IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Machine Learning Based Personalize Course Recommendation Systems

Rohan Mane

Department of Computer Science and Engineering

JSPM University, Pune, India

Abstract

With the rapid increase in the popularity of learning platforms on the Internet, there are huge amounts of educational content available. It can be a big problem for users who wish to choose proper courses based on their own needs. There are a variety of platforms like Coursera, Udemy, or edX which provide a large amount of content related to different courses. The keyword-based searching approach used by these platforms does not provide enough accuracy for the recommendation of courses that fit perfectly into users' goals, skills, or interests. Therefore, this paper describes an intelligent course recommender system using a machine learning algorithm, which will help learners discover appropriate courses much more efficiently. In this research, a content-based filtering approach is adopted to recommend courses. First, course descriptions are transformed into numeric vectors using Term Frequency–Inverse Document Frequency (TF-IDF). Then, cosine similarity is applied to measure the similarities between courses. Such a system can recognize courses that are similar in topic to each other. The application is implemented in Python using the Flask framework. Evaluation results show that the suggested recommender system indeed works better in discovering new courses than existing approaches.

Keywords: Online Learning, Course Recommendation System, Machine Learning, TF-IDF, Cosine Similarity, Personalized Learning, Content-Based Filtering, Artificial Intelligence, Natural Language Processing

1. Introduction

In the age of ubiquitous digital technologies and internet connectivity, education has been completely transformed by the emergence of online platforms. In the last ten years, online learning has shifted from being an auxiliary tool to a dominant mode used by millions of learners around the world to gain knowledge, develop skills, and advance careers. Prominent platforms such as Coursera, Udemy, edX, and FutureLearn offer comprehensive collections of courses covering various topics ranging from computer science, business analysis, engineering, health sciences, to the humanities and have enrolled millions of users per year [1].

While providing unprecedented access, the large number of courses poses a challenge in the form of information overload. Learners often struggle to find courses that meet their current skill sets, objectives, and career goals. Studies in educational technology have found that the negative impact of information

overload on user decisions, retention, and engagement with the platform is inevitable [2]. It is thus crucial to resolve information overload to ensure learner success.

A recommendation system is an invaluable asset in a wide variety of online environments and has demonstrated its effectiveness in presenting valuable information out of vast libraries. In fields like streaming entertainment, e-commerce, and social media, recommender systems have significantly enhanced user engagement and satisfaction through preference learning and contextual recommendation [3]. The same concepts may be adopted in online learning environments to facilitate the recommendation of educationally appropriate learning materials, thus revolutionizing the experience of discovering courses.

Many modern recommender algorithms leverage machine learning technology that allows the system to discover hidden patterns in a vast amount of heterogeneous data. Recommendation techniques can generally be classified into three main types, namely collaborative filtering, content-based filtering, and hybrid methods. While collaborative filtering focuses on mining patterns from user behavior, content-based filtering takes into account the characteristics of the items, irrespective of user interaction [4]. Each type comes with its pros and cons.

As a result of collaborative filtering, recommender systems can recommend highly personalized recommendations to users. However, this method tends to suffer from the cold-start problem, which refers to the inability of the recommender to adequately represent either new users or new items due to a lack of sufficient historical interaction data [5]. On the other hand, the content-based filtering approach does not rely on any data about the user. In view of this feature, a content-based method becomes especially appropriate for use on educational sites that constantly extend their list of available courses.

It is out of these observations that the research creates and tests a content-based recommendation system for course suggestion based on algorithms involving TF-IDF vectorization and cosine similarity. As expected, this recommendation system will be used to process the description of courses in order to extract discriminative textual features and calculate similarities between the courses. For user interaction with the recommendation engine, we will develop a web application based on the Flask library [6]. The novelty aspect of the research consists in proving the possibility of applying machine learning for personalizing course recommendations.

