Research Paper Comparative Analysis of Traditional and Deep Learning-based Recommendation Systems: A Case Study in E-Learning Platforms

Submitted in partial fulfillment of the requirements for the degree of

Diploma in University Technology

in

Computer Engineering

by

Rhouli Mohamed Mounim

S130169594

Under the guidance of

Dr. Abdelali Ibriz,

Higher School of Technology of Fez

Department of Computer Engineering

University Mohamed Ben Abdellah of Fes

جامعة سيدي محمد بن عبدالله بفاس
UNIVERSITÉ SIDI MOHAMED BEN ABDELLAH DE FES

المدرسة العليا للتكنولوجيا
ECOLE SUPÉRIEURE DE TECHNOLOGIE

April 2024

ACKNOWLEDGEMENTS

The project “Research Paper Comparative Analysis of Traditional and Deep Learning-based Recommendation Systems: A Case Study in E-Learning Platforms” was made possible because of valuable inputs from everyone involved, directly or indirectly. I would like to thank my supervisor, **Dr. Abdelali Ibriz**, fortheir invaluable guidance, support, and encouragement throughout the entire process of researching and writing this thesis. His expertise and insightful comments have been instrumental in shaping my work and helping me to achieve my academic goal.

I would also like to thank **the Higher School of Technology of Fez**, for providing us with a flexible choice in the execution of the project and for providing me with an excellent academic environment and the necessary resources.

Finally, I am immensely grateful to my **parents** for their unwavering love,encouragement, and support throughout my academic journey. Their sacrifices and dedication have been the driving force behind my success, and I cannot thank them enough for everything they have done for me.

ABSTRACT

This research paper provides a comprehensive comparison between traditional recommendation systems and those enhanced by deep learning methods, focusing on e-learning platforms. It initially explores traditional recommendation systems and subsequently examines the integration of deep learning within recommendation systems.

Subsequently, the paper delves into the implementation intricacies of recommendation algorithms employing both conventional and deep learning methodologies. Traditional methodologies scrutinized encompass knowledge-based recommendation leveraging rule-based systems, content-based recommendation employing TF-IDF, and collaborative filtering utilizing KNN. Conversely, deep learning methodologies are explored through sophisticated models including Latent Dirichlet Allocation (LDA), neural network-based architectures, LSTM with attention, self-attention models, and the Neural Collaborative Filtering (NCF) model. Additionally, the paper discusses Hybrid Recommendations Using a Switching Mechanism Between Models.

We thoroughly compare traditional recommendation methods with deep learning-based approaches in e-learning platforms. We highlight the drawbacks of traditional methods and the benefits of using deep learning for recommendations.

In summary, this research emphasizes the effectiveness of deep learning-based recommendation systems compared to traditional approaches in e-learning platforms. It highlights how deep learning methods can significantly improve recommendation accuracy, thereby enhancing user experiences and engagement in online learning contexts.

Keywords: Recommendation Systems, Traditional Methods, Deep Learning, E-Learning Platforms, Knowledge-Based Recommendation, Rule-Based Systems, Content-Based Recommendation, TF-IDF, Collaborative Filtering, Latent Dirichlet Allocation (LDA), Neural Networks, LSTM with Attention, Self-Attention Models, Neural Collaborative Filtering (NCF), Hybrid Recommendations, Switching Mechanism, Comparison, Effectiveness, Limitations.

LIST OF FIGURES

Table of Contents

INTRODUCTION

In today's digital landscape, Recommendation Systems (RS) stand as indispensable tools, especially within e-learning platforms, where the abundance of information poses a significant challenge. These systems act as invaluable guides, navigating through vast repositories of educational content to provide tailored recommendations aligned with individual learner preferences. Our research is dedicated to exploring the transformative potential of deep learning within recommendation systems, with a specific focus on e-learning platforms.

Through an in-depth investigation of both traditional and deep learning-based methodologies, our aim is to illuminate the inner workings of these systems within the dynamic realm of online education. Our study centers around an e-learning platform, where the quest for personalized learning experiences reigns supreme.

We embark on our exploration by delving into traditional approaches such as content-based filtering and collaborative filtering, unraveling the intricacies of recommendation algorithms tailored specifically to the e-learning domain. Furthermore, we venture into the realm of deep learning techniques, where neural networks harness their predictive capabilities to enhance the recommendations provided to learners.

Ultimately, our research seeks to enrich the ongoing discourse surrounding recommendation systems by providing a nuanced understanding of the profound impact and potential of deep learning within the dynamic domain of e-learning platforms.

1.Implementation of Recommendation Systems in E-Learning Platforms Using Traditional Methods

1.1Overview of rraditional recommendation systems

Recommendation Systems (RS) play a pivotal role in information retrieval, providing users with personalized recommendations tailored to their preferences. These systems operate within various contexts, including e-commerce, entertainment, social networking platforms, and e-learning platforms, where they play a vital role in enhancing learner engagement. Traditional recommendation systems encompass three primary categories: content-based filtering, collaborative filtering, and knowledge-based systems. These approaches have undergone extensive research and development over the years, contributing to their widespread application across diverse domains.

Content-Based Filtering: Content-based filtering recommends items based on their attributes or descriptions. It analyzes the features of items, such as keywords or genres, to generate recommendations. For example, in movie recommendations, films are categorized based on genres like action, comedy, or thriller, and users are recommended movies similar to those they have previously enjoyed.

Collaborative Filtering: Collaborative filtering focuses on user-item interactions, such as ratings or purchases. It identifies similarities between users or items based on their past interactions. Collaborative filtering techniques include user-based and item-based recommendation approaches. User-based recommendation identifies users with similar preferences and recommends items liked by those users. Item-based recommendation identifies items similar to those previously liked by the user.

Knowledge-Based Systems: Knowledge-based systems leverage domain knowledge, user preferences, and item attributes to provide recommendations. These systems are particularly useful when there is limited user interaction data available or when dealing with cold start problems. Knowledge-based systems may incorporate rule-based or case-based approaches to infer user preferences and generate recommendations. They often combine user ratings, item attributes, and domain knowledge to offer personalized recommendations.

1.2 Techniques used to Implement our Recommendation System

1.2.1 Knowledge-Based Recommendation Using a Rule-Based System

Presentation of the Algorithm:

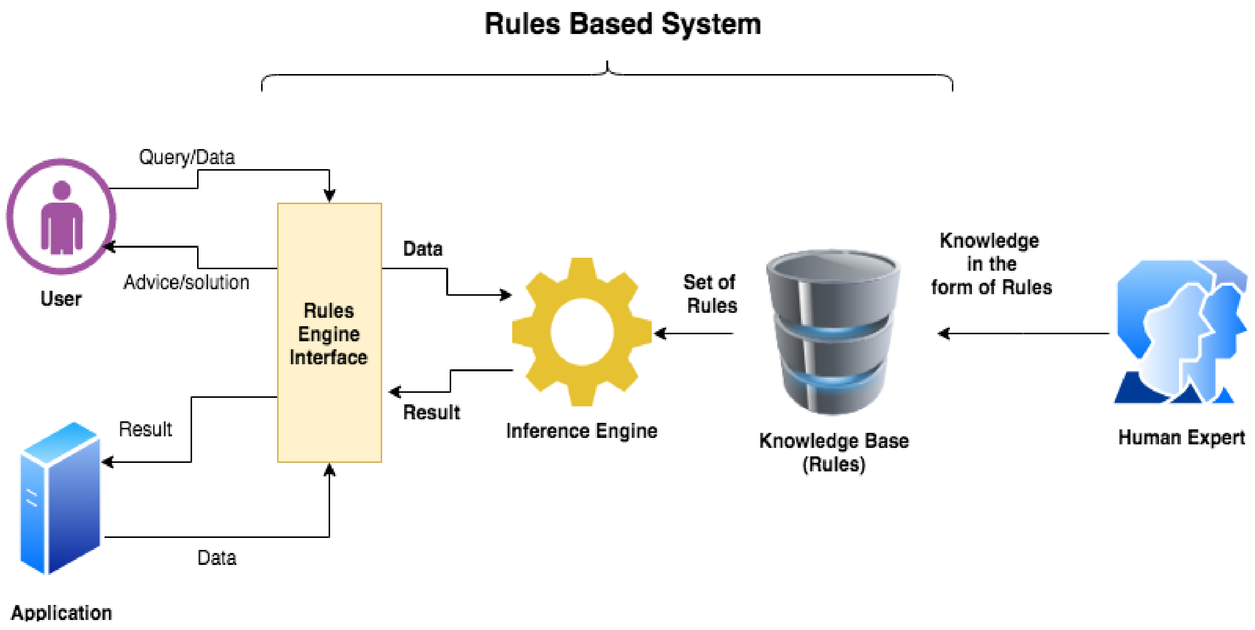


Figure 1 Rule-based system process

A rule-based recommendation system is a system that provides suggestions to users based on predefined rules rather than their past choices.

Recommendation Process for Courses Using this Algorithm:

1. User Category Identification: We began by identifying the category to which the user belongs by extracting this information from user data.

2. Course Filtering by Category: Next, we filtered course data to retain only the courses belonging to the same category as the user.

3. Course Recommendation: After obtaining the list of courses in the same category as the user, we recommended all these courses to the user.

1.2.2 Content-Based Recommendation with Term Frequency-Inverse Document Frequency (TF-IDF)

Presentation of the Algorithm:

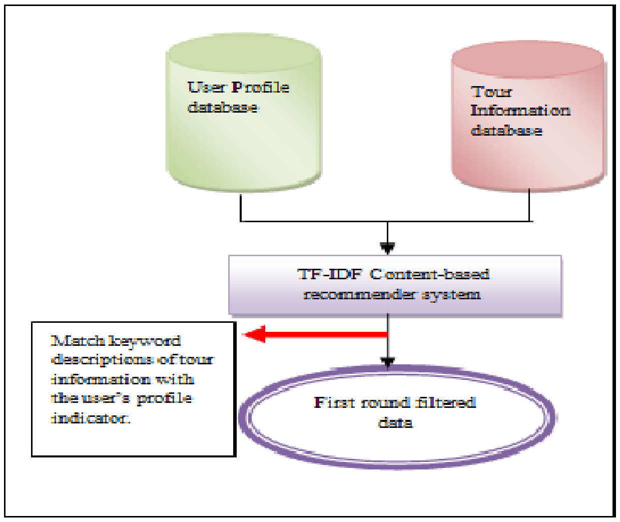

User Profile database
Tour Information database
TF-IDF Content-based recommender system
Match keyword descriptions of tour information with the user's profile indicator.
First round filtered data

Figure 2 TF-IDF Process

Term Frequency - Inverse Document Frequency (TF-IDF) is a widely used statistical method in natural language processing and information retrieval. It measures the importance of a term in a document relative to a collection of documents (i.e., a corpus).
Words in a text document are transformed into importance scores through a process of text vectorization. There are many different text vectorization scoring schemes, with TF-IDF being one of the most common.

As the name suggests, TF-IDF vectorizes/scores a word by multiplying the term frequency (TF) of the word by the inverse document frequency (IDF):
Term Frequency: The term frequency (TF) is the number of times the term appears in a document relative to the total number of words in the document.

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

Inverse Document Frequency: IDF of a term reflects the proportion of documents in the corpus that contain the term. Words unique to a small percentage of documents (e.g., technical jargon terms) receive higher importance values than words common to all documents (e.g., "a", "the", "and").

IDF(t) = log(Total number of documents / Number of documents with term t in it).

The TF-IDF of a term is calculated by multiplying the TF and IDF scores:

TF-IDF(t) = TF(t) * IDF(t)

Recommendation Process for Courses Using this Algorithm:

1. Creation of TF-IDF Vectors: Firstly, we utilized the `TfidfVectorizer` class from scikit-learn to convert course descriptions into TF-IDF vectors. TF-IDF is a term weighting technique that represents each course description as a numerical vector based on the relative importance of words in the corpus.

2. Cosine Similarity Calculation: Next, we calculated the cosine similarity between the TF-IDF vectors of the courses the user has viewed/completed and the TF-IDF vectors of all other courses. Cosine similarity measures the similarity between two vectors in terms of orientation rather than magnitude, making it a suitable measure for comparing text vectors.

