I.J. Intelligent Systems and Applications, **2023, 5, 28-40**
Published Online on October 8, 2023 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2023.05.03

An Enhanced Approach to Recommend Data Structures and Algorithms Problems Using Content-based Filtering

Aayush Juyal*
Department of Computer Science & Engineering, Sharda School of Engineering & Technology, Sharda University, Greater Noida, 201310, India
E-mail: aayushjuyal12@gmail.com
ORCID iD: https://orcid.org/0000-0003-4374-1119
*Corresponding Author

Nandini Sharma
Department of Computer Science & Engineering, Sharda School of Engineering & Technology, Sharda University, Greater Noida, 201310, India
E-mail: nandini.sharmaa007@gmail.com

Pisati Rithya
Department of Computer Science & Engineering, Sharda School of Engineering & Technology, Sharda University, Greater Noida, 201310, India
E-mail: pisatirithyareddy@gmail.com

Sandeep Kumar
Department of Computer Science & Engineering, Sharda School of Engineering & Technology, Sharda University, Greater Noida, 201310, India
E-mail: sandeep.csengg@gmail.com

Received: 03 May 2023; Revised: 23 June 2023; Accepted: 05 July 2023; Published: 08 October 2023

Abstract: Data Structures and Algorithms (DSA) is a widely explored domain in the world of computer science. With it being a crucial topic during an interview for a software engineer, it is a topic not to take lightly. There are various platforms available to understand a particular DSA, several programming problems, and its implementation. Hacckerank, LeetCode, GeeksForGeeks (GFG), and Codeforces are popular platforms that offer a vast collection of programming problems to enhance skills. However, with the huge content of DSA available, it is challenging for users to identify which one among all to focus on after going through the required domain. This work aims to use a Content-based filtering (CBF) recommendation engine to suggest users programming-based questions related to different DSAs such as arrays, linked lists, trees, graphs, etc. The recommendations are generated using the concept of Natural Language Processing (NLP). The data set consists of approximately 500 problems. Each problem is represented by the features such as problem statement, related topics, level of difficulty, and platform link. Standard measures like cosine similarity, accuracy, precision, and F1-score are used to determine the proportion of correctly recommended problems. The percentages indicate how well the system performed regarding that evaluation. The result shows that CBF achieves an accuracy of 83 %, a precision of 83 %, a recall of 80%, and an F1-score of 80%. This recommendation system is deployed on a web application that provides a suitable user interface allowing the user to interact with other features. With this, a whole E-learning application is built to aid potential software engineers and computer science students. In the future, two more recommendation systems, Collaborative Filtering (CF) and Hybrid systems, can be implemented to make a comparison and decide which is most suitable for the given problem statement.

Index Terms: Recommendation Systems, Content-Based Filtering, Text Classification, TF-IDF, NLP.

1. Introduction

Computer Science and its relatable domain is a complex field that demands continuous learning and updating of skillset. A solid understanding of DSA is critical for designing efficient and optimized code, improving performance, and creating robust applications in the field of computer science. With the rapid advancement of technology and the constantly evolving nature of programming languages, developers must stay up-to-date with the latest concepts and techniques in DSA. Numerous coding platforms often provide a static learning experience that may not be intuitive or engaging for users. As a result, a major issue arises where individuals understand the theoretical concepts but fail to grasp the practical application of DSA in real-world scenarios. To achieve expertise in DSA, developers must practice solving programming problems offered by online coding platforms like Hacckerank, LeetCode, GFG, and Codeforces. These platforms provide problems covering different DSA topics and varying difficulty levels. However, with the abundance of problems available, it can be overwhelming for users to identify the most effective problems to enhance their skills after studying the theory. This work looks to overcome this limitation by recommending effective programming problems using the proposed recommendation system. While several web applications encompass various DSA problems, most of them are problem-based applications, where there is no theory and only problems. As of now, approximately 40% of developers worldwide use Hacckerank for practicing coding problems, making it the most used coding platform. A limitation of this application is that while an abundance of problems is provided, beginners find it difficult to determine which problems to solve. The proposed approach includes the development of an E-Learning web application that focuses on providing an interactive and intuitive learning experience for DSA concepts. The web application will feature visualizations, more specifically animations, to aid in understanding theoretical concepts. It will offer a section that recommends programming problems to users after they read all about the theory through the help of visualization. By providing users with targeted and relevant questions, the platform will enable them to practice DSA concepts effectively and enhance their programming skills. NLP algorithms are utilized by the recommendation system to evaluate user behaviour to generate suggestions. The recommendation system considers several elements, including the user's proficiency level, the topic of interest, and the difficulty level of the questions. The system also considers the questions' popularity and similarity to real-world problems to ensure that the questions are relevant and practical. For effective recommendation of programming problems, one of several pre-existing recommendation systems will be used. The first step is to identify the most relevant and effective programming problems from popular. Data will be collected on these problems, including the problem statement, difficulty level, concept covered, and popularity. After collecting data, it will be analysed to identify patterns and trends using NLP. On the identified patterns, the data points will be assigned a similarity score indicating their similarity with other data points. User feedback is collected on the recommended problems. A survey is created that asks users to rate the effectiveness of the recommended problems and provide feedback on how the recommended problems have helped them enhance their skills. Finally, the top-N problems will be used to recommend effective programming problems. The system can predominantly be used for personalized learning, skill development, and progress tracking.

Recommendation systems are a tool used for filtering information that focuses on problems by displaying desired results to the user in a personalized manner. They are popular among applications like e-commerce [1], tourism [2], and entertainment [3]. For example, in e-commerce, the digital activity of the online user gets tracked and analysed to suggest products [4]. They tend to captivate the user's interest and consider contextual information to make recommendations. As powerful recommendation systems are, they suffer from issues such as data redundancy, the abundance of information, and context redundancy [5]. A regular recommendation problem is divided into two parts; the first involves estimating how much the user will like the proposed item, and the second part focuses on providing the top-N items that the user prefers. The following steps are followed to recommend any item for any type of recommendation system. First, the system will denote the user as u_i and the item as v_j and it will look to grasp this representation. A scoring function gets mapped as $f : u_i \times v_j \rightarrow \hat{y}_{i,j}$, that represents the preference of the item regarding the user. At last, the system recommends the item by arranging the items based on the preference score [5]. A utility function is often developed to measure how well the recommendation systems perform. The function is represented as a U* I→R. R is the positive integer; I represents an item set and U is a user set. The utility function, R(u,i), has to be calculated for all the users u for each item i, in which the utility function is not yet known. One or more items i will get chosen to maximize the utility function as shown in (1) [6].

$$\forall u \in U,\ i = \arg\max_{i \in I}\ R(u,i) \tag{1}$$

The way of calculating accuracy is different for different recommendation systems. For recommended products, the accuracy gets calculated in the proportion of the quantity of recommended and purchased products to entire products in the data set. The ratio is given in the below formula (2):

$$P = \frac{|B \cap R|}{|R|} \tag{2}$$

where R refers to the recommendation set, and B refers to the purchase set. There are many more performance metrics.

