KNOWLEDGE GRAPH BASED MULTIMODAL E-LEARNING RECOMMENDER SYSTEM, ONLINE COURSES DATA ANALYSIS

Minor Project I

Submitted By:

Aarush Gupta (22103030)

Saksham Purohit (22103017)

Shivaprasad Arunkumar Farale (22103012)

Under the supervision of – **Dr. Ankita Verma**

विद्या तत्व ज्योतिसमः

November – 2024

Submitted in partial fulfilment of the Degree of

Bachelor of Technology

Computer Science Engineering

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING & INFORMATION TECHNOLOGY

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA

TABLE OF CONTENT

DECLARATION

We hereby declare that this submission is our own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place:	Jaypee Institute of Information Technology, Noida
Date:	20 November 2024

Enrolment Number	**Name of Student**	**Signature**
22103030	Aarush Gupta	
22103017	Saksham Purohit	
22103012	Shivaprasad Arunkumar Farale	

STUDENTS' SELF DECLARATION FOR OPEN-SOURCE LIBRARIES AND OTHER SOURCE CODE USAGE IN MINOR PROJECT

We, **Aarush Gupta, Saksham Purohit, Shivaprasad Arunkumar Farale,** hereby declare the following usage of the open-source code and prebuilt libraries in our minor project in **5th** Semester with the consent of our supervisor. We also measure the similarity percentage of pre written source code and our source code and the same is mentioned below. This measurement is true with best of our knowledge and abilities.

1. List of pre build libraries (On next page)
2. List of pre build features in libraries or in source code. (On next page)
3. Percentage of pre written source code and source written by us.

Enrolment Number	**Name of Student**	**Signature**
22103030	Aarush Gupta	
22103017	Saksham Purohit	
22103012	Shivaprasad Arunkumar Farale	

Declaration by Supervisor (To be filled by Supervisor only)

I, **Dr. Ankita Verma** (Name of Supervisor) declares that I above submitted project with Titled "**Knowledge graph based multimodal e-learning recommender system, online courses data analysis"** was conducted in my supervision. The project is original and neither the project was copied from External sources not it was submitted earlier in JIIT. I authenticate this project.

(Any Remarks by Supervisor)

Signature (Supervisor)

1. List of Pre-Built Libraries

For the development of the **Knowledge Graph-Based Multimodal E-learning Recommender System**, the following pre-built libraries and tools were used to facilitate various functionalities:

- **Neo4j Desktop and Neo4j aura**: To connect with and query the Neo4j database using Cypher.
- **Scikit-learn**: For implementing collaborative filtering and content-based recommendation algorithms.
- **Pandas**: For data cleaning, preprocessing, and analysis.
- **Matplotlib/Seaborn**: For visualizing course-related trends.
- **NetworkX**: Used for creating and maintain Bipartite Graphs
- **Nltk**: Used for Text Processing
- **Stellargraph**: Used for Node embedding of graphs

2. List of Pre-Built Features in Libraries or Source Code

The following features were utilized directly from the libraries or tools:

- **Graph Algorithms in Neo4j**: Pre-built graph algorithms like **Degree Centrality**, **Betweenness Centrality** and others were used for analysing course relationships and user recommendations.
- **Matrix Factorization in Scikit-learn**: For implementing collaborative filtering without building an algorithm from scratch.
- **Cypher Query Functions in Neo4j**: Used for data analysis and retrieving course-related insights.
- **DataFrame Operations in Pandas**: Simplified data manipulation and preparation for various analyses.
- **Plotting Functions in Matplotlib/Seaborn**: Facilitated visualization of data trends like course popularity and durations.

CERTIFICATE

This is to certify that the work titled "**Knowledge Graph Based Multimodal E-Learning Recommender System, Online Courses Data Analysis**" submitted by "**Aarush Gupta, Saksham Purohit, Shivaprasad Arunkumar Farale**" in partial fulfilment for the award of degree of "**B.Tech Computer Science**" of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor: …………………………
Name of Supervisor: …………………………
Designation: …………………………
Date: …………………………

ACKNOWLEDGEMENT

We express my sincere gratitude to **Dr. Ankita Verma**, Assistant Professor (Senior Grade), Dept. of CSE/IT, India, for her stimulating guidance, continuous encouragement and supervision throughout the course of present work.

We would like to place on record our deep sense of gratitude to our mentor **Mr. Shardul Rastogi**, 4th year, Jaypee Institute of Information Technology, India for his generous guidance, help and useful suggestions.

We also wish to extend our thanks to our respective families and friends for their insightful comments and constructive suggestions to improve the quality of this project work. And we also thank God, who did not let us down and helped us completing this project by keeping up our moral and dedication towards this project.

Date: **19 November 2024**

Enrolment Number	**Name of Student**	**Signature**
22103030	Aarush Gupta	
22103017	Saksham Purohit	
22103012	Shivaprasad Arunkumar Farale	

SUMMARY

Content-Based Filtering:

Content-based filtering focuses on recommending items based on their inherent attributes. In our system, course metadata such as categories, subcategories, and skills were analyzed using cosine similarity to identify courses similar to those a user has previously engaged with. This method ensures that recommendations are tailored to the content characteristics of courses. While effective for sparse datasets and independent of user interaction history, its reliance on pre-defined attributes may limit diversity and novelty in recommendations, particularly for users exploring new interests.

Collaborative Filtering:

Collaborative filtering leverages user behavior to generate recommendations by identifying patterns in user-course interactions. We implemented a hybrid model combining Singular Value Decomposition (SVD) and user similarity metrics, balancing latent factor extraction with direct behavioral trends. This approach excels in capturing user preferences dynamically and offers diverse recommendations based on shared interests across similar users. However, it relies heavily on sufficient interaction data and may face challenges with data sparsity or cold starts.

Graph-Based Community Detection (Louvain Method):

Using graph-based methods, we employed the Louvain algorithm for community detection within a user-course interaction network. This approach identifies clusters of users with shared course preferences, enabling group-based recommendations. By leveraging graph structures, the system effectively captures both direct and contextual relationships among users and courses, enhancing personalization. This method also highlights user clusters, providing insights into broader learning trends and supporting adaptive course recommendations.

Neo4j:

Neo4j was instrumental in modeling and analysing complex relationships within the e-learning domain, using a graph database to represent entities like courses, skills, and users as nodes, and their interactions as edges. This structure enabled efficient querying and extraction of insights through Cypher, such as identifying popular courses, skill trends, and instructor expertise. By applying algorithms like Degree Centrality, Betweenness Centrality, and the Louvain method, we uncovered influential nodes and clustered users based on shared preferences. These graph-based insights significantly enhanced the recommendation system's personalization and adaptability, making it more context-aware and scalable.

Chapter – 1

INTRODUCTION

1.1 General Introduction

As online education platforms continue to expand, the abundance of available courses presents a challenge for learners in selecting the most suitable options. Traditional recommendation systems often rely on simplistic approaches like collaborative filtering or content-based filtering, which fail to capture the intricate relationships among course attributes such as skills, instructors, and categories. This lack of depth limits personalization and the relevance of course recommendations.

To address this, our project focuses on creating a Knowledge Graph-Based Multimodal E-learning Recommender System. By leveraging Neo4j, a graph database, we aim to represent and analyse complex relationships between entities such as courses, skills, instructors, categories, and user preferences. Graph-based insights allow us to enhance the recommendation process by uncovering connections that would otherwise remain hidden in traditional tabular data.

The system integrates data visualization and analysis techniques to explore trends like course popularity, skill distribution, and instructor expertise. These insights will form the backbone of a hybrid recommendation engine, combining graph-based insights with machine learning algorithms to deliver dynamic and personalized course suggestions. This innovative approach seeks to improve user engagement and facilitate a more tailored learning experience.

1.2 Problem Statement

The rapid growth of online learning platforms has led to an overwhelming number of courses, making it difficult for users to identify those that align with their interests, skills, and career aspirations. Existing recommendation systems suffer from several limitations:

1. Lack of Relationship Depth: Most systems fail to model the intricate relationships between entities like course prerequisites, instructor expertise, and category-specific skills.

2. Static Recommendations: Many systems offer static recommendations that do not adapt to a user's evolving preferences or learning history.

3. Limited Personalization: Simple filtering methods do not account for factors like skill level, preferred language, or learning objectives, resulting in generic recommendations.

4. Difficulty in Trend Analysis: Without proper data analysis, identifying trends in course durations, viewer engagement, and instructor effectiveness remains a challenge.

To overcome these issues, our project proposes a multimodal recommendation system that combines graph-based insights from Neo4j with machine learning techniques. This approach not only improves personalization but also provides a scalable solution for analysing and adapting to user preferences dynamically.

1.3 Significance/Novelty of the Project

The Multimodal E-learning Recommender System introduces a novel approach to online course recommendation by combining graph-based insights with machine learning techniques. This innovation lies in its ability to represent and analyse the intricate relationships between various entities, providing a highly personalized and dynamic user experience. The project's novelty can be categorized under the following key aspects:

1.3.1 Graph-Based Representation

The project leverages the Neo4j graph database to model relationships between key e-learning entities such as courses, students, skills, instructors, and categories. Unlike traditional relational databases, this approach enables the system to uncover meaningful insights from the network structure. For instance, relationships between instructors and courses help identify popular courses or those aligned with specific skill sets. Dependencies between courses, such as prerequisites, are efficiently managed through the graph's natural structure.

1.3.2 Biased Random Walk

To better capture the context of entities in the graph, the project employs community detection with a biased random walk algorithm. This allows the system to generate feature-rich embeddings for nodes (e.g., courses or students) and represent the network structure in a way that enhances the recommendation model's ability to identify relevant courses. The use of stellargraph ensures a balance between breadth-first and depth-first sampling, offering versatility in capturing both local and global graph relationships.

1.3.3. Hybrid Recommendation Model

The recommender system combines collaborative filtering with insights derived from graph embeddings. This hybrid approach stands out because it utilizes user-item interactions (collaborative filtering) to predict user preferences based on similar patterns and incorporates the contextual and structural relationships from graph embeddings, enhancing recommendation relevance and accuracy.

1.3.4. Adaptability and Scalability

The system's flexible schema design allows it to adapt to different datasets and data sources. By accommodating evolving data structures and content types, it ensures longevity and applicability across various e-learning platforms. This adaptability supports integration with new course categories or user behaviours and expansion to include new attributes, such as learning styles or time commitments.

1.3.5. Enhanced Personalization

The system addresses the challenge of personalized course recommendations by considering multiple factors, including user preferences like skill level, language, and category, and dynamic real-time updates based on user behaviour, ensuring recommendations remain relevant and insightful.

1.3.6. Data-Driven Insights

In addition to providing recommendations, the system incorporates Python-based data analysis and visualization tools to explore trends in user engagement, course popularity, and other critical metrics. This not only aids in fine-tuning the recommendation algorithms but also provides administrators with actionable insights to improve the overall platform.

1.3.7. Bridging the Gap in Current Systems

The novelty also lies in addressing limitations of existing systems. Unlike traditional systems that rely solely on collaborative or content-based filtering, this project merges graph-based techniques to provide a comprehensive solution. By considering relationships such as prerequisites and instructor popularity, the system goes beyond simple user-item similarity to deliver more meaningful recommendations.

In summary, the significance of this project lies in its ability to redefine how online learning platforms cater to user needs. By integrating advanced graph-based techniques, machine learning, and dynamic adaptability, the Multimodal E-learning Recommender System sets a new benchmark for personalized and intelligent course recommendations.

1.4 Empirical Study

1.4.1 Challenges in Existing E-Learning Platforms

The growing adoption of online learning platforms such as Coursera, edX, and Udemy has created an overwhelming pool of educational content for learners to choose from. However, despite the increasing popularity of these platforms, many users face significant challenges in identifying the most suitable courses. Existing recommendation systems on these platforms, primarily relying on collaborative or content-based filtering, often fail to provide highly personalized suggestions. These systems typically overlook deeper relationships between courses, such as prerequisites, instructor expertise, and skills taught. For example, while a course may be rated highly by users, it might not align with the learner's current skill set or career goals.

1.4.2 Limitations of Current Recommendation Systems

To better understand these limitations, we analysed popular e-learning platforms and their recommendation mechanisms. Coursera and edX, for instance, provide recommendations based on user ratings and simple collaborative filtering, which often results in generic suggestions. Similarly, Udemy focuses on popularity metrics like enrolments and ratings but rarely considers the evolving preferences of individual learners. This lack of contextualization creates a gap in providing meaningful recommendations tailored to specific user needs.

1.4.3 Dataset Insights and Learner Behaviour

In addition to studying existing systems, a preliminary analysis of our dataset revealed key insights into learner behaviour and course attributes. The dataset, consisting of thousands of courses and user interactions, highlighted patterns such as the dominance of certain categories (e.g., Data Science, Technology), the significance of instructor reputation, and the critical role of prerequisites in defining course relevance. For instance, courses with clear prerequisite relationships were observed to perform better in terms of user satisfaction, emphasizing the need to model these connections explicitly.

1.4.4 Evaluation of Graph-Based Technologies

To address these challenges, our study also evaluated the use of graph-based technologies like Neo4j. Graph databases allow us to represent and query complex relationships between entities such as students, courses, instructors, and skills. Experimental queries conducted on Neo4j demonstrated its efficiency in identifying related courses based on shared attributes, such as instructor expertise or skill progression. The ability to analyse such intricate relationships provides a significant advantage over traditional relational databases and simpler recommendation approaches.

1.4.5 Advancements in Recommendation System Trends

Furthermore, current trends in recommendation systems highlight a shift toward hybrid and graph-based methodologies. By integrating graph embeddings like those generated through louvain algorithm, we can capture both the structural and semantic relationships in the data. Unlike generalized models, this approach enables dynamic, real-time recommendations that adapt to the user's evolving preferences. For example, a learner who has completed beginner-level courses in Python could be recommended intermediate-level courses in Machine Learning, based on their progress and skill requirements.

1.4.6 Bridging the Gap with a Proposed Solution

This empirical study validates the need for a more sophisticated and personalized recommendation system. By combining insights from existing platforms, dataset analysis, and cutting-edge graph-based technologies, our proposed system aims to bridge the gap between learners' needs and the recommendations they receive. This approach not only enhances the user experience but also sets a new benchmark for e-learning personalization.

