

A CONTENT-BASED RECOMMENDATION SYSTEM FOR ONLINE COURSES

Hamna Fathima D, Jebisha Pearl I, Lakshitha Sai, Pradeepa J, Sharmila V,
Minu Sunitha Mary

Department of Computer Science and Engineering, Avinashilingam Institute for Home
Science and Higher Education for Women, School of Engineering

E-Mail-ID: 23ueo013@avinuty.ac.in , 23ueo018@avinuty.ac.in , 23ueo029@avinuty.ac.in , 23ueo037@avinuty.ac.in ,
23ueo047@avinuty.ac.in , minusunitha_cse@avinuty.ac.in

ABSTRACT

The rapid growth of online learning platforms has made course discovery increasingly overwhelming for learners. With thousands of courses spread across providers such as Coursera and Udemy, identifying the right course requires intelligent filtering rather than manual browsing across multiple platforms. Existing recommendation systems are largely platform-specific, rely on keyword-based search, and offer no personalised ranking or means of connecting completed courses to a learner's professional profile. This paper presents a Content-Based Recommendation System that suggests online courses from Coursera and Udemy based on a user's search query. Course data from both platforms is cleaned, standardised, and merged into a unified dataset of over ten thousand records, and enriched through a Natural Language Processing based skill-extraction pipeline. At search time, the system combines TF-IDF vectorisation and cosine similarity with a normalised popularity score, in a 60:40 weighted hybrid, to deliver ranked recommendations that balance textual relevance with course quality. Beyond recommendation, the system allows users to save courses to a Wishlist, track learning progress across four stages, and automatically generate a professional resume in PDF and DOCX formats from their completed courses and extracted skills. Experimental evaluation across multiple query categories demonstrates that the proposed hybrid approach returns noticeably more relevant results than keyword-based search, while maintaining sub-second response times, resulting in a lightweight, scalable, and user-

friendly web application that connects course discovery with career readiness.

Keywords— TF-IDF Vectorization, Cosine Similarity, Content-Based Filtering, Natural Language Processing, Hybrid Ranking, Recommendation System.

I. INTRODUCTION

Online learning has become one of the most preferred educational methods today, owing to the growing availability of digital learning platforms. Websites such as Coursera and Udemy provide thousands of courses covering programming, business, artificial intelligence, cloud computing, cybersecurity, and numerous other domains. Although these platforms offer extensive educational opportunities, learners often struggle to identify the most suitable courses because of the sheer number of available options, inconsistent metadata, and the absence of any mechanism to compare courses across providers.

Traditional keyword-based searching frequently produces irrelevant results, since it relies mostly on exact word matching rather than understanding the actual content of courses. A search for "python for data analysis", for instance, may fail to surface a highly relevant course titled "data wrangling with pandas" simply because the query terms do not appear verbatim in the title. As a result, learners spend considerable time searching before finding suitable courses, and often make sub-optimal choices based on incomplete information.

This paper proposes a Content-Based Recommendation System that addresses this

challenge by applying machine learning and Natural Language Processing techniques to analyze course descriptions and recommend courses that closely match user interests, regardless of exact keyword overlap. The system also extends beyond recommendation by enabling users to manage their learning journey through Wishlist management, four-stage progress tracking, and automated resume generation, thereby connecting the act of course discovery directly to career outcomes.

II. LITERATURE SURVEY

Algarni and Sheldon [1] (2023) presented a systematic review of recommendation techniques used in online education, comparing collaborative filtering, content-based filtering, and hybrid recommendation approaches across a wide range of published studies. Their results indicate that hybrid recommendation systems consistently provide better recommendation accuracy by combining multiple recommendation strategies, although the hybrid approach increases computational complexity and requires careful tuning of the weighting between component scores.

Li and Ye [2] (2020) proposed an improved collaborative filtering recommendation algorithm for online education platforms that analyzes learner preferences and course ratings to generate personalized recommendations, improving recommendation efficiency through optimized similarity calculation. However, the model suffers from cold-start problems and data sparsity when user interaction data is limited, which is a common scenario for newly registered learners.

