AUT Journal of Electrical Engineering

AUT J. Elec. Eng., 58(1) (2026) 75-100
DOI: 10.22060/eej.2025.24679.5746

Hybrid Deep Learning and Evolutionary Feature Selection for Real-Time Product Recommendations

Lazarus Nisha Evangelin[1]*, Ravichandran Devi[2], Sundar Raj Bharathi[3], Vallirathi Iyyadurai[4], Shyamalagowri Murugesan[5], Jehan Chelliah[6]

[1] Department of Computer Science and Engineering, Noorul Islam Centre for Higher Education, Kanyakumari, Tamil Nadu, India.
[2] Department of Artificial Intelligence and Data Science, R.M.K Engineering College, Kavaraipettai, Tamil Nadu 601206, India.
[3] Department of Electrical and Electronics Engineering, S.A Engineering College, Chennai, 600077, India.
[4] Rohini college of Engineering and Technology, Anjukramam, Tamil Nadu, India.
[5] Department of Electrical and Electronics Engineering, K.S.Rangasamy College of Technology, Tiruchengode, India.
[6] Department of Computer Science and Engineering, Vel Tech Multitech Dr. Rangarajan Dr. Sakunthala Engineering College, Avadi, Chennai-62, India.

ABSTRACT: The swift expansion of e-commerce has driven the creation of Recommendation Systems (RS) that help users navigate vast catalogues and make informed purchase decisions. This work presents a novel recommendation system framework integrating adaptive techniques for enhanced accuracy and efficiency. The system utilizes Adaptive Evolutionary Feature Selection (AEFS), a novel feature selection algorithm combining genetic algorithms and reinforcement learning to select the most relevant features from user interaction data, product details, and contextual data. The pre-processing stage comprises text tokenization, normalization, and stop-word removal, followed by feature extraction using Term Frequency-Inverse Document Frequency (TF-IDF) and Latent Factor Modelling. User profiling is performed using Graph-based Profiling and Behavioural Profiling, allowing for a holistic view of user inclinations and preferences. The Bidirectional Encoder Representations from Transformers for Recommendations (BERT4Rec) model, which uses transformer-based architectures, is used for generating recommendations by capturing complex sequential relationships in user behaviour. This hybrid approach combines Collaborative Filtering (CF) and Content-based Filtering (CBF) to deliver accurate and personalized recommendations. Real-time recommendations are provided using a distilled model, ensuring scalability and efficiency for large-scale e-commerce platforms. The system continuously adapts through a feedback loop based on user interactions, using reinforcement learning to improve performance. With an accuracy of 98%, BERT4Rec achieves improvements of up to 18.45% across key metrics. The proposed framework enhances recommendation accuracy, achieves a feature reduction rate of 70%, and ensures a robust user experience in modern e-commerce environments.

Review History:

Received: Sep. 04, 2025
Revised: Oct. 25, 2025
Accepted: Nov. 22, 2025
Available Online: Jan. 10, 2025

Keywords:

AEFS Algorithm
TF-IDF Vectorizer
BERT4Recmodel
Recommendation System
Collaborative Filtering
Content Based Filtering
Latent Factor Modelling

1- Introduction

Everyone now has access to vast repositories of knowledge because of the emergence of internet and modern web services over span of a few decades. However, it can be challenging for consumers to sift over all of data and retrieve the most important information [1]. Many online e-commerce businesses make product recommendations to their customers because there are millions of distinct goods available on a single website. The overwhelming amount of alternatives available to average user causes information overload. RSs aim to address the problem caused by data overload while simultaneously enhancing user experience by giving users specific, customised recommendations of products and amenities based on their preferences [2]. A RS aims to establish whether a product seems beneficial to a user according to information presented [3]. Retail and e-commerce businesses like Amazon [4], eBay, and others employ these systems, and usage of them has been quickly increasing in recent years. These companies accumulate a vast quantity of user data and alter RSs to meet user and commercial needs [5]. RSs are commonly classified according to the methodologies employed for recommendations or the types of services offered to users. Many applications have successfully implemented these techniques; yet conventional recommendation approaches exhibit certain drawbacks, including issues such as scarcity, the cold start problem, and overspecialization. Notably, the recommender systems research community has identified the reliance on a single rating for predictions as a significant limitation [6].