1.4.7 Evaluation of Neo4j as a Tool

Neo4j was evaluated for its efficiency and usability in modelling and querying complex relationships:

- Advantages: The graph-based approach provided a flexible and intuitive way to model and analyse relationships compared to traditional relational databases.
- Challenges: Initial schema design required careful consideration to ensure scalability and ease of querying. However, once designed, Neo4j proved effective in managing large, interconnected datasets.

1.5 Brief Description of the Solution Approach

Our course recommendation system integrates advanced machine learning and recommendation techniques to provide personalized and effective suggestions to users. The solution is built on three primary approaches: **Graph-based**, **Content Filtering**, and **Collaborative Filtering**, each tailored to complement the others and enhance the overall performance of the system.

1.5.1 Graph-Based Model:

- Constructs a knowledge graph using Neo4j to represent complex relationships between courses, users, skills, instructors, and categories.
- Utilizes community detection (Louvain algorithm) and biased random walks to identify user clusters and implicit relationships.
- Provides context-aware recommendations by modelling intricate dependencies like prerequisites, instructor expertise, and category-specific preferences.

1.5.2 Content Filtering:

This method focuses on the course metadata, such as descriptions, keywords, and topics. We employed **cosine similarity** to compute the similarity between courses based on their feature vectors. This approach ensures that users are recommended courses similar to the ones they have previously interacted with, based on course content alone.

1.5.3 Collaborative Filtering:

For collaborative filtering, we implemented a **hybrid model** combining:

70% Singular Value Decomposition (SVD): This matrix factorization technique captures latent factors in user-course interactions, providing robust predictions for unseen user-course pairs.

30% User Similarity: Cosine similarity was used to measure the similarity between users based on their historical course ratings. This ensures that recommendations are influenced by the preferences of users with similar interests.

The hybrid model balances personalized suggestions with global trends observed in user data.

This combined approach leverages the strengths of all three techniques. While graph-based model provides context-aware embeddings, content filtering ensures relevance based on course data, and collaborative filtering personalizes recommendations using user behaviour patterns. Together, these methods create a comprehensive, accurate, and scalable recommendation system.

1.6 Comparison of Existing Approaches to the Problem Framed

The problem of course recommendation has been extensively studied, with several approaches implemented to address the challenge of providing personalized and accurate recommendations. Below, we compare our integrated solution with existing approaches:

1.6.1. Traditional Content-Based Filtering

- **Description:** Focuses solely on course attributes such as keywords, descriptions, or metadata to recommend similar courses.
- **Advantages:** Simple and effective for well-tagged datasets; interpretable.
- **Limitations:** Cannot account for user preferences beyond content; suffers from the **cold-start problem** for new users and items.

Our Solution: While content filtering forms part of our approach (using cosine similarity), it is supplemented with collaborative and graph-based techniques to overcome cold-start limitations and improve personalization.

1.6.2. Collaborative Filtering

- **Description:** Uses user interaction data (e.g., ratings, clicks) to identify patterns in preferences and recommend items based on similar users or items.
- **Advantages:** Learns hidden relationships between users and items; highly personalized.
- **Limitations:** Requires sufficient interaction data to be effective; suffers from **data sparsity** and the **cold-start problem**.

Our Solution: We enhance collaborative filtering with a hybrid model combining SVD (to capture latent factors) and user similarity (to include direct behavioral patterns), achieving better handling of sparse data and cold starts.

1.6.3. Graph-Based Approaches

- **Description:** Models user-item interactions as a graph, applying algorithms like random walks or embeddings for recommendations.
- **Advantages:** Naturally captures complex relationships; effective for multi-modal data.
- **Limitations:** May struggle with large-scale datasets or require specialized frameworks.

Our Solution: By integrating Graph based recommendation techniques of community detection, we leverage graph-based strengths while combining them with content and collaborative filtering for improved accuracy and robustness.

Chapter – 2

LITERATURE SURVEY

2.1 Summary of papers studied

- **“Knowledge Graph-based Multimodal GNN Model for Recommendations System”**: The paper introduces **MKGAT**, a Multimodal Knowledge Graph-based Graph Neural Network model designed to enhance recommendation systems by integrating multimodal data like text and images into knowledge graphs. By combining multimodal data, demonstrating superior performance compared to existing methods. Its innovative use of knowledge graphs and advanced neural network techniques positions it as a powerful tool for solving information overload and improving user experience in recommendation tasks

- **The book "Graph Databases: New Opportunities for Connected Data" by Ian Robinson, Jim Webber, and Emil Eifrem, published by O'Reilly Media**, provides a comprehensive exploration of the principles and applications of graph databases, with a particular focus on Neo4j. It highlights the advantages of graph databases over traditional relational databases, especially when working with connected data, making them ideal for scenarios that require analysing complex relationships. The book delves into real-world use cases, including their application in recommendation systems and fraud detection, showcasing the versatility and potential of graph databases in diverse domains. Additionally, it offers valuable practical insights into schema design and graph modelling techniques, which are essential for effectively leveraging graph databases. This resource significantly contributed to understanding the conceptual framework for building a knowledge graph and applying Neo4j in the context of e-learning recommendations, enabling the design of an efficient and scalable recommendation system.

- **“Course Recommendations in Online Education Based on Collaborative Filtering Recommendation Algorithm”,** Jing Li, Zhou Ye: This research paper explores a collaborative filtering-based recommendation system tailored for online education. The proposed model enhances learning by suggesting courses aligned with users' preferences and historical behaviour. It integrates advanced filtering techniques to handle sparsity and cold-start challenges, ensuring personalized and effective recommendations. Through algorithmic optimizations, it demonstrates improved accuracy and scalability, contributing significantly to adaptive e-learning systems and addressing challenges in modern online education.

2.2 Integrated summary of the literature studied

The integration of knowledge graphs, graph neural networks (GNNs), and collaborative filtering techniques forms the foundation for advanced recommendation systems, particularly in personalized settings like e-learning.

The paper "Knowledge Graph-based Multimodal GNN Model for Recommendations System" introduces MKGAT, a model that integrates multimodal data (text, images) into knowledge graphs, using GNNs to enhance recommendation accuracy. By incorporating diverse data sources, MKGAT outperforms traditional models in handling information overload, offering more relevant and context-aware recommendations.

The book "Graph Databases: New Opportunities for Connected Data" explores the advantages of graph databases, particularly Neo4j, for modeling and querying connected data. It emphasizes how graph databases are ideal for recommendation systems due to their ability to represent complex relationships and offer insights into efficient schema design and graph modeling techniques. This is crucial for building knowledge graphs that power models like MKGAT.

The paper "Course Recommendations in Online Education Based on Collaborative Filtering" focuses on enhancing recommendations in e-learning through collaborative filtering, addressing challenges like sparsity and cold-start problems. By optimizing these techniques, the model improves personalization and scalability, ensuring more accurate course suggestions based on user preferences and behaviour.

Together, these resources highlight the synergy between multimodal knowledge graphs, GNNs, and collaborative filtering in recommendation systems. They demonstrate how graph databases like Neo4j can efficiently manage complex relationships, enabling personalized, scalable, and context-aware recommendations in dynamic environments such as online education. This integrated approach significantly enhances the user experience by delivering more relevant and accurate content suggestions.

Chapter – 3

REQUIREMENT ANALYSIS AND SOLUTION APPROACH

3.1 Overall description of the Project

The **"Knowledge Graph-Based Multimodal E-learning Recommender System**" combines three powerful recommendation models—graph-based, collaborative filtering, and content-based filtering—to deliver highly personalized and intelligent course recommendations for online learners. This hybrid approach addresses key challenges in traditional systems, such as limited personalization, sparse user data, and static recommendations, by leveraging advanced techniques and data-driven insights.

3.1.1 Graph-Based Model:

- Constructs a knowledge graph using Neo4j to represent complex relationships between courses, users, skills, instructors, and categories.
- Utilizes community detection (Louvain algorithm) and biased random walks to identify user clusters and implicit relationships.
- Provides context-aware recommendations by modelling intricate dependencies like prerequisites, instructor expertise, and category-specific preferences.

3.1.2 Collaborative Filtering Model:

- Employs a hybrid collaborative filtering approach:
    - Singular Value Decomposition (SVD) captures user-item interactions.
    - User Similarity-Based Filtering identifies similar users based on their course ratings.
- Combines these methods to predict user preferences effectively, even in sparse datasets.

3.1.3 Content-Based Filtering Model:

- Analyzes course metadata (e.g., descriptions, categories, skills) using cosine similarity to recommend courses similar to those previously interacted with by the user.
- Focuses on course attributes to ensure relevance and precision in recommendations.

How It Works

The system integrates these models into a unified recommendation engine, allowing each model to complement the others:

- The graph-based model captures structural and contextual relationships.
- The collaborative model provides personalized recommendations based on user behaviour.
- The content-based model ensures that recommendations align with course attributes.

Additionally, the system incorporates data visualization tools to analyse trends in course popularity, user engagement, and skill distributions. It adapts dynamically to user preferences and new data sources, ensuring scalability and relevance.

By combining these models, the project delivers a robust, personalized e-learning recommendation system that enhances user experience and supports diverse learning goals.

3.2 Requirement Analysis

3.2.1 Functional Requirements: Creation of User Dataset

In this phase, we developed a methodology to create a user-course dataset that simulates the behaviour of 1000 users interacting with online courses. The goal was to ensure that the dataset reflects diverse user interests, proficiency levels, and language preferences. We used a combination of random sampling, weighted probabilities, filtering, and rating mechanisms to make the dataset realistic and relevant. The following methodologies were key in generating the dataset:

1. Random Sampling:
    - Random sampling was employed to simulate diverse user profiles by selecting interests, skill levels, and language preferences. This process ensures that the dataset represents a wide variety of behaviours, mirroring real-world user interactions with online platforms. Multiple categories or skills were randomly chosen for each user, which helped maintain diversity and avoid over-representation of any single attribute. This approach ensures that users have varied backgrounds, interests, and learning preferences, providing a more realistic and dynamic dataset.

2. Weighted Probabilities:

    - Weighted probabilities were applied to the selection of primary and secondary attributes such as categories and skills. Users are more likely to engage with courses in their primary area of interest (e.g., a specific category) but may also explore secondary skills or sub-categories. To simulate this, we weighted the dominant attributes (e.g., category) higher than secondary attributes (e.g., skills), but with enough flexibility to allow users to explore courses outside their main interests. This approach replicates typical online learning behaviour, where users often balance exploration with core subject matter.

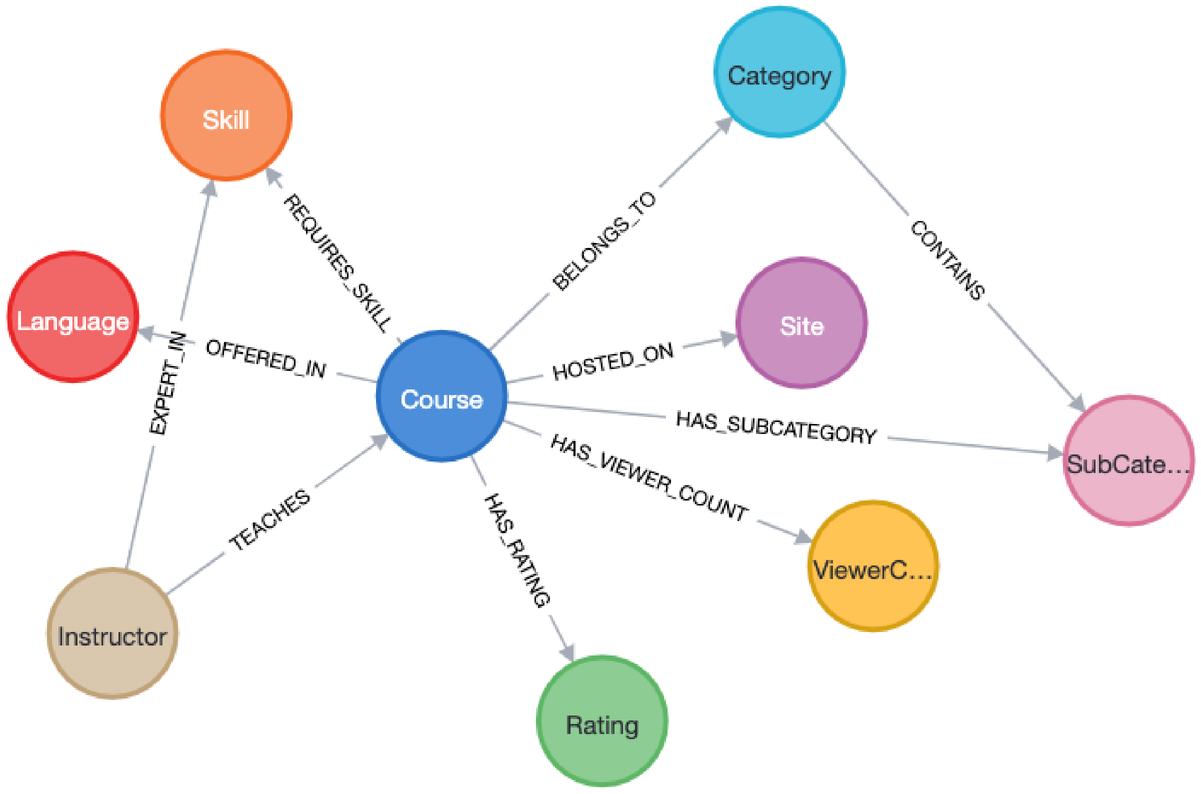
3. Filtering and Matching:

    - After generating the user profiles, courses were filtered based on user preferences such as interests, language, and skill levels. This filtering mechanism ensures that users are only recommended courses that align with their specific preferences and learning needs. By ensuring that courses match the user's selected category, skills, and language preferences, the dataset remains relevant and personalized. Language preferences, in particular, ensure that users receive content in a language they are comfortable with, further enhancing their learning experience.