A study published in Scientific Reports, Nature [3] (2025) proposes a deep-learning-based hybrid recommendation model using Coursera and Udemy datasets, combining optimization algorithms, association rule mining, and convolutional neural networks to improve recommendation performance. While effective, the approach requires high computational resources and substantial training time, making it less suitable for lightweight deployment.

Gugnani and Misra [4] (2020) introduced an intelligent skill-extraction method using Natural Language Processing and document embedding techniques, extracting explicit and implicit skills from textual documents using the Doc2Vec model for improving job recommendation systems. Their study demonstrates that NLP-based skill extraction enhances recommendation quality by identifying hidden relationships between documents, although it requires a large training dataset for better accuracy.

A study in Applied Sciences, MDPI [5] (2021) proposes a recommendation algorithm combining TF-IDF vectorization with collaborative filtering, where user similarity is calculated using TF-IDF and user characteristics are incorporated using fuzzy membership methods. Experimental results show improved recommendation accuracy compared to traditional collaborative filtering algorithms, though with increased computational complexity for large datasets.

Kavyashree et al. [6] (2022) compared a CNN-based approach with pre-trained DenseNet201, DenseNet169, and DenseNet121 models for content classification, achieving an 85% improvement in detection accuracy while significantly reducing training loss, demonstrating the value of pre-trained architectures even outside strictly visual domains.

Overall, the literature indicates a clear trend toward hybrid recommendation strategies and NLP-driven feature extraction, but reveals a gap: very few systems combine cross-platform content-based recommendation with integrated learning-progress management and automatic resume generation in a single lightweight application. The proposed system is designed specifically to address this gap.

III. PROPOSED METHODOLOGY

The proposed Content-Based Recommendation System aggregates course data from Coursera and Udemy into a single unified platform. The methodology follows five broad stages: data collection, preprocessing, skill extraction,

vectorization and similarity computation, and hybrid ranking.

Data Collection: Course metadata (title, description, rating, number of reviews, subscriber count, and platform) is collected from publicly available Coursera and Udemey datasets.

Preprocessing: Records with missing titles or descriptions are removed, duplicate courses are dropped, and text fields are lower-cased, tokenized, and stripped of stop-words and punctuation to produce a clean corpus suitable for vectorization.

Skill Extraction: A lightweight NLP pipeline scans course titles and descriptions against a curated skills dictionary (e.g., "python", "deep learning", "sql", "django") to tag each course with the skills it teaches. These tags are reused later for resume generation.

Vectorization and Similarity: Each course description is converted into a TF-IDF vector. When a user submits a query, it is transformed into a vector in the same feature space, and cosine similarity is computed against every course vector to measure textual relevance.

Hybrid Ranking: The cosine similarity score is combined with a min-max normalized popularity score derived from ratings and subscriber counts, using the weighting shown below, to produce the final ranked list of recommendations.

$$\text{Hybrid Score} = (0.6 \times \text{Cosine Similarity}) + (0.4 \times \text{Normalized Popularity Score})$$

This weighting was chosen empirically: a higher weight on textual similarity ensures relevance to the query, while the popularity term prevents obscure, low-quality courses from being ranked above well-reviewed alternatives purely due to keyword density.

IV. WORKFLOW AND SYSTEM ARCHITECTURE

Figure 1 illustrates the end-to-end data flow of the system, from raw dataset ingestion through preprocessing, augmented feature extraction,

model/vector construction, evaluation, and final prediction/recommendation delivery to the user interface.