In past, Machine Learning (ML) methods and algorithms

*Corresponding author's email: nishakingston607@gmail.com

are used to implement Collaborative Filtering (CF) RS, most notably memory-based K closest neighbour process and, more recently, Matrix Factorization (MF) method [7]. Matrix Factorization stands out as a powerful model-based approach that provides accurate recommendations. Known for its simplicity in comprehension and implementation, MF operates on the foundational concept of dimension reduction. This entails encapsulating ratings data within a more compact set of latent variables, resulting in a streamlined and effective strategy for generating precise suggestions [8]. A significant flaw in MF forecasts is that linear dot product misses intricate non-linear relationships between groups of hidden components. Since both content-based and collaborative RSs depend on historical data to provide accurate predictions, they will always have "new product" or "new consumer" issues. However, some of these restrictions are lessened by the hybridization techniques [9, 10]. Finding the ideal weights for individual algorithms is the consequence of hybridization strategy. The hybridization challenge is simplified to a multi-objective optimisation problem by treating each dimension as an independent objective. This allows for the search of ideal weighting scheme that maximises accuracy, diversity, and novelty [11]. The leading drawback of this methodology is that there hasn't been any user-item interaction with it, which makes it unsuitable for making endorsements for new goods. By building independent models for individuals or objects, respectively, the Restricted Boltzmann Machine (RBM) like strategies expressly propose either the user or item side. However, this model only take into account the correlation from one side, either item-item or user-user, completely ignoring other. Additionally, RBM-like methods are incapable of recording complicated features because they are not deep enough to record item-user interaction [12].

Neural models are chosen to drive in RS because they are not constrained by this limitation. The enormous potential of neural networks to simulate complicated relationships between products and users has drawn researchers to these groups of methodologies. Neural system-based models is broadly categorised as Shallow network and Deep network depending on number of inserted layers. The success of several embedding techniques in Natural Language Processing (NLP), such as word embedding, served as inspiration for shallow network, which moves assets to a low-dimension space [13]. Convolutional neural network (CNN) for computer vision and Recurrent Neural Network (RNN) for NLP are prominent instances of neural networks with customised architecture tailored to specific tasks that perform better than standard neural networks [14].

Nonparametric Probabilistic Principal Component Analysis (NPCA) [15], Singular Value Decomposition (SVD), and Probabilistic Matrix Factorization (PMF) are actual research that uses neural network methodology. But often, in circumstances where rating matrix is quite sparse, the latent features obtained by these techniques are insufficient. Latent features that are used to predict consumer preferences for products are found using the well-liked technique known as SVD in RSs. However, utilising SVD in feature selection for RSs has some disadvantages, including cold start challenge, scalability, interpretability, and excessive fitting [16]. Alternative methods for feature selection in RSs, such as Non-negative Matrix Factorization (NMF) and deep learning-based models, have been proposed to address these disadvantages. These methods enhance the accuracy and throughput while addressing some of SVD's drawbacks, such as interpretability and scalability [17]. NMF is better than SVD in some instances, but it also has significant disadvantages, such as limited flexibility, initialization sensitivity, overfitting, and comprehensibility. NMF has been extended and modified in different ways to overcome these limitations, including sparse NMF [18], non-negative tensor factorization, and deep learning models. As a result, these techniques overcome some of the limitations of NMF, while at the same time improving a RSs precision and performance. However, the aforementioned methods possess limitations when it comes to capturing complicated relationships, insufficient domain expertise, dimensionality reduction, and exposure to noise. Henceforth, deep learning approaches like Autoencoders, CNN, RNN [19], Graph Neural Networks (GNN) [20] and Deep Reinforcement Learning (DRL) [21] are introduced. A detailed analysis of several RSs is provided in Table 1.

