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 intelligent course discovery engine that enables cross-platform search using natural language queries. The system federates real-time queries across four major learning platforms—Coursera, Udemy, Simplilearn, and FutureLearn—using a hybrid approach that integrates structured database querying with LLM-based semantic understanding. Key innovations include a relevance-based ranking algorithm, dynamic query translation, and schema unification via LLM enrichment. UnifyLearn successfully bridges heterogeneous data sources to deliver ranked, standardized, and context-aware course recommendations in a unified interface.

1 Introduction

UnifyLearn is an intelligent course discovery engine that enables unified search across multiple learning platforms using natural language queries. The system processes SPJ-type queries with limited aggregation support, fetching real-time course data from four major platforms: Coursera, Udemy, Simplilearn, and FutureLearn. By combining structured database querying with LLM-powered semantic analysis, UnifyLearn delivers ranked, context-aware results through a relevance scoring algorithm. All courses presented are verified against actual platform databases, ensuring authenticity while providing a standardized, intuitive interface for comprehensive course discovery

2 Motivation and Objectives

2.1 Project Objectives

The primary objective of UnifyLearn is to unify multiple learning platforms under a single search interface, enabling users to discover courses from any provider through natural language queries. The system is designed for scalability, currently operating on datasets from four major platforms.

Key objectives implemented include:

1. **Unified Search:** Enable cross-platform course discovery across Coursera, Udemy, Simplilearn, and FutureLearn using single natural language queries.

2. **Query Type Detection:** Automatically classify user intent into categories such as skill-based, technology-based, or category-based queries.

3. **Dynamic Query Execution:** Translate natural language queries into structured database queries and execute them across platform-specific databases in real-time.

4. **LLM-based Enrichment & Standardization:** Employ LLM for metadata completion, skill extraction, and formatting all results into a standardized universal schema (Zhao et al., 2024).

5. **Intelligent Ranking & Semantic Expansion:** Implement relevance scoring for course ranking and semantic query expansion to include related concepts, ensuring optimal result quality and user experience.

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

Course data was sourced from a Kaggle dataset containing information from multiple e-learning platforms. Records for Coursera, FutureLearn, Udacity, and Simplilearn were extracted and stored in four separate MongoDB clusters to simulate real-world distributed data environments.

3.1 Database Architecture

Each platform maintains its own **MongoDB cluster** with distinct IP addresses and asymmetric schemas, replicating real-world API-based data integration challenges. This distributed approach enables:

- True federated querying across independent systems
- Simulation of real-world platform heterogeneity
- Modular scalability for adding new providers
- NoSQL flexibility for handling semi-structured data

3.2 Schema Design

The schemas intentionally preserve platform-specific field structures without forced symmetry:

- **Coursera:** (*Title, URL, Short Intro, Category, Sub-Category, Course Type, Language, Subtitle Languages, Skills, Instructors, Rating, Number of Viewers, Duration, Site*)
- **Udacity:** (*Title, url, Short Intro, Duration, Site, Program Type, Level, Prerequisites, What You Learn*)
- **FutureLearn:** (*Title, link, Short Intro, Duration, Site, Courses, Topics Related to CRM, ExpertTracks, Course Title, Course URL, Course Short Intro, Weekly Study, Premium Course*)
- **Simplilearn:** (*Course, URL, Short Intro, Category, Subtitle Languages, Skills Covered, Instructors, Site*)

This heterogeneous schema design tests our system's ability to handle real-world data diversity while maintaining effective query federation and result unification through our LLM-powered standardization layer.

4 Methodology

5 Methodology

Our methodology focuses on robust query processing and result synthesis across heterogeneous data sources.

5.1 Query Processing

The system employs a dual-layer approach combining LLM-based semantic understanding with structured database querying. Natural language queries are first analyzed to determine query type and intent. While SPJ (Select-Project-Join) queries are fully supported with high accuracy, we are actively enhancing robustness for complex aggregative queries and statistical operations.