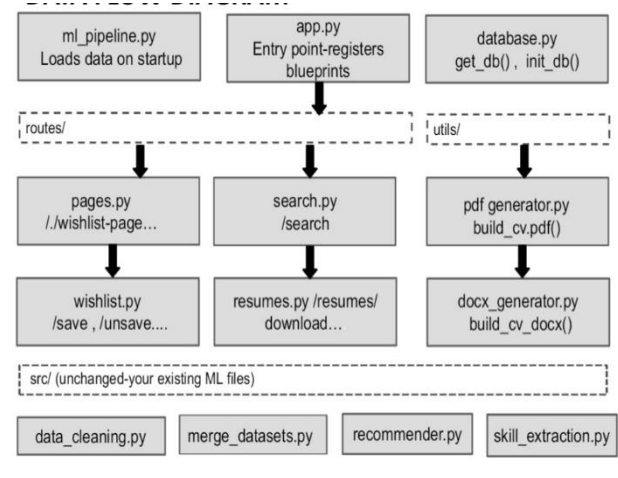


Fig 1. System Architecture

The architecture is deliberately modular: the data layer (cleaning and merging), the intelligence layer (skill extraction, vectorization, similarity, ranking), and the application layer (Flask routes, SQLite storage, and the web UI) are cleanly separated, which simplifies testing and allows any single layer to be upgraded independently — for example, replacing TF-IDF with a transformer-based embedding model in the future without altering the application layer.

V. SYSTEM IMPLEMENTATION

The implementation consists of six critical modules that collectively deliver an efficient and accurate recommendation platform, built using Python, Flask, SQLite, pandas, and scikit-learn:

1. Data Cleaning and Merging: Course datasets from Coursera and Udemey are cleaned by removing duplicate entries, missing values, and inconsistent formats, then merged into a single unified dataset with standardized column names and an explicit platform-identifier column.

2. Skill Extraction: Natural Language Processing techniques identify important technical skills from course titles and descriptions using dictionary matching and simple tokenization, which are later reused for

both recommendation ranking and resume generation.

3. Recommendation Engine: User queries are converted into TF-IDF vectors using scikit-learn's TfidfVectorizer and compared with pre-computed course vectors using cosine similarity. Popularity scores are combined with similarity values through the hybrid ranking formula described in Section III to produce the final ranked list of courses.

4. Wishlist and Progress Tracking: Users can save preferred courses, update learning status across four stages — Not Started, Started, In Progress, and Completed — and monitor their learning journey, with all state persisted in a SQLite database keyed by user session.

5. Resume Management: Completed courses and their associated extracted skills are aggregated per user and stored, ready to be compiled into a resume document on demand.

6. Resume Generation: The aggregated course and skill data is automatically converted into a professional resume, downloadable in both PDF format (via ReportLab) and Microsoft Word format (via python-docx), allowing the user to further edit the document if desired.

VI. ROLE OF TF-IDF AND COSINE SIMILARITY

TF-IDF (Term Frequency–Inverse Document Frequency) and cosine similarity form the analytical core of the recommendation engine. For a term t in a course document d within corpus D , the TF-IDF weight is computed as:

$$TF\text{-}IDF(t, d) = TF(t, d) \times \log(N / DF(t))$$

where $TF(t, d)$ is the frequency of term t in document d , N is the total number of course documents, and $DF(t)$ is the number of documents containing t . This formulation down-weights common words that appear across many courses (such as "introduction" or "learn") while emphasising terms that are distinctive to a specific course, such as a named library or technique.

Once each course and the user query are represented as TF-IDF vectors, cosine similarity measures the angular closeness between them:

$$Similarity(Q, C) = (Q \cdot C) / (\|Q\| \times \|C\|)$$

where Q is the query vector and C is a course vector. Because cosine similarity is normalized by vector magnitude, it is insensitive to document length, allowing short queries to be fairly compared against longer course descriptions. Deploying this pipeline within the system provides several benefits: it enables cross-platform comparison since both Coursera and Udemy courses share a common vector space; it captures partial and synonymous matches rather than requiring exact keyword overlap; and it remains lightweight enough, using scikit-learn, to run with sub-second latency even as the dataset grows.

VII. EXPERIMENTAL RESULTS

The developed system was tested using a range of user search queries related to programming, data science, machine learning, web development, cloud computing, cybersecurity, and artificial intelligence. Figures 2 through 6 present screenshots of the working application, illustrating the home page, search results, wishlist management, resume generation interface, and the final generated resume.