3. Recommendation of Similar Courses: After calculating cosine similarity, we identified the 5 most similar courses to each course viewed/completed by the user. We excluded the same course from the list of recommendations to avoid recommending a course already viewed/completed.

1.2.3 Collaborative Filtering with K-Nearest Neighbors (KNN)

Presentation of the Algorithm:

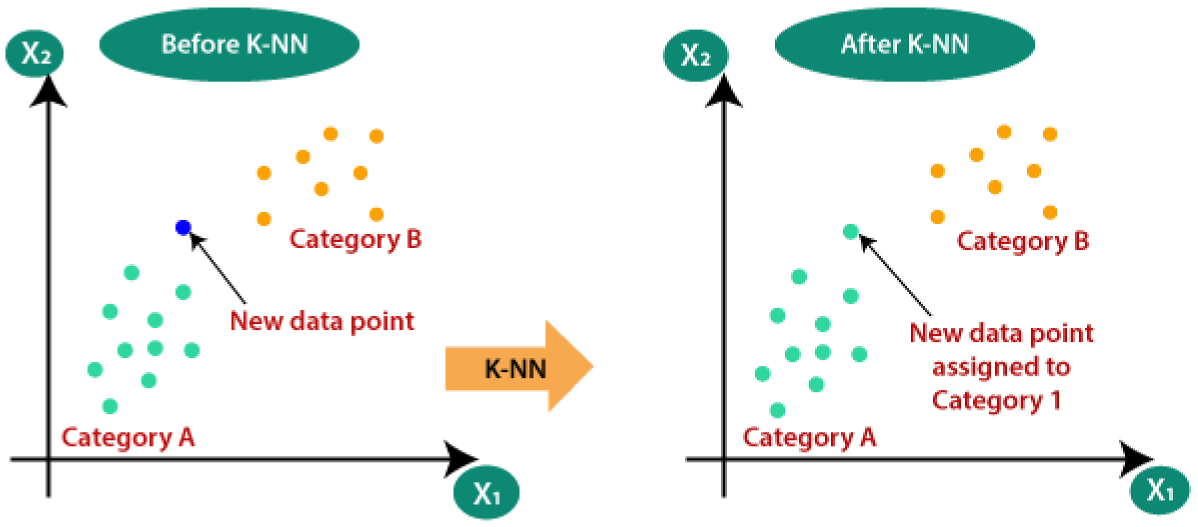


Figure 3 KNN Process

K-nearest neighbors (KNN) is a type of supervised learning algorithm used for regression and classification. KNN attempts to predict the correct class for test data by calculating the distance between the test data and all training points. It then selects the K nearest points to the test data. The KNN algorithm computes the probability that the test data belongs to the classes of the 'K' training data, and the class with the highest probability is selected.

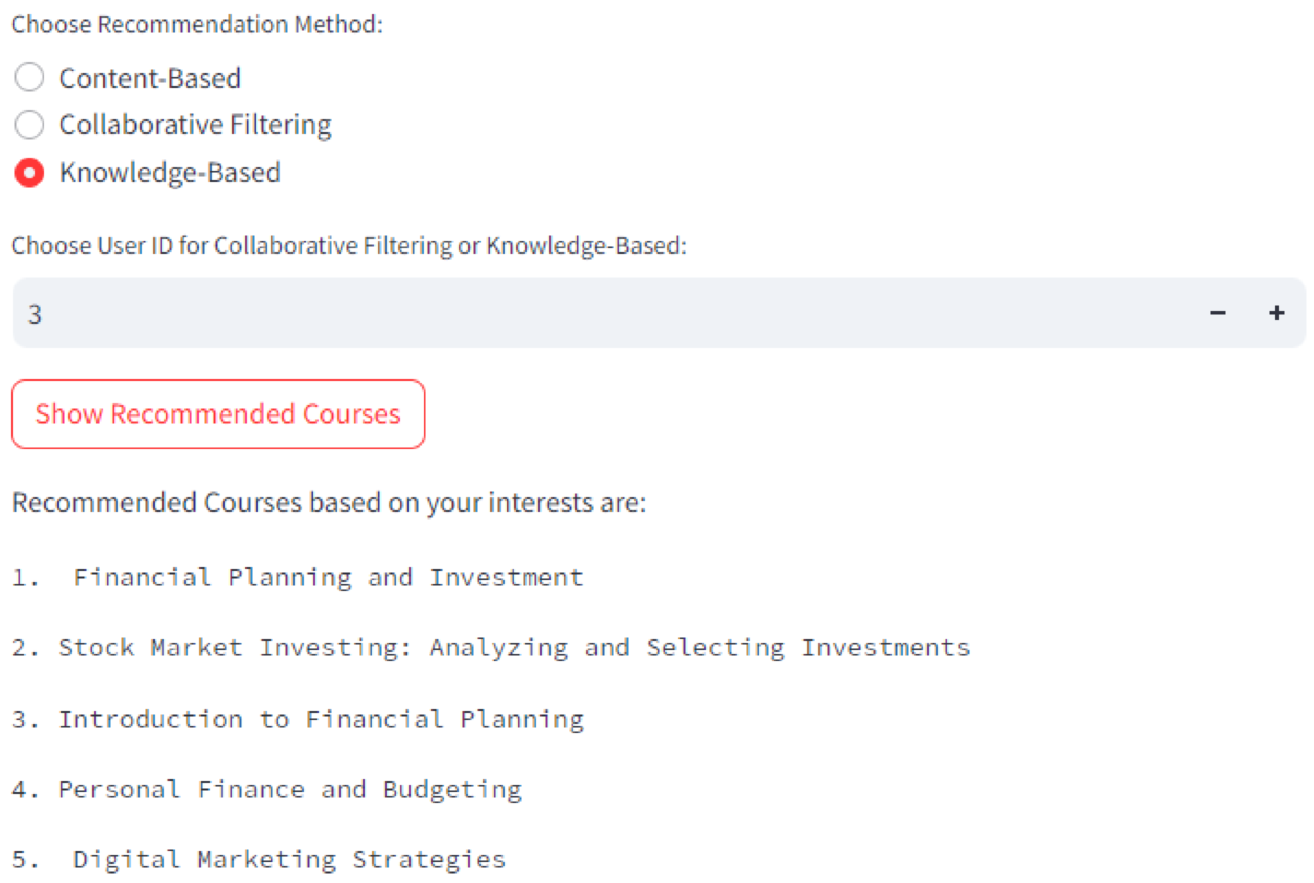
Course Recommendation Process Using the Algorithm:

1. Data Preprocessing:
   - Loading the dataset.
   - Filtering to include only positive interactions, where users both viewed and completed a course.
   - Removing rows with missing values in the 'user_name' or 'course_title' columns.

2. Training the KNN Model:
   - Converting the data into numeric format.
   - Training the KNN model on the filtered data.

3. Recommendation for Similar Users:
   - Defining a recommendation function based on similar users.
   - Using the KNN model to find similar users.
   - Extracting the courses viewed and completed by these similar users.
   - Returning the recommended courses for the given user.

1.3 Demonstration



Choose Recommendation Method:

Content-Based

Collaborative Filtering

[✓ Knowledge-Based]

Choose User ID for Collaborative Filtering or Knowledge-Based:

3

Recommended Courses based on your interests are:

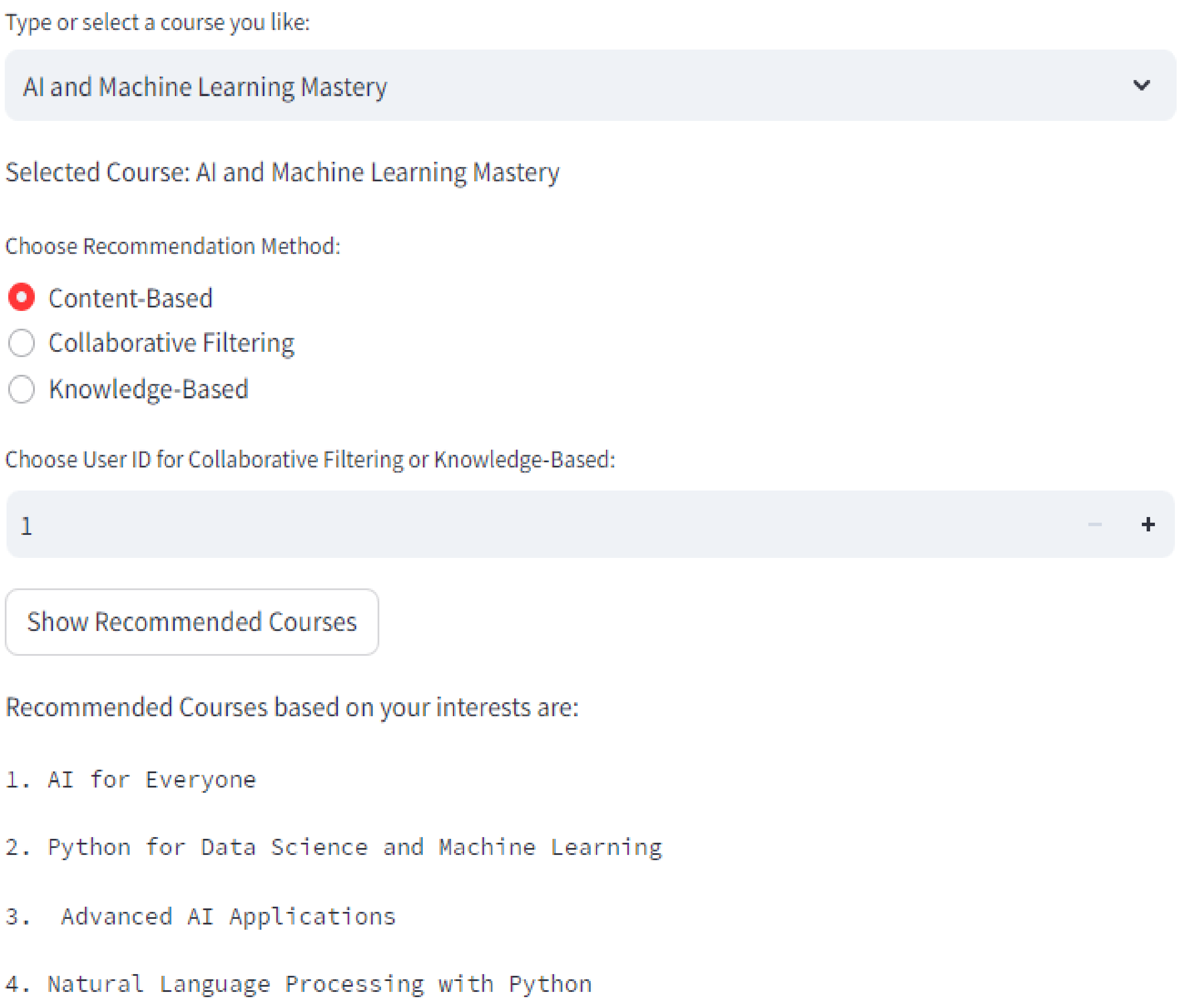
```
1.  Financial Planning and Investment

2. Stock Market Investing: Analyzing and Selecting Investments

3. Introduction to Financial Planning

4. Personal Finance and Budgeting

5.  Digital Marketing Strategies
```


Type or select a course you like:

AI and Machine Learning Mastery

Selected Course: AI and Machine Learning Mastery

Choose Recommendation Method:

[✓ Content-Based] Collaborative Filtering Knowledge-Based

Choose User ID for Collaborative Filtering or Knowledge-Based:

