UnifyLearn : A Intelligent Cross-Platform Course Discovery Engine
Group No: 18

Gour Krishna Dey
Department of Computer Science
IIIT-Delhi
New Delhi, India
gour24035@iiitd.ac.in

Parul
Department of Computer Science
IIIT-Delhi
New Delhi, India
parul24128@iiitd.ac.in

Abstract

UnifyLearn is an AI-powered unified course search system that interprets natural language queries to retrieve and standardize learning content from multiple platforms such as Coursera, Udemy, Simplilearn, and FutureLearn. It integrates structured and unstructured data, dynamically generates database queries, enriches results using LLM-based metadata completion and skill extraction, and delivers context-aware, comprehensive learning recommendations through a universal schema.

1 Introduction

In today's rapidly evolving digital education landscape, learners face the challenge of navigating multiple platforms to find relevant and high-quality courses. Each platform maintains distinct data formats, metadata, and search mechanisms, making cross-platform discovery inefficient and inconsistent. UnifyLearn addresses this gap by providing an intelligent unified course search system that interprets natural language queries and retrieves structured, enriched results from diverse sources. By combining database-driven search with large language model (LLM) reasoning, UnifyLearn ensures context-aware understanding, automatic metadata completion, and standardized presentation. This enables users to efficiently explore comprehensive learning opportunities across platforms through a single, intuitive interface.

2 Motivation and Objectives

2.1 Project Objectives

The primary objective of UnifyLearn is to design and implement an intelligent unified course search platform capable of understanding and processing natural language queries to deliver comprehensive and standardized learning content. The system integrates both structured and unstructured data sources to ensure context-aware and complete search results.

Key objectives include:

1. Enable users to search across multiple course providers (Coursera, Udemy, Simplilearn, and FutureLearn) using a single natural language query.
2. Automatically detect the query type (e.g., category-based, skill-based, or technology-based).
3. Dynamically generate and execute structured database queries for relevant course retrieval.
4. Employ LLM-based enrichment to fill missing metadata fields, extract skills, and enhance contextual relevance.
5. Present unified results in a standardized universal schema across all platforms.

2.2 Motivation

Online learning platforms contain vast amounts of data organized in diverse and inconsistent formats, making unified access challenging. Currently, no system enables seamless natural language querying across multiple course sources simultaneously. UnifyLearn aims to bridge this gap by combining the precision of structured database querying with the flexibility of LLM-based reasoning. This approach aligns with emerging research in federated querying and LLM-assisted data integration, offering both innovation and practical value.

3 Data Collection and Schema

3.1 Data Sources

We utilized the Kaggle dataset titled *"Online Course Dataset (EdX, Udacity, Coursera)"* [1] as the

[1] https://www.kaggle.com/datasets/nomanturag/online-course-datasetedx-udacity-coursera

primary data source, which contains course information from multiple e-learning platforms. From this unified dataset, course records corresponding to Coursera, FutureLearn, Udacity, and Simplilearn were extracted and filtered using platform identifiers and metadata attributes.

The curated data was segregated by platform and stored in four independent MongoDB clusters — `Coursera_DB`, `FutureLearn_DB`, `Udacity_DB`, and `Simplilearn_DB`. This design supports modular scalability and enables efficient querying from the respective clusters based on user requests.

Data curation involved:

- Removing duplicate and incomplete entries.
- Mapping provider-specific fields to a unified schema.
- Handling missing or inconsistent data using heuristic and LLM-based enrichment methods.
- Ensuring consistency and interoperability across all databases for seamless integration.

All databases are maintained in distinct MongoDB clusters as per the project requirements, with carefully segregated schema structures to introduce variability and support flexible query federation across platforms.

3.2 Database Schema

Each course provider maintains a dedicated MongoDB cluster designed to support modular scalability and flexible schema adaptation. The databases—`Coursera_DB`, `Simplilearn_DB`, `Udacity_DB`, and `FutureLearn_DB`—store structured and semi-structured course data with provider-specific fields mapped to a unified schema for integration and querying.

Database: Coursera_DB

- `Title` (String)
- `URL` (String)
- `Short Intro` (Text)
- `Category` (String)
- `Sub-Category` (String)
- `Course Type` (String)
- `Language` (String)
- `Subtitle Languages` (String)
- `Skills` (Text)
- `Instructors` (Text)
- `Rating` (String)
- `Number of Viewers` (Integer)
- `Duration` (String)
- `Site` (String)

Database: Udacity_DB

- `Title` (String)
- `url` (String)
- `Short Intro` (Text)
- `Duration` (String)
- `Site` (String)
- `Program Type` (String)
- `Level` (String)
- `Prerequisites` (Text)
- `What You Learn` (Text)

Database: FutureLearn_DB

- `Title` (String)
- `link` (String)
- `Short Intro` (Text)
- `Duration` (String)
- `Site` (String)
- `Courses` (String)
- `Topics Related to CRM` (Text)
- `ExpertTracks` (String)
- `Course Title` (String)
- `Course URL` (String)
- `Course Short Intro` (Text)
- `Weekly Study` (String)

- `Premium Course` (String)

Database: Simplilearn_DB

  - `Course` (String)
  - `URL` (String)
  - `Short Intro` (Text)
  - `Category` (String)
  - `Subtitle Languages` (String)
  - `Skills Covered` (String / Integer)
  - `Instructors` (String)
  - `Site` (String)

All four schemas were designed to retain platform-specific richness while mapping to a unified metadata structure. This modular schema separation across distinct MongoDB clusters ensures high flexibility, scalability, and efficient cross-platform query federation within the UnifyLearn system.