The proposed RS improves upon the methodologies reviewed in the literature by addressing key limitations in feature selection, sequential modelling, and scalability. By incorporating the AEFS algorithm, the system optimizes feature selection dynamically, overcoming issues like noise and scalability seen in models such as Factorization Machines and Autoencoders. In order to explain the innovation of AEFS, this paper differentiates itself from earlier evolutionary-RL methods by presenting a two-phase optimization procedure. In contrast to conventional techniques that utilize genetic algorithms or reinforcement learning independently, AEFS incorporates both within a reward-based feedback loop in which reinforcement learning continuously varies mutation and crossover rates according to reward signals obtained from recommendation accuracy. This dynamic control allows AEFS to adapt to shifting data distributions and user patterns of behaviour in real-time. In addition, AEFS integrates domain knowledge heuristics specific to recommendation systems, like feature importance weighting according to user-item interaction density and temporal saliency. These updates cause AEFS to surpass static feature selection approaches and earlier hybrid models in both scalability and accuracy, especially under sparse and cold-start conditions. Furthermore, the use of BERT4Rec enhances the ability to capture complex sequential patterns in user behaviour, providing significant improvements over older methods like Neural Collaborative Filtering and Knowledge-aware Graph Neural Networks, which struggle with long-range dependencies and temporal dynamics. The hybrid approach combining CF and CBF further tackles issues such as the cold start challenge and sparse data, providing more accurate and personalized recommendations compared to traditional CF-based methods. In addition, the system supports real-time

Table 1. Comprehensive Review of Key Methodologies in RSs.

Ref	Author/Year of Publication	Methodology	Dataset	Merits	Demerits
[22]	Eric Appiah Mantey *et al*, 2022	Block Chain	Kaggle Chest X-Ray	Selection process is easier. Improved security.	Possibility for data leakage. Scalability concerns.
[23]	Xuguang Zhang *et al*, 2024	Hybrid Attention-based Long Short-Term Memory (HA-LSTM)	Retail time-series product demand data	Captures temporal dependencies and dynamic attention for improved forecasting accuracy	Requires extensive hyperparameter tuning and large training data for generalization
[24]	Tanmay Thorat *et al*, 2023	Transition Probability Function and CNN	Fertilizer dataset	Improved accuracy in identifying optimal nutrients.	Error free nodes are essential to define threshold.
[25]	Sanjeev Dhawan *et al*, 2025	Bidirectional Gated Recurrent Unit (GRU)-LSTM	MovieLens and Amazon Reviews	Enhances prediction accuracy by combining sequential and semantic features	Involves extensive preprocessing and may struggle with real-time scalability
[26]	Amany Sami *et al*, 2024	Hybrid Recommendation System for Internet Users using Deep Learning (HRS-IU-DL)	Internet user activity logs	Combines content-based and collaborative filtering for improved personalization	Necessitates frequent retraining to adapt to evolving user behaviour
[27]	Abolfazl Mehbodniya, *et al*., 2022	Deep Belief Network	Online product reviews	Improves sentiment classification accuracy using hybrid optimization and deep learning	Computationally intensive and sensitive to parameter initialization
[28]	Swathi Angamuthu, *et al*., 2023	Multi-Criteria Decision Making (MCDM)	Amazon and IMDb reviews	Develops sentiment analysis by integrating decision logic with deep learning	Entails careful criteria weighting and may be sensitive to subjective bias
[29]	Yan-e Hou, *et al*., 2022	Deep Reinforcement Learning (DRL) with Long- and Short-Term Preference Modeling	E-commerce user interaction logs	Detentions both recent and historical user preferences for real-time personalized recommendations	Face convergence issues in dynamic environments
[30]	Bufan Liu, *et al*., 2022	Adaptive Parallel Feature Learning and Hybrid Feature Fusion	Machining condition monitoring data	Improves feature representation and classification accuracy through parallel learning and fusion	High computational cost and complexity in tuning fusion parameters

recommendations, using a distilled model for efficiency and scalability, making it suitable for large-scale e-commerce platforms, unlike many previous works with scalability concerns.