Diversity refers to the amount of dissimilarity among the items recommended. The diversity will be higher if the system can recommend items as different from each other as possible, while still staying related to the user's interest. Coverage is defined as the percentage of items that the user can recommend from the dataset. Novelty can determine how different the current recommendation is from the previous recommendation. Serendipity is a mixture of several aspects, including usefulness, novelty, and unexpectedness. It is defined as the competence of the system to suggest items that are new and surprising to the user that they might not have discovered otherwise [7]. Most often, recommendation systems look to suffer from the problem of scalability. As the size of the dataset looks to increase, the performance of the system looks to degrade as it is not trained upon the varying type of data provided. For example, in movie recommendations, there can be different kinds of genres. A remedy to this issue is to reduce dimensionality using Single Value Decomposition (SVD) [8]. The different types of recommendation systems are discussed in the subsections below.

1.1. Content-based Recommendation System

CBF systems propose items to users based on their past interactions with similar items [9]. These systems rely on analysing some features and characteristics of items to generate recommendations [10]. In the scenario of DSA questions, a CBF system analyses some attributes of questions to suggest questions that are comparable to ones that a user has seen before.

The CBF systems, as shown in Fig.1, analyses user data such as their past responses to DSA questions, the topics they have studied, their preferred learning style, and the level of difficulty of the questions they have attempted. Based on this data, the system can recommend questions that match the user's level of proficiency, interests, and learning style. This kind of recommendation system can be mainly useful in the context of learning, where it can give personalized suggestions for students depending on the user's learning history and interests. The use of the CBF system to recommend questions to the users related to their prior question history is common in sectors like e-commerce. They are also commonly used in music and movie streaming services, where they are used to suggest new songs and movies to users based on their past preferences. One of the key benefits of CBF systems is that they do not need a huge amount of user data to generate accurate recommendations [11].

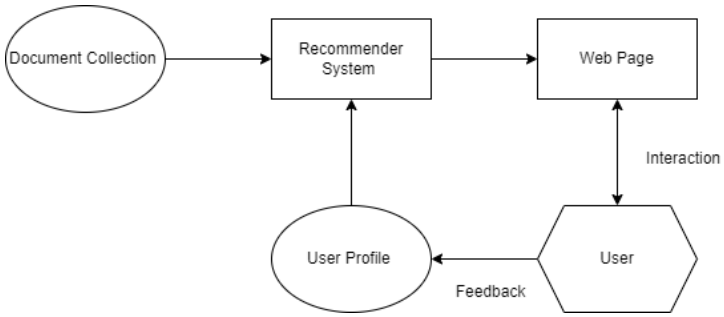


Fig.1. CBF recommendation system base architecture

As long as the system has enough dataset on the attributes of the items it is recommending, it can generate accurate recommendations based on that limited data alone. Another advantage of CBF systems is that they are highly transparent. Since the suggestions are established on some attributes of the items being recommended, users can easily understand why they are being recommended a particular item. This transparency can be especially valuable in the context of learning, where students need to understand why they are being recommended a particular question to learn more effectively.

1.2. Collaborative Filtering Recommendation System

CF is a technique used to make predictions about the preferences or interests of a user by analysing the behaviours of other similar users. Fig.2 represents the architecture of a CF system to recommend items to users. Through this analysis, the system can make predictions about what a particular user might be interested in, even if they have never interacted with that item before. CF is separated into two categories: user-based and item-based. With user-based CF, the system detects users who reflect the target user's tastes. It suggests items that the users looked to have enjoyed or connected with. In item-based CF, the system discovers questions that are similar to the ones that a target user has already liked or engaged with and recommends those similar items. Another recommendation name for CF looks to filter items based on the preference of similar users. For example, to recommend movies, firstly the movie ratings are collected through various users, and then the movies are suggested based on similar history [12].

One of the major advantages of CF is that it doesn't need any explicit knowledge or information about the items being recommended. Instead, the system learns from the behaviours and preferences of users, which makes it very useful in situations where users may not be able to provide detailed feedback or ratings on items. However, CF also has some limitations. For example, it can be susceptible to the "cold start" issue, during which the system lacks data and knowledge of new users or objects. In addition, it can be difficult to find users or items that are truly similar, which can lead to inaccurate recommendations.

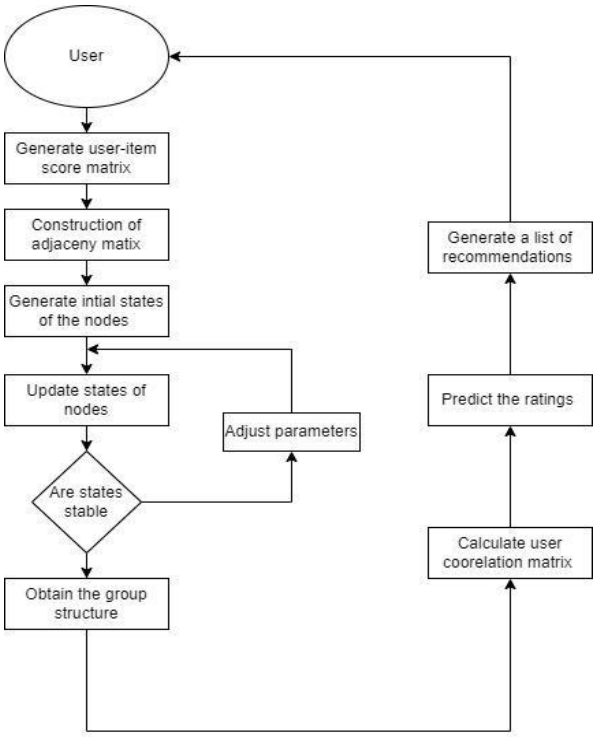


Fig.2. CF recommendation systems architecture

To overcome these limitations, researchers have developed a wide range of techniques to enhance the accuracy and efficiency of CF. Some of these techniques include using hybrid recommendation systems that combine CF with other approaches, such as CBF or demographic filtering. Other methods lean towards using deep learning (DL) techniques to analyze more complex data, such as user-generated content or social network interactions [13].

Overall, CF is a powerful technique for making recommendations that are personalized. They can aid users in discovering novel products, services, or content that they might not have otherwise discovered. As the amount of data generated by users and devices continues to grow, CF is likely to become even more important in helping users navigate the vast landscape of digital content and services.