1

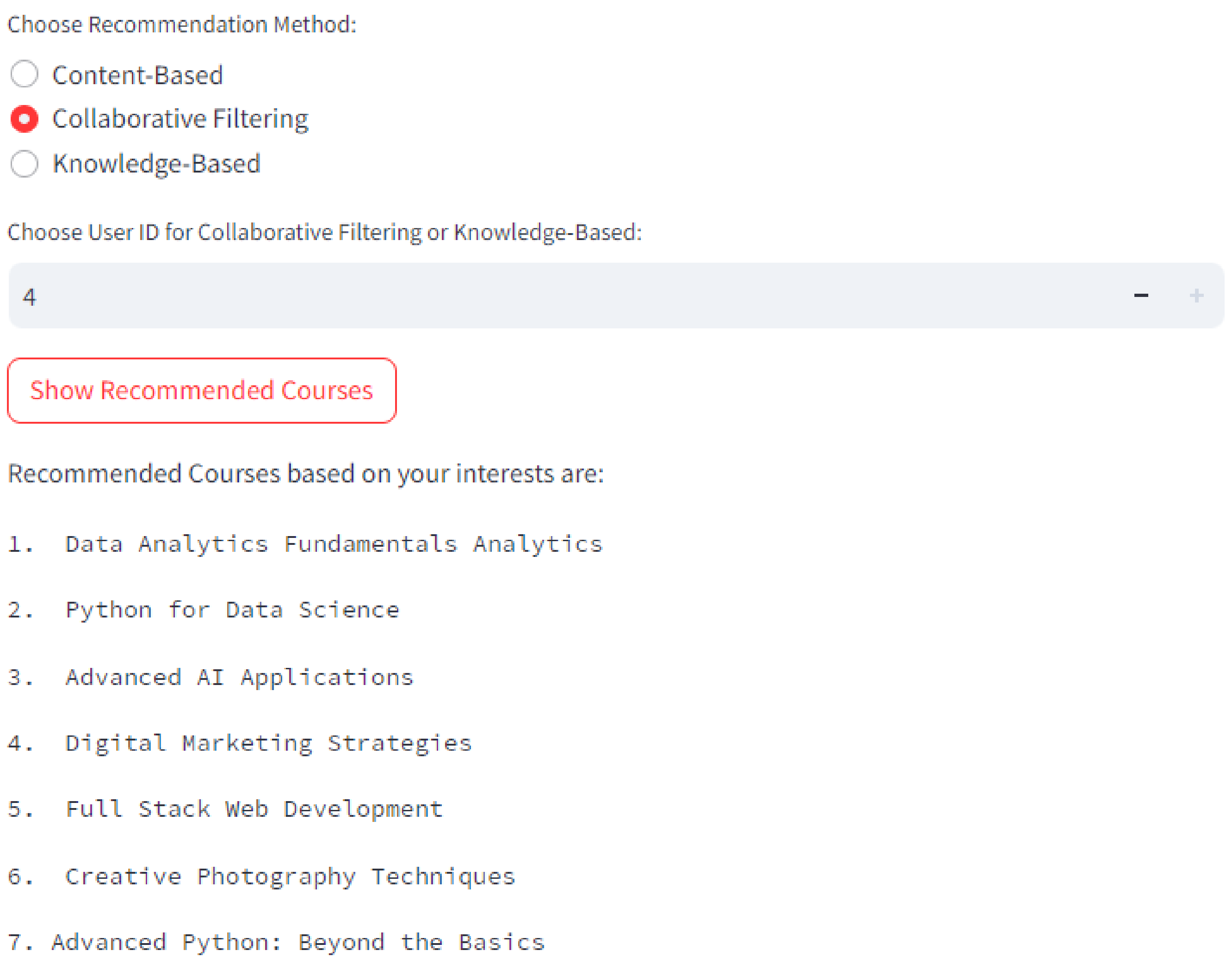
Recommended Courses based on your interests are:

```
1. AI for Everyone

2. Python for Data Science and Machine Learning

3.  Advanced AI Applications

4. Natural Language Processing with Python
```


Choose Recommendation Method:

Content-Based [✓ Collaborative Filtering] Knowledge-Based

Choose User ID for Collaborative Filtering or Knowledge-Based:

4

Recommended Courses based on your interests are:

```
1.  Data Analytics Fundamentals Analytics

2.  Python for Data Science

3.  Advanced AI Applications

4.  Digital Marketing Strategies

5.  Full Stack Web Development

6.  Creative Photography Techniques

7. Advanced Python: Beyond the Basics
```

1.4 Limitations of Traditional Approach

Recommendation Method	**Limitations**
Knowledge-Based Recommendation with Rule-Based Systems	Limited scalability: Rule-based systems rely heavily on manually defined rules, which can become complex and difficult to maintain as the dataset grows or as new features need to be incorporated. This can limit scalability and increase maintenance overhead. Lack of adaptability: Rule-based systems may struggle to adapt to changes in user preferences or trends without manual intervention to update rules. They often lack the ability to learn from data and adjust recommendations dynamically.

Content-Based Recommendation with TF-IDF	Limited personalization: TF-IDF-based content recommendation relies primarily on textual features of items. While it can capture similarities between items based on their content, it may not fully capture user preferences or provide highly personalized recommendations beyond textual information. Cold start problem: Content-based methods may face challenges when dealing with new users or items with limited interaction data. Without sufficient historical data, these methods may struggle to provide meaningful recommendations.
Collaborative Filtering with K-Nearest Neighbors (KNN)	Sparsity issues: KNN-based collaborative filtering relies on user-item interaction data. In scenarios where the dataset is sparse or there are few interactions between users and items, KNN may struggle to find meaningful similarities, leading to less accurate recommendations. Lack of interpretability:KNN does not provide explicit explanations for recommendations, reducing user trust.

2.Implementation of Recommendation Systems in E-Learning Platforms Using Deep Learning

2.1 Overview of Deep Learning in Recommendation Systems

Deep learning, a subset of machine learning, has revolutionized various fields, including image processing, natural language processing, and computer vision. Its impact on recommendation systems has been profound, offering novel approaches to personalized recommendations.

Broadly, the life-cycle of deep learning for recommendation can be split into two phases: training and inference. In the training phase, the model is trained to predict user-item interaction probabilities (calculate a preference score) by presenting it with examples of interactions (or non-interactions) between users and items from the past.

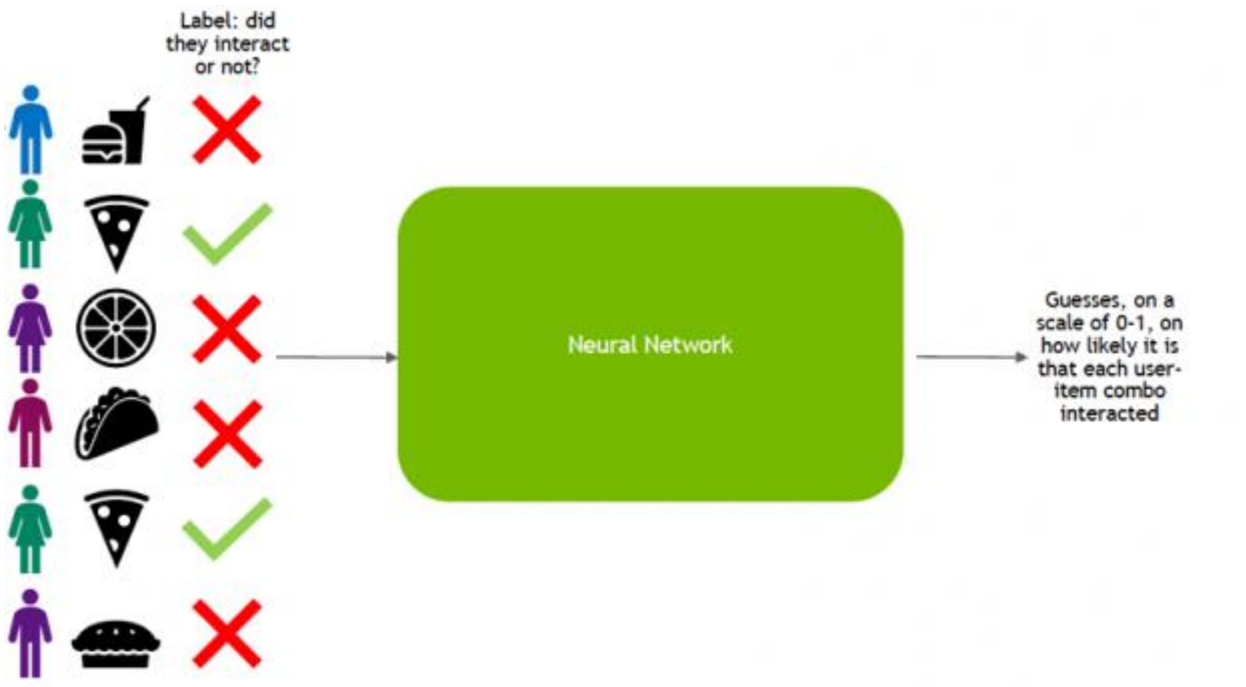

Label: did they interact or not?
Neural Network
Guesses, on a scale of 0-1, on how likely it is that each user-item combo interacted

Figure 4 Deep learning for recommendation training

Once it has learned to make predictions with a sufficient level of accuracy, the model is deployed as a service to infer the likelihood of new interactions.

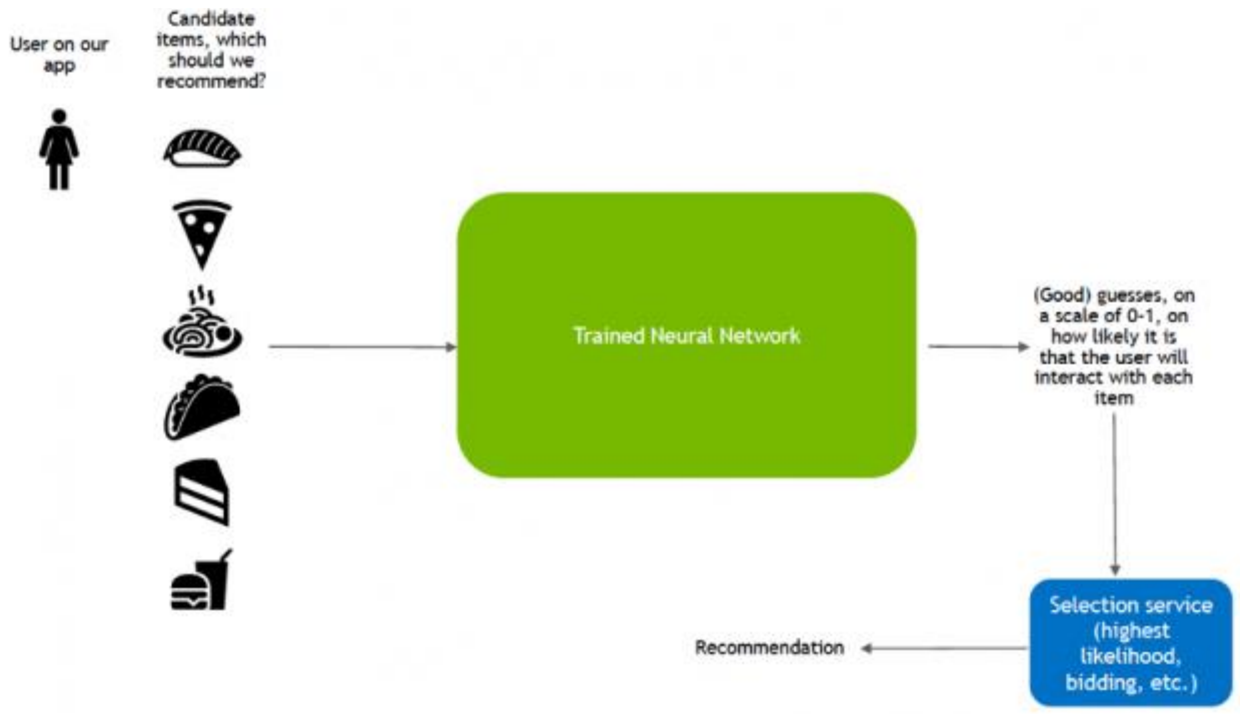

User on our app
Candidate items, which should we recommend?
Trained Neural Network
(Good) guesses, on a scale of 0-1, on how likely it is that the user will interact with each item
Selection service (highest likelihood, bidding, etc.)
Recommendation

Figure 5 Deep learning for recommendation inference

This inference stage utilizes a different pattern of data consumption than during training:

Candidate generation: pair a user with hundreds or thousands of candidate items based on learned user-item similarity.
Candidate ranking: rank the likelihood that the user enjoys each item.
Filter: show the user the item they are rated most likely to enjoy.