4 Methodology

4.1 Query Types

UnifyLearn supports federated text-based queries across all four MongoDB clusters, combining structured filtering with LLM-powered semantic interpretation. Currently, the system performs best on SPJ (Select–Project–Join) queries, while aggregate queries are not fully implemented yet.

4.2 Sample Queries

UnifyLearn currently supports federated SPJ (Select–Project–Join) queries across all four MongoDB clusters, combining structured filtering with LLM-powered semantic interpretation. These queries allow users to retrieve courses based on category, duration, skills, and platform.

Example SPJ queries:

- Retrieve courses with the highest ratings across all platforms.
- Find Data Science courses with a duration of less than 3 months.
- List Cloud courses covering both AWS and Azure from the Udacity platform.
- Fetch Data Science courses requiring Python and R programming skills.

Note: Queries involving aggregation or ranking, such as "Top 10 highest-rated courses from all platforms," are not fully supported yet and will be implemented in future phases.

4.3 SQL Queries

Example 1: *"Find top 10 highest-rated courses across all platforms."*

Generated MongoDB Query:

```
Enter your query (or 'quit' to exit): Top 10 highest rated courses from all platforms

Processing query: 'Top 10 highest rated courses from all platforms'
--------------------------------------------------
DEBUG: Raw LLM response: ```json
{
  "query_type": "AGGREGATE",
  "description": "Find the top 10 highest rated courses from all platforms",
  "providers": {
    "coursera": {
      "$sort": {
        "Average Rating": -1
      },
      "$limit": 10
    },
    "udacity": {
      "$sort": {
        "Average Rating": -1
      ...
Query Type: AGGREGATE
Description: Find the top 10 highest rated courses from all platforms
--------------------------------------------------

Provider: COURSERA
LLM Generated Query (Schema Fields):
{
  "$sort": {
    "Average Rating": -1
  },
  "$limit": 10
}

Translated Query (Database Fields):
{
  "$sort": {
    "Rating": -1
```

Example 2: *"List Cloud courses covering both AWS and Azure from the Udacity platform."*

Generated MongoDB Query:

```
Enter your query (or 'quit' to exit): Cloud courses that cover both AWS and Azure from Udacity Platform

Processing query: 'Cloud courses that cover both AWS and Azure from Udacity Platform'
--------------------------------------------------
DEBUG: Raw LLM response: ```json
{
  "query_type": "SPJ",
  "description": "Find cloud courses covering both AWS and Azure from Udacity",
  "providers": {
    "udacity": {
      "$and": [
        {
          "$or": [
            {"Skills Covered": {"$regex": "\\bAWS\\b", "$options": "i"}},
            {"Course Title": {"$re...
Query Type: SPJ
Description: Find cloud courses covering both AWS and Azure from Udacity
--------------------------------------------------

Provider: UDACITY
LLM Generated Query (Schema Fields):
{
  "$and": [
    {
      "$or": [
        {
          "Skills Covered": {
            "$regex": "\\bAWS\\b",
            "$options": "i"
          }
        },
        {
          "Course Title": {
            "$regex": "\\bAWS\\b",
            "$options": "i"
```

Explanation: The model first identifies entities (e.g., "Data Science", "Cloud", "AWS", "Azure") and schema fields (e.g., `Category`, `Skills`, `Duration`, `Rating`) using semantic parsing. These are then mapped to their corresponding MongoDB collections and attributes through metadata alignment.

Cross-platform queries, such as retrieving "Top 10 highest-rated courses across all providers," are handled using federated queries across multiple MongoDB clusters. Future phases will extend this framework to support advanced cross-cluster aggregation and ranking pipelines for more comprehensive analytics.

5 System Architecture

5.1 Current Architecture: Virtualization-Based Federated Querying

The current implementation of the system follows a **virtualization-based federated query architecture**. In this design, user queries are executed in real time across multiple MongoDB clusters, without pre-storing or caching intermediate results. This ensures that data remains up-to-date and consistent across platforms such as Coursera, Udacity, and EdX.

The choice of a virtualization approach was driven by several key factors:

- **Lightweight and easy to implement:** No need for maintaining redundant data across clusters.
- **Compatibility with MongoDB:** Works efficiently with MongoDB's flexible JSON schema and nested document structures.
- **Dynamic data handling:** Ideal for frequently changing fields such as ratings, views, and course costs.