1.3. Hybrid Recommendation System

Hybrid recommendation systems combine multiple recommendation techniques to offer more precise and effective recommendations to users. They leverage the strengths of different recommendation approaches to overcome their weaknesses, resulting in a more comprehensive and personalized recommendation system [14]. There are several reasons why hybrid recommendation systems are becoming increasingly popular. First, they are better able to handle the diversity of user preferences and the complexity of recommendation tasks, as no single recommendation technique can provide optimal recommendations for all users and situations. Second, they are more robust to changes in the input data and can adapt to new user preferences or trends in real time. Finally, they can help mitigate the "cold start" issue, during which the system has no data or information related to new users or items, by using other data sources to make predictions. Scenarios like these occur for new items that have gotten no ratings from the user, thus having no past knowledge [14]. There are different types of hybrid recommendation systems, depending on how they combine recommendation techniques similar to that displayed in Fig. 3. One common approach is to use weighted averaging, where the predicted ratings or recommendations from different techniques are combined based on their respective accuracy or relevance. Another approach is to use switch-based methods, where different recommendation techniques are selected based on the context or characteristics of the user or item being recommended. Yet another approach is to use feature-based methods, where the recommendations are generated by modelling the user and item features in a joint space.

One popular type of hybrid recommendation system is the content-based CF method which merges CBF with CF. In this approach, the system first analyses the content of items and identifies their features and attributes, such as genre, author, or keywords. Then, it uses CF to find equivalent users or items based on their interactions and preferences and combines the results with content-based recommendations to generate the final recommendations. This approach can provide more accurate and diverse recommendations, as it leverages both the content and the social context of the items being recommended. CF and CBF systems tend to showcase the communication between users and items statically and only captivate the general preference of users. On the other hand, a sequential recommendation system looks to recommend items by capturing the dependencies of the sequential process to take into consideration the current and recent likes of a user to improve the accuracy of the system [15].

Another type of hybrid recommendation system is the knowledge-based approach, which merges domain-specific knowledge with user preferences to generate recommendations. In this approach, the system first acquires domain knowledge about the items being recommended, such as their properties, relationships, or constraints. Then, it uses this knowledge to generate recommendations that satisfy the user's preferences while also satisfying the domain-specific constraints. This approach can be particularly useful in domains where the items being recommended have complex relationships or dependencies, such as in healthcare or finance [16]. Overall, hybrid recommendation systems provide a powerful and flexible framework for building personalized recommendation systems that can adapt to a wide range of user preferences and contexts. They combine the benefits of many recommendation techniques and data sources while also mitigating the limitations of individual techniques. As the amount and complexity of data continue to grow, hybrid systems are likely to play an increasingly major role in helping users navigate the vast landscape of digital content and services.

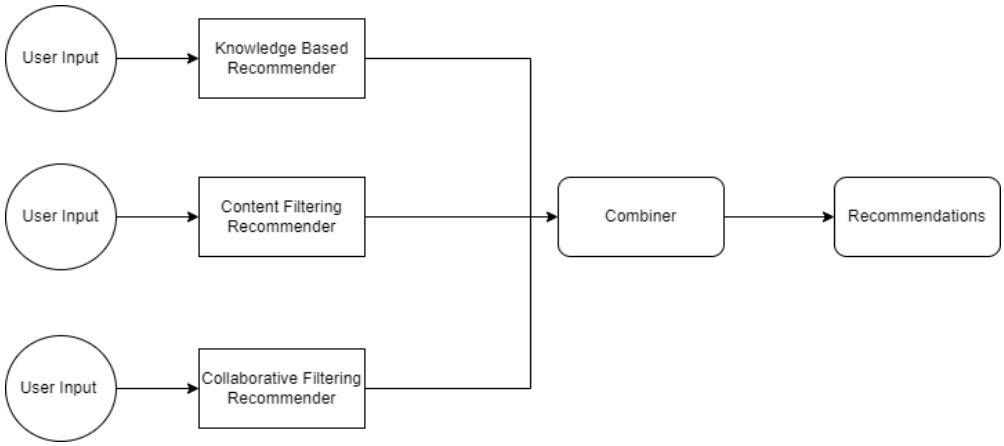


Fig.3. Hybrid recommendation system base architecture

The primary goal of this work is to develop a CBF recommendation engine that suggests users' DSA problems that are module-specific, i.e., arrays, linked lists, trees, etc. A dataset of over 500 problems was considered, and features like problem domain, problem statement, and difficulty level were considered to recommend the problems. This manuscript provides the following inputs.

- Propose a CBF recommendation system that recommends DSA problems for different modules.
- The recommendation system incorporates the Term Frequency – Inverse Document Frequency (TF-IDF) to determine the importance of words in a DSA problem statement.
- Four evaluation metrics, namely, accuracy, precision, recall, and F1-score, are used to determine the performance of the system.
- The integration of recommendation systems within the E-Learning web application.

The rest of the article is structured as follows. Section 2 presents the literature review. Section 3 illustrates the proposed architecture along with its description. Section 4 outlines the results and discussion. Section 5 concludes the work and future scope of work.

2. Background Study

In this section, several state-of-the-art methods are reviewed involving CBF, CF, and hybrid recommender systems.

CF systems are popular to use and are often considered for use in multiple domains. For example, Nassar et al.[17] present a CF recommendation system incorporated along Deep Learning (DL). The architecture is composed of two components where the former retrieves the features of users and items, and the latter is a deep neural network (DNN) using these features to predict the ratings of criteria.

Similarly, Bobadilla et al.[18] incorporate the prediction loss in the DL layers of a DNN to raise the recommendation quality. The DL architecture is composed of three components, namely, real prediction loss, predicted loss, and estimated ratings.

Aljunid and Dh [19] used a DL-based CF system to recommend movies by working on two popular datasets: Movielens 100K and 1M MovieLens. Using an evaluation metric referred to as Root Mean Squared Error (RMSE), they achieved an RMSE of 0.917 on the Movielens 100K datasets and an RMSE of 0.903 on the MovieLens 1M dataset.

Cui et al. [20] addressed the issue of recommendation systems neglecting the relationship between time and user preference by insinuating a recommendation system that is dependent on the time correlation coefficient along with an improved K-means clustering algorithm with cuckoo search. The system is evaluated on two datasets, namely, Movelens and Douban.

Often CF systems suffer from the 'cold start problem' preventing it from working on new, unseen items. This was resolved by Natarajan et al. [21] where they found a solution to data sparsity and the 'cold start' problem by proposing a Recommender System with Linked Open-data (RS-LOD) and Matrix Factorization technique with Linked Open Data (MF-LOD), respectively.

Zheng et al. [22] proposed Deep Conn, a joint DL model that incorporates user and item reviews to make personalized recommendations. The authors argue that CF approaches do not take advantage of the content in reviews, which can provide valuable information about users' preferences and item characteristics.

H.Wang et al. [23] insinuated a collaborative DL approach recommendation systems that jointly learn user and item visualizations. The model, called Collaborative Deep Learning (CDL), combines a CF approach with a deep neural network to captivate both user-item interactions and their latent characteristics.