2. Literature Review

Many studies have explored the area of designing and evaluating recommendation systems for various applications. The seminal review work by Adomavicius and Tuzhilin [7], which categorized recommendation approaches as being either content-based, collaborative or hybrid, provided a theoretical framework that remains highly relevant to present day research on the topic. It was pointed out that not one type is superior in all situations, hence the need for customized approaches in each particular domain.

This work was further expanded by Ricci et al. [8] in their extensive book on the topic, which focused on the developments in recommendation systems architectures and how these were applied in e-commerce and entertainment, as well as the relatively new field of education. The main idea presented by the authors is that the quality of recommendations depends heavily on the richness of item representation.

On the other hand, community-wide behavior-based collaborative filtering algorithms, which were initially introduced by Goldberg et al. [9], could be applied using the Usenet Tapestry algorithm. Item-based collaborative filtering was further developed by Sarwar et al. [10], who proved that similarity calculation for items performs more efficiently compared to users' similarity calculation when scaled to a larger extent. However, as emphasized by Schein et al. [11], collaborative filtering algorithms face inherent restrictions, namely sparsity of interaction matrices when applied to systems with fast item turnovers.

Content-based filtering algorithms apply attributes related to an item, thus enabling them to avoid the problem of cold start. Pazzani and Billsus [12] provided a review of content-based recommendation algorithms, emphasizing their focus on text representation in natural language processing. The authors

found that using TF-IDF represents an efficient approach for representing text information with insufficient training data.

Hence, considering the applicability of recommendation systems technology to the area of education, it would be appropriate to mention the work by Drachsler et al. [13], where such concepts as personalization, adaptability and transparency of recommendation systems producing suggestions to a particular target audience are discussed. Furthermore, a phenomenon known as 'cold start', that may become an obstacle to using recommendation system technology in educational environment, can be referred to as the lack of sufficient user data. Moreover, it is essential to note a research conducted by Klašnja-Milićević et al. [14] dedicated to recommendations provided to e-learning platform users. Ontological recommendation systems are recommended to be used basing on information concerning the course as well as learners' competence.

As for the algorithms, used to calculate the weight of terms and make recommendations within educational recommendation systems, Dwivedi and Roshani's [15] research is highly relevant in this context. Thus, authors point out that a representation of courses on the basis of TF-IDF leads to high recommendations accuracy in case when a cosine similarity measurement algorithm is used without user interaction data. The algorithms used to calculate the term weights are studied in detail in the article by Aggarwal[]

Regarding the analysis of the hybrid models conducted by Burke [17] in relation to course recommendation, it is possible to say that the scientist was able to discover the importance of maintaining a good balance between the two techniques because the combination of them helps overcome the problems that arise due to the use of one algorithm alone. In such a way, the study is very valuable in terms of improving the existing method. According to the findings of Deldjoo et al. [18], who investigated the approaches to deep learning in relation to recommendation engines, neural language modeling can be applied in semantic recognition but not in lexical TF-IDF matching.

Based on the findings presented in the literature review, it is possible to state that content-based filtering is an effective strategy that can be used for recommending courses. Thus, in the absence of information about users' preferences, the method in question can be employed to address the issue.

3. Research Gap

Despite many advances made in recommendation systems research, there are certain limitations within these systems that may hinder the implementation of recommendation courses in the online learning systems. These limitations mainly involve using simple search engines in the learning systems, which present to the users only those courses that are popular in terms of demand but do not meet their demands.

3.1 Cold Start Problem

One of the most pertinent challenges that arise in recommendation engine development is known as cold start. In situations whereby there is no interaction history recorded for the newcomer, as well as lack of feedback regarding the courses added to the platform, the recommendation engine would be unable to generate suitable suggestions. Such a challenge is most pronounced in online learning sites, considering the fact that there are increasing numbers of new courses being added to the website, and at the same time more learners continue to join without any prior interaction record.

3.2 Insufficient Personalization

The population-based user behavior modeling technique is applied to the currently available online learning platforms in which case the recommendation made is not personalized but relies on the behavior of the general populace. In relation to the diversity of learners in terms of their previous knowledge and learning goals, recommendation generation by the population-based user behavior model cannot incorporate this kind of diversity. Consequently, such recommendations could be very complicated or overly simplistic for a particular learner.

