages to be queued. Once the processing is complete, these containers are removed, and, as needed, when there is new data to process, they are re-created and restarted.

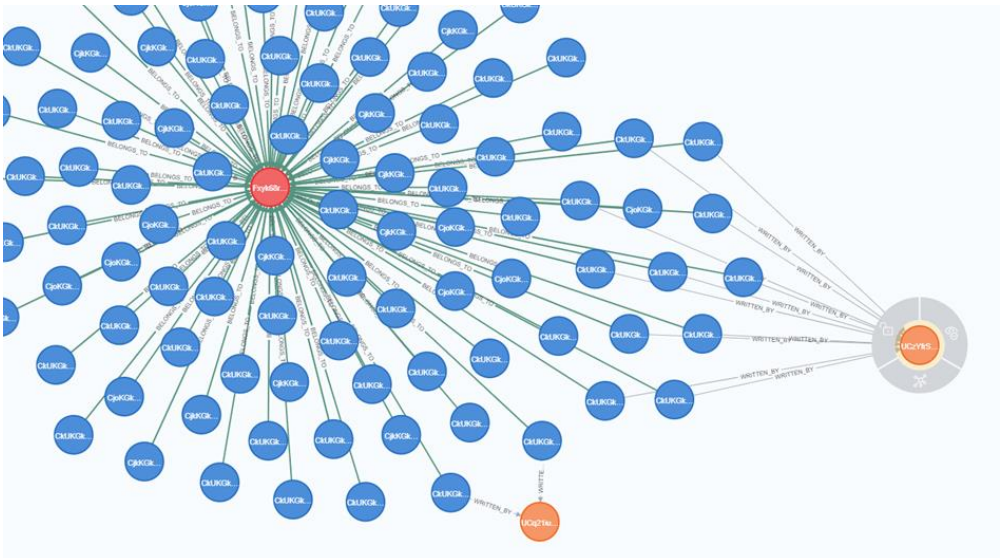
The following is code for creating and managing processes in a Kubernetes cluster, specifically for running a task to be executed by a certain number of containers until they are finished.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: job-wq-2
  namespace: new-order
spec:
  parallelism: 2
  template:
    metadata:
      name: job-wq-2
    spec:
      containers:
      - name: c
        image: docker.io/…
      restartPolicy: OnFailure
```

2.4 Creating a knowledge graph database

All data extracted from YouTube is stored in the Neo4j graph database. Figure 3 illustrates two triples in a knowledge graph. The graph filled with YouTube data is shown below, demonstrating the practical application of knowledge graphs in analyzing and managing data from this platform. Each element of the graph, or node and edge, reflects specific information extracted from video content, comments, or user interactions, allowing for deeper analysis and a better understanding of the data context.

Figure 3. *Graph database populated with YouTube data*



Source: *Own creation.*

3. Conclusions

The research used knowledge graphs to analyze social dynamics on the YouTube platform, a new contribution to understanding and modeling social interactions in digital spaces. Knowledge graphs allow for a deeper understanding of the complex relationships between users, their content, and their comments, which is essential for developing more effective recommendation systems and analytical tools.

Data management using the Neo4j database enabled efficient and scalable processing of large data sets while queuing mechanisms in Redis and distributed processing in Kubernetes provided the necessary operational flexibility. These technologies played a crucial role in ensuring the system could dynamically adapt and evolve in response to the constant influx of new data.

Implementing knowledge graphs for content analysis on YouTube demonstrated the possibility of understanding commented content's social and topic structure, directly impacting shaping opinions and trends.

In conclusion, applying knowledge graphs to the analysis of YouTube interactions represents an essential step toward better understanding and utilizing the wealth of data generated in digital ecosystems. This research opens up new perspectives for the further development of data analysis technologies and artificial intelligence systems that cannot only passively observe but also actively participate in shaping the digital social space.

References:

Allen, D., Hodler, A., Hunger, M., Knobloch, M., Lyon, W., Needham, M., Voigt, H. 2019. Understanding Trolls with Efficient Analytics of Large Graphs in Neo4j. Datenbanksysteme für Business, Technologie und Web.

Che, X., Ip, B., Lin, L. 2015. A Survey of Current YouTube Video Characteristics, in IEEE MultiMedia, vol. 22, 2, 56-63. doi: 10.1109/MMUL.2015.34.

Grover, P., Kar, A.K. 2017. Big Data Analytics: A Review on Theoretical Contributions and Tools Used in Literature. Glob J Flex Syst Manag 18, 203-229. https://doi.org/10.1007/s40171-017-0159-3.

Gupta, R., Pandey, I., Mishra, K. Seeja, K. 2022. Applied Information Processing Systems. New York, NY, USA, Springer.

Gutierrez, C., Sequeda, J.F. 2021. Knowledge graphs. Commun. ACM 64, 3, 96-104. https://doi.org/10.1145/3418294.

Heist, N., Hertling, S., Ringler, D., Paulheim, H. 2020. Knowledge Graphs on the Web - an Overview. ArXiv, abs/2003.00719.

Hoobyar, T., Dotz, T., Sanders, S. 2013. NLP: the essential guide to neuro-linguistic programming. Harper Collins.

Jordan, G. 2014. Querying. In Practical Neo4j (pp. 39-48). Apress. doi:10.1007/978-1-4842-0022-3.

Polyakova, A.G., Loginov, M.P., Serebrennikova, A.I., Thalassinos, E.I. 2019. Design of a socio-economic processes monitoring system based on network analysis and big data. International Journal of Economics and Business Administration, 7(1), 30-139.

Rotman, D., Golbeck, J., Preece, J. 2009. The community is where the rapport is -- on sense and structure in the YouTube community. In: Proceedings of the fourth international conference on Communities and technologies (C&T '09). Association for Computing Machinery, New York, NY, USA, 41-50. https://doi.org/10.1145/1556460.1556467.

Rotman, D., Preece, J. 2010. The 'WeTube' in YouTube – creating an online community through video sharing. Int. J. Web Based Communities 6, 3, 317-333. https://doi.org/10.1504/IJWBC.2010.033755.

Shanmukhaa, G.S., Nandita, S.K., Kiran, M.V.K. 2020. Retracted: Construction of Knowledge Graphs for video lectures, 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 127-131. doi: 10.1109/ICACCS48705.2020.9074320.

Spathis, P., Gorcitz, R.A. 2011. A data-driven analysis of YouTube community features. In: Proceedings of the 7th Asian Internet Engineering Conference (AINTEC '11). Association for Computing Machinery, New York, NY, USA, 12-18. https://doi.org/10.1145/2089016.2089019.

Tkachenko, V. Lukianiuk S. 2021. Analysis of the use of the Redis in the distributed order processing system in the restaurant network. Technology audit and production reserves. 5. 39-43. 10.15587/2706-5448.2021.238460.

Thurgood B., Lennon, R.G. 2019. Cloud Computing With Kubernetes Cluster Elastic Scaling. In: Proceedings of the 3rd International Conference on Future Networks and Distributed Systems (ICFNDS '19). Association for Computing Machinery, New York, NY, USA, Article 5, 1-7. https://doi.org/10.1145/3341325.3341995.

Zampeta, V., Chondrokoukis, G. 2023. A Comprehensive Approach through Robust Regression and Gaussian/Mixed-Markov Graphical Models on the Example of Maritime Transportation Accidents: Evidence from a Listed-in-NYSE Shipping Company. Journal of Risk and Financial Management, 16(3), 183.