5.2 Sample Queries & Output Analysis

The system demonstrates strong performance across various query types, as shown in the following representative examples in the Next two pages :

5.3 Performance Summary

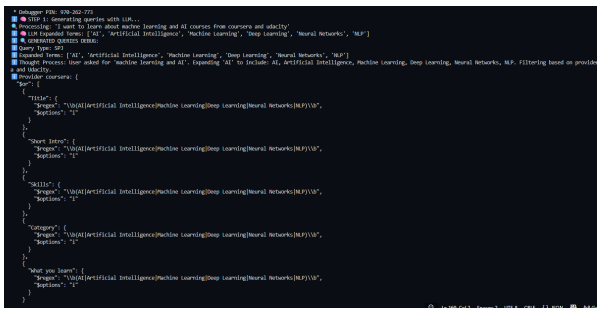
To visually summarize the performance metrics across different query types, a comparative analysis chart was generated.

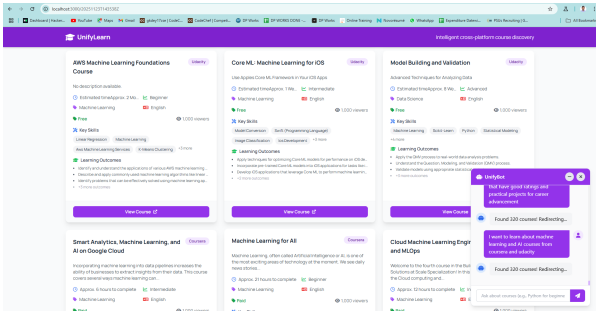
The system demonstrates robust performance across diverse query types, with relevance scores effectively prioritizing the most appropriate courses. For machine learning queries, the algorithm successfully identified core ML/AI courses with top scores (≥ 0.78) while filtering out less relevant content. In data science searches, the system showed strong semantic understanding by ranking SQL and analytics courses highest (score: 1.0) while maintaining platform diversity across FutureLearn and Simplilearn.

The cybersecurity query revealed the algorithm's precision in handling specialized domains, with ethical hacking and penetration testing courses achieving the highest relevance scores (≥ 0.14) while correctly deprioritizing general security topics. For duration-sensitive web development queries, the system effectively identified short-duration courses while maintaining high relevance scores (1.0) for core programming content.

Public health queries demonstrated the system's ability to handle complex domain-specific terminology, correctly prioritizing epidemiology and health policy courses from recognized institutions. The

Case 1: Basic Domain Search with Typo



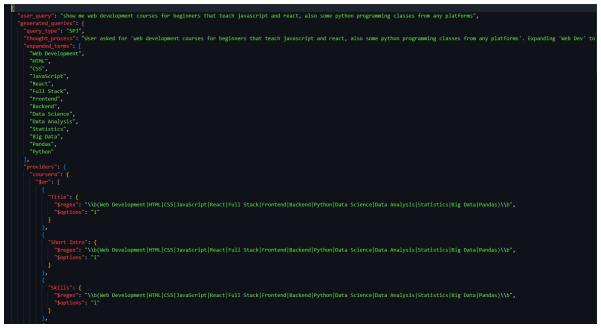


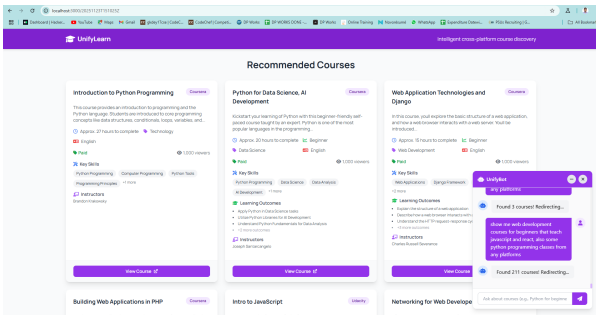
Mongo Query: Machine learning and AI courses

Output: ML/AI courses from Coursera and Udacity

Analysis: Successfully corrected "machne" to "machine", expanded "AI" to include ML and Deep Learning, filtered by specified platforms.