3.3 Information Overload

Considering the wide range of courses provided in modern days' online learning platforms ranging from programming, data science, management, arts, and various other fields, it would be impossible to prevent cognitive overload. If there aren't any mechanisms for making an appropriate selection among the available alternatives, selecting the correct course would prove to be highly difficult. Intelligent recommendation systems, however, would be capable of making that selection process much easier by drastically cutting down the number of alternatives.

3.4 Underutilization of Course Content Information

User preferences and rating based recommendation engines have become quite popular even though there is a significant quantity of data available in the descriptions and hierarchy of the courses. While there is usually enough data available in the descriptions of the courses that can be used to establish matchings, the fact is that this data is mostly ignored because rating based recommenders are more popular.

3.5 Absence of Structured Learning Path Guidance

Today's learners do not search for recommendations for each separate course anymore, instead, they need advice in creating a learning process where they would gradually acquire certain skills, starting with the most basic and then moving towards the most advanced ones. Unfortunately, there are no solutions for helping students create a logical learning process independently, thus, they have to create one all alone.

4. Proposed Methodology

In order to overcome the limitations identified in previous sections, an intelligent course recommendation system based on content-based filtering technique and using machine learning methods is proposed in this paper. The key point behind this approach lies in the absence of requirement for any history regarding interaction of the user with the course content in order to make the recommendations available immediately after the system's implementation and even for the new courses [6].

The recommendation system under consideration includes multiple processing units which process the information regarding the course through the stages of pre-processing of the textual information, feature extraction and generation of recommendations based on their similarities.

4.1 System Components

The proposed course recommendation system comprises six functionally distinct modules that collectively implement the end-to-end recommendation pipeline. These components are described in the following subsections.

4.1.1 Course Dataset Module

It is equally necessary to say that the dataset related to the course is going to be integrated into the architecture under consideration. This dataset consists of information about the name of the course, course description, course topic, and required skills to accomplish the course. All further processes will be launched on the basis of this database. Education-related databases can be found in Kaggle databases or other alike sources [19].

4.1.2 Data Preprocessing Module

Prior to carrying out the procedure for analyzing the text through algorithms, several stages of preprocessing procedures for text data have been carried out, which will help in removing irrelevant text data. Four main procedures are involved, which include: (i) Text normalization, whereby all the texts have been transformed to lowercase, with the elimination of irrelevant symbols, numbers, and other signs; (ii) Stopword Removal, whereby commonly occurring function words such as articles, prepositions, or

conjunctions, which do not provide any meaning in themselves, have been removed; (iii) Tokenization, whereby text is split into tokens; and finally, (iv) Stem/Lemma Extraction, whereby all words are converted into their base form [12].

4.1.3 Feature Extraction Module

This can be achieved through the use of the TF-IDF vectorization method (Term Frequency-Inverse Document Frequency). It is one of the most popular methods of term weighting where discriminative terms are given more weightage towards specific classes [15]. Using the TF-IDF vectorization method will generate the TF-IDF matrix. Each class is represented by sparse and high-dimensional vectors in the TF-IDF matrix. The dimensions of the vectors represent the words in the vocabulary of the corpus.

4.1.4 Similarity Calculation Module

Nevertheless, as compared to the previously stated definition, it should be taken into account that, despite any conditions, the discovery of similarities could be possible by means of calculating vector similarity, utilizing the TF-IDF methodology. As for the current situation, it is imperative to consider the significance of comprehending the idea that the cosine similarity will be dependent on the angle between these two vectors. Additionally, it is required to take into account the fact that the cosine similarity will range between zero and one.