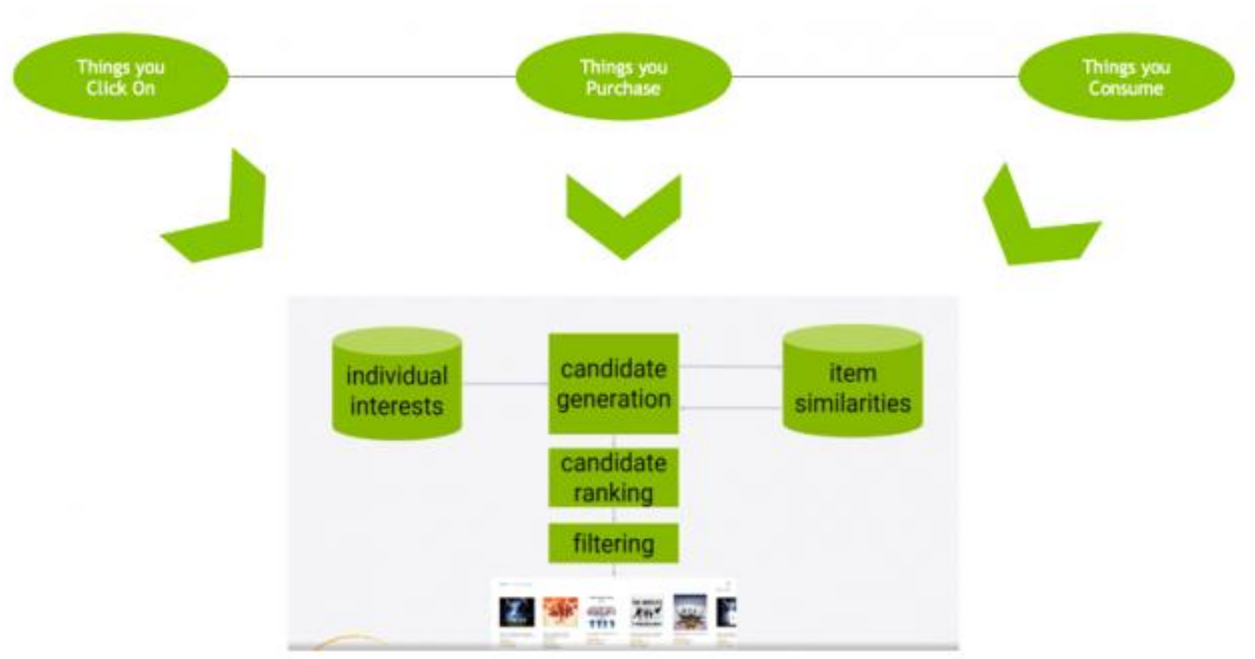


Figure 6 Deep learning for recommendation inference: candidate generation, ranking and filtering

Deep Neural Network Models for Recommendation:

Deep learning (DL) recommender models build upon existing techniques such as factorization to model the interactions between variables and embeddings to handle categorical variables. An embedding is a learned vector of numbers representing entity features so that similar entities (users or items) have similar distances in the vector space. For example, a deep learning approach to collaborative filtering learns the user and item embeddings (latent feature vectors) based on user and item interactions with a neural network.

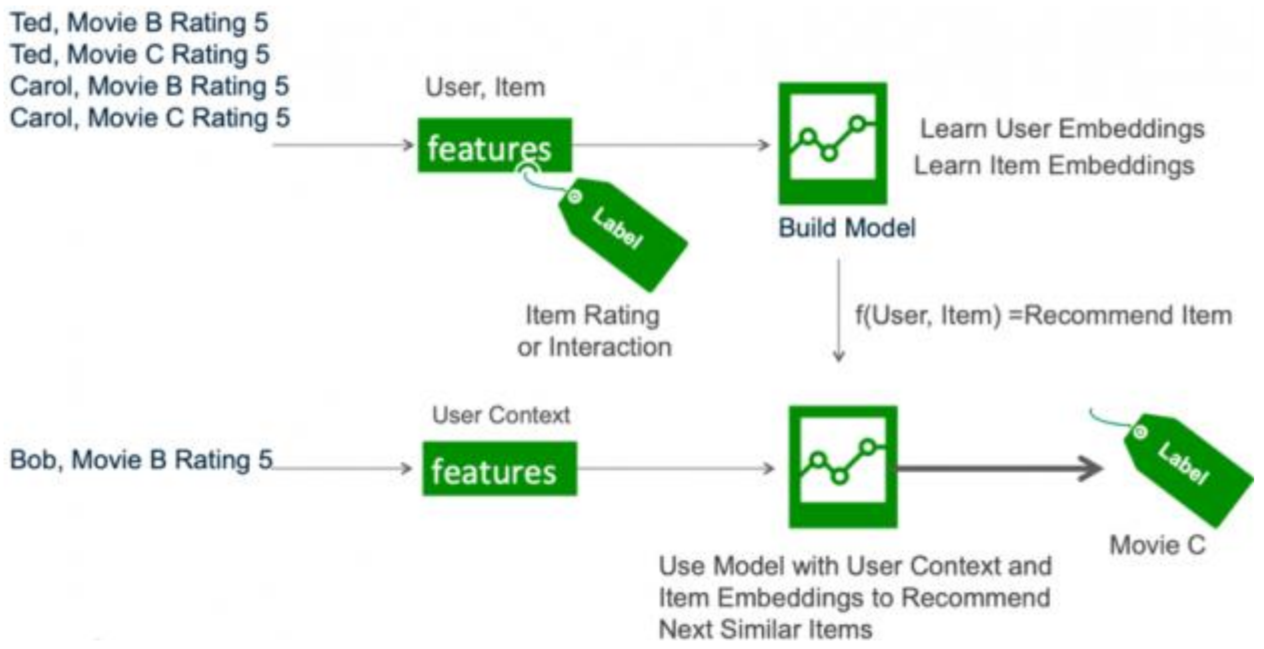


Figure 7 A deep learning approach to collaborative filtering learns the user and item embeddings based on user and item interactions

DL techniques also tap into the vast and rapidly growing novel network architectures and optimization algorithms to train on large amounts of data, use the power of deep learning for feature extraction, and build more expressive models. DL–based models build upon the different variations of artificial neural networks (ANNs), such as the following:

Feedforward neural networks are ANNS where information is only fed forward from one layer to the next. Multilayer perceptrons (MLPs) are a type of feedforward ANN consisting of at least three layers of nodes: an input layer, a hidden layer, and an output layer. MLPs are flexible networks that can be applied to a variety of scenarios.
Convolutional Neural Networks are the image crunchers to identify objects.
Recurrent neural networks are the mathematical engines to parse language patterns and sequenced data.

Neutral Collaborative Filtering

The Neural Collaborative Filtering (NCF) model is a neural network that provides collaborative filtering based on user and item interactions. The NCF model treats matrix factorization from a non-linearity perspective. NCF TensorFlow takes in a sequence of (user ID, item ID) pairs as inputs, then feeds them separately into a matrix factorization step (where the embeddings are multiplied) and into a multilayer perceptron (MLP) network.

The outputs of the matrix factorization and the MLP network are then combined and fed into a single dense layer which predicts whether the input user is likely to interact with the input item.

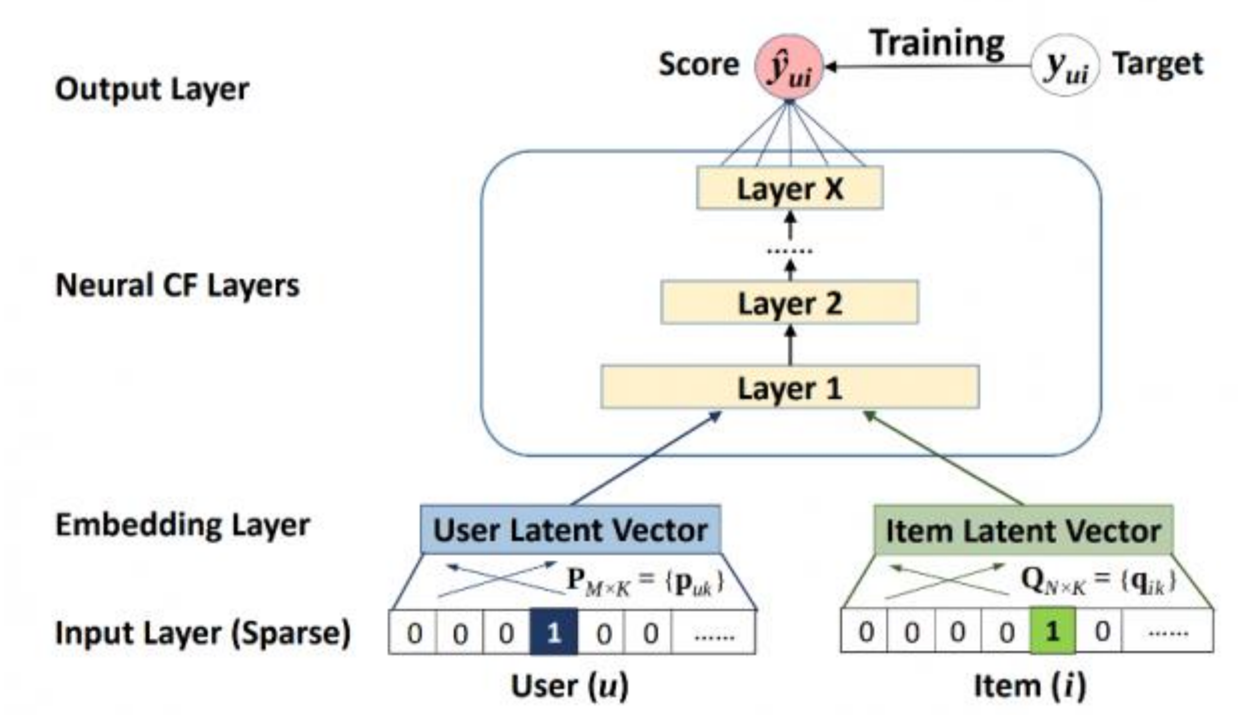


Figure 8 NCF model

Variational Autoencoder for Collaborative Filtering

An autoencoder neural network reconstructs the input layer at the output layer by using the representation obtained in the hidden layer. An autoencoder for collaborative filtering learns a non-linear representation of a user-item matrix and reconstructs it by determining missing values.

VAE-CF is a neural network that provides collaborative filtering based on user and item interactions. The training data for this model consists of pairs of user-item IDs for each interaction between a user and an item.

The model consists of two parts: the encoder and the decoder. The encoder is a feedforward, fully connected neural network that transforms the input vector, containing the interactions for a specific user, into an n-dimensional variational distribution. This variational distribution is used to obtain a latent feature representation of a user (or embedding). This latent representation is then fed into the decoder, which is also a feedforward network with a similar structure to the encoder. The result is a vector of item interaction probabilities for a particular user.

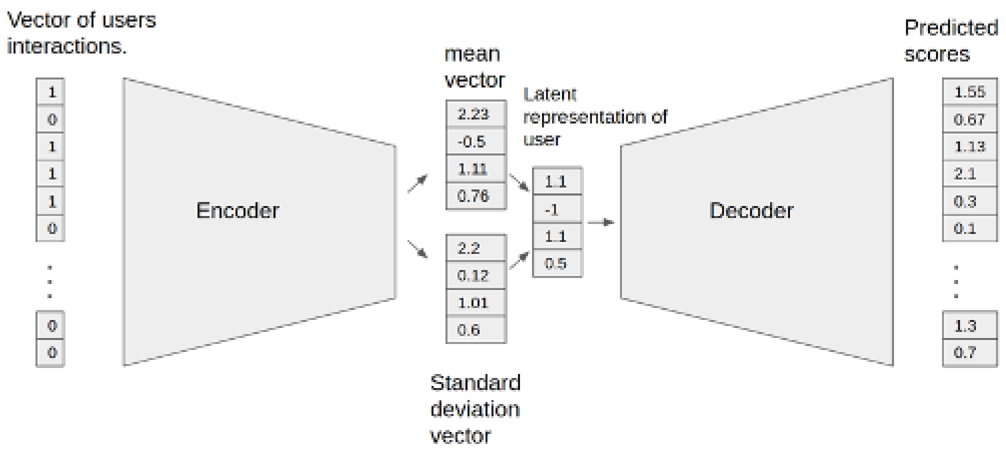


Figure 9 VAE-CF model

<u>Wide and Deep</u>

Wide & Deep refers to a class of networks that use the output of two parts working in parallel—wide model and deep model—whose outputs are summed to create an interaction probability. The wide model is a generalized linear model of features together with their transforms. The deep model is a Dense Neural Network (DNN), a series of hidden MLP layers, each beginning with a dense embedding of features. Categorical variables are embedded into continuous vector spaces before being fed to the DNN via learned or user-determined embeddings.

