# 1. Introduction

With the exponential growth of academic publications, researchers struggle to keep up with emerging trends, influential papers, and topic evolution. Traditional citation-based metrics fail to capture the deeper relationships between concepts, methodologies, and research communities. To address this, we propose a **Knowledge Graph-based system** that models research data in a way that enables richer exploration and analysis.

This project operates within the **domain of scientific research analytics**, using data from **OpenAlex**, a comprehensive, openly available metadata database of research publications. We construct a knowledge graph that includes entities such as **papers, authors, topics, and citations**, capturing both the semantic content and structural relationships of academic output. The goal is to transform this symbolic information into a navigable and intelligent data structure suitable for advanced querying, analysis, and learning.

The result is a usable **analytics service** built on top of a KG:

- Users can analyze **emerging research areas** by clustering similar papers using embeddings and publication years.
- They can **forecast citation impact** using our trained GNN.
- The system supports **complex queries** via Cypher and could be extended into a full recommendation engine to explore emerging or declining research topics, identify collaboration networks, and track citation trajectories. (e.g., *“Which authors have published in fast-growing topics over the past five years?”* or *“What papers are similar to a given publication?“*).

These services show how **KGs enable real-world applications (LO9)** and support **data-driven services (LO11)** such as research analytics, predictive modeling, and semantic search interfaces.

# 2. KG Construction

## 2.1. Data Collection and Preprocessing

Our Knowledge Graph is built using the **OpenAlex** dataset, a large, openly available collection of scholarly metadata encompassing academic papers, authors, venues, topics, institutions, and citation relationships. OpenAlex provides JSON-formatted data dumps, which we parsed and integrated into our graph. Our goal was to create a graph focused on recent research in Artificial Intelligence and Computer Vision. We filtered the dataset to include only works from two relevant subfields:

- **Artificial Intelligence** (OpenAlex subfield ID: `1702`)

- **Computer Vision and Pattern Recognition** (OpenAlex subfield ID: `1707`)

In addition, to ensure the quality and relevance of our data, we restricted the source of publications to a selected set of venues that are well-regarded in these fields:

- **IEEE Access** (`s2485537415`)
- **CVPR Conferences** (`s4306512817,s4210176548,s4363607701`)
- **CVPR Workshops** (`s4363607748`)

This filtering was implemented via the OpenAlex API using the following query:

```
OPENALEX_URL = "https://api.openalex.org/works"
FILTER_QUERY = (
    "primary_location.source.id:s2485537415|s4306512817|s4210176548|"
    "s4363607748|s4363607701,"
    "primary_topic.subfield.id:subfields/1702|subfields/1707"
)
```

The resulting dataset was saved in a structured JSON format for further processing.

## 2.2. Technologies and Architecture

Our architecture is designed **(LO5)** to enable flexible querying, learning, and reasoning over the knowledge graph, while also supporting downstream machine learning tasks.

- **Preprocessing and Ingestion**: We use **Python (Pandas, JSON)** for parsing and cleaning the raw data, and the **Neo4j Python driver** for bulk insertion.
- **Storage and Query**: While alternative graph databases such as Amazon Neptune, Azure Cosmos DB, or TigerGraph offer similar functionality, **Neo4j** was chosen as the database system for this project due to its native support for graph data structures, intuitive Cypher query language, and efficient querying capabilities. Most importantly, its integration with the **Graph Data Science (GDS) library**—which provides a rich suite of graph embedding and machine learning algorithms within the same runtime—eliminates the need for costly data migrations between systems or formats. This makes Neo4j especially well-suited for both the construction and analysis of Knowledge Graphs.
- **Graph Machine Learning**: We employ **PyTorch Geometric** and **DGL** for training Graph Neural Networks on sampled subgraphs extracted from Neo4j.

### 2.3. Knowledge Graph Construction

The KG was built **(LO7)** by parsing the cleaned OpenAlex data and transforming it into nodes and relationships suitable for Neo4j. We defined the following main node types:

- `Paper` (with properties: title, abstract, year, citation count)
- `Author` (with properties: name, ORCID, affiliation)
- `Topic` (OpenAlex subfield names)

And the following relationship types:

- `WROTE` (Author → Paper)
- `HAS_TOPIC` (Paper → Topic)
- `RELATED` (Paper → Paper): the papers do not cite each other but are content-related. This relation is already given by OpenAlex.
- `CITES` (Paper → Paper)

This structured schema allowed us to effectively perform both semantic and analytical queries over the graph, facilitating services such as trend discovery and citation prediction.