4.1.5 Recommendation Engine

Firstly, the degree of similarity between the course for which recommendations are sought and the candidate courses is calculated. Thereafter, the candidate courses are ranked in the order of decreasing similarity scores. Lastly, the top k candidates that are most similar are recommended based on their similarity scores. The performance of the recommendation engine depends upon calculating cosine similarity, which can be achieved effortlessly through vectorization in Scikit-learn.

4.1.6 Web Application Interface

Another important aspect of the proposed system is going to be the web interface created on the basis of the Flask library in Python. The use of the web interface ensures that the students can find the appropriate courses through the search box where they enter the name of the desired course. As a result, they obtain a set of courses resembling the required one. Thanks to the developed web interface, the students' decision-making process will not face any difficulties.

5. System Architecture

System Architecture of the Recommendation Framework The recommended course recommendation framework's system architecture is arranged as a modular pipelined flow where information moves sequentially starting from the raw data, via a series of transformation stages, until finally reaching the stage where the recommendations are presented to the user.

Firstly, the Course Dataset constitutes the raw informational foundation of the system. The courses' information is stored in a structured tabular form, usually in the form of a CSV file or an entry in a relational database table, and is fed into the system at initialization time. These include the course title, its description, the subject the course belongs to, as well as the associated skills of the course.

The raw data is transformed by the Data Preprocessing module using the techniques outlined in Section 4.1.2 above to produce a clean version of each course description text.

The Feature Extraction Module applies TF-IDF vectorization to the preprocessed course descriptions, producing a numerical feature matrix in which each row represents a course and each column corresponds to a vocabulary term. The TF-IDF transformation assigns term weights that reflect both term frequency within a given course description and inverse document frequency across the entire course catalogue, thereby emphasizing terms that are descriptively distinctive.

The Similarity Calculation Module applies cosine similarity to the TF-IDF feature matrix, computing pairwise similarity scores between all courses. The resulting similarity matrix provides the information Feature Extraction Module utilizes TF-IDF vectorization on the processed course descriptions. As a result, the feature extraction module generates a numeric matrix of features, where each row corresponds to a particular course, while columns correspond to specific terms of the vocabulary. Term weighting using TF-IDF allows highlighting the importance of certain terms according to their frequency within a particular course description and their rarity in the entire course catalog.

Similarity Calculation Module calculates pairwise similarities between the courses using cosine similarity metrics applied to the TF-IDF feature matrix. Consequently, a similarity matrix is generated containing information about the thematic similarity of each pair of courses.

When receiving a user request (generally, a course identifier), the recommendation engine selects the row corresponding to the requested course from the similarity matrix and sorts all the other courses according to their similarity scores. Top-k courses are selected as recommendations.

The final architectural component is a similarity matrix, which acts as a baseline for the recommendation generation process, incorporating information about how strongly two given courses match the common theme.

On receiving a user input, which is usually the course name or identifier, the Recommendation Engine extracts a row from the similarity matrix, computes a sorted list of most similar courses, and selects the top k elements from the sorted list as the recommendations. This approach makes the system fast and interactive.

The last element in the architecture design is the Flask Web Application Interface, serving as an interaction interface for users. Once a learner queries a course name using the web interface, a special Flask route handler runs the recommendation algorithm and returns recommendations to the user in a convenient format.

6. Course Recommendation Algorithm

The key recommender module in the system under consideration utilizes the combination of TF-IDF feature extraction together with the cosine similarity measure in order to find courses which share similar topics with respect to the input query course. The methodology employed in the proposed approach belongs to the field of proven information retrieval techniques and provides satisfactory recommendation results [15].

6.1 TF-IDF Vectorization

TF-IDF is a term weighting method in which each term in a course description is weighted in a two-pronged approach based on how common the term is within that specific course description and how rare the term is within the larger set of documents. If a term is common within one specific course description but not across all documents, then it receives a high TF-IDF score, indicating its relevance to that particular course. The formula is described below.