5.2 Planned Upgrade: Hybrid Architecture (Virtualization + Materialization)

While the current prototype uses a fully virtualized setup, the next stage of development aims to transition toward a **hybrid architecture** that combines both *virtualization* and *materialization* for optimal efficiency and scalability.

The planned hybrid design includes:

- **Virtualization Layer** – To manage dynamic fields like ratings, views, and pricing, ensuring real-time freshness and up-to-date insights.
- **Materialization Layer** – To store relatively static metadata such as course titles, instructors, categories, and skill mappings, reducing redundant computations and lowering query latency.

This hybrid approach will achieve a balanced trade-off between **data freshness** and **query performance**, making it better suited for large-scale or production-level deployment. Currently, the implemented prototype operates purely on the virtualization model, with the hybrid architecture planned as part of future enhancements.

6 Algorithm

The core federated query processing pipeline of **UnifyLearn** combines LLM-assisted query understanding with dynamic query translation and execution across multiple MongoDB clusters. The system operates in real time without materialization, ensuring freshness of data.

6.1 Overall Workflow

Input: user_query (text)
Output: federated_response (JSON)

1. Parse the `user_query` using an LLM to extract *intent*, *filters*, and *keywords*.
2. Map extracted terms to universal schema fields using a global `FIELD_MAPPING` dictionary.
3. Identify the relevant MongoDB clusters (e.g., Coursera, Udacity, Simplilearn, FutureLearn).
4. For each cluster:
   (a) Construct a MongoDB filter or aggregation pipeline using the schema.
   (b) Fetch matching records via database connectors or APIs.
5. Merge all retrieved datasets into a unified response.
6. Rank or aggregate results depending on query type (SPJ or Aggregate).
7. Return a structured JSON response to the frontend.

6.2 Federated Query Processor Pseudocode

```
Algorithm FederatedQueryProcessor(user_query):

    # Step 1: Extract Query Intent using LLM
    parsed_query = LLM.generate_content(prompt = build_prompt_with_schema(user_query))
    query_json = extract_balanced_json(parsed_query)

    # Step 2: Determine Query Type (SPJ or Aggregate)
    query_type = query_json["query_type"]

    # Step 3: Iterate over target providers
    for provider in query_json["providers"]:
        provider_query = query_json["providers"][provider]["query"]

        # Translate schema fields to actual DB fields
        translated_query = translate_query_to_db_fields(provider_query, provider)

        # Step 4: Execute Query on respective MongoDB cluster
        results[provider] = execute_mongo_query(cluster=provider, query=translated_query)

    # Step 5: Combine and Standardize Results
    unified_results = merge_and_standardize(results)

    # Step 6: Perform Ranking or Aggregation (if needed)
    if query_type == "AGGREGATE":
        unified_results = perform_cross_cluster_aggregation(unified_results)

    return unified_results
```

Figure 1: Federated Query Processor Pseudocode

6.3 Query Decomposer Logic

The **Query Decomposer** module decomposes a natural language query into multiple platform-specific subqueries. It uses the Gemini-based LLM model for semantic parsing and generates structured MongoDB query objects.

Key Steps:

- Identify whether the query is SPJ (Select–Project–Join) or Aggregate.
- Generate JSON-formatted MongoDB filters for each provider.
- Use schema samples (retrieved via `getSchemasAndSamples()`) to align query fields.
- Translate logical schema fields to physical database fields using `translate_query_to_db_fields()`.
- Return the fully formed query object to the execution engine.

7 Summary and Future Work

This algorithm enables dynamic, schema-aware query generation across heterogeneous data sources. The LLM-based query builder interprets user queries, while the translator ensures provider-specific accuracy. The federated processor unifies all responses into a structured JSON output, supporting seamless cross-platform course discovery.

7.1 Current Work

We have implemented a federated query system across four MongoDB clusters (Coursera, FutureLearn, Udacity, Simplilearn) using a unified schema. The system performs well on SPJ queries, allowing structured filtering and semantic expansion via the Google Gemini AI Studio API. Queries are parsed, mapped to the global schema, executed across clusters, and returned as consistent, enriched JSON responses.

7.2 Future Work

Planned improvements include:

- **Distributed Query Execution:** Concurrently send decomposed queries to multiple clusters for faster results.
- **Aggregation Support:** Handle advanced queries such as top-rated courses, average ratings, and popularity metrics across clusters.
- **Enhanced Query Analyzer:** Incorporate unstructured data and semantic search for queries combining free-text descriptions and structured filters.
- **Hybrid Architecture:** Materialize relatively static fields (titles, instructors, skills) while virtualizing dynamic fields (views, ratings, price) to optimize freshness and performance.

These enhancements will enable support for both SPJ and aggregate queries, cross-platform analytics, and richer results combining structured and unstructured data.