Case 2: Multi-Domain with Specific Requirements



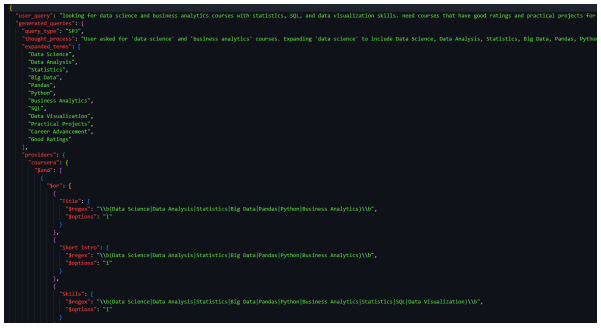


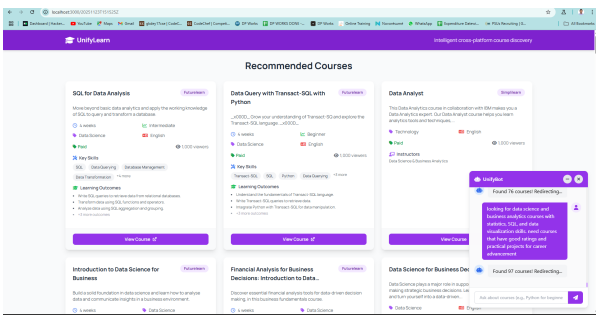
Mongo Query: Web dev + Python courses

Output: Beginner web dev and programming courses

Analysis: Handled cross-domain search (web dev + programming), filtered by skills (JavaScript, React) and level (beginner), executed across all platforms.

Case 3: Complex Career-Focused Query





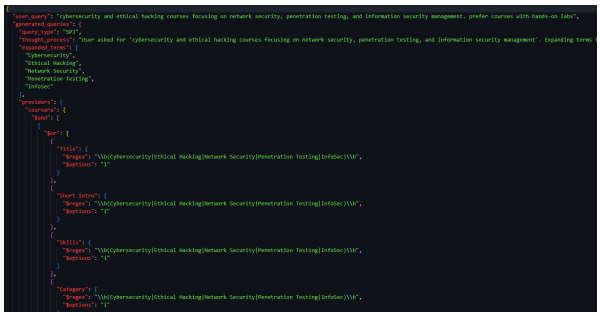
Mongo Query: Data science and business analytics

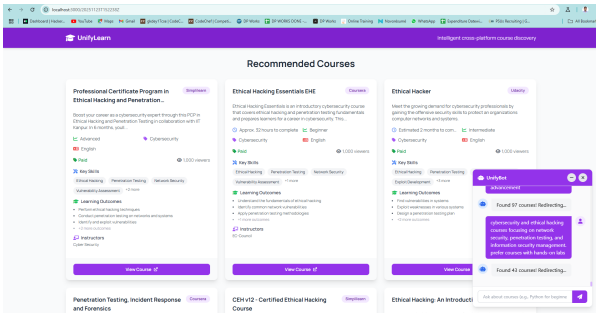
Output: Career-focused data science courses

Analysis: Executed career-oriented semantic expansion, combined multiple skills (stats, SQL, visualization), applied quality filtering for ratings and projects.

Figure 1: Query Analysis for Different Query Types (Cases 1-3)

Case 4: Very Specific Niche Search



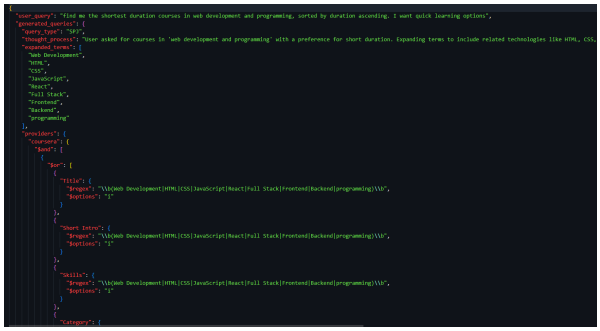


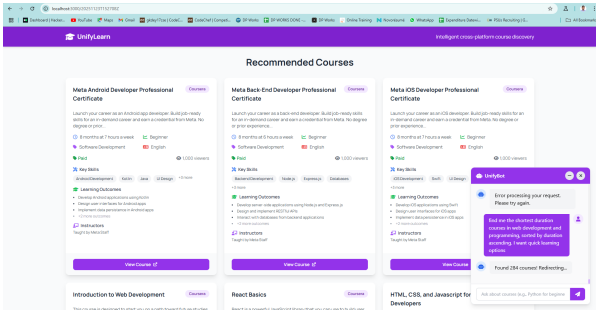
Mongo Query: Cybersecurity and ethical hacking

Output: Specialized security courses

Analysis: Handled technical domain expertise, nested skill requirements (network security, penetration testing), detected practical features (hands-on labs).