4. Rating System:

    - A rating system was introduced to evaluate courses based on their relevance to the user's selected attributes, including interest areas, instructors, and skill levels. The rating behaviour mirrors real-world patterns, where users are more likely to give higher ratings to courses that meet their personal needs or are taught by instructors they prefer. Additionally, the skill alignment was considered, ensuring that more advanced users were recommended challenging courses. This rating system helps simulate user satisfaction and engagement with courses in a realistic manner, providing insights into the quality of the recommendations.

3.2.3 Logical Dataset Requirements



Knowledge Graph Schema

Nodes (Entities):

1. Course: Represents the individual courses offered on various platforms, including attributes like title, rating, and duration.
2. Category: Represents the broader domain or field to which a course belongs.
3. SubCategory: A more specific classification under a category for detailed grouping.
4. Skill: Captures the key competencies required to complete or gained from a course.
5. Instructor: Represents the experts responsible for teaching courses, with inferred expertise based on their taught courses.
6. Language: Indicates the mediums of instruction for courses, such as English or Spanish.
7. Rating: Represents the average rating of courses, useful for identifying course quality.
8. Duration: Represents the time commitment required to complete courses, aiding in filtering based on user preferences.
9. Site: Represents the platform hosting the courses (e.g., Coursera, Udemy).

Relationships:

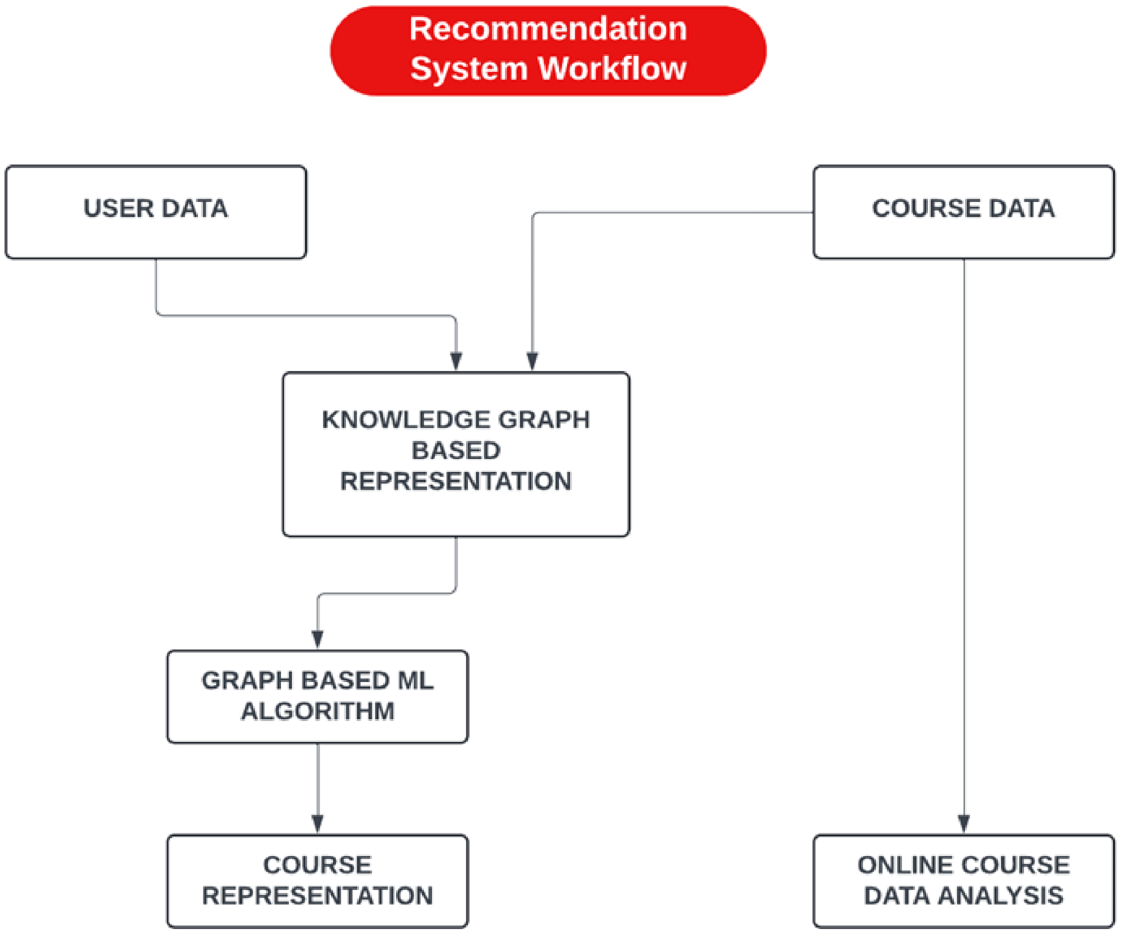
1. Course → Category: Links each course to its associated category, indicating its primary field.
2. Course → SubCategory: Connects courses to their specific subcategories within broader categories.
3. Category → SubCategory: Establishes hierarchical relationships, showing the structure of categories and subcategories.
4. Course → Skill: Indicates the skills required or imparted by a course.
5. Instructor → Course: Links instructors to the courses they teach, allowing mapping of their expertise.
6. Instructor → Skill: Connects instructors to skills inferred from their taught courses.
7. Course → Language: Shows the languages in which courses are offered.
8. Course → Rating: Connects courses to their rating nodes to evaluate quality.
9. Course → Duration: Links courses to their duration nodes, aiding in user filtering based on time availability.
10. Course → Site: Identifies the platform hosting a course, useful for platform-specific recommendations.

Chapter 4

MODELING AND IMPLEMENTATION DETAILS

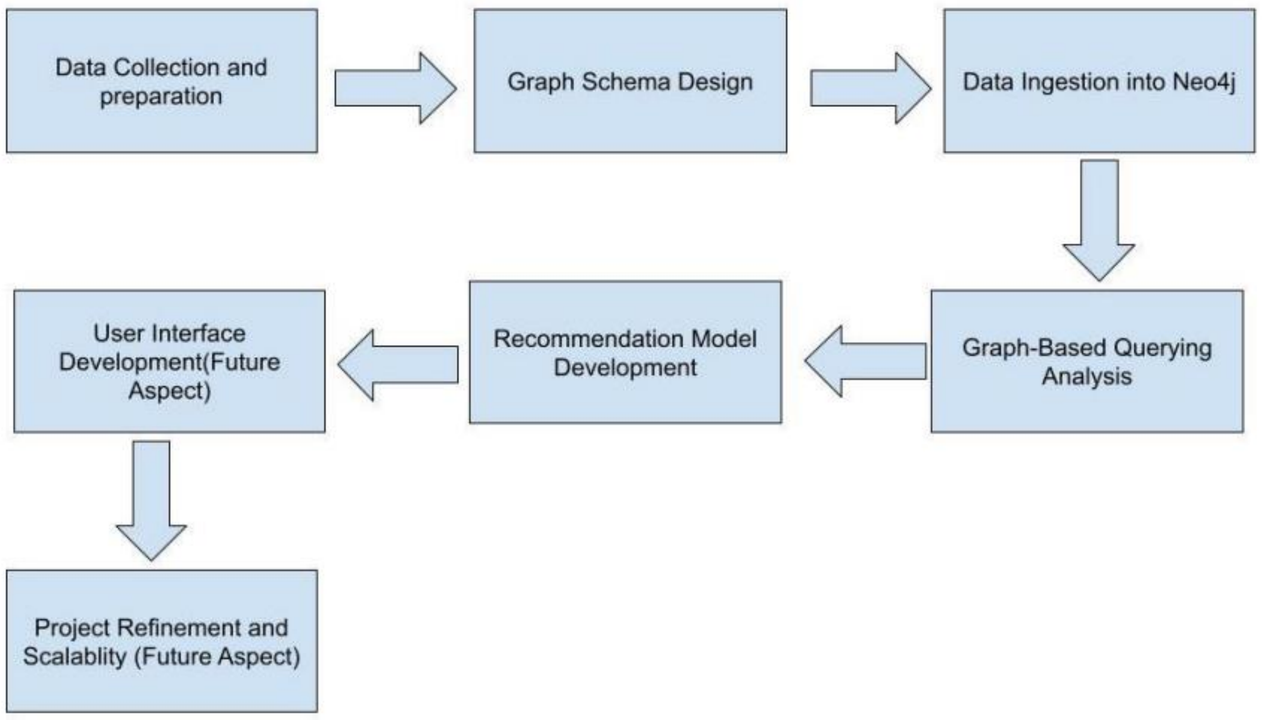
4.1 Design Diagrams

4.1.1 Control Flow Diagram


Recommendation System Workflow
USER DATA
COURSE DATA
KNOWLEDGE GRAPH BASED REPRESENTATION
GRAPH BASED ML ALGORITHM
COURSE REPRESENTATION
ONLINE COURSE DATA ANALYSIS

This flowchart outlines the workflow for your Knowledge Graph-Based Multimodal E-Learning Recommender System. It combines user data (e.g., preferences, skills) and course data (e.g., categories, content) into a knowledge graph, representing relationships between users and courses. A graph-based machine learning algorithm analyses this graph to recommend personalized courses. Simultaneously, course data analysis extracts insights to improve the recommendation process.

4.1.2 Workflow Diagram


Data Collection and preparation
Graph Schema Design
Data Ingestion into Neo4j
Graph-Based Querying Analysis
Recommendation Model Development
User Interface Development(Future Aspect)
Project Refinement and Scalablity (Future Aspect)

1. Data Collection & Preparation: Collect course data, including content, instructor profiles, user interactions, and skill requirements. Clean and structure the data for efficient use in graph modelling and machine learning.

2. Graph Schema Design: Define a schema in Neo4j with nodes (e.g., Course, Instructor, Skill) and relationships (e.g., TAUGHT_BY, REQUIRES_SKILL). Ensure its flexible for future data expansions.

3. Data Ingestion: Import the cleaned data into Neo4j, maintaining the defined schema and relationships to enable effective querying and visualization.

4. Querying & Analysis: Use Cypher queries to extract insights like popular courses or skills, setting the foundation for the recommendation model.

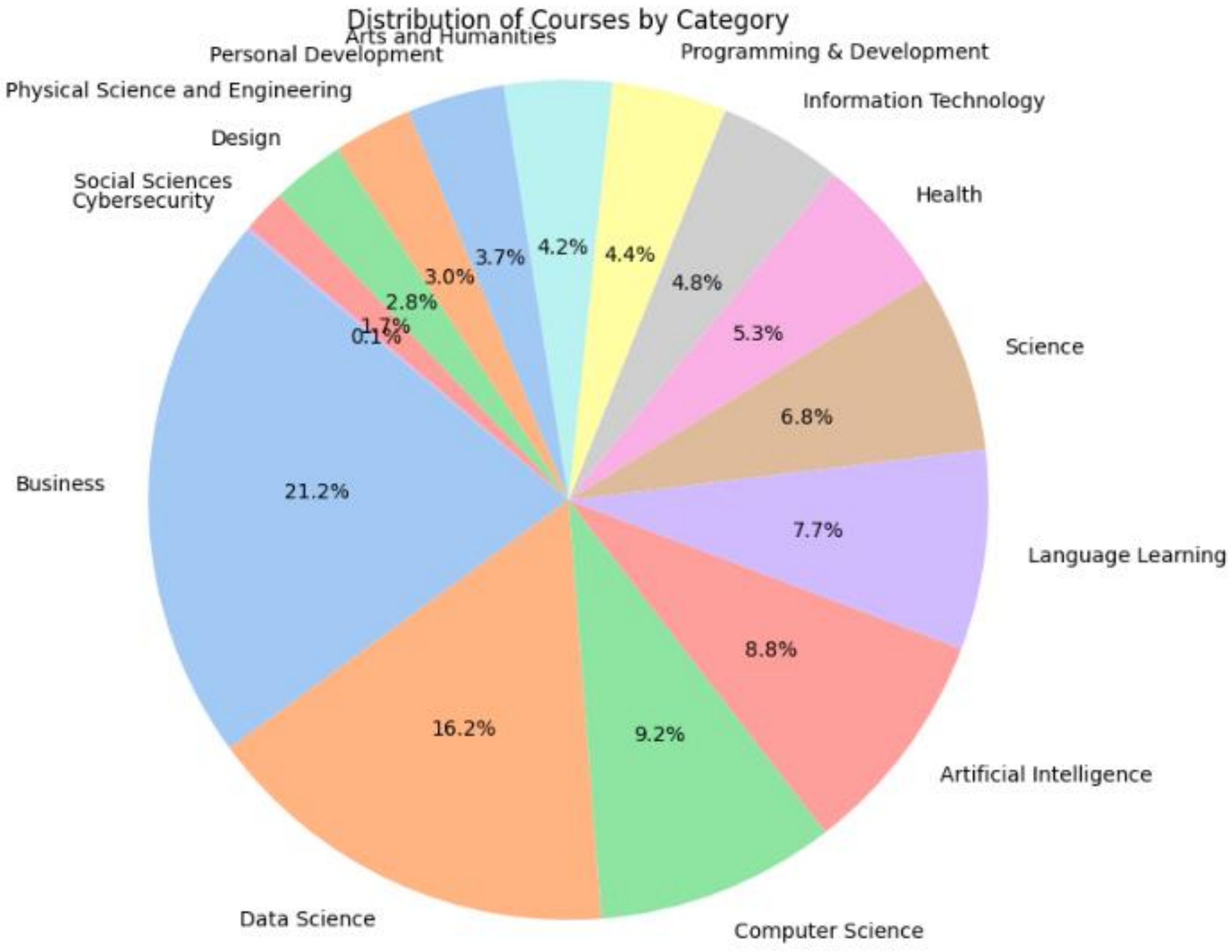
5. Recommendation Model: Build a hybrid recommendation engine combining collaborative filtering with graph-based insights for personalized course suggestions. Optimize the model for accuracy.

6. User Interface: Develop an interactive web UI that dynamically displays personalized course recommendations and allows users to filter by skills, categories, or instructors.

7. Testing & Evaluation: Assess the recommendation engine using metrics like precision, recall, and user engagement. Gather user feedback to refine the system.

8. Refinement & Scalability: Improve the model based on feedback, focusing on scalability and future-proofing. Plan for real-time feedback loops and additional data integration to enhance recommendations.