J.Tan et al. [24] designed a neural network-based technique to quote recommendations in texts, which aims to automatically suggest relevant and meaningful quotes to users as they write. The authors argue that existing methods rely on simple rules or statistics studies and do not review the context and semantics of the writing.

Z.Xu et al. [25] developed a hybrid deep-semantic matrix factorization approach for a tag-aware personalized recommendation. The authors argue that existing methods either ignore the tag information or rely on simple tagging models that do not capture the complex relationships between tags and items.

Y.Wu et al. [26] incorporated the Collaborative Denoising Auto-Encoder (CDAE) approach in [19] for Top-N recommendation systems. The authors argue that traditional CF approaches suffer from data sparsity and cannot effectively capture complex user-item interactions.

V.Kumar et al. [27] implemented a DL architecture for news recommendation that leverages both content and context information. The authors argue that existing news recommendation methods mainly rely on CF or CBF approaches and do not consider the temporal dynamics of news and user preferences.

A.Van Den Oord et al. [28] proposed a deep content-based music recommendation approach that leverages the audio content of music tracks. The proposed model, called Deep Content-based Music Recommendation (DCMR), works on CNN to learn the audio attributes of the music tracks and a deep neural network to model the user preferences.

Singhal et al. [29] provided a summary of recent works on the use of DL in modern recommendation systems, where they showed DL techniques have promising results in capturing complex user-item interactions and improving the accuracy and diversity of recommendations.

S.Li et al. [30] portrayed a deep CF approach that leverages a marginalized denoising auto-encoder (MDAE) to learn the user and item representations.

H.Liang et al. [31] introduce a probabilistic rating auto-encoder (PRAE) model for personalized recommendation systems. The PRAE model aims to learn user and item embeddings that can capture the underlying user-item interactions by jointly optimizing the reconstruction error of the observed ratings and the prior distribution of the embeddings.

R.Devoought et al. [32] depicted the use of a CF model that leverages recurrent neural networks (RNNs) to capture the sequential dependencies in user-item interactions.

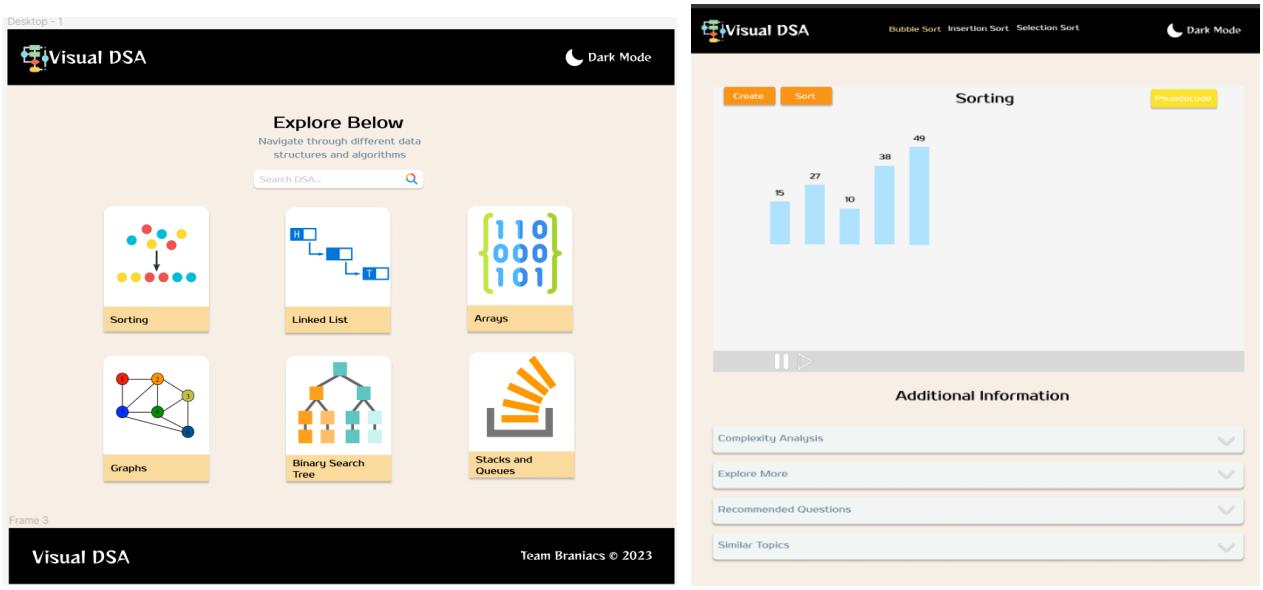
S. Deng et al. [33] suggested a DL-based approach for trust-aware recommendations in social networks. The authors evaluate the TARS-DL model on a real-world social network dataset and show that it outperforms several baseline recommendation algorithms. The paper presents a promising approach to a trust-aware recommendation that leverages DL to incorporate trust relationships in social networks.

While DL methods are becoming more popular, genetic algorithms are also vital methods used for solving optimization problems. Aljhijawi and Kilani [34] proposed a genetic-based CF recommendation system that revolves around semantic details and the past data of ratings.

In this paper, a novel CBF system is proposed to suggest DSA problems to the user, integrated within the E-Learning application.

3. Methodology

CBF technique was applied to custom-made data to suggest users' DSA problems based on the given topic. This will allow the students to know which problems to focus on and follow a sequence of problems to solve. The recommendation engine looks to be integrated within the architecture across all the different modules corresponding to different DSA. The idea behind CF, as depicted in Fig. 2, is that users who have comparable prior preferences or objectives are likely to have identical future preferences and interests. Fig. 4-a depicts the interface for the home page from where the user may select any of the DSA they wish to explore, and Fig. 4-b represents the DSA-specific webpage from where the user can get information about that DSA. Here all the information, along with the results of the recommendation engine, is displayed. Fig. 5 represents the architecture of the entire e-learning application, starting from the page loading to navigating through the different modules.