During graph construction, we iterate through each record in the dataset and create or update nodes for **Authors**, **Papers**, and **Topics**. To maintain consistency and avoid duplicates, we use Cypher's `MERGE` command, which ensures that each entity is uniquely represented. This approach is critical for preserving the integrity of the Knowledge Graph.

## 3. ML-based Representation

### 3.1. Paper Similarity and Clustering

We applied the pre-trained `all-MiniLM-L6-v2` model[1] from **Sentence Transformers** to encode the abstracts into dense vector embeddings. To discover semantically similar papers, we used **K-Nearest Neighbors (KNN)** with `k=10` on the abstract embeddings. For each paper, we identified the 10 most similar papers in the embedding space and added a `SIMILAR_TO` edge to the knowledge graph, enriching the graph with **implicit semantic links** not originally present **(LO8)**.

To further explore the structure of the research domain, we applied **KMeans clustering** to the abstract embeddings. The goal was to identify whether papers naturally group into clusters that correspond to **meaningful subtopics** within the AI and Computer Vision domains. After clustering (e.g., with `k=10`), we analyzed the distribution of known paper subfields within each cluster. In many cases, the clusters showed coherent topical groupings. Below, we highlight the interpretation of two representative clusters:

1 https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

**Cluster 0.** This cluster primarily groups topics related to cryptography, data protection, and information security, including:

- Chaos-based Image/Signal Encryption (760 papers)
- Cryptography and Data Security (516)
- Steganography and Watermarking Techniques (436)
- Privacy-Preserving Technologies (344)
- Advanced Malware Detection (146)

**Cluster 1.** This cluster is dominated by NLP and text analysis topics, including:

- Topic Modeling (835 papers)
- Natural Language Processing Techniques (624)
- Sentiment Analysis and Opinion Mining (339)
- Text and Document Classification (232)
- Semantic Web and Ontologies (143)

While clustering did not directly modify the graph, it provided insights for potential ontology refinement or graph restructuring **(LO8)**. For instance, if many papers labeled under a single subfield split cleanly into multiple clusters, this may indicate that the current taxonomy is too coarse.

## 3.2. Graph Neural Networks

To demonstrate the predictive power of graph-based models, we trained a **Graph Convolutional Network (GCN)** to estimate the **number of citations (regression model)** a paper would receive using Pytorch Geometric **(LO3)**. The GCN is well-suited for this task because it can capture relational information between papers—how they cite each other—alongside their intrinsic features.

For this task, we first queried Neo4j to extract papers (nodes) that had abstract embeddings (previously computed using a SentenceTransformer), a non-zero citation count, and were published between 2008 and 2022. This range was chosen because older records often had missing values, while data from 2023 onward was incomplete or not yet fully updated. Additionally, we also extracted undirected citation-based connections (edges), including `CITES`, `RELATED`, and `SIMILAR_TO` relationships.

Each node was represented by a feature vector consisting of the **abstract embedding concatenated with the publication year**. Features were standardized to have zero mean and unit variance. To address the skewed distribution of citation counts, we applied a **log1p transformation** to the target variable.

We split the data into an 80% training and 20% test set, using boolean masks to identify training and test nodes within the graph, the Mean Squared Error as the loss function and Adam optimizer with learning rate 0.008, chosen empirically to avoid underfitting and overfitting.

The model was evaluated using Root Mean Squared Error (RMSE), achieving an RMSE of 1.62, indicating reasonable performance given the variance in citation counts.

Example predictions demonstrate that the model performs well overall: for instance, it predicted 4.57 citations for a 2016 biomedical classification paper that received 5 citations and 43.43 for a 2018 watermarking paper that received 60. However, some outliers remain, such as underestimating a 2009 theoretical hash function paper with only 1 citation (predicted: 4.62) or overestimating a 2018 cloud security paper with 96 citations (predicted: 51.89).

# 4. Logic-based Representation

In this section, we demonstrate our understanding and practical application of **logical knowledge** in Knowledge Graphs (KGs), aligned with **(LO2)**. We focus on expressing logical rules, recursive exploration, and object creation adopting **Cypher**, the declarative query language for Neo4j, to express logical knowledge.

### Example 1: Most Popular Topics

```
MATCH (:Paper)-[:HAS_TOPIC]->(t:Topic)
RETURN t.name AS topic, count(*) AS paper_count
ORDER BY paper_count DESC
LIMIT 10
```

**Purpose**: Identifies the most frequently discussed topics across all papers.

**Logic**: Aggregation logic is applied here to count the number of papers per topic. This is a simple deductive rule based on existing facts in the graph.