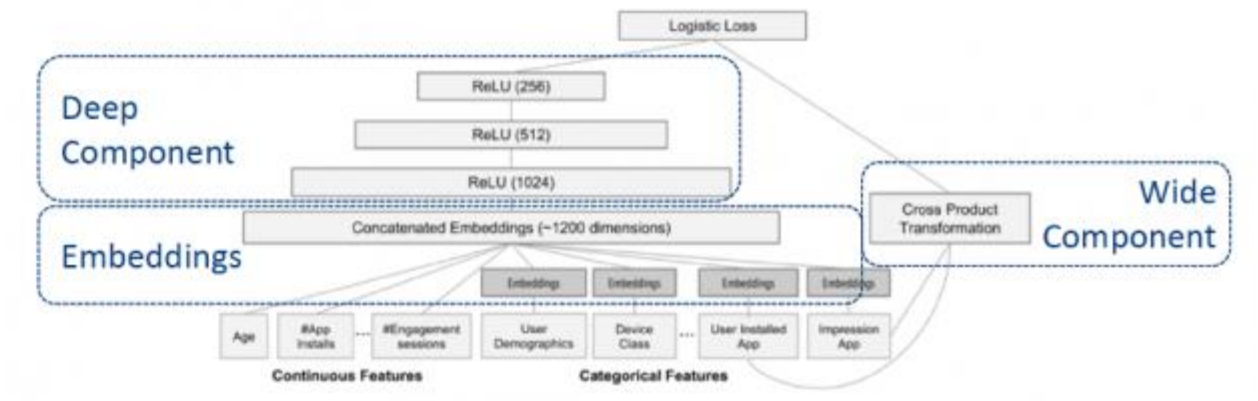


Figure 10 Wide and Deep model

What makes this model so successful for recommendation tasks is that it provides two avenues of learning patterns in the data, “deep” and “shallow”. The complex, nonlinear DNN is capable of learning rich representations of relationships in the data and generalizing to similar items via embeddings but needs to see many examples of these relationships in order to do so well. The linear piece, on the other hand, is capable of “memorizing” simple relationships that may only occur a handful of times in the training set.

DLRM

DLRM is a DL-based model for recommendations introduced by Facebook research. It’s designed to make use of both categorical and numerical inputs that are usually present in recommender system training data. To handle categorical data, embedding layers map each category to a dense representation before being fed into multilayer perceptrons (MLP). Numerical features can be fed directly into an MLP.

At the next level, second-order interactions of different features are computed explicitly by taking the dot product between all pairs of embedding vectors and processed dense features. Those pairwise interactions are fed into a top-level MLP to compute the likelihood of interaction between a user and item pair.

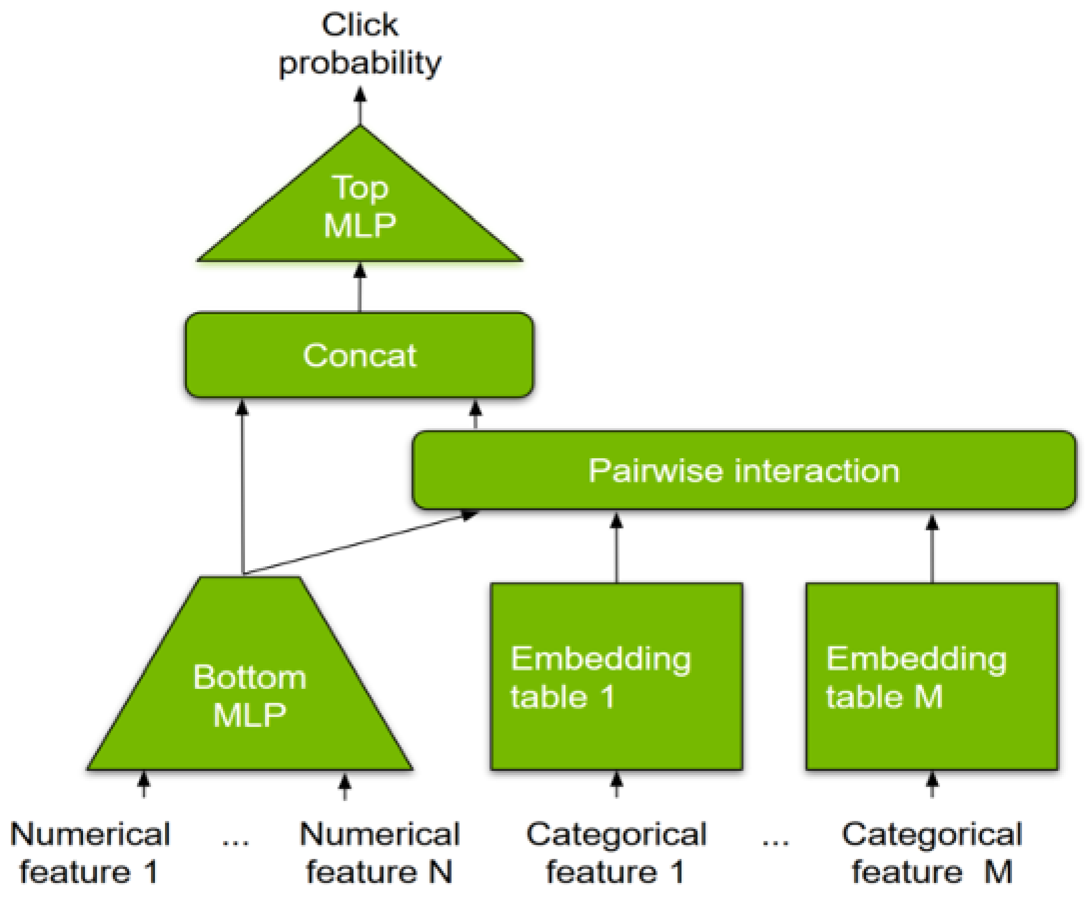


Figure 11 DLRM model

Compared to other DL-based approaches to recommendation, DLRM differs in two ways. First, it computes the feature interaction explicitly while limiting the order of interaction to pairwise interactions. Second, DLRM treats each embedded feature vector (corresponding to categorical features) as a single unit, whereas other methods (such as Deep and Cross) treat each element in the feature vector as a new unit that should yield different cross terms. These design choices help reduce computational/memory cost while maintaining competitive accuracy.

<u>**Contextual Sequence Learning**</u>

A Recurrent neural network (RNN) is a class of neural network that has memory or feedback loops that allow it to better recognize patterns in data. RNNs solve difficult tasks that deal with context and sequences, such as natural language processing, and are also used for contextual sequence recommendations. What distinguishes sequence learning from other tasks is the need to use models with an active data memory, such as LSTMs (Long Short-Term Memory) or GRU (Gated Recurrent Units) to learn temporal dependence in input data. This memory of past input is crucial for successful sequence learning. Transformer deep learning models, such as BERT (Bidirectional Encoder Representations from

Transformers), are an alternative to RNNs that apply an attention technique—parsing a sentence by focusing attention on the most relevant words that come before and after it. Transformer-based deep learning models don't require sequential data to be processed in order, allowing for much more parallelization and reduced training time on GPUs than RNNs.

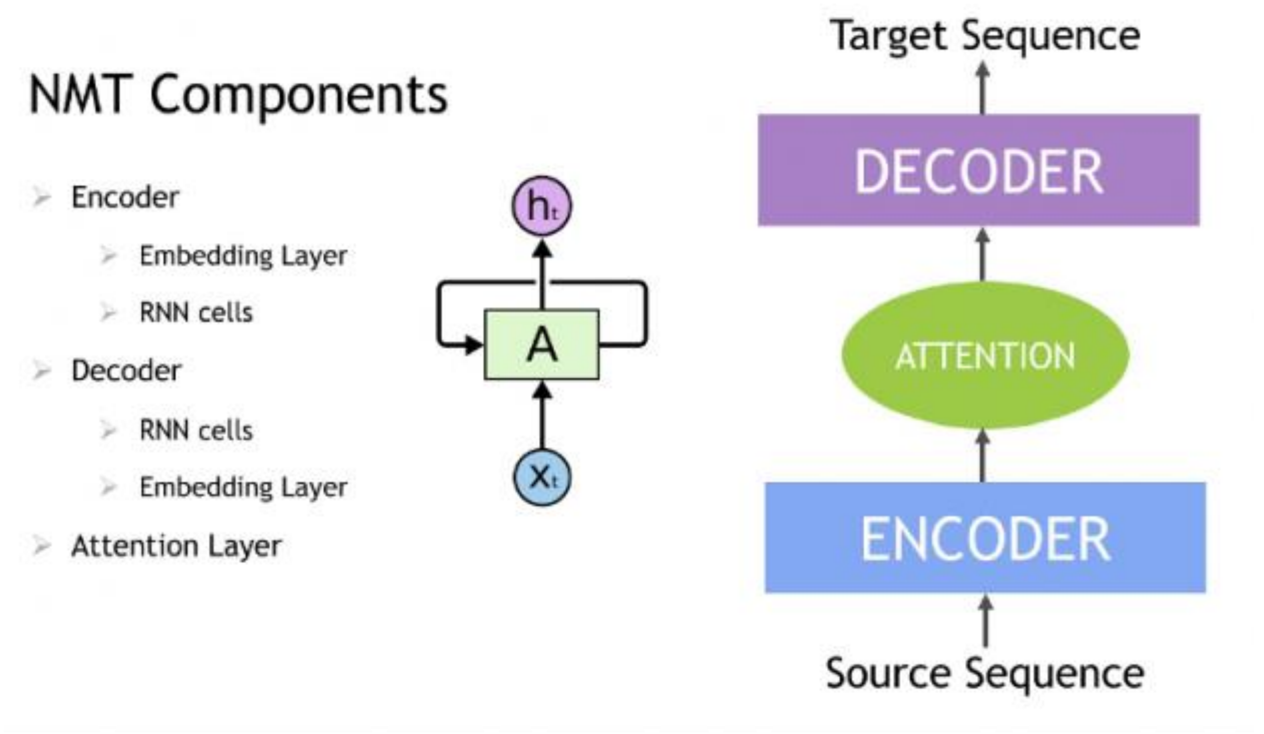


Figure 12 Neutral machine translation Model

In an NLP application, input text is converted into word vectors using techniques, such as word embedding. With word embedding, each word in the sentence is translated into a set of numbers before being fed into RNN variants, Transformer, or BERT to understand context. These numbers change over time while the neural net trains itself, encoding unique properties such as the semantics and contextual information for each word, so that similar words are close to each other in this number space, and dissimilar words are far apart. These DL models provide an appropriate output for a specific language task like next-word prediction and text summarization, which are used to produce an output sequence.