(a) (b)

Fig.4. User interface for E-Learning web application showing (a) Home page and (b) DSA-specific page

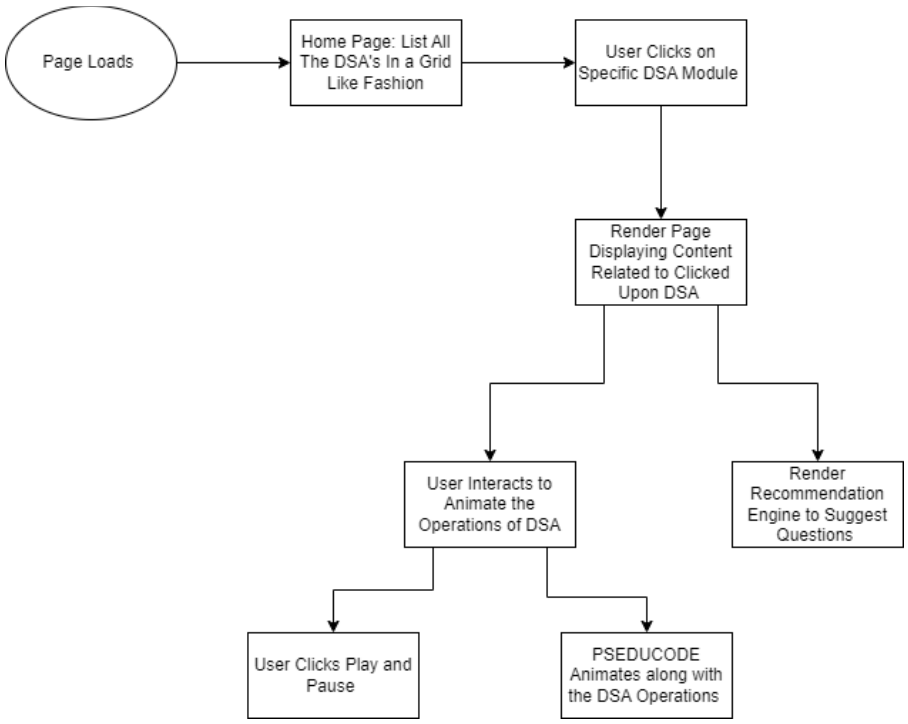


Fig.5. E-Learning application flowchart

3.1. Dataset Formation

Since there was no available dataset for this problem, a CSV data file was created that included more than 500 problems and their relevant topics, difficulty, problem statement, and the link to the coding platform on which that problem exists.

3.2. Creating a TF-IDF Vectorizer

Several attributes or features have to be retrieved for the item getting recommended. These features can be retrieved through a popular model called the vector space model associated with frequency-inverse document frequency weighting. TF-IDF algorithm looks to measure the significance of each word in a corpus of text. A word is considered more important if it looks to appear in a greater number of items in that particular textual corpus. The algorithm is split into two components: TF and IDF. TF evaluates the frequency of the term, whereas IDF evaluates the importance of the term [35]. TF-IDF is an approach that allows weighing the significance of every term in a document. The weight of TF-IDF weight is corresponding to the frequency of the terms in the document; however, it is inversely proportionate to the frequency of

the term in the corpus. The higher the TF-IDF score, the more important that term is [35]. TF-IDF allows the generation of a vector that looks to represent each item (DSA Problem). This vector will look to captivate the most important terminologies in the item's description. The similarity between the vectors can be calculated.

The TF-IDF vector gets generated by first pre-processing the text through tokenizing it. Tokenizing refers to the process of splitting words into useful and meaningful units [36]. Secondly, the stop words get removed. These are the words that hold no real value in terms of meaning. Words like he, she, it, etc., can be considered stop words [37]. Text normalization gets performed as well to reduce the randomness and bring it in proximity to predefined standards.

The TF-IDF weight gets computed for the important term in the item's description by the following equations:

$$tf_i = \frac{n_i}{\sum_{k=i}^{n} n_k} \tag{3}$$

Where n_i represents the number of times the term i is present, and n_k is considering all the terms in the corpus.

$$IDF = \log\left(\frac{|D|}{|\{d_j : t_i \in d_j\}|}\right) \tag{4}$$

D refers to a set of the total number of documents, and t_i refers to the documents belonging to set d_j containing the i term.

$$W_{x,y} = tf_{x,y} * \log\frac{N}{dfx} \tag{5}$$

where $W_{x,y}$ refers to the TF-IDF, which is a product of TF and IDF, $tf_{x,y}$ is the frequency of x in y, df_x is the number of documents consisting of x, and N is the total amount of documents present. The output will be a TF-IDF matrix that contains each word and its associated score concerning each document or item. Thus, each item is now represented concerning its description.

3.3. Vector Space Model

A vector will represent the items, the DSA problem statement, or the question, and it allows us to evaluate the similarity between the items and users based on their history. The vector will represent the item as a high-dimensional vector, as shown in Fig. 6. Each dimension is associated with a specific feature of that item. The different dimensions that can correspond to the DSA question can be the relevant keywords in the DSA topic, difficulty level, and user ratings. Each dimension gets depicted by a numerical value, which can be categorical or continuous. The recommendation will be made depending on the degree of resemblance between the item and the preference of the user. The similarity is computed using Cosine similarity, which looks to measure the cosine angle between the two vectors. More precisely, the cosine takes the angle between the user profile vector, denoted by U_i and the document vector. Fig. 7 represents the graph showing the cosine angle between two DSA-related problems. The value obtained ranges from -1 to 1, where 1 indicates that the vectors are identical, 0 refers to them being orthogonal, and -1 indicates that they are opposite. Equation (6) illustrates the way to calculate the cosine similarity

$$sim(A,B) = \cos(x,y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \tag{6}$$

where x and y are the two document vectors.

3.4. Making Recommendation

Based on the TF-IDF vector calculated and the cosine similarity calculated between the different documents, the system will be able to suggest a list of items. The DSA problems with the highest similarity score to the given input will be recommended. Since the output is a matrix showing the likeness between the items, that will be challenging for a user to interpret; hence, it is significant to reverse map the topics to the matrix. Fig. 8 portrays the architecture of the CBF systems for suggesting DSA problems from the 500+ dataset.

```
array([0.76393042, 0.76169129, 0.76188086, 0.76167808, 0.7619731 ,
       0.76189184, 0.76182575, 0.76159416, 0.76192402, 0.7621033 ,
       0.76162185, 0.76232014, 0.76184372, 0.76286434, 0.76170008,
       0.76176955, 0.76159416, 0.76165578, 0.76163257, 0.76247176,
       0.76217505, 0.76207472, 0.7618235 , 0.7616694 , 0.76205301,
       0.76189914, 0.76182032, 0.7616852 , 0.7623866 , 0.76164435,
       0.76163112, 0.76179245, 0.76167313, 0.76163263, 0.76178763,
       0.76159416, 0.7617345 , 0.76159416, 0.76165275, 0.76167425,
       0.76163515, 0.76167055, 0.76165428, 0.76161717, 0.76188711,
       0.76164719, 0.7616387 , 0.76164521, 0.76166742, 0.76165307,
       0.76160415, 0.76162087, 0.76162148, 0.76164361, 0.761729  ,
       0.76160904, 0.76159416, 0.76172257, 0.76214622, 0.76217007,
       0.76230518, 0.76159781, 0.76185216, 0.76170384, 0.76162644,
       0.76159416, 0.76159416, 0.76160823, 0.76165345, 0.76184424,
       0.76232528, 0.76215756, 0.76207881, 0.76165718, 0.76165718,
       0.76165718, 0.76163448, 0.76163271, 0.76163601, 0.76162824,
```