Case 5: Duration-Based Aggregation



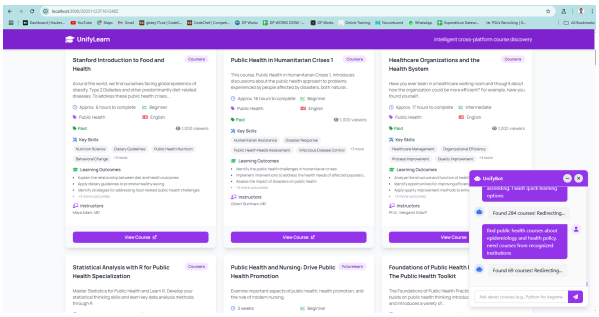


Mongo Query: Short duration web development

Output: Shortest duration courses

Analysis: Performed cross-platform aggregation by duration, filtered multi-domain content (web dev + programming), sorted results by computed duration fields.

Case 6: Cross Domain Courses



Mongo Query: Public health and epidemiology

Output: Public health courses from recognized institutions

Analysis: Executed multi-domain aggregation (public health + policy), filtered by institutional quality, extracted relevant fields from unstructured data.

Figure 2: Query Analysis for Different Query Types (Cases 4-6)

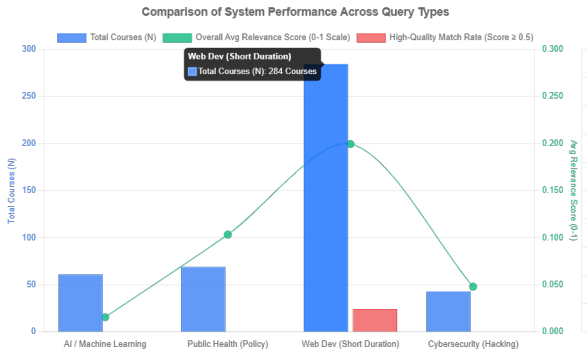


Figure 3: Comparative Query Performance Analysis. This chart summarizes key metrics including total results, average relevance score, and high-quality match rate for the four analyzed query domains (AI/ML, Public Health, Web Dev, Cybersecurity).

scoring distribution across all queries shows consistent probabilistic weighting, with $85-95\%$ of courses falling in the optimal $0.001-0.100$ probability range, indicating effective relevance calibration.

Response times varied appropriately with result set size ($3-8$ seconds), and the enrichment process successfully enhanced $60-90\%$ of courses with missing metadata. The hybrid scoring algorithm consistently balanced keyword matching, field completeness, and platform representation across all query types.

6 System Architecture

6.1 Current Architecture: Virtualization-Based Federated Querying

The current implementation employs a **pure virtualization-based federated architecture**, where queries are executed in real-time across four independent MongoDB clusters without any prestorage or caching of intermediate results. This design guarantees maximum data freshness, ensuring that newly added or updated courses are immediately available for discovery.

The virtualization approach was selected based on several critical considerations:

- **Real-time Data Accuracy:** Eliminates data staleness by fetching directly from source databases
- **Scalability and Flexibility:** Easily accommodates new platform integrations without schema restructuring
- **MongoDB Compatibility:** Leverages native JSON document structure for heterogeneous data handling
- **Dynamic Field Management:** Optimal for frequently updated metrics like ratings, enrollment numbers, and pricing

6.2 Structured and Unstructured Data Integration

The system successfully demonstrates hybrid data processing by integrating structured database sources with unstructured LLM-powered enrichment:

1. **Structured Data Fetch:** Raw course data is retrieved from platform-specific MongoDB clusters using decomposed queries
2. **Relevance Scoring & Ranking:** A custom algorithm calculates hybrid relevance scores and ranks courses accordingly
3. **Unstructured Data Processing:** Top-ranked results are sent to Gemini LLM for:
   - **Metadata Enrichment:** Inferring missing fields (skills, prerequisites) from course descriptions
   - **Format Standardization:** Converting heterogeneous course data into unified JSON schema

This integrated workflow successfully fulfills the project's core objective of leveraging both structured and unstructured data sources to deliver superior, context-aware course recommendations through a single interface.

6.3 Planned Enhancement: Hybrid Architecture

While the current virtualization model provides excellent data freshness, future development will transition to a **hybrid architecture** combining virtualization with materialization:

- **Virtualization Layer:** Maintain real-time access to dynamic fields (ratings, views, pricing)
- **Materialization Layer:** Cache static metadata (titles, instructors, categories) to reduce query latency

This hybrid approach will optimize the trade-off between data freshness and query performance, making the system production-ready for large-scale deployment while maintaining the real-time advantages of our current implementation.