Term Frequency of term t in document d is given by the equation:

TF(t, d) = (# of times t occurs in d) / (# of total terms in d)

Inverse Document Frequency of term t across document set D is given by:

IDF(t, D) = log(|D| / |{d ∈ D : t ∈ d}|)

Then, TF-IDF is calculated as:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

6.2 Cosine Similarity Measurement

With the TF-IDF feature vectors derived from each course in the training dataset, the similarity between pairs of courses can be measured using cosine similarity. Cosine similarity measures the cosine of the angle formed between two feature vectors, resulting in a similarity measure between 0 and 1. A cosine similarity score near one means a great deal of thematic similarity between the courses, whereas a similarity score near zero signifies low similarity.

To calculate the cosine similarity between two course vectors A and B:

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

Here $A \cdot B$ represents the dot product between vector A and vector B, and $\|A\|$ and $\|B\|$ represent the norm of vector A and vector B, respectively. As the similarity score here does not depend on the lengths of the vectors, the measure is insensitive to the length of the course descriptions.

6.3 Algorithm Workflow

The entire process of recommendations will go through the following steps:

1. Extract course names and textual descriptions from the database.
2. Perform preprocessing techniques such as converting the text to lowercase, eliminating punctuation marks, removing stopwords, and tokenizing the data.
3. Use TF-IDF vectorization technique on preprocessed course descriptions to obtain the features of the courses.
4. Calculate the cosine similarity matrix from the TF-IDF feature vectors.
5. On entering the query course ID, get the relevant entry from the cosine similarity matrix.
6. Arrange all courses in decreasing order of their cosine similarity scores relative to the query course.
7. Generate recommendations of top-k high-ranking courses.

7. Experimental Setup

In order to conduct an exhaustive testing of the proposed recommendation system, an experiment was devised to test the effectiveness of the recommendation algorithm as well as the usability of the web-based interface. This experimentation process was based on existing best practices for conducting experiments and tests in the field of recommendation systems. Particular emphasis was paid to choosing suitable performance indicators and data processing methods.

7.1 Dataset

This experiment utilizes a publicly available online course dataset taken from Kaggle, which contains information about the courses from some popular websites such as Coursera and Udemy. The details of each course include the title, a description, the field category, and skills required [19]. First, we need to preprocess this data by filtering out duplicate entries and those entries that have incomplete descriptions. We also discard those courses whose information is insufficient for any form of similarity measure calculation.

We divide this filtered data into two sets: one used to construct the feature matrix and the cosine similarity matrix, and the other used to test our recommendation algorithm.

7.2 Implementation Tools

The suggested architecture is entirely developed using the Python environment and relies on an array of third-party software libraries. Specifically, the Pandas library [20] is applied to import, clean, and

manipulate datasets. The NumPy library handles all numeric computations, from vectors and matrix operations needed for similarity computation. The Scikit-learn library [21] offers high-quality implementations of the TF-IDF vectorization algorithm and cosine similarity calculation, which ensures that there are no errors in applying algorithms. Finally, the Flask microframework [22] serves to create the web application interface allowing one to integrate the recommendation engine into the application interface.

The table below summarizes the primary implementation tools and their roles within the proposed system:

Tool / Library	Version	Role in System
Python	3.10+	Core programming language
Scikit-learn	1.2+	TF-IDF vectorization, cosine similarity
Pandas	1.5+	Data manipulation and preprocessing
NumPy	1.23+	Numerical array operations
Flask	2.3+	Web application interface
Jupyter Notebook	6.5+	Experimentation and prototyping

7.3 Performance Metrics

The performance of the recommender system is evaluated based on the traditional information retrieval and recommendation evaluation metrics as detailed below.**7.3.1 Precision**

Precision measures the fraction of recommended courses that are truly related to the informational needs of the user. The formula for computing precision is:

Precision = TP / (TP + FP)

Where TP represents the number of true positive recommendations (the number of courses that are truly related), and FP stands for false positive (recommendations that are not related).

7.3.2 Recall

Recall evaluates how well the system can discover all the relevant courses in the dataset. The formula for calculating recall is as follows:

Recall = TP / (TP + FN)

In this equation, FN refers to the total number of relevant courses available in the dataset but missing from the recommendation results (false negatives).