Fig.6. Vector space model for a DSA recommendation problem

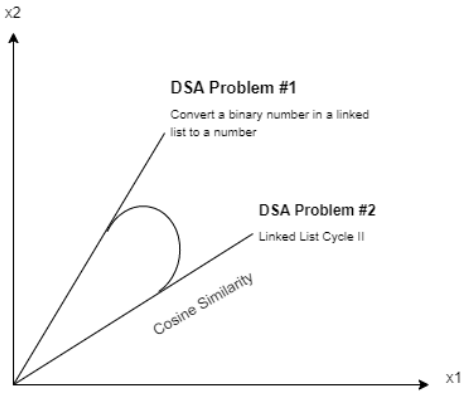


Fig.7. Graph representing the cosine angle between two similar DSA problems

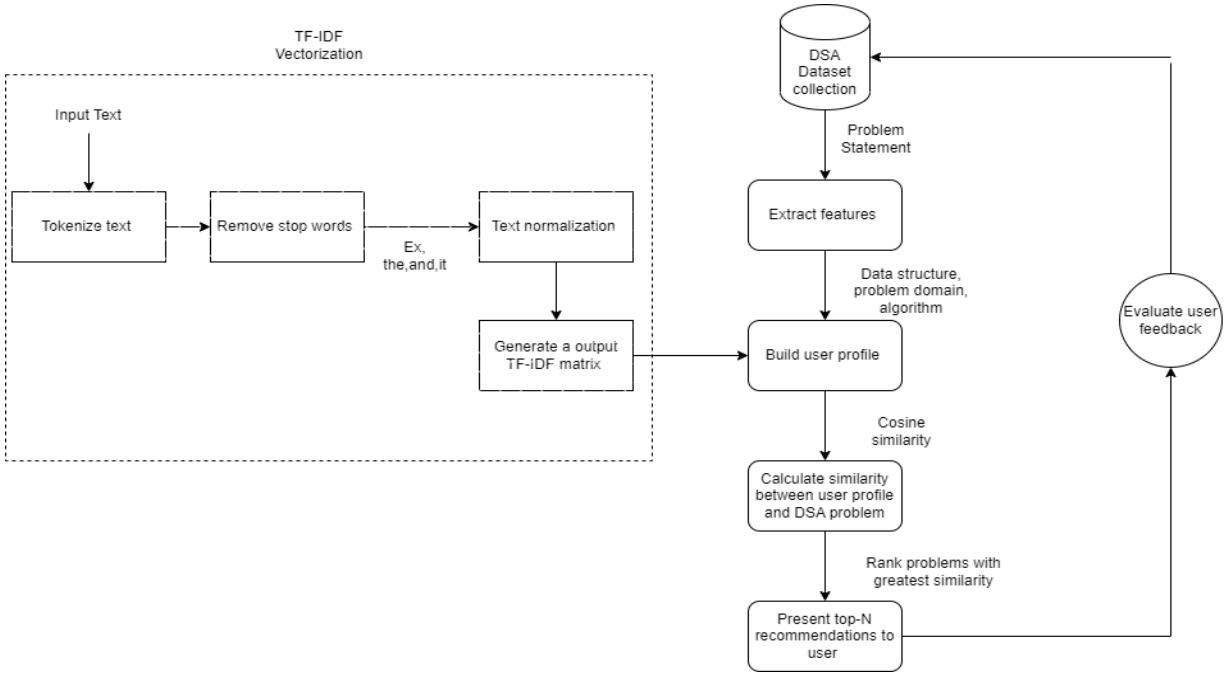


Fig.8. DSA CBF architecture

3.5. Evaluating Recommendation Engine

For CBF systems, similarity metrics are often used to calculate the performance of the systems. Cosine Similarity shown by (5) can be used and is most preferred for high-dimensional features. Accuracy is a popular evaluation metric that determines the proportion of instances classified correctly. Equation (7) represents the way to calculate the Accuracy.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

where TP refers to true positives, TN is true negatives, FP is false positives, and FN is false negatives. Precision is a similar metric that calculates the proportion of predictions that are true positive out of the total quantity of positive predictions. Equation (8) indicates the way to compute the value

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

Recall computes the ratio of true positive predictions to the entire set of positive instances. Equation (9) represents the formula for calculating recall.

$$Recall = \frac{TP}{TP + FP} \tag{9}$$

F1-score looks to merge both recall and precision, providing a trade-off between the two-evaluation metrics. The value ranges from 0 to 1, where a higher score indicates better precision and recall. Equation (10) calculates the F1-score.

$$F_1 - score = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall} \right) \tag{10}$$

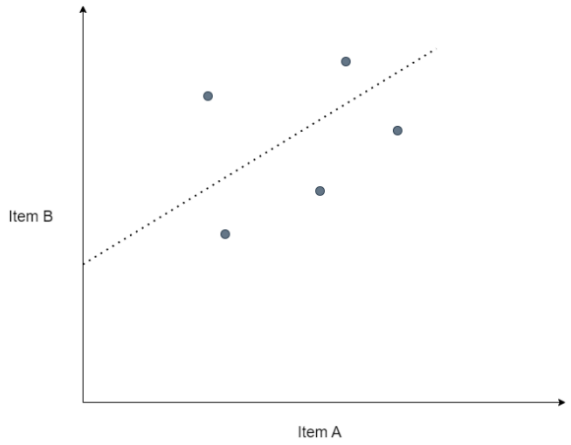


Fig.9. Correlation between item A and item B

Fig. 9 looks to depict the positive correlation between two items, thus indicating that the suggested item, whomever it may be, A or B, is correct.

4. Results and Discussion

By using the CBF recommendation, it was observed that it operated well on the small dataset of 500 problems. Fig. 10 illustrates the cosine matrix showing the degree of similarity between all the problems. As Equation (6) shows the formulation of cosine similarity, the higher the value indicates that the two problems are more similar, hence more likely to be recommended. Subsequently, all the diagonals are having a cosine similarity of 1, insinuating that the row and column at that diagonal contain the same problem. A value of below 0.5 indicates that the two respective problems are not so similar in any of the features, such as problem statement, topic, and difficulty level.

```
[[1.         0.04140753 0.12226158 ... 0.15735125 0.01260105 0.01766376]
 [0.04140753 1.         0.76640207 ... 0.00457901 0.01835213 0.00960912]
 [0.12226158 0.76640207 1.         ... 0.0332264  0.01168297 0.011247  ]
 ...
 [0.15735125 0.00457901 0.0332264  ... 1.         0.14026823 0.21374148]
 [0.01260105 0.01835213 0.01168297 ... 0.14026823 1.         0.15115105]
 [0.01766376 0.00960912 0.011247   ... 0.21374148 0.15115105 1.        ]]
```

Fig.10. Cosine Matrix indicating the degree of similarity between all the 500 problems

A classification report has been built, as shown in Table 1, on a test dataset of 40 tree problems, where it shows how well the system was able to recommend tree-related problems while the user is on the tree module in the E-Learning application. Similarly, several classification reports were built for different DSAs, indicating their recommendation with respective problems. An overall accuracy of 83% is achieved with a precision of 83%, recall of 80%, and F1-score of 80%. An accuracy of 80% is interpreted as out of the 500 problems, roughly 400 problems were correctly recommended. Cosine similarity is also calculated since it is more suitable for sparse data and more robust to the variable document lengths. Fig. 11 represents the visualization of various evaluation metrics on the CBF system. It is observed that while the system is accurate and precise, it falls short, as compared to accuracy and precision, in terms of f1-score and recall.