4.2 Data Analysis & Insights



The pie chart represents the distribution of courses by category in an e-learning platform. Each segment of the pie chart shows the proportion of courses available in a particular category.

Interpretation:

- **Business** has the largest share at 21.2%, indicating it is the most popular category in terms of the number of courses.
- Other notable categories include:
  - **Data Science** (16.2%)
  - **Computer Science** (9.2%)

- **Artificial Intelligence** (8.8%)
- **Language Learning** (7.7%)

Smaller categories, such as **Cybersecurity** (0.1%) and **Social Sciences** (1.7%), have much fewer courses. The chart provides insight into the variety and focus areas of the e-learning courses offered

4.3 Neo4j Graph Analysis

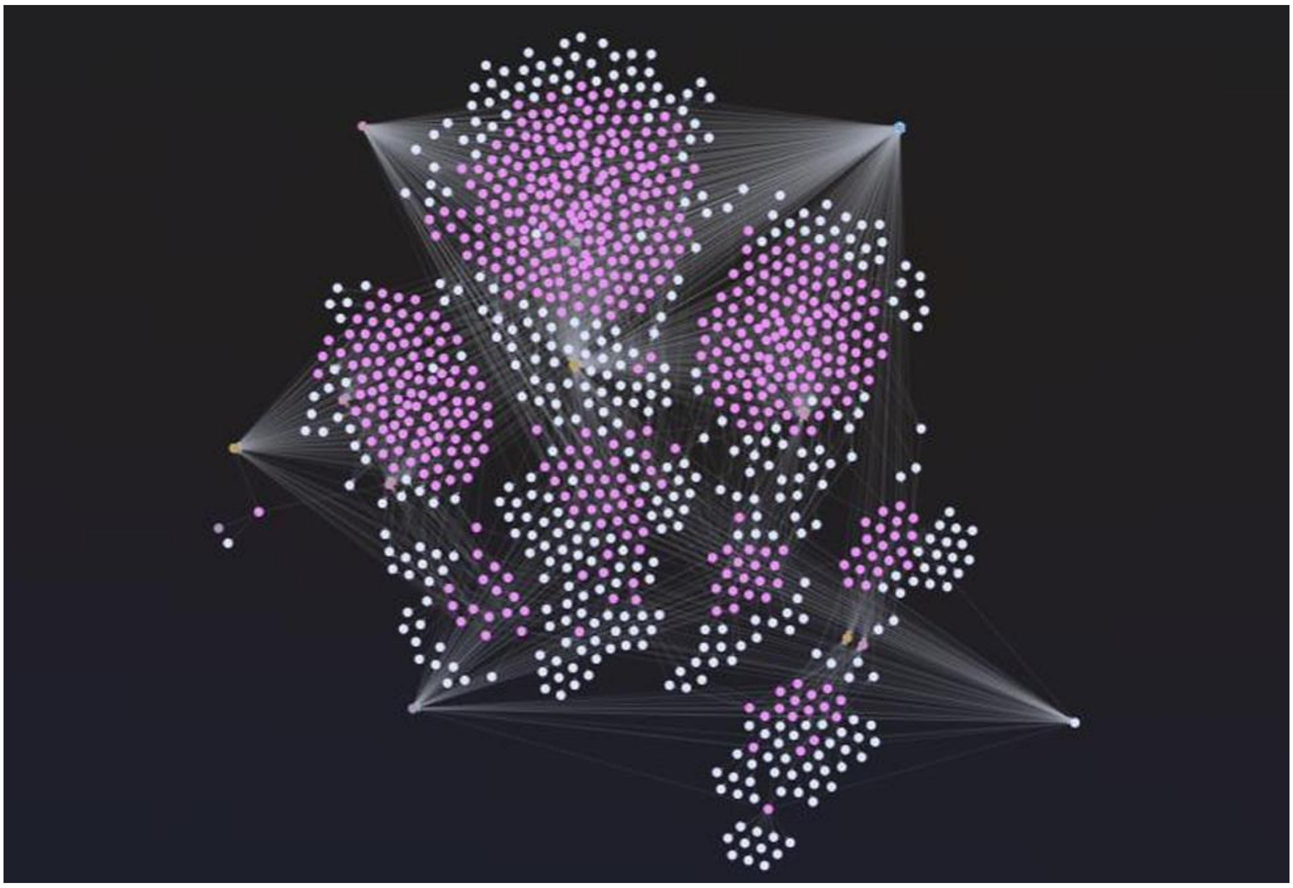
4.3.1 Neo4j Graph Analysis

Objective:
Neo4j is utilized to represent the relationships between various entities such as courses, skills, instructors, and other attributes in a graph format. This enables the efficient representation and querying of connections, making it easier to extract insights such as course recommendations, skill distributions, and trends in instructor expertise.

Schema Overview:
The schema (as outlined) supports the analytical goals of the project by structuring entities and their relationships effectively. Here's a brief overview:

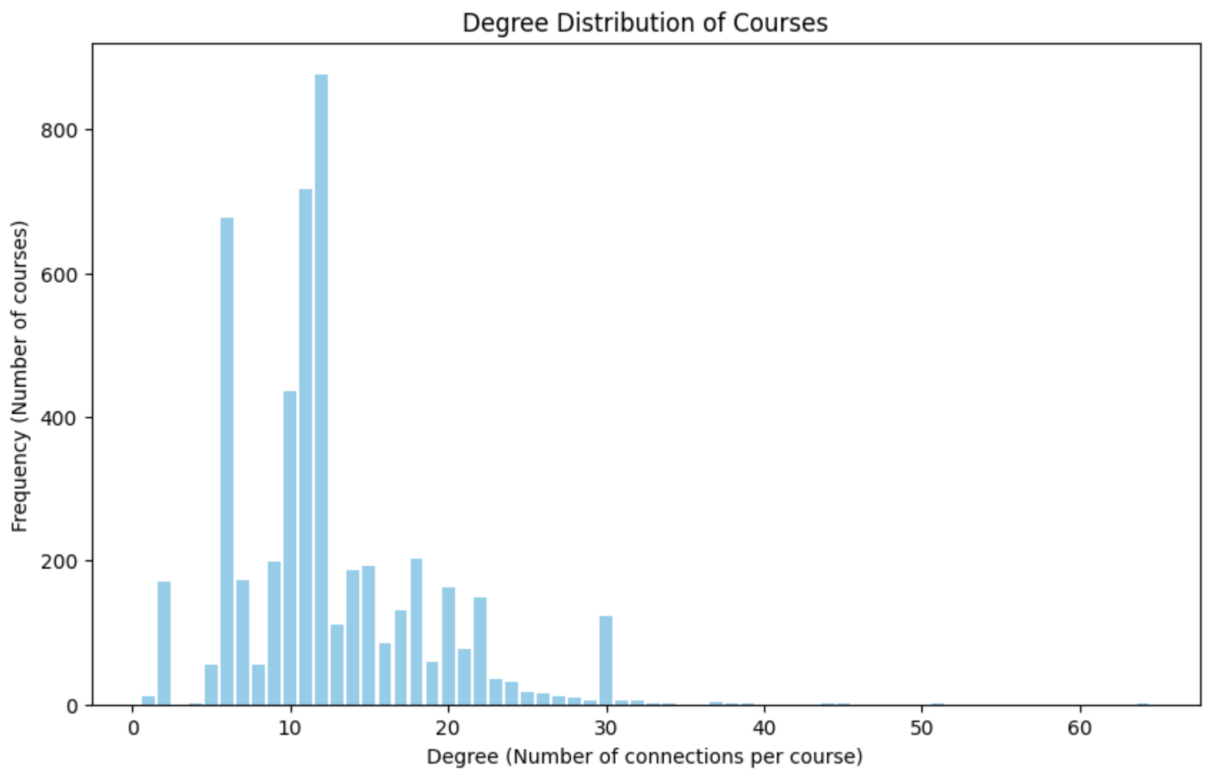
- Nodes (Entities): Represent key elements like courses, categories, instructors, etc., each containing attributes like rating, duration, and language.
- Relationships: These define how the entities interact or relate to each other, such as courses being linked to categories and instructors teaching certain courses.

This structure supports various analytical tasks, such as discovering highly rated courses, mapping instructor expertise, and identifying skill trends, which are essential for delivering personalized course recommendations and insights.



This is the realisation of the schema showcased above. It helps understand the vastness of data and visualise how nodes within the schema are connected and provides life to a mundane and difficult to imagine tabular data.

4.3.2 Queries and Insights



1. Degree Centrality:

Degree centrality is a measure used in network analysis to quantify the importance of a node (in this case, a course) based on its number of direct connections (or edges) to other nodes. It is calculated by counting the number of edges connected to a node.

Formula:

For an undirected graph, degree centrality for a node v is calculated as:

$C_D(v)$=degree(v)=number of edges connected to v

Where:

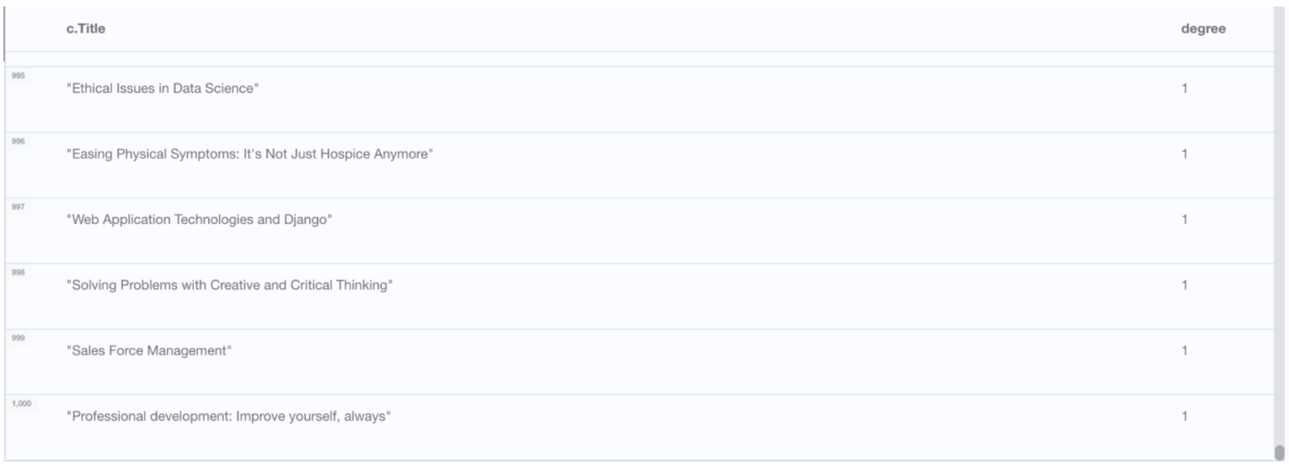
- v is the node (e.g., a course in your schema),
- Degree is the number of edges connected to the node.

```
MATCH (c:Course)

WITH c, [(c)--() | 1] AS relationships

RETURN c.Title, COUNT(relationships) AS degree

ORDER BY degree DESC;
```

Interpretation:

C.T		**Degree**
1	"AWS Cloud Technical Essentials"	7
2	"Google Cloud Fundamentals: Core Infrastructure"	7
3	"Microsoft Future Ready: Designing and Implementing a Data Science Solution on Azure"	7
4	"Teaching Digital Literacy: Videogames in Education"	6
5	"UX Design Fundamentals: Delivering Value to Users"	6

High Degree Centrality: A course with high degree centrality is one that has many direct relationships with other entities. This could indicate a course with broad coverage, greater interactivity, and potential popularity.



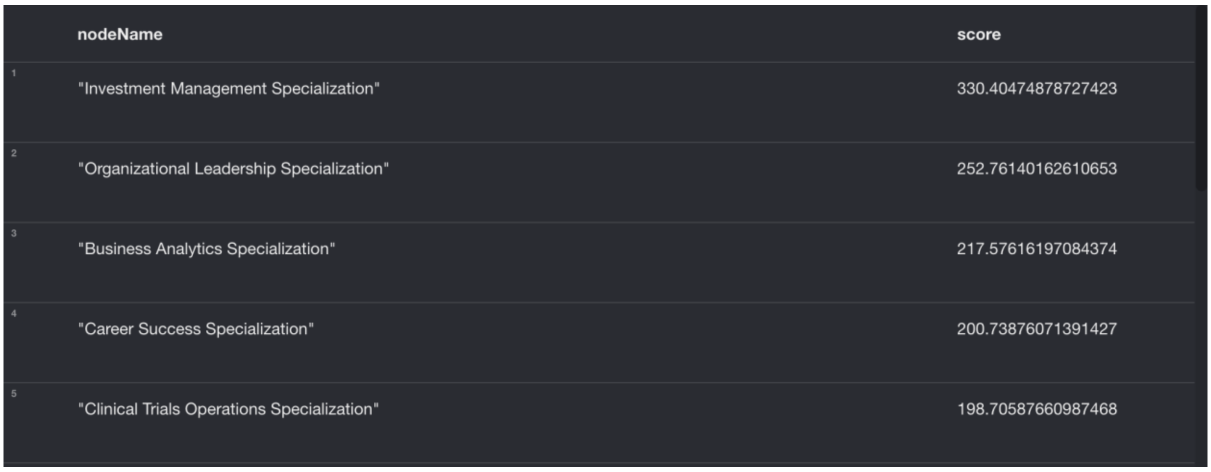
	c.Title	degree
995	"Ethical Issues in Data Science"	1
996	"Easing Physical Symptoms: It's Not Just Hospice Anymore"	1
997	"Web Application Technologies and Django"	1
998	"Solving Problems with Creative and Critical Thinking"	1
999	"Sales Force Management"	1
1,000	"Professional development: Improve yourself, always"	1

Low Degree Centrality: A course with low degree centrality has fewer direct connections, suggesting it might be more niche or specialized.