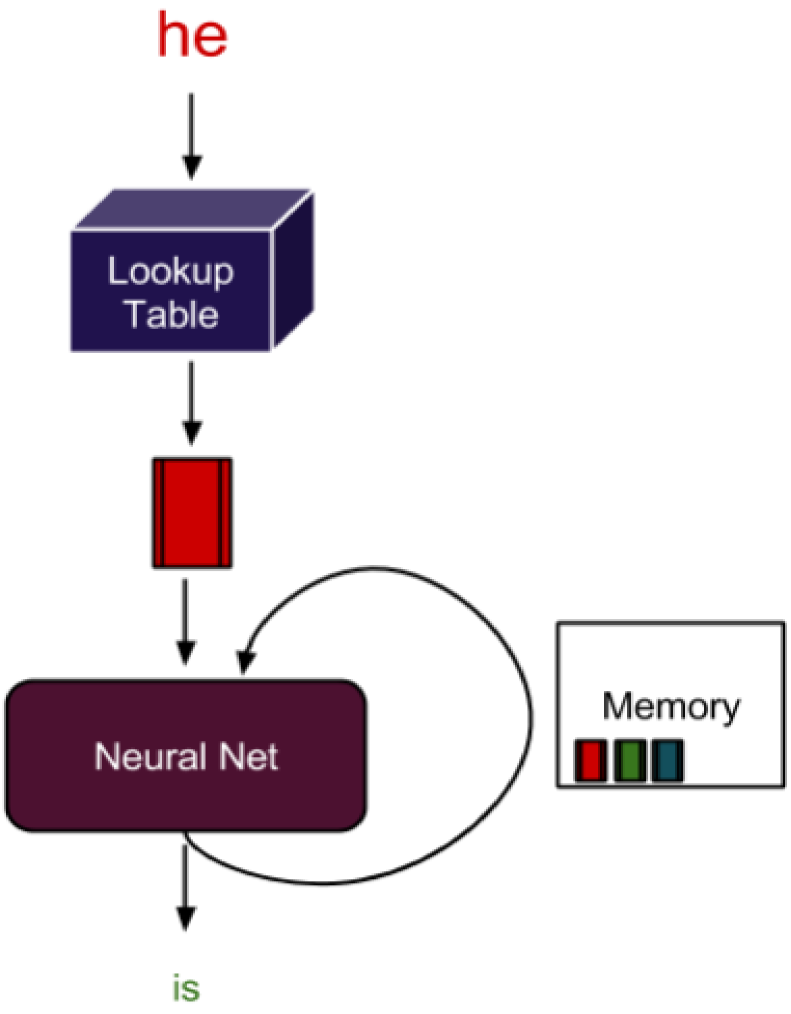


Figure 13 A recurrent neural network has memory of past experiences. The recurrent connection preserves these experiences and helps the network keep a notion of context

Session-based recommendations apply the advances in sequence modeling from deep learning and NLP to recommendations. RNN models train on the sequence of user events in a session (e.g. products clicked, date, and time of interactions) in order to predict the probability of a user clicking the candidate or target item. User item interactions in a session are embedded similarly to words in a sentence before being fed into RNN variants such as LSTM, GRU, or Transformer to understand the context. For example, Square's deep learning-based product recommendation system shown below leverages the transformer-based model BERT, GRUs, and NVIDIA GPUs to create a vector representation of their sellers.

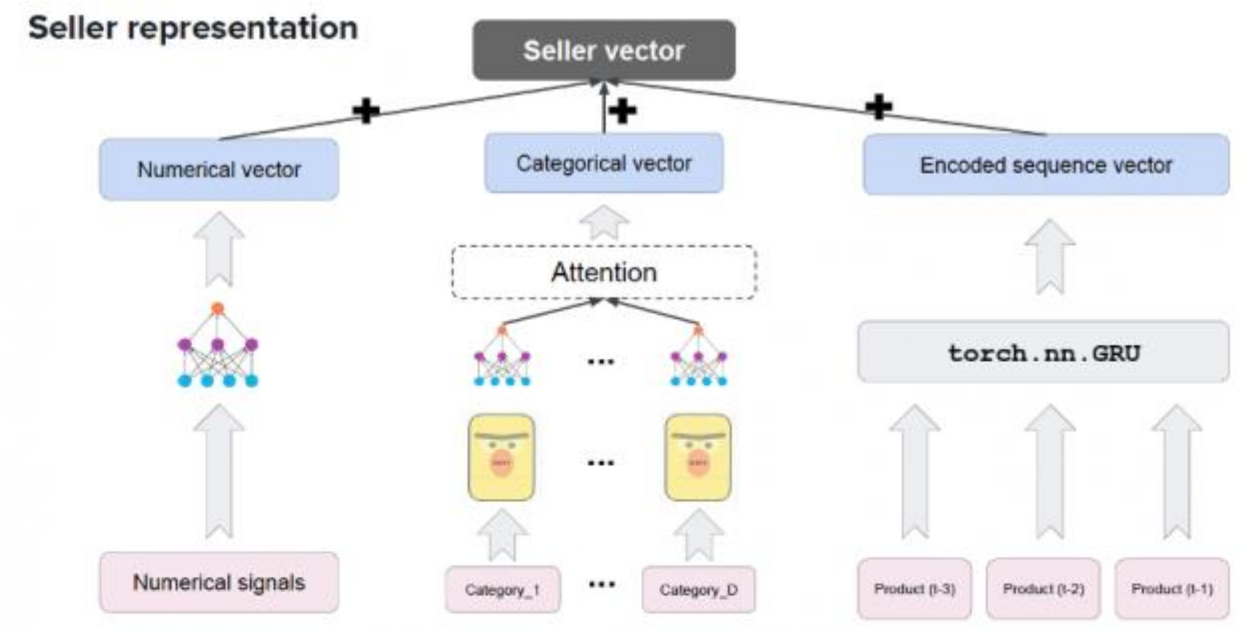


Figure 14 Square's model architecture leverages the transformer-based model BERT and GRUs to create the vector representation of their sellers

2.2 Models Used to Implement Our Recommendation System

2.2.1 Knowledge-Based Recommendation with LDA and NLPModel Models

Presentation of the Models:

Latent Dirichlet Allocation (LDA): A technique used to extract topics from textual data, enabling the identification of underlying themes in a set of documents.

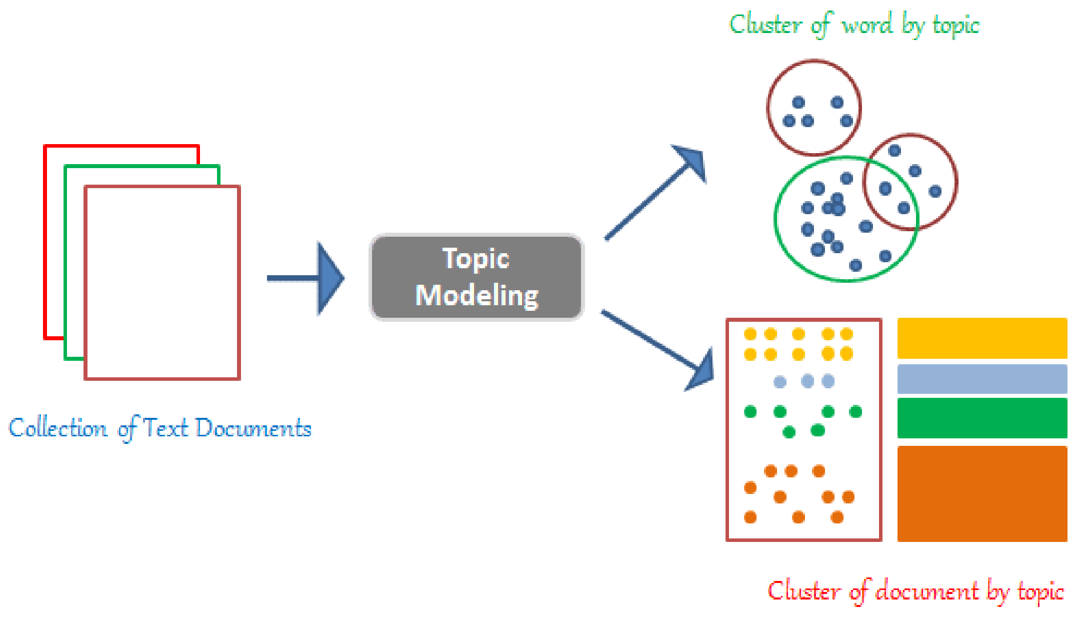


Figure 15 LDA Process

Neural Network (NLPModel): A model used to classify the topics extracted by LDA. This network takes as input the topic distributions generated by LDA and predicts the category to which each example belongs.

Recommendation Process Using These Models:

1. Data Preprocessing:
   - Load course and user data from CSV files.
   - Combine course skills and user categories into a single dataset.
   - Split the data into training and testing sets.

2. NLP Preprocessing:
   - Use TF-IDF to convert textual data into document-term matrices.
   - Apply Latent Dirichlet Allocation (LDA) to extract topics from the data.

3. Model Definition and Training:
   - Define a neural network model taking into account the topic features.
   - Train the model with the training set using a cross-entropy loss function and an Adam optimizer.
   - Utilize early stopping to prevent overfitting.

4. Data Augmentation:
   - Add new samples by randomly selecting existing samples and adding random noise.
   - Retrain the model with augmented data.

5. Model Evaluation:
   - Use the test set to evaluate the model's performance by calculating accuracy.

6. Course Recommendation:
   - We can use our model to recommend courses based on the similarity between the user's category and the course skills, thereby providing personalized recommendations based on each user's specific interests and needs.

2.2.2 Content-Based Recommendation with LSTMWithAttention and SelfAttention Models

Presentation of the Models:

SelfAttention: It is used as an attention mechanism in the LSTMWithAttention model. The SelfAttention model calculates attention weights over the outputs of the LSTM layer, allowing the model to focus on important parts of the input data sequence when predicting the next course.

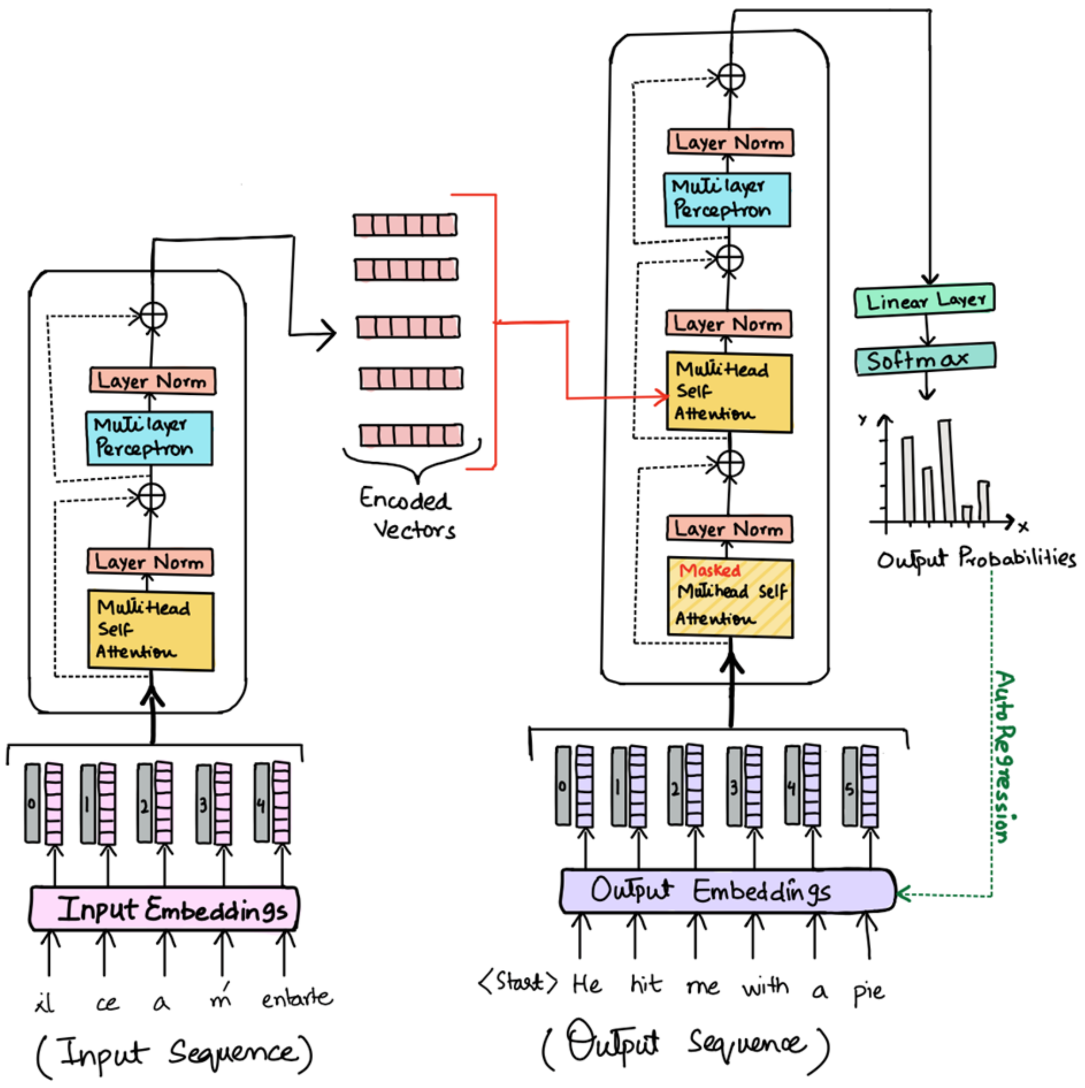


Figure 16 Self-Attention architecture

LSTMWithAttention: This model combines an LSTM layer with a self-attention layer. It takes input size, hidden size, and the number of LSTM layers as input and generates outputs by applying self-attention to the LSTM layer outputs.