Table 1. Classification Report for tree problems

Semester	Precision	Recall	F1-score	Support
Binary Tree Inorder Traversal	0.82	0.80	0.83	10
Binary Tree Level Order	0.83	0.80	0.84	10
Binary Tree Maximum Path Sum	0.82	0.78	0.77	0
Binary Tree Preorder Traversal	0.83	0.79	0.84	10
Diameter of a Binary	0.84	0.80	0.85	10
Accuracy			0.83	40
Macro Avg	0.83	0.80	0.80	40
Weighted Avg	0.83	0.80	0.83	40

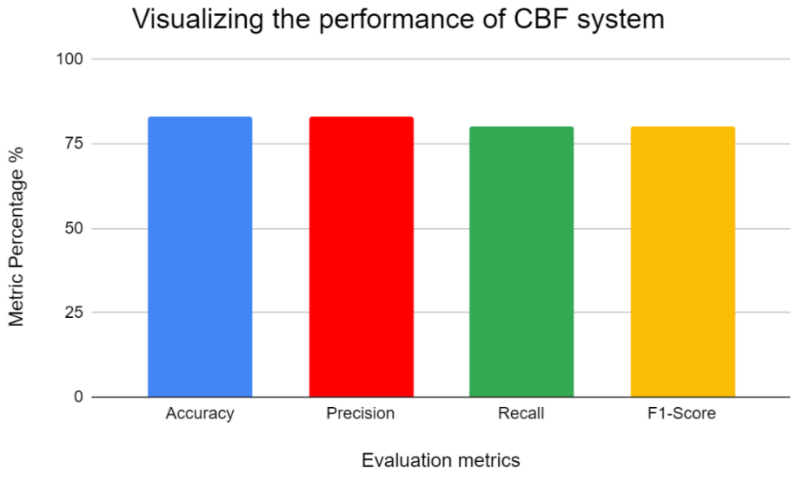


Fig.11. Bar graph illustrating the performance of CBF on various evaluation metrics

5. Conclusions

From the work conducted, it can be observed that recommendation systems act as ubiquitous tools in a world of information overload. With current coding platforms being problem-centric, with minimal focus on which problems to solve for a beginner in DSA, this proposed system addresses that limitation by suggesting DSA-specific problems in the E-Learning application. These systems analyse the data provided, learn from it, and present an appropriate recommendation for the user. This work focused on implemented CBF, where a dataset of approximately 500 problems was used. It achieved an overall accuracy of 83%, a precision of 83%, an F1-score of 80%, and a recall of 80%. In addition, a cosine similarity matrix is constructed to show the relationship between all the problems. A higher value, like 0.7 or above, indicates that the two respective problems are similar and more likely to be recommended. A classification report in Table 1 for all the tree problems depicts the accuracy, precision, recall, and f1-score for predicting the problems correctly. This recommendation system is integrated within a web application that serves as an E-learning application for many computer science enthusiasts. The recommendation system will recommend problems with whichever module (Arrays, Sorting, Graphs, etc) the user has interacted with. This web application was tested in real-time to observe the working of the recommendation engine within the web application. For different modules, it was decided that the top 5 problems would be displayed to the user on the web page. This recommendation has the main use of helping aspiring computer science students and aspiring software engineers who have begun their DSA journey. Future research looks to implement more recommendation systems, such as CF and Hybrid recommendation systems, to compare the performances and deduce which system is most preferred for the E-Learning application. Moreover, the dataset size can be increased considering a wider range of topics such as Dynamic Programming, Heaps, Shortest Path Algorithms, etc from various coding platforms.

References

[1] Karthik, R., & Ganapathy, S. (2021). A fuzzy recommendation system for predicting the customer's interests using sentiment analysis and ontology in e-commerce. *Applied Soft Computing*, *108*, 107396. https://doi.org/10.1016/j.asoc.2021.107396

[2] Ray, B., Garain, A., & Sarkar, R. (2021). An ensemble-based hotel recommender system using sentiment analysis and aspect categorization of hotel reviews. *Applied Soft Computing*, *98*, 106935. https://doi.org/10.1016/j.asoc.2020.106935

[3] Batmaz Z. Yurekli A. Bilge A. & Kaleli C. (2019). A review on deep learning for recommendation systems: challenges and remedies. Artificial Intelligence Review 52 1-37.

[4] Cabrera-Sánchez J. P. Ramos-de-Luna I. Carvajal-Trujillo E. & Villarejo-Ramos Á. F. (2020). Online recommendation systems: Factors influencing use in e-commerce. Sustainability 12(21) 8888.

[5] Guo Q. Zhuang F. Qin C. Zhu H. Xie X. Xiong H. & He Q. (2020). A survey on knowledge graph-based recommendation systems. IEEE Transactions on Knowledge and Data Engineering 34(8) 3549-3568.

[6] Shambour Q. (2021). A deep learning based algorithm for multi-criteria recommendation systems. Knowledge-based systems 211 106545.

[7] Raza S. & Ding C. (2022). News recommendation system: a review of recent progress challenges and opportunities. Artificial Intelligence Review 1-52.

[8] Singh R. H. Maurya S. Tripathi T. Narula T. & Srivastav G. (2020). Movie recommendation system using cosine similarity and KNN. International Journal of Engineering and Advanced Technology 9(5) 556-559.

[9] Javed U. Shaukat K. Hameed I. A. Iqbal F. Alam T. M. & Luo S. (2021). A review of content-based and context-based recommendation systems. International Journal of Emerging Technologies in Learning (iJET) 16(3) 274-306.

[10] Zhang Q. Lu J. & Jin Y. (2021). Artificial intelligence in recommendation systems. Complex & Intelligent Systems 7 439-457.

[11] Dhawan S. (2019 February). Comparision of Recommendation System Approaches. In 2019 International Conference on Machine Learning Big Data Cloud and Parallel Computing (COMITCon) (pp. 76-78). IEEE.

[12] Ahuja R. Solanki A. & Nayyar A. (2019 January). Movie recommendation system using k-means clustering and k-nearest neighbor. In 2019 9th International Conference on Cloud Computing Data Science & Engineering (Confluence) (pp. 263-268). IEEE.