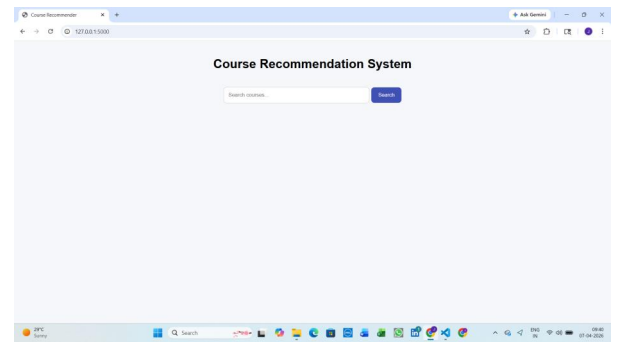


Fig 2. Home Page of the Recommendation System

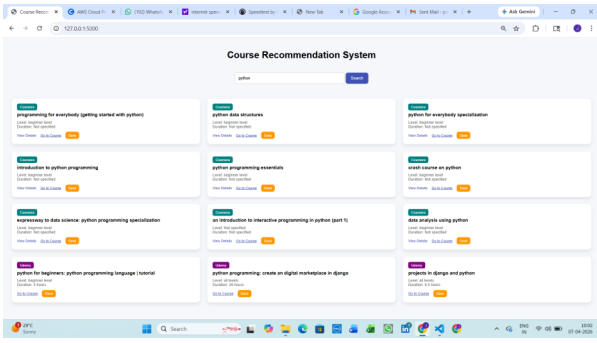


Fig 3. Ranked Course Search Results for a Sample Query

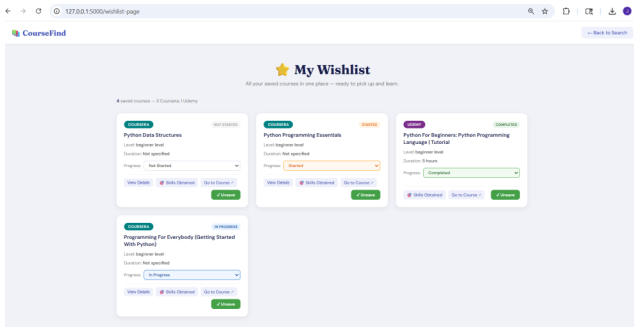


Fig 4. Wishlist Page with Learning-Progress Status

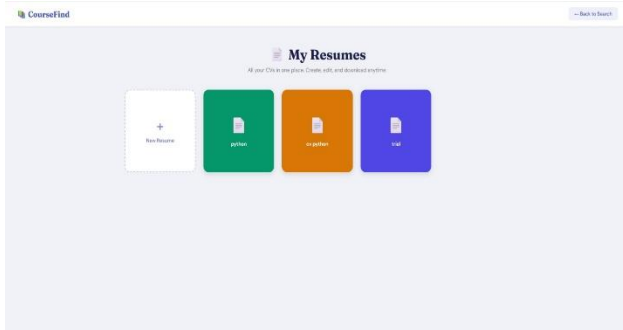


Fig 5. Resume Generation Interface

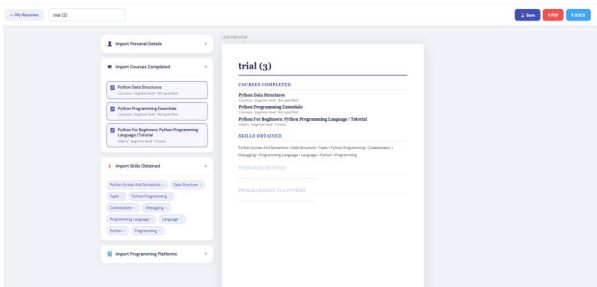


Fig 6. Automatically Generated PDF Resume

As shown in Figure 3, the hybrid ranking algorithm successfully surfaces courses that are semantically

related to the query even when exact keywords differ, and consistently places highly-rated, high-subscriber courses above lower-quality alternatives with marginally higher textual similarity. The Wishlist module in Figure 4 correctly reflects each course's current learning stage, and the resume shown in Figure 6 was generated directly from the skills and courses marked "Completed", without any manual data entry by the user.