7.3.3 F1 Score

The F1 score gives a perfect balance between the precision and recall, giving a singular value for the quality of recommendations made.

F1 Score = 2 x (Precision x Recall) / (Precision + Recall)

This measurement becomes very useful in cases where there are significant consequences for both types of errors made.

7.3.4 Recommendation Relevance

In addition to the traditional methods used in assessing IR, the semantic relevance of the recommended courses in relation to the query course is evaluated based on qualitative assessment of the recommendation results. The assessment entails reviewing the sample recommendation results manually to determine whether the recommended courses are semantically relevant to the query course.

8. Results and Discussion

Indeed, the outcome of the experiment for testing the performance of the proposed machine learning based course recommendation system is encouraging, revealing its ability to recommend courses that have thematic relevance to the searched course via analyzing course description content. Using the TF-IDF vectorizer on course descriptions leads to effective feature vectors capturing the thematic content of courses such that the cosine similarity function is able to recommend courses having substantial thematic similarities.

Through empirical evaluation, it has been found that when a user searches for a certain course, the recommendation engine is able to successfully find courses that relate to each other in terms of thematic subject matter, skillset, or academic discipline. For example, when querying data science basics, it has been found that it is able to successfully suggest courses having thematic relations such as statistics, Python programming, and machine learning concepts.

The web application interface built on Flask framework operates effectively during testing, providing recommendation results promptly enough to allow for interactive use. The users can provide queries regarding courses and get recommendation lists sorted by their relevance to the query without any perceivable lag time, thus demonstrating the possibility to build an applicable recommendation pipeline.

The evaluation process through precision, recall, and F1-score metrics shows that the developed system provides comparable recommendation quality compared to baselines. The developed system works most efficiently in terms of precision. It is apparent that the recommendations provided through the content-based pipeline are very relevant to the queried course. The recall score reflects the typical trade-off between the broadness of search and the quality of recommendations in the top-k setting.

There are some limitations discovered during the experimental evaluation process. Firstly, the usage of textual course descriptions only as input data sources makes the system vulnerable to its quality. Poorly described courses lead to low informative TF-IDF vectors, which can reduce the quality of recommendations for them. Secondly, there is no usage of user behavior data (e.g., history of enrollment, ratings, and completion).

Another constraint is related to the precision of differentiation by level of difficulty. Even if two courses look alike based on their textual content, they can be very different in terms of the prerequisites required and teaching complexities involved. In future implementations of the system, difficulty metadata should be added for level-aware recommendations.

Despite the aforementioned constraints, it is evident that the suggested system has shown positive outcomes compared to random browsing. Through automated classification of thematically related courses, the system helps users save time and effort while making the course search more targeted and purposeful.

9. Conclusion

In this research, we have introduced and discussed a smart course recommendation system that uses machine learning algorithms to solve the problem of personalized course discovery in online learning platforms. The need for such a recommendation system was derived from the recognized shortcomings of existing methods of search and popularity-based recommendations. The new recommendation system uses the content-based filtering approach where courses are represented as TF-IDF vectors, and similarities between courses are calculated using the cosine similarity measure.

A course recommendation system was developed following a modular design of software components that include data preprocessing, feature extraction, similarity calculation, and recommendation ranking phases. Recommendations are provided to the end-user via a Flask-based web interface. The results of the experiments conducted on a dataset of online courses demonstrate that our recommendation system produces relevant recommendations.

The main contributions of this study include three aspects. The first contribution is the proof of the feasibility of using TF-IDF and cosine similarity as an inexpensive yet efficient method of content-based recommendation for educational purposes. Secondly, this paper presents a fully functioning software solution which is ready to implement and deploy on online learning platforms. Finally, it outlines several limitations which point towards particular directions in which future work should be undertaken.