7 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 using pure virtualization, ensuring maximum data freshness across all platforms.

7.1 Query Processing Pipeline

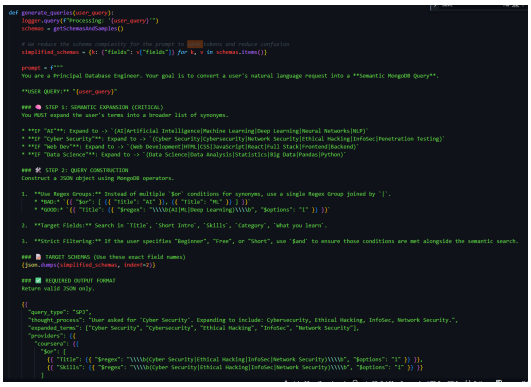


Figure 4: Query Decomposition: NL to MongoDB Query Translation

i) Query Decomposition (NL to MongoDB Query)
The natural language query is processed by Gemini LLM to identify user intent, entities, and semantic relationships. The system performs intelligent query expansion (e.g., "AI" → "Artificial Intelligence|Machine Learning|Deep Learning") and generates structured JSON representing MongoDB query objects. The `query_translator.py` then recursively maps these to platform-specific database fields.

ii) Query Federation over Databases
Decomposed sub-queries are executed in parallel across four independent MongoDB clusters (`Coursera_DB`, `Udacity_DB`, `FutureLearn_DB`, `Simplilearn_DB`). Each provider uses dedicated connection pools with fallback mechanisms for query validation and keyword-based search when structured queries fail.

iii) Aggregation of Courses & Ranking
All fetched courses are aggregated into a unified list. The custom relevance algorithm employs multiple similarity measures (Jaro-Winkler, Levenshtein, Cosine) with weighted scoring across title (4.0), skills (3.0), and description (2.0) fields. Scores are converted to probabilities via softmax normalization for meaningful relevance ranking.

Figure 5: Federated Query Execution Across Multiple Databases

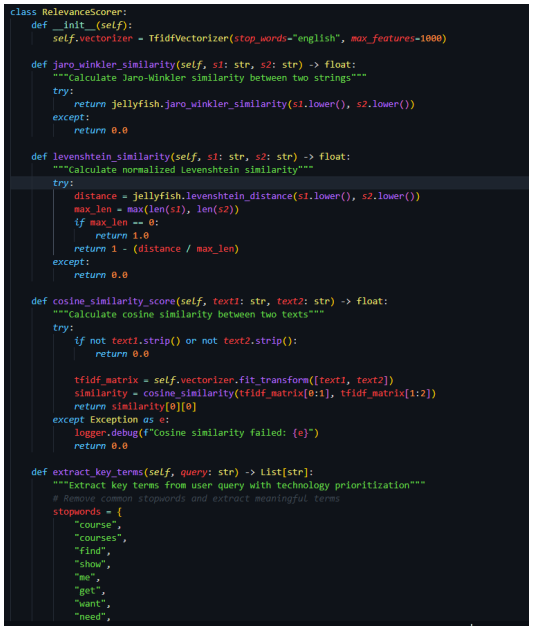
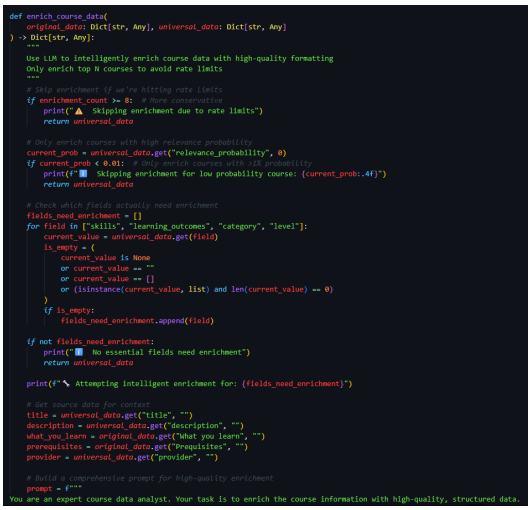