2.Betweenness Centrality

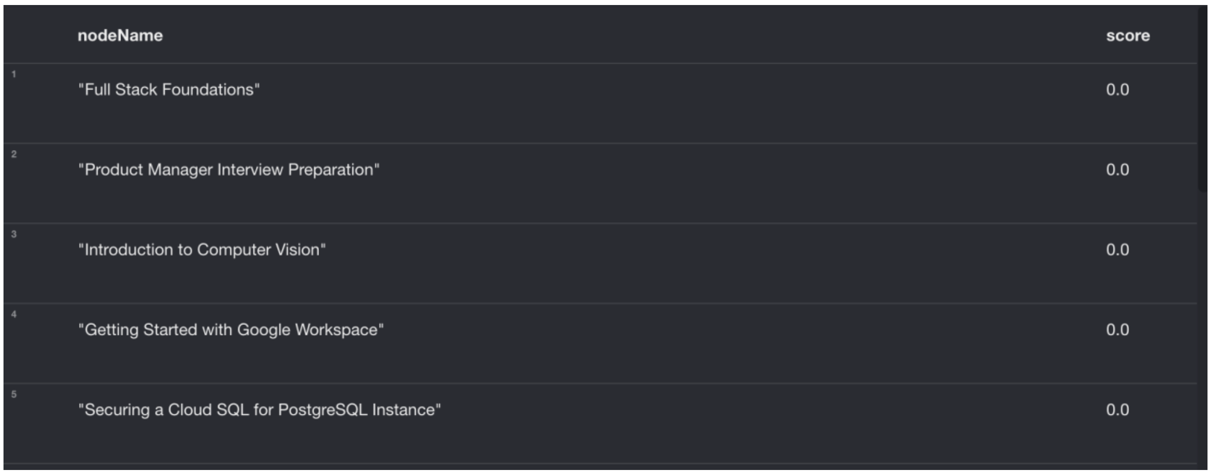
Betweenness Centrality is a network measure that identifies the nodes (in this case, courses) that serve as bridges or intermediaries between other nodes in the network. In simple terms, a node with high betweenness centrality has more control over the flow of information or connections between other nodes. It acts as a "connector" or "broker" in the network. Betweenness centrality is particularly useful for detecting the "important" nodes in a network that may not have the highest degree but still play a crucial role in connecting different parts of the network.

```
CALL gds.betweenness.stream('fullGraph') YIELD nodeId, score
RETURN gds.util.asNode(nodeId).Title AS nodeName, score
ORDER BY score DESC
LIMIT 20
```



	nodeName	score
1	"Investment Management Specialization"	330.40474878727423
2	"Organizational Leadership Specialization"	252.76140162610653
3	"Business Analytics Specialization"	217.57616197084374
4	"Career Success Specialization"	200.73876071391427
5	"Clinical Trials Operations Specialization"	198.70587660987468

```
CALL gds.betweenness.stream('fullGraph') YIELD nodeId, score
RETURN gds.util.asNode(nodeId).Title AS nodeName, score
ORDER BY score ASC
LIMIT 20;
```



	nodeName	score
1	"Full Stack Foundations"	0.0
2	"Product Manager Interview Preparation"	0.0
3	"Introduction to Computer Vision"	0.0
4	"Getting Started with Google Workspace"	0.0
5	"Securing a Cloud SQL for PostgreSQL Instance"	0.0

Courses with high **betweenness centrality**, such as "Investment Management Specialization", "Organisational Leadership Specialization", and "Business Analytics Specialization", serve as critical bridges connecting different groups of courses. These **core courses** play a central role in navigating between various topics and are likely to be popular, widely recommended, and serve as gateways for users to explore related subjects.

In contrast, **peripheral courses** with lower betweenness centrality, like "Diseño y Gestión de Proyectos de Desarrollo Specialization" or "Cloud Computing Law Specialization", are more specialised and connect with fewer courses, serving niche audiences.

Key takeaway: High betweenness centrality courses are crucial for course recommendations and for fostering broader learner engagement across different subjects.

3. Eigenvalue centrality

Eigenvector centrality for a node v is calculated using the principle of eigenvalues and eigenvectors from linear algebra. The centrality score of a node is proportional to the sum of the centrality scores of its neighbours. This recursive property makes eigenvector centrality a powerful tool for ranking nodes in terms of their overall influence in the network.

For a graph with n nodes, eigenvector centrality for a node v is computed as:

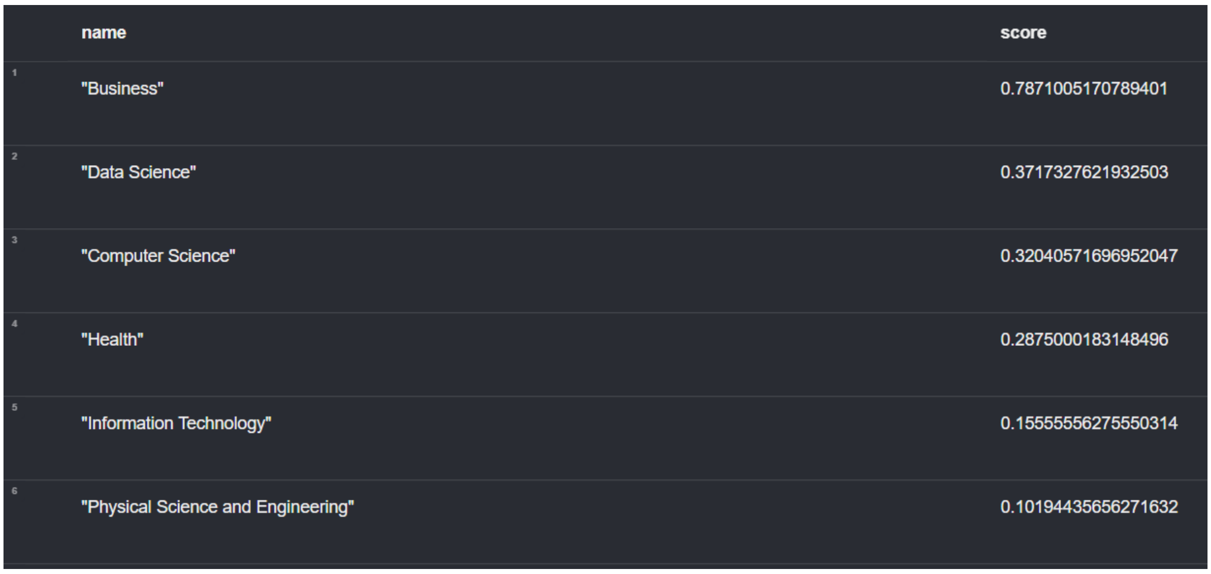
$$C_E(v) = \frac{1}{\lambda} \sum_{u \in N(v)} C_E(u)$$

Where:

- $C_E(v)$: Eigenvector centrality of node v.

- λ: The largest eigenvalue of the graph's adjacency matrix.
- N(v): The set of neighbours directly connected to v.
- $C_E(u)$: Eigenvector centrality of a neighbour node u.

The formula can be rewritten in matrix form, where the centrality scores for all nodes form an eigenvector of the graph's adjacency matrix.



	name	score
1	"Business"	0.7871005170789401
2	"Data Science"	0.3717327621932503
3	"Computer Science"	0.32040571696952047
4	"Health"	0.2875000183148496
5	"Information Technology"	0.15555556275550314
6	"Physical Science and Engineering"	0.10194435656271632

"Business" (0.7871):

- It is the most influential category in the network.
- Likely connected to many courses or categories that themselves have high centrality, suggesting strong interconnections and a high degree of importance.

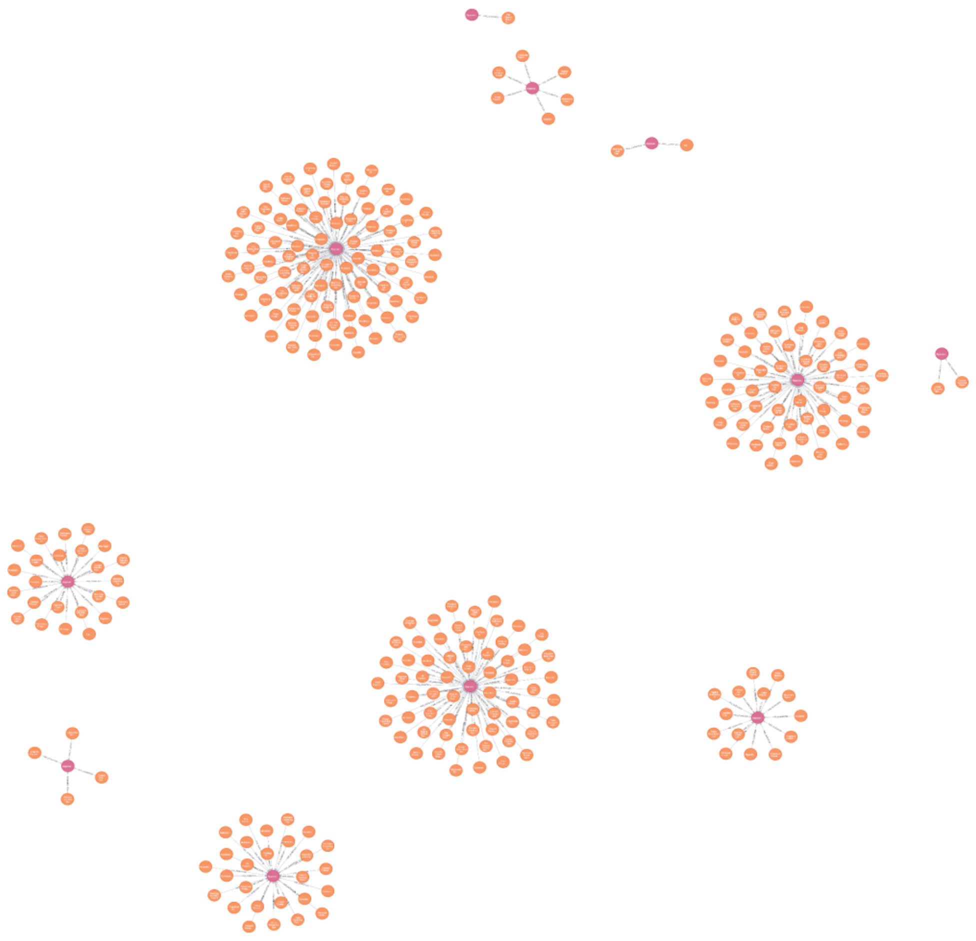
"Data Science" (0.3717) and "Computer Science" (0.3204):

- These are moderately influential categories.
- They have significant connections to other important nodes but are less central compared to "Business."

"Health" (0.2875):

- Slightly less central than "Data Science" and "Computer Science."
- May indicate a more specialised or niche presence in the network, connected to moderately influential nodes.\

5. Courses Linked to Durations (Duration Nodes)



Interpretation:

- The graph groups courses by their durations, revealing time commitments for each course.
- For example:
    - A course with a Duration of "Approximately 3 months to complete" will be connected to that specific duration node.
    - If multiple courses share the same duration, they will all connect to the same Duration node, showing popular time lengths.
- **Insights Derived:**
    - Identifies common course durations, such as whether most courses require 3, 5, or 7 months.
    - Helps users filter courses based on their available time.
    - Reveals patterns or trends in course lengths—for example, shorter durations might appeal to users looking for quick certifications.

4.4 Content-Based Filtering

This solution applies a content-based collaborative filtering approach to predict course ratings based on their textual features (such as category, sub-category, and skills). The process involves using a Cosine Similarity measure between course tags to find similarities, and then predicting the rating based on the weighted average of similar courses. We also evaluate the model using Root Mean Squared Error (RMSE) and F1 Score.

Steps Involved:

1. Data Preprocessing:

   - Combining Tags: The Category, Sub-Category, and Skills columns are concatenated to form a new tag column. This column is a textual representation of the course's primary attributes that can be used for vectorization.

```
course['tag'] = course['Category'] + ' ' + course['Sub-Category'] + ' ' + course['Skills']
```

2. Data Splitting:

   - The dataset is split into a training and testing set using Stratified Sampling, ensuring that the distribution of ratings in both sets is similar.
   - Ratings are binned into broader categories (Rating_Binned) to facilitate stratified sampling and ensure there is sufficient representation of each class in both training and testing sets.

```
new_df['Rating_Binned'] = pd.cut(new_df['Rating'], bins=[0, 2, 3, 4, 5], labels=[1, 2, 3, 4])
```

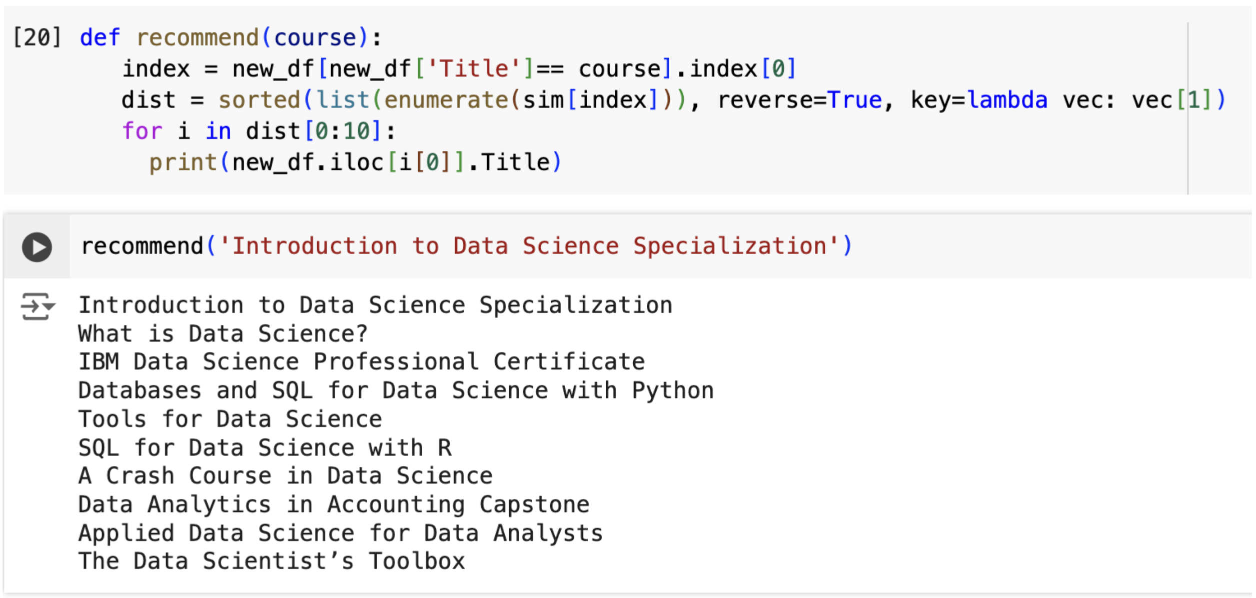
3. Vectorization:

   - The textual tag column is transformed into numerical data using CountVectorizer. This converts the words in the tags into a matrix of token counts.
   - The CountVectorizer is fitted to the training set, ensuring the vocabulary is learned from the training data.

cv = CountVectorizer(max_features=10000, stop_words='english')

train_vectors = cv.fit_transform(train['tag'].values.astype('U')).toarray()

OUTPUT:



```
[20] def recommend(course):
         index = new_df[new_df['Title']== course].index[0]
         dist = sorted(list(enumerate(sim[index])), reverse=True, key=lambda vec: vec[1])
         for i in dist[0:10]:
           print(new_df.iloc[i[0]].Title)
```

```
recommend('Introduction to Data Science Specialization')
```

```
Introduction to Data Science Specialization
What is Data Science?
IBM Data Science Professional Certificate
Databases and SQL for Data Science with Python
Tools for Data Science
SQL for Data Science with R
A Crash Course in Data Science
Data Analytics in Accounting Capstone
Applied Data Science for Data Analysts
The Data Scientist’s Toolbox
```

4. Cosine Similarity:

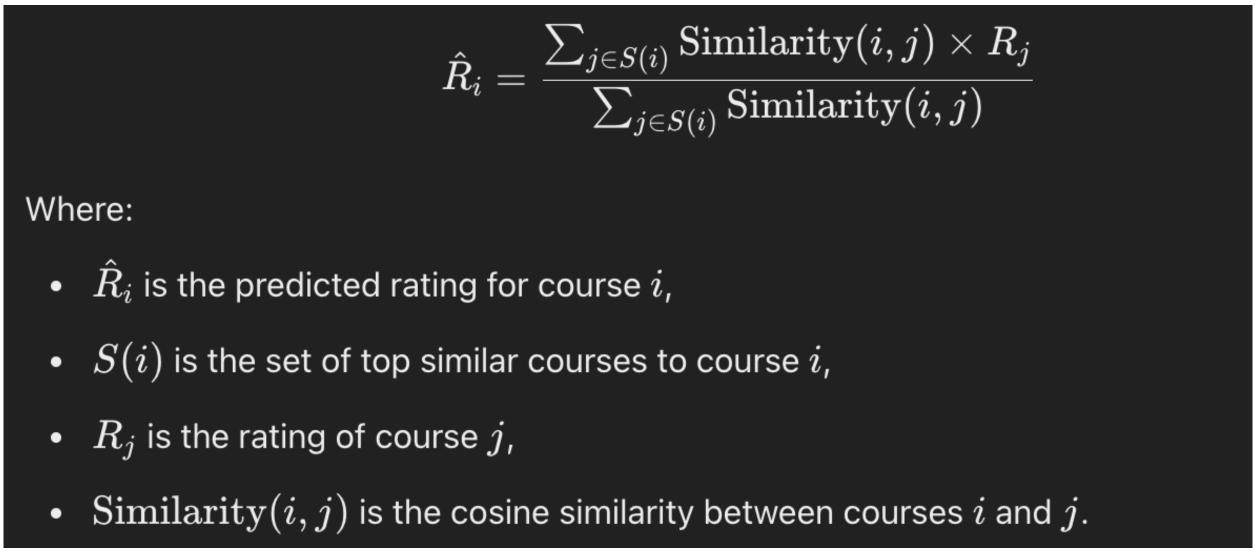
- A Cosine Similarity matrix is computed to measure the similarity between courses based on their vectorized tags. Cosine similarity is defined as:

$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

This matrix helps in identifying similar courses to a given course. The higher the cosine similarity, the more similar the courses are.
python

sim = cosine_similarity(train_vectors)

5. Rating Prediction:
   - For each course in the test set, a prediction is made based on the weighted average of the ratings of the top 10 most similar courses in the training set. The formula for the weighted average is:



$$\hat{R}_i = \frac{\sum_{j \in S(i)} \text{Similarity}(i, j) \times R_j}{\sum_{j \in S(i)} \text{Similarity}(i, j)}$$

Where:

- $\hat{R}_i$ is the predicted rating for course i,
- $S(i)$ is the set of top similar courses to course i,
- R_j is the rating of course j,
- $\text{Similarity}(i, j)$ is the cosine similarity between courses i and j.

If no similar courses are found, the mean rating of the training set is used as the fallback prediction.

```
def predict_ratings(title, train, sim):
```

4.5 Collaborative Filtering

The collaborative filtering approach predicts user preferences based on interactions with courses, utilizing a hybrid model combining:

1. SVD-Based Filtering (70%): Captures latent factors in user-course interactions.
2. User Similarity-Based Filtering (30%): Explores similarity between users using Pearson correlation.

1. Data Loading and Setup

The script starts by importing necessary libraries (pandas, numpy, sklearn) and loading the dataset.

- The dataset is read into a ratings_matrix using pandas. It has a shape of (1000, 5000), indicating 1000 users and 5000 courses.

```
data = pd.read_csv('collab_dataset.csv')
```

```
ratings_matrix = data.values
```

2. Data Splitting

The dataset is split into training, validation, and test sets:

- Training Set: 80% of the data
- Validation Set: 10% of the data
- Test Set: 10% of the data

This split ensures that there is no overlap between sets.

```
train_size = int(len(non_nan_indices) * 0.8)
```

3. Data Preprocessing

To handle missing values (NaN), the script computes a global mean rating and fills missing entries with centered values (ratings minus the global mean).

```
global_mean = np.nanmean(train_matrix)
train_matrix_filled = np.nan_to_num(train_matrix - global_mean, nan=0)
```

4. Latent Factor Model (SVD)

The SVD (Singular Value Decomposition) method is used to factorize the rating matrix into latent components:

- It extracts 100 latent features from the training matrix.
- The svd_predict() function predicts ratings by reconstructing the matrix from these latent features.

```
predicted_matrix_svd = svd_predict(train_matrix_filled)
```

5. Item-based Collaborative Filtering

Item-based filtering uses the cosine similarity between courses (items):

- The item similarity matrix is computed using cosine_similarity().
- Predicted ratings are generated by weighting the user's ratings with item similarities.

```
item_similarity_matrix = cosine_similarity(np.nan_to_num(train_matrix.T, nan=0))
predicted_matrix_item_based = np.dot(np.nan_to_num(train_matrix, nan=0),
item_similarity_matrix)
```

6. User-based Collaborative Filtering

User-based filtering uses the cosine similarity between users:

- The user similarity matrix is calculated.
- Ratings are predicted based on similar users' ratings.

```
user_similarity_matrix = cosine_similarity(np.nan_to_num(train_matrix, nan=0))
predicted_matrix_user_based = np.dot(user_similarity_matrix, np.nan_to_num(train_matrix,
nan=0))
```

7. Combining Predictions

The script combines the SVD predictions and user-based predictions using weighted averaging. This helps leverage both higher-order patterns (from SVD) and direct similarities (from user-based filtering).

```
predicted_matrix = 0.7 * predicted_matrix_svd + 0.3 * predicted_matrix_user_based
```

8. Threshold Tuning for F1 Score

The script fine-tunes the threshold for predicting positive ratings using the F1 score on the validation set:

- Various thresholds (from 3.0 to 4.5) are tested.
- The threshold with the highest F1 score is selected.

```
if f1 > best_f1:
    best_f1, best_threshold = f1, threshold
```

9. Evaluation on Test Data

The best threshold is applied to the test data to compute the final F1 score, ensuring that the model performs well on unseen data.

```
f1_test = f1_score(test_actual, test_predicted, zero_division=1)
```

10. Finding Similar Users

A helper function identifies the top 5 most similar users for a given user based on the user similarity matrix. This can be useful for explaining recommendations.

```
top_users, scores = top_5_similar_users(user_id, user_similarity_matrix)
```

11. Course Recommendation

The script concludes by recommending the top 5 courses for a specific user based on the predicted ratings. The recommendations are based on the highest predicted ratings for courses the user has not yet rated.

```
top_courses = recommend_courses(user_id, train_matrix, item_similarity_matrix)
```

4.6 Graph-based Recommendation

In this graph-based recommendation system, a hybrid approach is used that combines graph structures, item embeddings, and personalized scoring to recommend products to users based on item similarities and user preferences.

4.6.1 Dataset Import and Preprocessing

Loading Datasets: Two datasets, user_course_ratings_final1_updated.csv and Online_courses_final1_updated.csv, are loaded for analysis.

Data Merging: The datasets are merged on course_id to combine user ratings and course metadata.

Text Processing:

Description fields (Short_Intro and Skills) undergo tokenization, stopword removal, and lemmatization using NLTK.

New processed fields (Short_Intro_processed, Skills_processed) are created.

Encoding Categorical Data: Label encoding is applied to categorical features (user_id, course_id, etc.), converting them into numerical representations.

4.6.2 Graph Creation

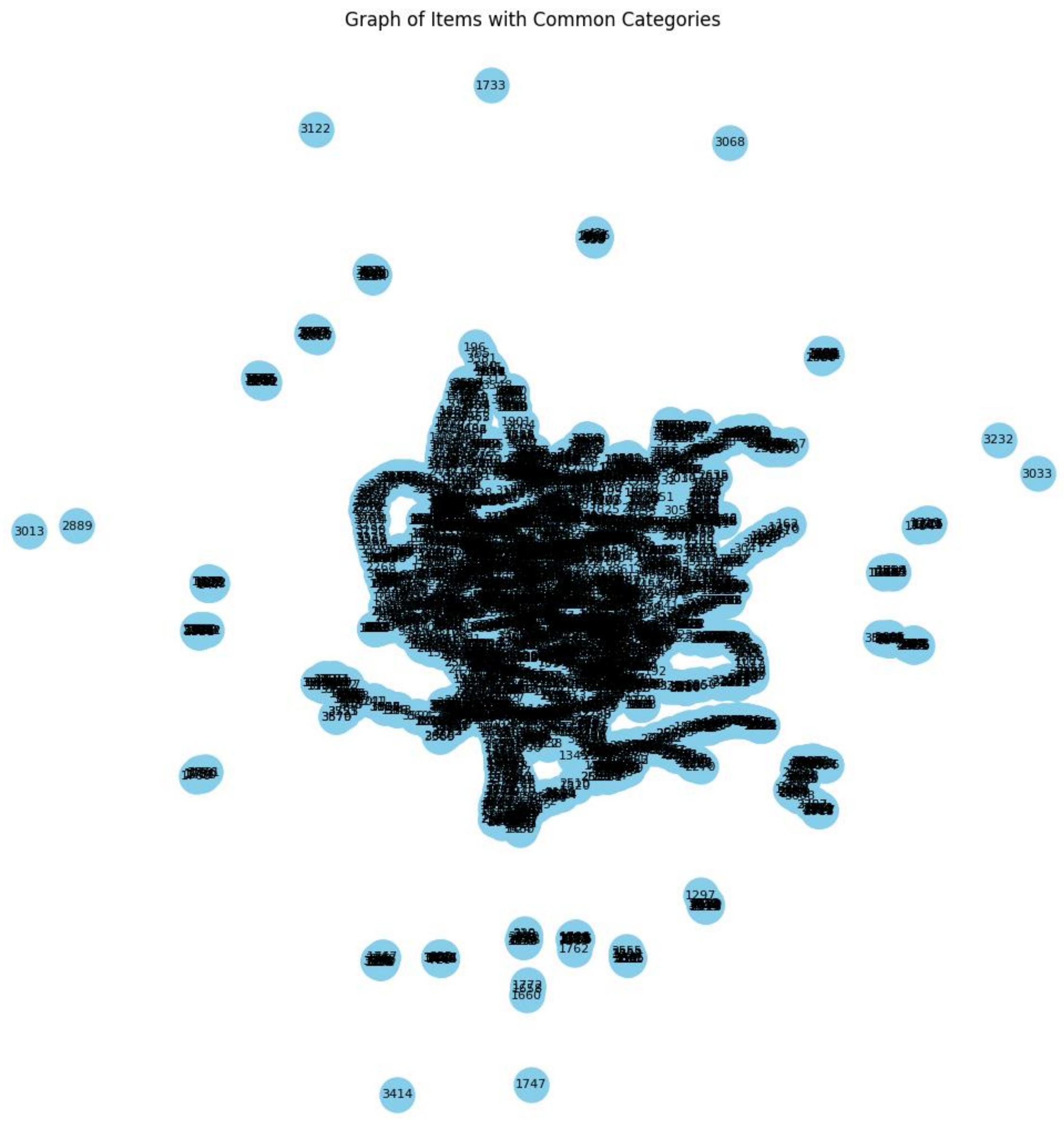
Using NetworkX:

A graph (G) is created, where nodes are courses (course_id_encoded) and edges are formed based on shared categories or correlated ratings.

Pearson correlation is calculated to determine similarity between course ratings, with a threshold for edge creation.

Edge Reduction: The maximum degree of a node is limited to a specific value (max_degree) for simplicity.