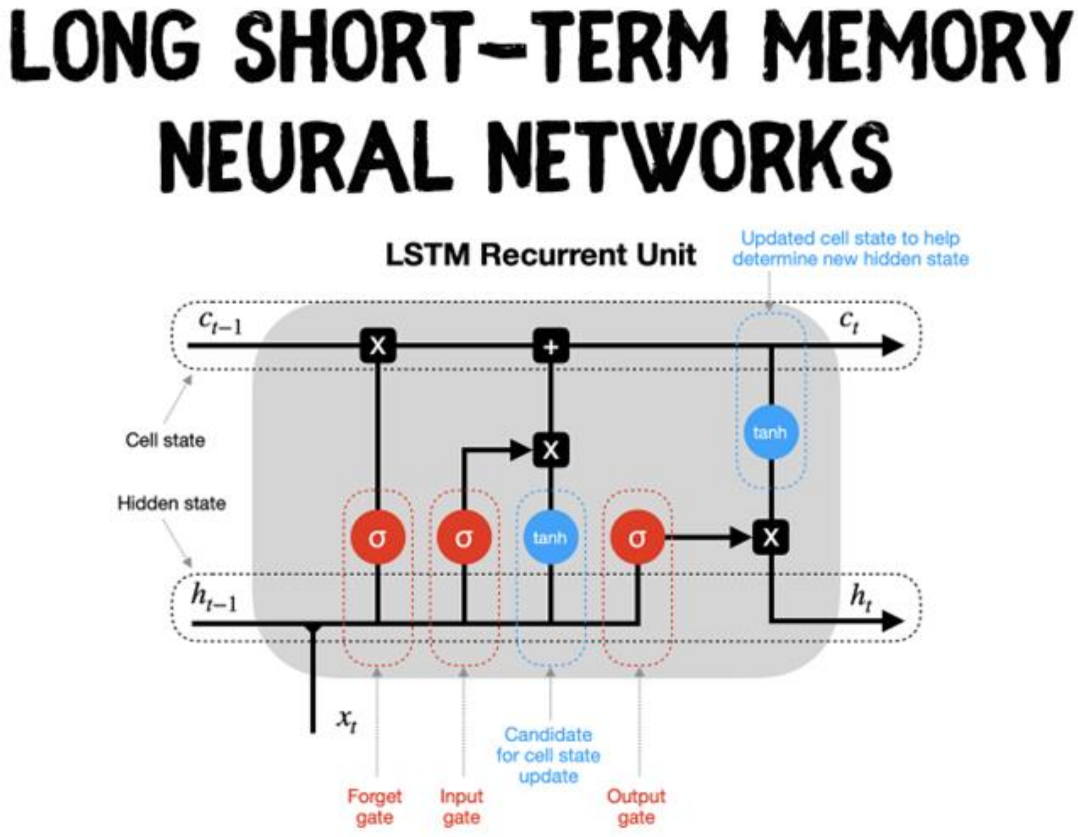

LONG SHORT-TERM MEMORY NEURAL NETWORKS
LSTM Recurrent Unit
Updated cell state to help determine new hidden state
c_{t-1}
c_t
Cell state
Hidden state
tanh
σ
σ
tanh
σ
h_{t-1}
h_t
x_t
Forget gate
Input gate
Candidate for cell state update
Output gate

Figure 17 LSTM architecture

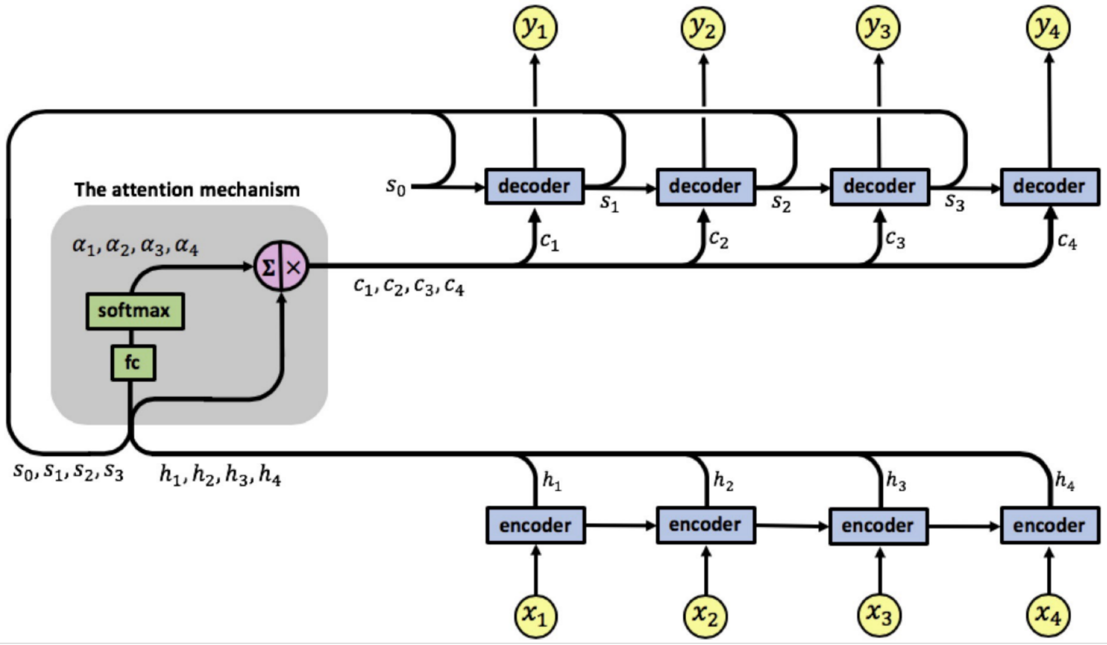


Figure 18 LSTM with Attention

Recommendation Process Using These Models:

1. Data Preprocessing:
   - Load data on courses and user viewing/completion history data.
   - Filter data to obtain the viewing/completion history of courses for a specific user.
   - Join course data with user history to obtain descriptions of courses viewed/completed by the user.

2. Course Representations Creation:
   - Use course descriptions to create TF-IDF vectors using the `TfidfVectorizer` model.
   - Transform descriptions of courses viewed/completed by the user into TF-IDF vectors.

3. LSTM Model Training with Attention:
   - Train the LSTM model with attention, composed of an LSTM layer and an attention mechanism.
   - The LSTM layer learns patterns in sequences of course descriptions.
   - The attention mechanism, based on the SelfAttention model, allows the model to focus on important parts of the data sequence.

4. Course Recommendation:
   - Use the trained LSTM model with attention to predict the next course based on courses viewed/completed by the user.
   - The model takes as input the TF-IDF vectors of viewed/completed courses and predicts the TF-IDF vector of the next course.
   - Calculate cosine similarity between the predicted next course and all course descriptions to obtain similar courses.
   - The most similar courses are selected as recommendations for the user.

2.2.3 Recommendation Using the Neural Collaborative Filtering (NCF) Model

Presentation of the Models:

This model consists of interconnected layers and is used for classification, regression, and prediction from labeled data. Data flows from input to output without loops, passing through input layers, hidden layers with activation functions, and an output layer to generate predictions. This model is effective for modeling complex relationships between data and output labels.

Recommendation Process Using These Models:

1. Data Preparation:
   - Load the dataset containing user-course interactions.
   - Filter positive interactions, i.e., interactions where the user viewed and completed the course.
   - Split the dataset into training and testing sets.
2. Model Training:
   - Instantiate the NCF model with the appropriate number of users, courses, and training data.
   - Train the model using PyTorch Lightning.

3. Model Evaluation:
   - Calculate the model's accuracy on the test set.
   - For each user in the test set, generate a list of 100 courses (99 not interacted with by the user and 1 interacted with by the user).
   - For each user-course pair, use the model to predict the probability that the user interacts with the course.
   - Rank the courses based on the predicted probabilities and extract the top 10 courses.
   - Calculate the Hit Ratio @ 10, which measures the proportion of actual interactions that are among the top 10 recommendations of the model.

4. Recommendations for a Specific User:
   - Randomly select a user from the test set.
   - Use the model to generate recommendations for this specific user.
   - Display the top 10 recommended courses for this user.

2.2.4 Hybrid Recommendations Using a Switching Mechanism Between Our Models

<u>Presentation of the Mechanism:</u>

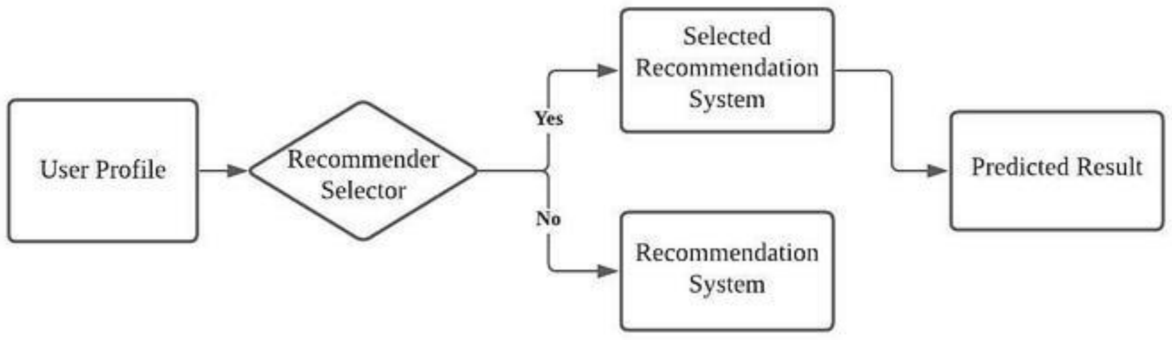


Figure 19 Switching mechanism

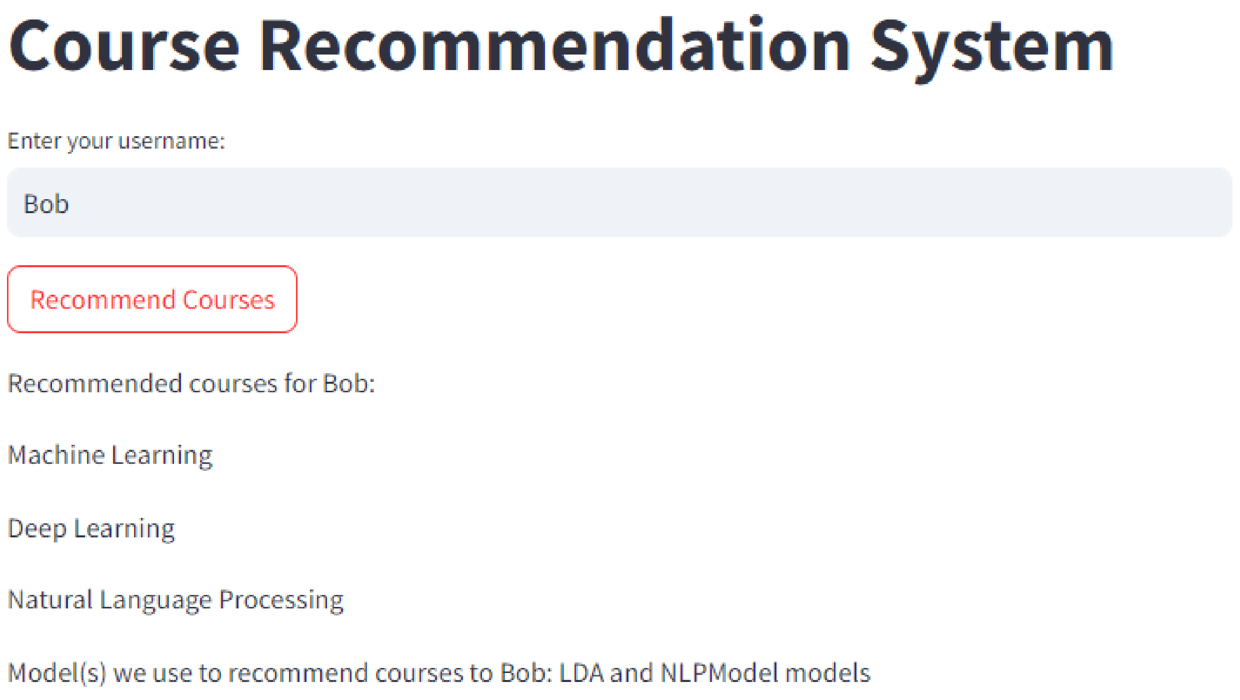
The switching hybrid system selects a single recommendation system based on the situation. The model is designed to be built on item-level sensitive data; therefore, we should define the criteria for selecting the recommendation system based on user profile or other characteristics.