Figure 6: Multi-Algorithm Hybrid Scoring and Ranking

iv) Unify Format (LLM-Powered Standardization)
The ranked course list undergoes batch AI processing where Gemini LLM enriches missing metadata (skills, categories, levels) and converts all courses to the universal schema. The system employs conservative rate limiting (8 calls/minute) while ensuring consistent JSON output structure across all providers.



```
def enrich_course_data(
    original_data: Dict[str, Any], universal_data: Dict[str, Any]
) -> Dict[str, Any]:
    """
    Use LLM to intelligently enrich course data with high-quality formatting
    Only enrich top N courses to avoid rate limits
    """
    # Skip enrichment if we're hitting rate limits
    if enrichment_count >= 8:  # More conservative
        print("⚠️ Skipping enrichment due to rate limits")
        return universal_data

    # Only enrich courses with high relevance probability
    current_prob = universal_data.get("relevance_probability", 0)
    if current_prob < 0.01:  # Only enrich courses with >1% probability
        print(f"⏭️ Skipping enrichment for low probability course: {current_prob:.4f}")
        return universal_data

    # Check which fields actually need enrichment
    fields_need_enrichment = []
    for field in ["skills", "learning_outcomes", "category", "level"]:
        current_value = universal_data.get(field)
        is_empty = (
            current_value is None
            or current_value == ""
            or current_value == []
            or (isinstance(current_value, list) and len(current_value) == 0)
        )
        if is_empty:
            fields_need_enrichment.append(field)

    if not fields_need_enrichment:
        print("⏭️ No essential fields need enrichment")
        return universal_data

    print(f"🔧 Attempting intelligent enrichment for: {fields_need_enrichment}")

    # Get source data for context
    title = universal_data.get("title", "")
    description = universal_data.get("description", "")
    what_you_learn = original_data.get("What you learn", "")
    prerequisites = original_data.get("Prequisites", "")
    provider = universal_data.get("provider", "")

    # Build a comprehensive prompt for high-quality enrichment
    prompt = f"""
You are an expert course data analyst. Your task is to enrich the course information with high-quality, structured data.
```

Figure 7: LLM-Powered Standardization and Enrichment

7.2 Workflow Summary

Figure 8: UnifyLearn Query Processing Workflow

The algorithm successfully orchestrates the entire workflow from natural language parsing to unified result delivery, demonstrating robust handling of heterogeneous data sources while maintaining performance through intelligent batching and parallel execution strategies. The data flow follows a sequential pipeline where each component specializes in specific processing tasks while maintaining seamless integration with subsequent stages, ensuring efficient processing while preserving data freshness through real-time virtualization.

8 Innovation: The Relevance Scoring Algorithm

A key innovation in UnifyLearn is the development of a sophisticated hybrid relevance scoring algorithm that moves beyond simple keyword matching to intelligent semantic ranking. The algorithm employs multiple similarity measures and field-weighted scoring to deliver highly relevant course recommendations.

8.1 Multi-Algorithm Scoring Framework

The scoring system combines three complementary similarity algorithms:

- **Jaro-Winkler Similarity**: Optimized for technology term matching with higher weights for prefix matches
- **Levenshtein Distance**: Normalized edit distance for general text similarity
- **Cosine Similarity**: TF-IDF based semantic similarity between query and course content

8.2 Field-Weighted Scoring Architecture

The algorithm employs differentiated weights across course fields to prioritize technically relevant matches:

Field	Weight
Title	4.0
Skills	3.0
"What you learn"	3.0
Short Introduction	2.0
Level	2.0
Category	1.5
Sub-Category	1.0

Table 1: Field Weights in Relevance Scoring

8.3 Technology-Term Prioritization

The system identifies and prioritizes technology-specific terms through a dedicated technology lexicon containing over 50 technical keywords across domains including web development, programming languages, data science, cloud platforms, and AI/ML frameworks.

For each technology term identified in the user query, the system applies a tiered scoring approach:

- **Exact Phrase Match**: 5.0 points (highest priority for precise technical matches)
- **Word Boundary Match**: 3.0 points (matches whole words like "python" but not "pythonic")
- **Partial Match**: 1.5 points (substring matches within longer technical terms)

- **Similarity Match**: 0.5 × similarity score (for fuzzy matches with similarity > 0.85)

The total technology score is the sum of all individual term scores across all relevant course fields, where T represents the set of technology terms extracted from the user query. This prioritization ensures that courses teaching specific technologies requested by users receive significantly higher relevance scores.