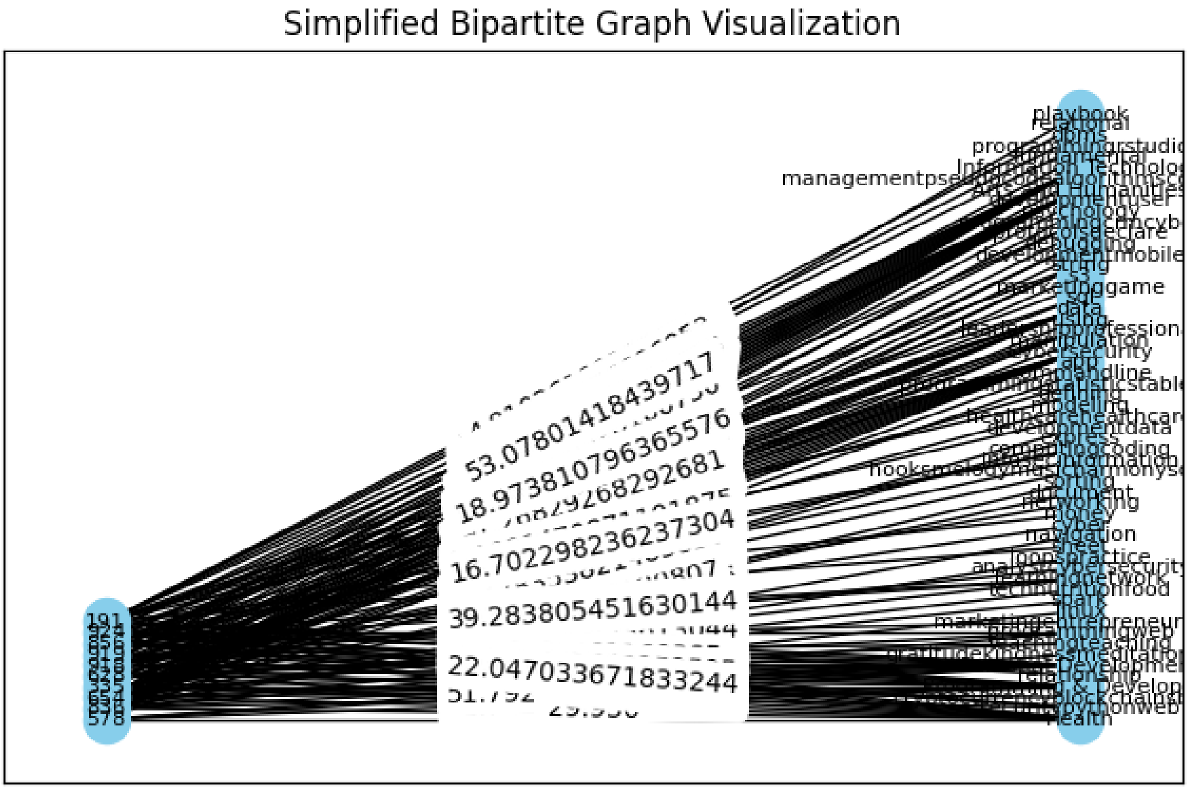
Visualization: Matplotlib is used to draw the graph, displaying relationships among courses based on shared attributes.



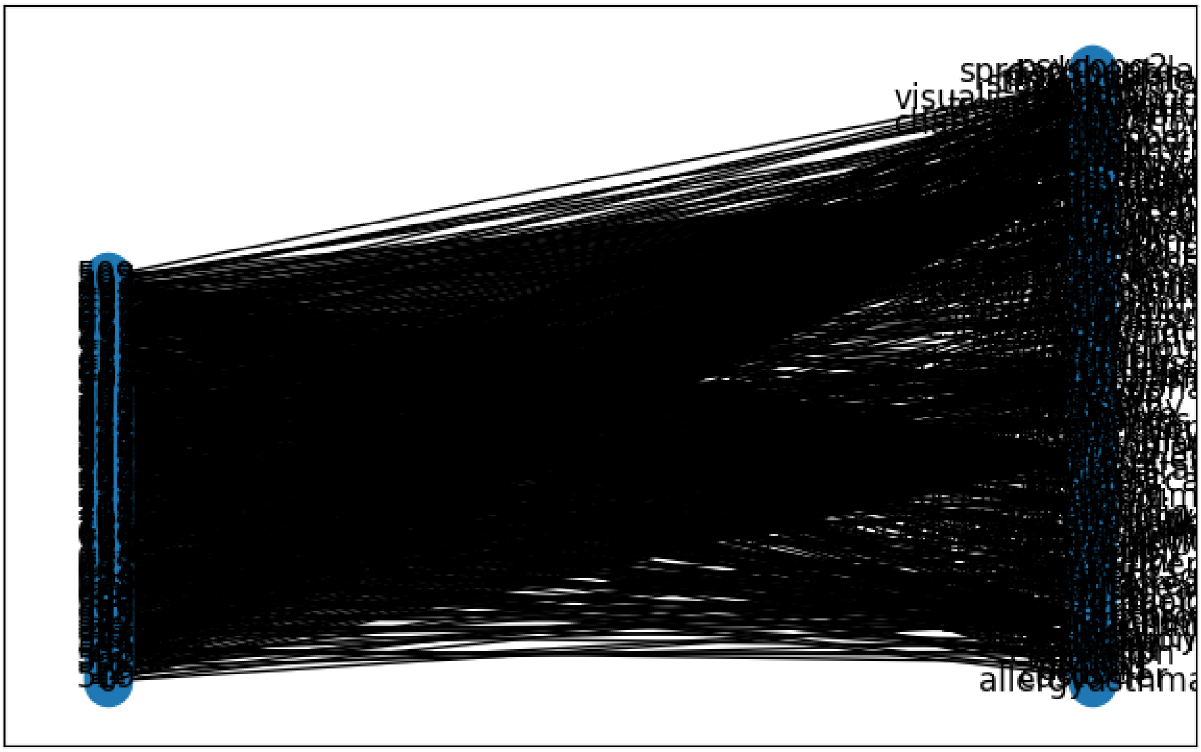
The graph illustrates courses as nodes, with edges representing shared categories. Nodes closer together indicate greater category similarity, forming dense clusters. Sparse nodes on the periphery represent courses with unique or less commonly shared categories.

4.6.3 Bipartite Graph of Users and Skills

Description: This graph captures the relationship between users and the skills they gain by enrolling in courses. Users form one set of nodes, while skills form the other. Edges connect a user to a skill if they have taken a course that imparts that skill. This structure highlights how users interact with skills in the dataset and helps identify popular or niche skills based on the number of users connected to them.



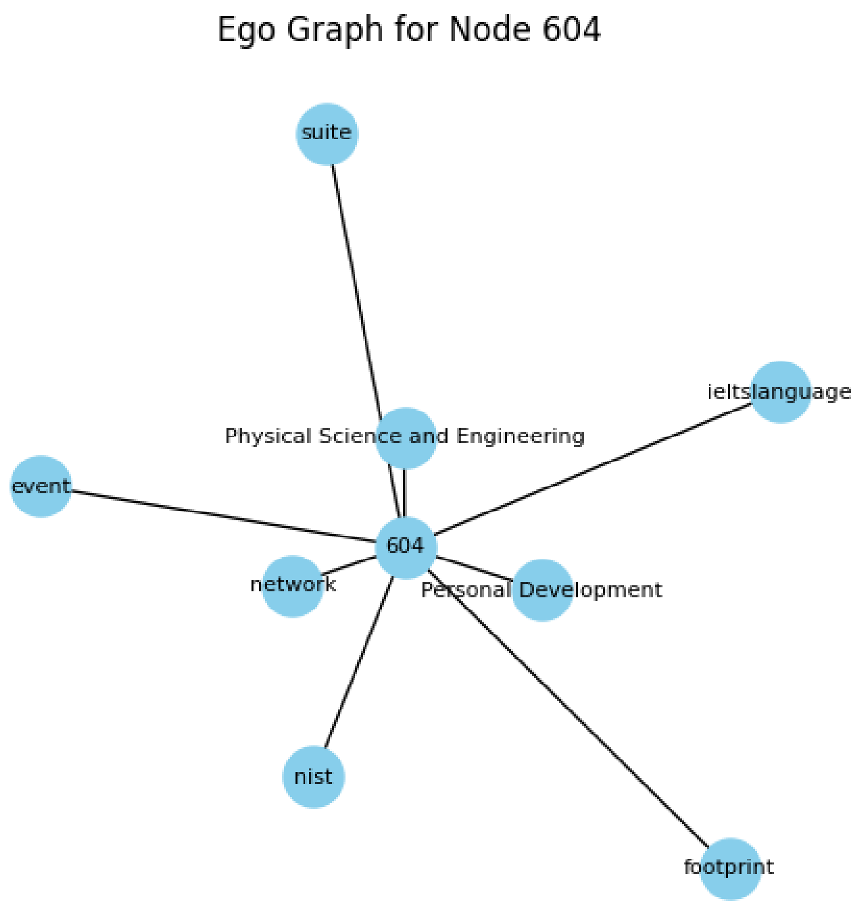
A scaled-down version of the bipartite graph is created for better visualization, showcasing only a small subset of users and skills to clearly demonstrate the user-skill relationships.



The complete version of the graph includes all users and skills in the dataset. It reveals the true complexity of interactions, providing insights into skill demand patterns and user distribution.

4.6.4 Ego Graph for a User

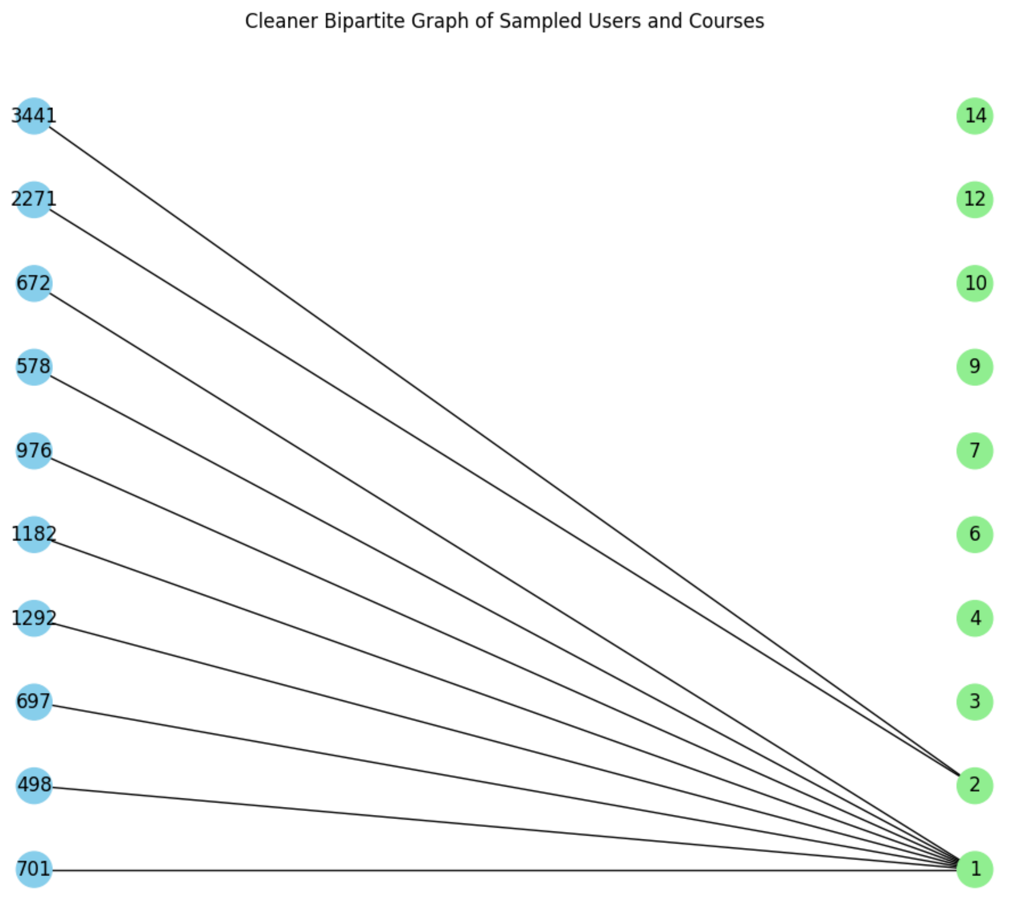
Description: An ego graph is a focused visualization centred on a single user, showing only their immediate connections. In this graph, the central node represents the selected user, and the surrounding nodes represent the skills they have acquired through various courses. The edges connecting the user to the skills highlight their learning journey.



This graph helps isolate and analyse the learning profile of an individual user, showing the breadth or specialization of their acquired skills. It also provides a personalized perspective that can be used for recommendations or user-specific insights.

4.6.5 Bipartite Graph of Users and Courses

Description: This graph represents the enrolment relationships between users and courses. Users and courses form two distinct sets of nodes, and an edge between a user and a course indicates that the user has taken that course. This graph helps visualize how users interact with different courses in the dataset, revealing trends in course popularity and user preferences.