Recommendation Process Using the Mechanism:

Opting for the "Switching" approach, I have decided to dynamically select the recommendation model to use based on the user's interaction level with the system.

- If the user has little or no interaction with the system, I have chosen to use the knowledge-based model to recommend relevant courses.
- If the user has moderate interaction with the system, I have opted for the content-based model to recommend courses based on course features and user preferences.
- Lastly, if the user interacts frequently with the system, I have preferred to use the collaborative-based model to recommend courses similar to those enjoyed by other similar users.

2.3 Demonstration



Course Recommendation System

Enter your username:

Bob

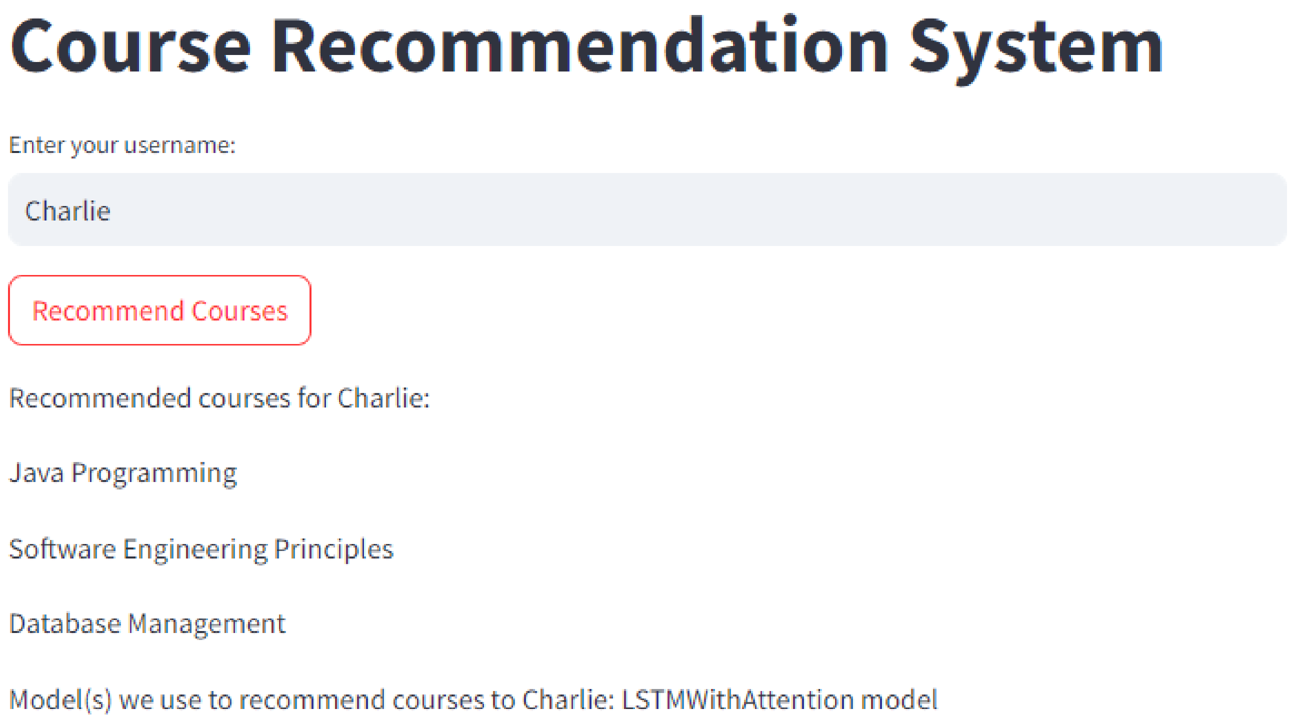
Recommended courses for Bob:

Machine Learning

Deep Learning

Natural Language Processing

Model(s) we use to recommend courses to Bob: LDA and NLPModel models


Course Recommendation System

Enter your username:

Charlie

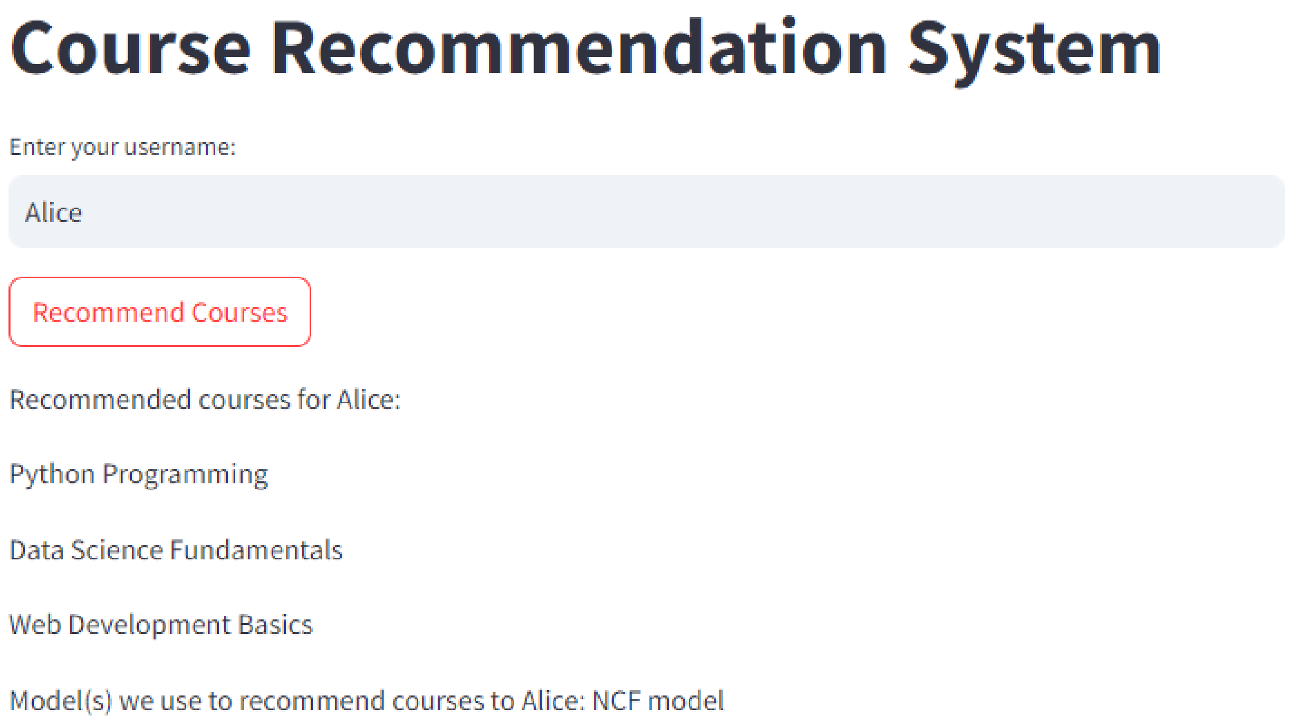
Recommended courses for Charlie:

Java Programming

Software Engineering Principles

Database Management

Model(s) we use to recommend courses to Charlie: LSTMWithAttention model


Course Recommendation System

Enter your username:

Alice

Recommended courses for Alice:

Python Programming

Data Science Fundamentals

Web Development Basics

Model(s) we use to recommend courses to Alice: NCF model

2.4 Advantages of Deep Learning over Traditional Recommendation Methods

Advantages	**Description**
Enhanced Representation Learning	Deep learning models, such as neural networks, can learn rich and complex representations of data, capturing intricate patterns and relationships that traditional methods may struggle to grasp.
Automatic Feature Extraction	Deep learning models automatically extract relevant features from raw data, eliminating the need for manual feature engineering, which is often required in traditional methods. This leads to more robust and adaptable models.
Scalability	Deep learning models can efficiently process large-scale datasets and handle complex computations, making them suitable for recommendation systems dealing with massive amounts of data and interactions.
Improved Personalization	Deep learning models excel at capturing individual user preferences and behavior patterns, enabling highly personalized recommendations tailored to each user's specific interests and needs.
Flexibility and Adaptability	Deep learning architectures, such as neural networks, are highly flexible and adaptable, allowing them to accommodate various types of data and adapt to changing trends and user preferences over time.
Better Handling of Complex Data Relationships	Deep learning models can effectively model intricate relationships and dependencies in data, including nonlinear and dynamic patterns, which traditional methods may struggle to capture.
Ability to Learn from Unstructured Data	Deep learning models can handle unstructured data types, such as text and images, allowing recommendation systems to leverage diverse sources of information for better recommendations.
Higher Accuracy and Performance	Deep learning models often achieve superior performance and accuracy compared to traditional methods, especially in tasks requiring complex data processing and pattern recognition.

CONCLUSION

In conclusion, this research paper has shed light on the limitations of traditional recommendation systems and highlighted the potential of deep learning-based approaches in the context of e-learning platforms. Traditional methods often struggle to capture the intricacies of user preferences and behavior patterns effectively. These systems rely on simplistic algorithms and predefined rules, which may not adapt well to evolving user interests or provide personalized recommendations tailored to individual needs. Looking ahead, the integration of leisure activities, hobbies, and daily life activities into recommendation algorithms presents an opportunity to enhance the effectiveness of recommendation systems by providing more holistic and personalized recommendations that resonate with users on a deeper level.

In contrast, deep learning-based recommendation systems offer significant advantages in their ability to analyze complex data and provide more accurate and personalized recommendations. Leveraging advanced techniques in machine learning, deep learning models can extract meaningful features from diverse datasets, leading to more nuanced and context-aware recommendations.

Looking ahead, to further enhance the effectiveness of recommendation systems, it is essential to consider integrating users' leisure activities, hobbies, and daily life activities into our algorithms. By broadening the scope of user interests beyond academic or professional domains, recommendation systems can deliver more holistic and personalized recommendations that resonate with users on a deeper level. This shift towards lifestyle-oriented recommendations reflects a growing recognition of the diverse needs and preferences of users in e-learning platforms.

REFERENCES

- Deep Learning in Recommendation Systems: https://developer.nvidia.com/blog/how-to-build-a-winning-recommendation-system-part-2-deep-learning-for-recommender-systems/#deep_learning_for_recommendation

- Recommendation System and Its Types: https://www.coursera.org/learn/recommendation-models-gcp

- Hybrid Recommendation System: Switching: https://medium.com/analytics-vidhya/7-types-of-hybrid-recommendation-system-3e4f78266ad8