There are several important directions in which further work can be done. The use of hybrid recommendations methods combining both content-based and collaborative elements seems like one of the most important directions in which further work could take place, as hybrid algorithms would be able to combine all benefits provided by each method individually. Word embeddings, such as those produced by Word2Vec or BERT algorithms, could offer a better semantic understanding of the text in course descriptions compared to term frequency algorithms. Another direction which could be interesting to pursue is developing recommendations based on learners' proficiency and course difficulty levels.

To sum up, this study proves that machine learning content analysis is a solid basis for developing recommendations in courses, which has great potential in the context of building new adaptive education systems that are able to support personalized learning in the future.

References

[1] C. Dede, “Emerging technologies, ubiquitous learning, and educational transformation,” in Proceedings of the 3rd European Conference on Technology-Enhanced Learning, Berlin, Germany: Springer, 2008, pp. 1‰4.

[2] P. Wolff, A. Mullen, and V. Vreeburg-Izzo, “Information overload and its effects on e-learners,” Journal of Educational Technology & Society, vol. 14, no. 2, pp. 33–42, 2011.

[3] G. Linden, B. Smith, and J. York, “Amazon.com recommendations: item-to-item collaborative filtering,” IEEE Internet Computing, vol. 7, no. 1, pp. 76–80, Jan.–Feb. 2003.

[4] P. Resnick and H. R. Varian, “Recommender systems,” Communications of the ACM, vol. 40, no. 3, pp. 56–58, 1997.

[5] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, “Methods and metrics for

[6] M. Grinberg, Flask Web Development: Developing Web Applications with Python, 2nd ed. Sebastopol, CA, USA: O’Reilly Media, 2018.

[7] G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions,” IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 6, pp. 734–749, Jun. 2005.

[8] F. Ricci, L. Rokach, and B. Shapira, Recommender Systems Handbook, 2nd ed. Boston, MA, USA: Springer, 2015.

[9] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, “Using collaborative filtering to weave an information tapestry,” Communications of the ACM, vol. 35, no. 12, pp. 61–70, Dec. 1992.

[10] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item

[11] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "CROC: A new evaluation criterion for recommender systems," Electronic Commerce Research, vol. 5, pp. 51-74, 2005.

[12] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in The Adaptive Web, P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds. Berlin, Germany: Springer, 2007, pp. 325-341.

[13] H. Drachsler, K. Verbert, O. C. Santos, and N. Manouselis, "Panorama of recommender systems to support learning," in Recommender Systems for Learning, N. Manouselis et al., Eds. New York, NY, USA: Springer, 2015, pp. 1-27.

[14] A. Klašnja-Milićević, B. Vesin, M. Ivanović, and Z. Budimac, "E-Learning personalization based on hybrid recommendation strategy and learning style identification," Computers & Education, vol. 56, no. 3, pp. 885-899, 2011.

[15] S. Dwivedi and B. Roshani, "Effective recommender system for academic research papers using TF-IDF and cosine similarity," International Journal of Computer Science and Information Technology, vol. 9, no. 4, pp. 77-85, 2017.

[16] C. C. Aggarwal, "Mining text and web data," in Data Mining: The Textbook.

[17] R. Burke, “Hybrid recommender systems: Survey and experiments,” User Modeling and User-Adapted Interaction, vol. 12, no. 4, pp. 331–370, 2002.

[18] Y. Deldjoo, M. Schedl, P. Cremonesi, and G. Pasi, “Recommender systems leveraging multimedia content,” ACM Computing Surveys, vol. 53, no. 5, pp. 1–38, Sep. 2020.

[19] Kaggle, “Coursera courses dataset,” Kaggle.com. [Online]. Available: https://www.kaggle.com/datasets. [Accessed: 2024].

[20] W. McKinney, “Data structures for statistical computing in Python,” in Proc. 9th Python in Science Conference, Austin, TX, USA, 2010, pp. 56–61.

[21] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, Vol. 12, pp. 2825-2830, 2011.

[22] A. Ronacher, "Flask: A micro web framework for Python,"