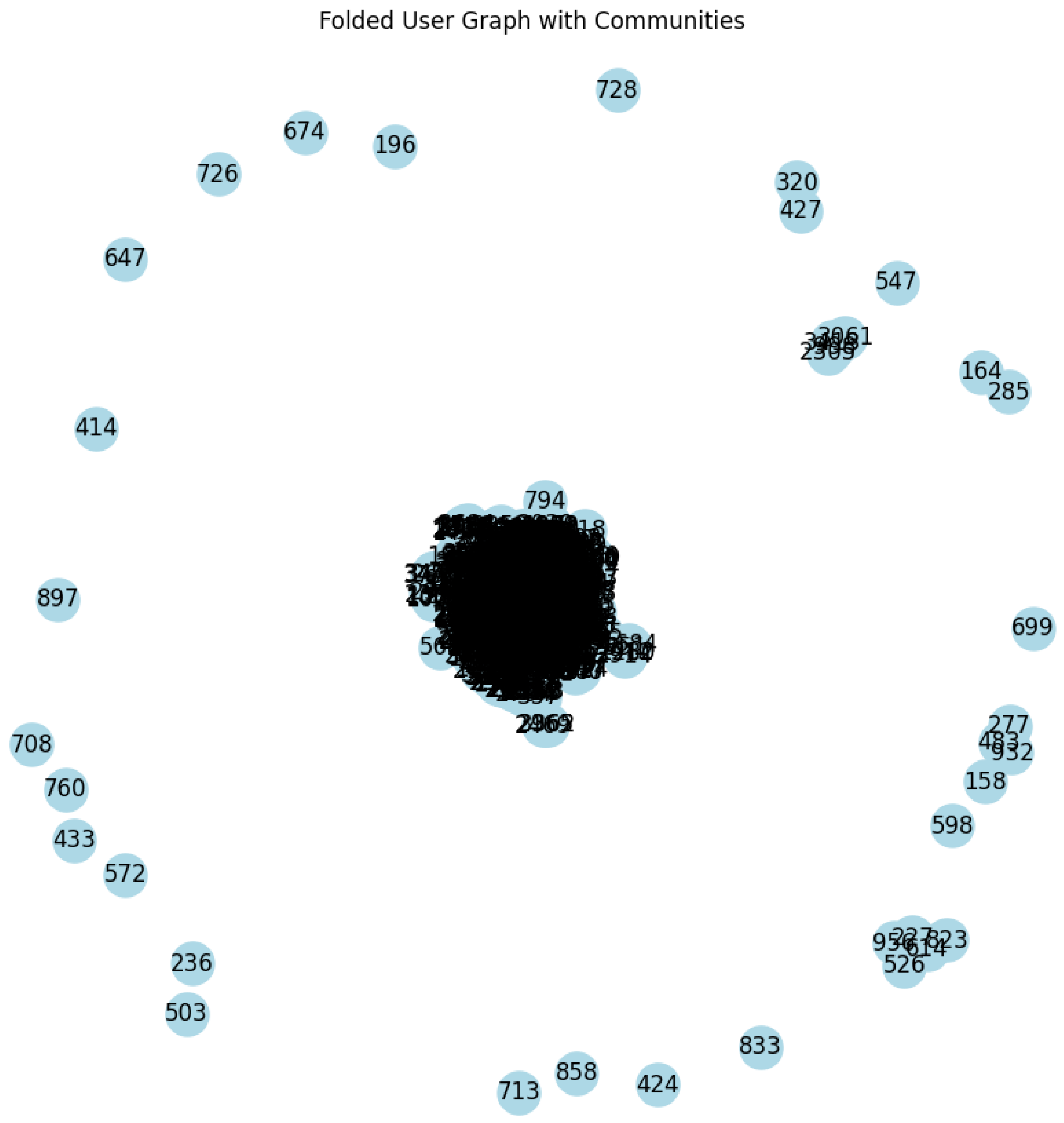
The graph allows for a deeper understanding of course participation patterns, identifying highly subscribed courses or user clusters with similar learning paths. Since this is a cleaned and pruned graph it is able to show a clearer picture for visualisation.

4.6.6 Folded User Graph with Communities

Description: This graph is a projection of the user-course bipartite graph onto the user nodes. In this folded graph, edges between users are weighted by the number of courses they have in common. Users who have taken similar courses are more strongly connected. A threshold (T) is applied to filter out weak edges, leaving only significant relationships.

Community Detection: By applying the Louvain algorithm, the graph is divided into communities, where each community represents a group of users with shared interests or learning habits. These communities provide valuable insights into user clusters based on their course-taking patterns.

Visualization: The folded user graph is visualized with a spring layout, showing nodes (users) clustered into communities. The layout highlights user groupings, making it easier to identify dense clusters and isolated nodes.



This graph provides an in-depth understanding of user behaviour, allowing for targeted recommendations and the identification of trends in user interests. It also showcases how collaborative learning patterns emerge within the dataset.

4.6.7 Recommendation

Random Walk:

A biased random walk on the graph captures proximity-based relationships.

Nodes visited during the walk form the basis for item recommendation.

$$P(u \to v) = \frac{\pi_{uv} \cdot w_{uv}}{\sum_{z \in \mathcal{N}(u)} \pi_{uz} \cdot w_{uz}}$$

Where:

- π_{uv} is the bias factor, defined as:

$$\pi_{uv} = \begin{cases} 1/p, & \text{if } d_{t,v} = 0 \text{ (returning to the source)} \\ 1, & \text{if } d_{t,v} = 1 \text{ (neighboring node)} \\ 1/q, & \text{if } d_{t,v} = 2 \text{ (exploration to farther nodes)} \end{cases}$$

- w_{uv}: Weight of the edge between nodes u and v.
- $\mathcal{N}(u)$: The neighbors of node u.

The random walk generates sequences of nodes W as:

$$W = [v_1, v_2, \dots, v \quad v_{i+1} \sim P(v_i \to v_{i+1})$$

Feature Extraction with CountVectorizer:

A vocabulary is created using product IDs.

Walk sequences are vectorized with CountVectorizer, enabling representation learning for recommendations.

$$\text{Vocabulary} = \{v_1, v_2, \ldots, v_n\}$$

Vectorization is performed using the frequency of co-occurrences of nodes within a fixed context window c. For a sequence W:

$$X_{ij} = \text{Frequency}(v_i, v_j \mid |i - j| \leq c)$$

Where:

- X_{ij}: Entry in the co-occurrence matrix, representing the number of times item v_i appears within the context window of v_j.
- c: The size of the context window.

Recommendation Logic:

Similar items are identified based on embeddings derived from random walks.

$$\text{Similarity}(i, j) = \frac{\vec{X}_i \cdot \vec{X}_j}{||\vec{X}_i||\,||\vec{X}_j||}$$

Where:

- $\vec{X}_i, \vec{X}_j$: Vector embeddings of items i and j derived from random walks.
- $\cdot$: Dot product.
- $||\vec{X}_i||$: Euclidean norm of $\vec{X}_i$.

Top-k Recommendations: To recommend items similar to a given item v, sort all items j by their similarity to v and select the top k:

$$\text{Recommended Items} = \text{arg top-k}_j \ \text{Similarity}(v, j)$$

Chapter – 5

TESTING

5.1 Evaluation Metrics

Root Mean Squared Error (RMSE):

- RMSE is calculated to evaluate the model's performance. It measures the average magnitude of the errors in prediction. It is defined as:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Where:

- y_i is the true rating for course i,
- $\hat{y}_i$ is the predicted rating for course i,
- n is the total number of courses in the test set.

F1 Score:

- The F1 Score is used to measure the model's performance in predicting whether a course will have a high rating (≥ 4). This is a binary classification problem (relevant vs irrelevant). F1 Score is calculated using:

$$\mathrm{F1} = 2 \times \frac{\mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

Where:

- Precision is the proportion of positive identifications that were actually correct.
- Recall is the proportion of actual positives that were correctly identified.

5.2 Errors and Exception Handling

In our project, we encountered and handled the following key errors:

5.2.1 Null and NaN Values:

Null or NaN values in critical columns caused errors during operations like filtering or grouping.

Handling: We replaced missing numeric data with 0 and textual data with empty strings or skipped rows with invalid data. For example:

```
data['rating'] = pd.to_numeric(data['rating'].replace({'\$': '', ',': ''}, regex=True), errors='coerce').fillna(0)
```

5.2.2 Invalid Data Formats:

Columns like Skills_processed contained improperly formatted data, causing SyntaxError or ValueError.

Handling: Used try-except blocks to log and skip invalid entries during processing. For example:

```
try:
    keywords = ast.literal_eval(Skills_processed)
except (SyntaxError, ValueError) as e:
    print(f"Skipping invalid Skills_processed: {Skills_processed}, Error: {e}")
    continue
```

5.2.3 Missing Attributes:

Attempting to access non-existent or improperly renamed columns raised AttributeError or KeyError.

Handling: Added pre-checks to ensure the column existed before processing. For example:

```
if 'rating' in data.columns:
    data['rating_scaled'] = scaler.fit_transform(data[['rating']])
```

5.2.4 Type Mismatch Errors:

Operations requiring specific data types failed due to mismatches.

Handling: Enforced consistent type conversions during processing. For example:

```
data['user_id_encoded'] = data['user_id'].astype('category').cat.codes
```

5.2.5 Visualization Errors:

Graph visualizations failed when nodes or edges were missing.

Handling: Validation checks ensured the graph was non-empty before visualization. For example:

```
if len(G.edges()) > 0:
    nx.draw(G, with_labels=True)
```

These measures ensured smooth execution and improved the system's reliability by addressing potential interruptions effectively.

5.3 Limitations of the Solution

Limitations of Content, Collaborative, and Graph-Based Community Detection Using Louvain Algorithm

5.3.1 Content-Based Limitations

Dependence on Content Quality: Poor or missing metadata/content can lead to inaccurate communities.

Cold Start Problem: Fails to effectively group new or inactive nodes with insufficient content.

Scalability Issues: Processing large-scale textual or attribute data is resource-intensive.

Content Ambiguity: Nodes with similar content but different contexts may be misclassified.

5.3.2 Collaborative-Based Limitations

Data Sparsity: Low interaction density leads to weak or incomplete community detection.

Popularity Bias: Heavily interacted nodes dominate community structures, overshadowing smaller groups.

Cold Start Problem: Struggles with inactive users/nodes or those with no prior interactions.

Overfitting to Existing Patterns: Fails to adapt well to new or changing relationships.

5.3.3 Graph-Based Limitations

Attribute Ignorance: Ignores rich node-specific features, relying only on topology.

Noisy or Dense Graphs: Performance degrades in networks with excessive or random connections.

Resolution Limit: Louvain can miss small communities due to its resolution bias.

Computational Intensity: For very large graphs, Louvain may face scalability and efficiency challenges.

General Limitations of Louvain Algorithm

Deterministic Output: Slight differences in the input order can lead to different results.

Global Optimization Issues: May get stuck in local optima, missing the true modularity maximum.

Resolution Limit: Struggles to identify smaller communities within large networks.

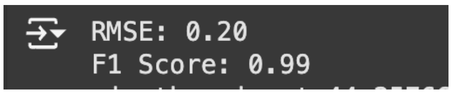
These limitations highlight areas where refinements or complementary approaches could improve performance and applicability.

Chapter-6

FINDINGS, CONCLUSIONS AND FUTURE WORK

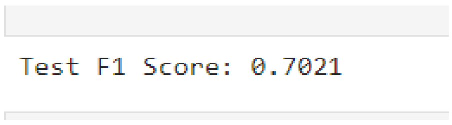
6.1 Findings

For Content-Based Filtering:



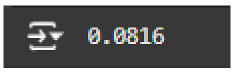
```
RMSE: 0.20
F1 Score: 0.99
```

For collaborative filtering:



```
Test F1 Score: 0.7021
```

For Graph-based recommendation:

F1 Score: 

```
0.0816
```

6.2 Comparison

6.2.1 Content-Based

- Pros:
  - Can work even with sparse network connections.
  - Leverages rich metadata for detailed insights.
  - Useful in personalized recommendations.

- Cons:
  - Requires high-quality content or features.
  - Struggles when content is noisy or missing.
  - Limited by the "cold start" problem for new nodes.

6.2.2 Collaborative-Based

- Pros:
  - Captures implicit relationships through interactions.
  - Effective even with minimal node attributes.
  - Scales well with large datasets.

- Cons:
  - Requires sufficient interaction data.
  - Suffers from sparsity in low-activity networks.
  - Vulnerable to bias from dominant user behaviors.

6.2.3 Graph-Based

- Pros:
  - Works on purely structural data.
  - Can detect hidden communities through topology.
  - Applicable to diverse networks without node attributes.

- Cons:
  - Ignores node-specific features (content, attributes).
  - May struggle in dense or noisy graphs.
  - Computationally intensive for large networks.

6.3 Conclusion

Our project successfully demonstrated the integration of content-based, collaborative, and graph-based methods to build a robust and dynamic course recommender system. By leveraging the strengths of each technique, we addressed key challenges such as data sparsity, limited personalization, and evolving user preferences. The graph-based approach further enriched our system by uncovering complex relationships and user communities, enabling highly targeted recommendations. Moving forward, scalability, real-time data integration, and enhanced user feedback mechanisms will be pivotal in refining and expanding the system to cater to a wider audience and evolving trends in e-learning.

6.4 Future Work

6.4.1 Interactive Course Search and Filtering:

- Users will be able to search for courses based on various attributes, such as skills, categories, instructors, or platforms.
- Advanced filtering options will help narrow down choices based on duration, ratings, or specific learning objectives.

6.4.2 Personalized Recommendations:

- The system will provide tailored course suggestions derived from graph-based insights and user preferences.
- Recommendations will adapt dynamically to a user's evolving interests and learning history.

6.4.4 Visualization of Course Relationships:

- Users can view graphical representations of how courses, skills, instructors, and categories are interconnected, helping them make informed decisions.

6.4.5 User Profiles:

- The application will include user accounts where individuals can save their preferences, track their learning progress, and receive customized course updates.

6.4.6 Ensuring Scalability and Continuous Improvement

- The application will be designed to accommodate a growing number of users and datasets, it remains relevant and effective as new courses and trends emerge.
- User feedback will play a key role in refining recommendations and enhancing the overall user experience.

6.4.7 Finetuning the graph-based algorithm

- Future work involves refining the Louvain algorithm by optimizing modularity calculations and enhancing the handling of weighted edges to better capture community structures. Adjustments in edge weight thresholds will ensure more meaningful relationships are represented, leading to improved detection quality.

- Additionally, scalability improvements, such as parallelizing computations, will enable the model to handle larger datasets and more complex relationships efficiently. These enhancements aim to boost prediction accuracy and provide more precise recommendations in dynamic e-learning scenarios.

REFERENCES

[1] Atef, Khaled, "Online Courses Dataset," Kaggle, 2023. [Online]. Available: https://www.kaggle.com/datasets/khaledatef1/online-courses/data. [Accessed: Nov. 19, 2024].

[2] Li, J. and Z. Ye, "Course Recommendations in Online Education Based on Collaborative Filtering Recommendation Algorithm," Dec. 28, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1155/2020/6619249

[3] M. Shimpi, "recommendation_system," GitHub, [Online]. Available: https://github.com/mayurishimpi/recommendation_system. [Accessed: 20-Nov-2024].

[4] Neo4j, "Neo4j documentation," 2023. [Online]. Available: https://neo4j.com/docs. [Accessed: Nov. 19, 2024].

[5] Robinson, J. Webber, and E. Eifrem, *Graph Databases: New Opportunities for Connected Data.* Sebastopol, CA, USA: O'Reilly Media, 2015.

[6] Sinclair, K. A., J. Callahan, S. Donovan, and L. Beckett, “Knowledge Graph-based Multimodal GNN Model for Recommendation Systems,” Research Square, preprint, version 1, Nov. 2024. [Online]. Available: https://doi.org/10.21203/rs.3.rs-3969137/v1