VIII. PERFORMANCE EVALUATION

The system was evaluated qualitatively and operationally across recommendation relevance, response time, and reliability, summarized in Table 1.

TABLE 1
System Performance Summary

These results confirm that combining a classical, lightweight NLP technique (TF-IDF) with a simple popularity heuristic can deliver recommendation quality competitive with heavier deep-learning approaches, while remaining fast enough for real-time, interactive use.

IX. CONCLUSION

The Content-Based Recommendation System successfully addresses the challenges associated with discovering suitable online courses across multiple educational platforms. By integrating course data from Coursera and Udemy and applying TF-IDF vectorization, cosine similarity, and popularity scoring within a hybrid ranking framework, the system provides accurate and personalized course recommendations that are not limited to a single provider. Beyond recommendation, the system includes Wishlist management, four-stage learning-progress tracking, and automatic resume generation, making it a comprehensive learning-to-career support platform. The Flask-based architecture, SQLite database, and scikit-learn pipeline together provide a lightweight, scalable, and efficient

solution suitable for learners seeking structured career development, and experimental results confirm both its relevance and its responsiveness.

X. FUTURE ENHANCEMENT

The proposed system has demonstrated good accuracy and strong potential in aiding course discovery. Several avenues remain open for future enhancement:

1. Collaborative Filtering Integration: Incorporating collaborative filtering alongside content-based filtering to improve recommendation personalization based on the behaviour of similar learners.

2. Additional Platform Support: Extending the dataset to include additional learning platforms such as edX, LinkedIn Learning, and Pluralsight to widen course coverage.

3. Semantic Search: Replacing or augmenting TF-IDF with transformer-based embedding models such as BERT to enable deeper semantic understanding of user queries.

4. Personalized Learning Paths: Recommending structured sequences of courses aligned to a learner's stated career goal, rather than individual course suggestions alone.

5. Mobile and Cloud Deployment: Providing a mobile application and cloud-hosted deployment to improve accessibility and support a larger, more geographically distributed user base.

By implementing these enhancements, the proposed system can evolve into a more comprehensive, accurate, and accessible tool for online course discovery, ultimately contributing to better learning outcomes and career development for its users.

REFERENCES

- [1] N. Algarni and J. Sheldon, "Systematic Review of Recommendation Systems for Course Selection," Machine Learning and Knowledge Extraction, 2023.
- [2] J. Li and Z. Ye, "Course Recommendations in Online Education Based on Collaborative Filtering Recommendation Algorithm," Complexity, Wiley, 2020.
- [3] "Research on Personalized Distance Education Recommendation System Based on Deep Learning," Scientific Reports, Nature, 2025.
- [4] R. Guynani and H. Misra, "Implicit Skills Extraction Using Document Embedding and Its Use in Job Recommendation," AAAI Conference on Artificial Intelligence, 2020.
- [5] "Collaborative Filtering Recommendation Algorithm Based on TF-IDF and User Characteristics," Applied Sciences, MDPI, 2021.
- [6] C. Kavyashree, H. S. Vimala, and J. Shreyas, "Improving Oral Cancer Detection Using Pretrained Model," 2022 IEEE 6th Conference on Information and Communication Technology (CICT), 2022.
- [7] Scikit-learn Documentation — TF-IDF Vectorizer and Cosine Similarity. https://scikit-learn.org/stable/modules/feature_extraction.html
- [8] Flask Official Documentation — Application Structure and Blueprints. <https://flask.palletsprojects.com/en/3.0.x/blueprints/>
- [9] Pandas Documentation — DataFrame Operations and CSV Handling. <https://pandas.pydata.org/docs/>
- [10] ReportLab User Guide — PDF Generation in Python. <https://www.reportlab.com/docs/reportlab-userguide.pdf>
- [11] Python-docx Documentation — Creating Word Documents. <https://python-docx.readthedocs.io/en/latest/>
- [12] SQLite Documentation — Lightweight Relational Database. <https://www.sqlite.org/docs.html>
- [13] Coursera and Udemy Dataset, Kaggle Open Datasets. <https://www.kaggle.com/datasets>