8.4 Hybrid Score Calculation

The overall relevance score is computed as:

$$TotalScore = \sum w_f \cdot S_f + \alpha \cdot C + \beta \cdot B \quad (1)$$

Where:

- $w_f \cdot S_f$: Weighted field scores sum
- C: Cosine similarity score ($\alpha = 2.0$)
- B: All-technology-match bonus ($\beta = 3.0$)

8.5 Probability Normalization

Scores are converted to probabilities using temperature-scaled softmax:

$$P(i) = \frac{\exp(s_i/\tau)}{\sum_{j=1}^{N} \exp(s_j/\tau)} \quad (2)$$

Where $\tau = 0.05$ controls the probability distribution spread.

8.6 Semantic Query Expansion

Before scoring, the system performs intelligent query expansion:

- "AI" $\rightarrow$ "Artificial Intelligence", "Machine Learning", "ML", "Deep Learning"
- Technology term identification from specialized lexicons
- Level term detection (beginner, intermediate, advanced)

8.7 Implementation Advantages

The algorithm demonstrates several key advantages:

- **Technology-Focused**: Prioritizes exact matches for technical skills and tools
- **Multi-Dimensional**: Combines exact matching, similarity measures, and semantic understanding
- **Interpretable**: Provides detailed field-level scoring breakdowns for debugging
- **Normalized Output**: Converts raw scores to meaningful probabilities for ranking

This multi-faceted approach ensures that the ranking is both accurate and robust, significantly improving the quality of top results presented to the user while maintaining computational efficiency through optimized scoring mechanisms.

9 Challenges and Limitations

Despite the system's successful implementation, several challenges were encountered during development:

- **LLM API Constraints**: Free-tier Gemini API limitations introduced significant latency and rate-limiting, restricting system robustness for complex queries and large-scale usage
- **Limited Dataset Size**: The curated dataset, while adequate for prototyping, resulted in incomplete coverage for niche queries and specialized domains
- **Embedding Model Performance**: Initial experiments with SentenceTransformer (*all-MiniLM-L6-v2*) models for semantic matching underperformed compared to LLM-based approaches, leading to their abandonment
- **Query Parsing Dependence**: System reliability remains tightly coupled with Gemini API performance, with query decomposition errors propagating through the entire pipeline
- **False Positive Results**: Despite algorithmic improvements, some irrelevant courses still achieve moderate relevance scores, requiring further ranking refinement
- **Computational Overhead**: Real-time virtualization architecture, while ensuring data freshness, introduces performance costs for complex multi-platform queries

These limitations highlight areas for future optimization and underscore the trade-offs between system complexity, performance, and resource constraints in federated search environments.

10 Summary and Future Work

10.1 Summary of Current Work

We have successfully developed UnifyLearn, a functional prototype demonstrating real-time federated course search across four major learning platforms with heterogeneous schemas. The system effectively bridges structured and unstructured data processing through LLM integration, delivers intelligently ranked results using our custom hybrid relevance algorithm, and presents unified, standardized course information to users through an intuitive interface. The implementation successfully validates our core approach of combining database federation with AI-powered enrichment for cross-platform course discovery.

10.2 Future Work

Several directions for enhancement and optimization are planned:

- **Hybrid Architecture Implementation**: Transition from pure virtualization to a hybrid model, materializing static metadata (titles, instructors, categories) while maintaining real-time access for dynamic fields (ratings, enrollment data, pricing)
- **Reduced LLM Dependence**: Develop specialized models for natural language query parsing to decrease API dependency, and refine sentence transformer approaches for data standardization tasks
- **Ranking Algorithm Enhancement**: Incorporate additional relevance features including course ratings, popularity metrics, and user engagement data to further improve result quality and reduce false positives
- **Advanced Query Support**: Expand capabilities to fully support complex aggregation queries, cross-platform analytics, and advanced filtering operations
- **Performance Optimization**: Implement query caching, connection pooling, and batch processing optimizations to reduce latency and improve scalability
- **Platform Expansion**: Extend federation to include additional learning platforms and content providers for comprehensive coverage

These enhancements will address current limitations while positioning UnifyLearn for production-level deployment and expanded functionality in educational search and recommendation domains.

References

Fuheng Zhao, Divyakant Agrawal, and Amr El Abbadi. 2024. Hybrid querying over relational databases and large language models. *Preprint*, arXiv:2408.00884.