[13] Ali S. Hafeez Y. Humayun M. Jamail N. S. M. Aqib M. & Nawaz A. (2022). Enabling recommendation system architecture in virtualized environment for e-learning. Egyptian Informatics Journal 23(1) 33-45

[14] Urdaneta-Ponte M. C. Mendez-Zorrilla A. & Oleagordia-Ruiz I. (2021). Recommendation systems for education: systematic review. Electronics 10(14) 1611.

[15] Wang S. Hu L. Wang Y. Cao L. Sheng Q. Z. & Orgun M. (2019). Sequential recommendation systems: challenges progress and prospects. arXiv preprint arXiv:2001.04830.

[16] S. Meng et al., "Privacy-Aware Factorization-Based Hybrid Recommendation Method for Healthcare Services," in IEEE Transactions on Industrial Informatics, vol. 18, no. 8, pp. 5637-5647, Aug. 2022, doi: 10.1109/TII.2022.3143103.

[17] Nassar, N., Jafar, A., & Rahhal, Y. (2020). A novel deep multi-criteria collaborative filtering model for recommendation system. *Knowledge-Based Systems*, *187*, 104811. https://doi.org/10.1016/j.knosys.2019.06.019

[18] Bobadilla, J., Alonso, S., & Hernando, A. (2020). Deep Learning Architecture for Collaborative Filtering Recommender Systems. *Applied Sciences*, *10*(7), 2441. https://doi.org/10.3390/app10072441

[19] Aljunid, M. F., & Dh, M. (2020). An Efficient Deep Learning Approach for Collaborative Filtering Recommender System. *Procedia Computer Science*, *171*, 829-836. https://doi.org/10.1016/j.procs.2020.04.090

[20] Z. Cui et al., "Personalized Recommendation System Based on Collaborative Filtering for IoT Scenarios," in IEEE Transactions on Services Computing, vol. 13, no. 4, pp. 685-695, 1 July-Aug. 2020, doi: 10.1109/TSC.2020.2964552.

[21] Natarajan, S., Vairavasundaram, S., Natarajan, S., & Gandomi, A. H. (2020). Resolving data sparsity and cold start problem in collaborative filtering recommender system using Linked Open Data. *Expert Systems with Applications*, *149*, 113248. https://doi.org/10.1016/j.eswa.2020.113248

[22] Zheng Lei Noroozi Vahid and Yu Philip S.. 2017. "Joint Deep Modeling of Users and Items Using Reviews for Recommendation". In Proceedings of the 10th ACM International Conference on Web Search and Data Mining (WSDM'17). ACM 425–434. DOI: 10.1145/3018661.3018665

[23] H. Wang N. Wang and D.-Y. Yeung "Collaborative Deep Learning for Recommendation Systems" in Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2018 pp. 1235–1244.

[24] J. Tan X. Wan and J. Xiao "A Neural Network Approach to Quote Recommendation in Writings" Proc. 25th ACM Int. Conf. Inf. Knowl. Manag. - CIKM '16 pp. 65–74 2016

[25] Z. Xu C. Chen T. Lukasiewicz and Y. Miao "Hybrid Deep-Semantic Matrix Factorization for Tag-Aware Personalized Recommendation" Aug. 2020.

[26] Y. Wu C. DuBois A. X. Zheng and M. Ester "Collaborative Denoising Auto-Encoders for Top-N Recommendation Systems" in Proceedings of the Ninth ACM International Conference on Web Search and Data Mining - WSDM '16 2019 pp. 153–162.

[27] V. Kumar D. Khattar S. Gupta and M. Gupta "Deep Neural Architecture for News Recommendation" in Working Notes of the 8th International Conference of the CLEF Initiative Dublin Ireland. CEUR Workshop Proceedings 2020.

[28] A.van den Oord S. Dieleman and B. Schrauwen "Deep content-based music recommendation" Electron. Inf. Syst. Dep. p. 9 2022.

[29] Use of Deep Learning in Modern Recommendation System: A Summary of Recent Works Ayush Singhal Pradeep Sinha Rakesh Pant 2017

[30] S. Li J. Kawale and Y. Fu "Deep Collaborative Filtering via Marginalized Denoising Auto-encoder" in Proceedings of the 24th ACM International on Conference on Information and Knowledge Management - CIKM '15 2015 pp. 811–820.

[31] H. Liang and T. Baldwin "A Probabilistic Rating Auto- encoder for Personalized Recommendation Systems" in Proceedings of the 24th ACM International on Conference on Information and Knowledge Management - CIKM '15 2015 pp. 1863–1866.

[32] R. Devooght and H. Bersini "Collaborative Filtering with Recurrent Neural Networks" Aug. 2016.

[33] S. Deng L. Huang G. Xu X. Wu and Z. Wu "On Deep Learning for Trust-Aware Recommendations in Social Networks" IEEE

Trans. Neural Networks Learn. Syst. vol. 28 no. 5 pp. 1164–1177 2017.
[34] Alhijawi, B., & Kilani, Y. (2020). A collaborative filtering recommender system using genetic algorithm. *Information Processing & Management*, *57*(6), 102310. https://doi.org/10.1016/j.ipm.2020.102310
[35] Chiny M. Chihab M. Bencharef O. & Chihab Y. (2022). Netflix Recommendation System based on TF-IDF and Cosine Similarity Algorithms. no. Bml 15-20.
[36] Ozaydin B. Zengul F. Oner N. & Delen D. (2017). Text-mining analysis of mHealth research. Mhealth 3.
[37] Fan Y. Arora C. & Treude C. (2023). Stop Words for Processing Software Engineering Documents: Do they Matter?. arXiv preprint arXiv:2303.10439.

Authors' Profiles

Aayush Juyal is a student pursuing his B.Tech in Computer Science with specialization in Artificial Intelligence and Machine Learning at Sharda University, India. His area of focus is primary Software Engineering, Machine Learning, and Deep Learning.

Nandini Sharma is a student pursuing her B.Tech in Computer Science with a specialization in Artificial Intelligence and Machine Learning at Sharda University, India. Her area of research involves Blockchain, Machine Learning, Artificial Intelligence, and Deep Learning.

Pisati Rithya is a student pursuing her B.Tech in Computer Science with a specialization in Artificial Intelligence and Machine Learning at Sharda University, India. Her area of research involves Deep Learning Computer Vision, and Natural Language Processing.

Dr. Sandeep Kumar is an associate professor in the Department of Computer Science Engineering, School of Engineering and Technology, Sharda University. He holds a Ph.D. Degree in Computer Science. His areas of research include Data Mining, Fractal Graphics, and Artificial Intelligence.

How to cite this paper: Aayush Juyal, Nandini Sharma, Pisati Rithya, Sandeep Kumar, "An Enhanced Approach to Recommend Data Structures and Algorithms Problems Using Content-based Filtering", International Journal of Intelligent Systems and Applications(IJISA), Vol.15, No.5, pp.28-40, 2023. DOI:10.5815/ijisa.2023.05.03