---

### Example 2: Recursive Exploration — Top Authors in Similar Papers

```
MATCH (a1:Author)-[:WROTE]->(p1:Paper)-[:SIMILAR_TO]->(p2:Paper)<-[:WROTE]-
(a2:Author)
WHERE a1 <> a2
WITH a1, a2, count(*) AS similarity_score
ORDER BY similarity_score DESC
RETURN a1.name AS author1, a2.name AS author2, similarity_score
LIMIT 20
```

**Purpose**: Recursively traverses the `SIMILAR_TO` edges between papers to find co-involved authors across similar works.

**Logic**: This rule performs a recursive-like join on the graph, allowing us to infer implicit author collaboration through similarity of papers.

---

### Example 3: Inference with Object Creation — Linking Similar Authors

```
MATCH (a1:Author)-[:WROTE]->(p1:Paper)-[:SIMILAR_TO]->(p2:Paper)<-[:WROTE]-
(a2:Author)
WHERE a1 <> a2
MERGE (a1)-[r:SIMILAR_AUTHOR]->(a2)
ON CREATE SET r.similarity_count = 1
ON MATCH SET r.similarity_count = r.similarity_count + 1
```

**Purpose**: This rule materializes a new relation in the KG :SIMILAR_AUTHOR, based on a pattern observed in existing nodes and edges.

**Logic**: This demonstrates **existential quantification** by creating new knowledge — a new edge — from inferred similarity. The MERGE clause ensures the edge is created only once, and updated if it already exists.

---

### Example 4: Most Influential Authors by Topic

```
MATCH (a:Author)-[:WROTE]->(p:Paper)-[:HAS_TOPIC]->(t:Topic)
WHERE p.citations IS NOT NULL
RETURN a.id AS author, t.name AS topic, sum(p.citations) AS total_citations
ORDER BY total_citations DESC
LIMIT 50
```

**Purpose**: Ranks authors by impact within specific topics using citation counts.

**Logic**: Encodes a logical rule aggregating citation evidence over a chain of relationships, linking authors to the impact of their works by topic.

---

### Example 5: Emerging Topics Over Time

```
MATCH (p:Paper)-[:HAS_TOPIC]->(t:Topic)
WHERE p.year IS NOT NULL
RETURN t.name AS topic, p.year AS year, count(*) AS papers_published
ORDER BY year ASC, papers_published DESC
```

**Purpose**: Identifies topics that are gaining popularity over time.

**Logic**: This rule applies temporal reasoning, revealing trends by evaluating relationships conditioned on the year property. While not recursive, it captures evolution over the KG timeline.

# 5. Conclusions and Future Work

## 5.1. Conclusions

This project demonstrates how knowledge graphs can be combined with machine learning to extract insights from academic research data. Through Neo4j, OpenAlex, and PyTorch Geometric, uur system demonstrated real-world potential for supporting researchers, institutions, and funding bodies in navigating the growing body of scientific literature. From forecasting citation impact to identifying fast-growing fields, the framework can support evidence-based decision-making and discovery **(LO11)**.

Ultimately, the project illustrates the promise of KGs as a unifying platform for organizing, reasoning over, and learning from scientific knowledge — bridging the gap between human-understandable structure and machine-driven inference.

## 5.2. Future Work

In future work, we propose shifting from relying solely on abstract text embeddings to utilizing node similarity derived from knowledge graph embeddings **(LO1)**. This approach incorporates structural information such as shared topics, co-authors, and citation patterns, offering a more comprehensive understanding of paper similarity. By combining text embeddings with graph-based embeddings, we can enhance the accuracy and relevance of paper recommendations, capturing not only the semantic meaning of the text but also the contextual relationships within the graph.

In this work, we used a Graph Convolutional Network (GCN), but we could enhance it by incorporating temporal information, as each paper has a publication date. In future work, we could apply a Temporal Graph Neural Network (TGNN) to model the evolution of the graph over time. By considering the temporal aspect, such as research trends and citation patterns, we could improve paper recommendations by capturing how relationships between papers change over time **(LO4)**.

As future work, we plan to strengthen the integration between logic-based reasoning and machine learning within the Knowledge Graph **(LO12)**. One direction is to use logical rules as constraints or guidance for ML models — for example, helping embedding models respect known hierarchies or relationships. We also see potential in learning some logical patterns automatically, such as discovering similarity rules between authors based on co-authorship and citation data. Additionally, ML outputs could be validated or refined using logical consistency checks. By tightening the interaction between symbolic and statistical methods, we aim to improve both the accuracy and interpretability of the system.