2- Proposed System

The presented RS, as seen in Fig. 1, uses raw Amazon reviews and user interaction data to deliver accurate, personalized product suggestions. The process begins with data collection, where raw reviews, user interactions, product details, and contextual information are gathered. The raw data undergoes pre-processing to prepare data for analysis. Key features are extracted from the reviews using TF-IDF and Latent Factor Modelling, which transforms textual and structured data into numerical representations. The AEFS algorithm works in a two-stage process: a population of subsets of features is first evolved with genetic operations, and then reinforcement learning agents check the subsets according to a reward function associated with recommendation performance indicators like precision and recall. The introduction of a dynamic learning rate and exploration-exploitation trade-off, adapting to feedback from the recommendation engine, sets AEFS apart from traditional evolutionary-RL hybrids. This keeps feature selection context-sensitive and reactive to changes in user behaviour. The design of the algorithm is specifically tuned

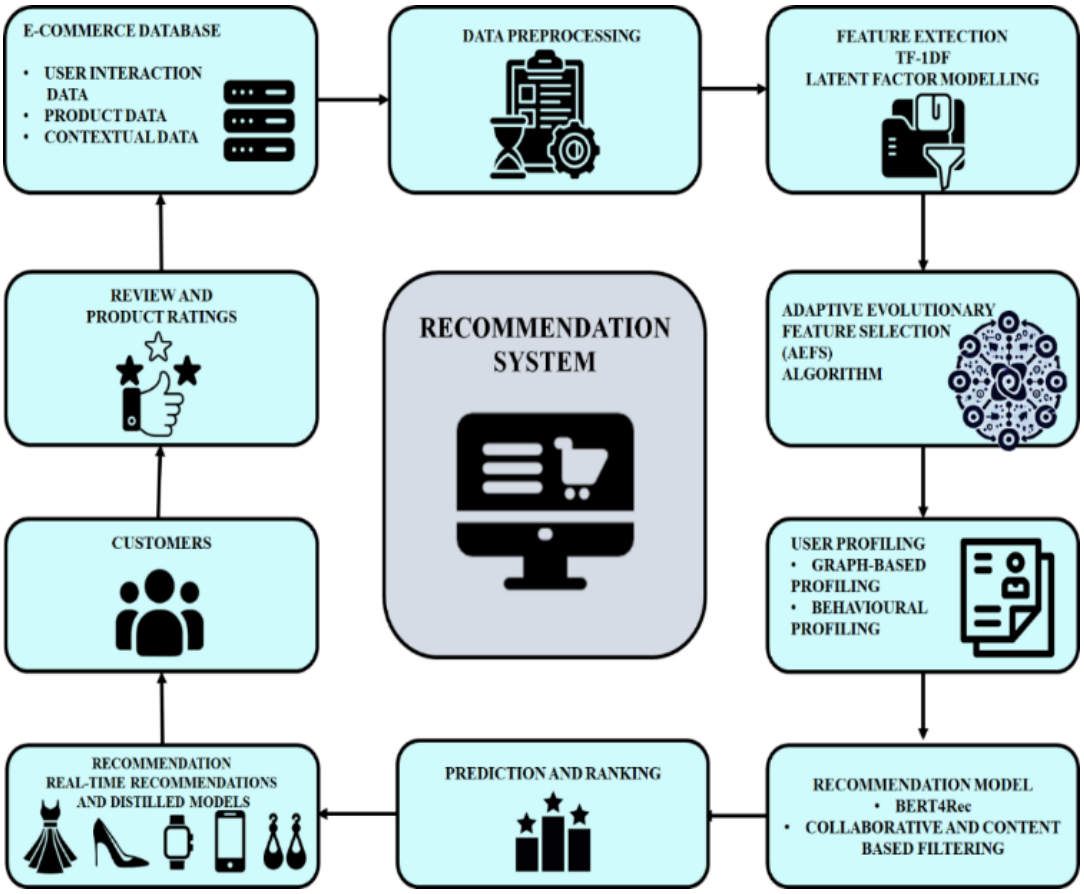

Fig. 1. Proposed Recommendation System Architecture.

for recommendation systems, leveraging interaction sparsity and sequential dependencies to guide feature relevance scoring. User profiling is performed through graph-based and behavioural profiling, which helps understand user preferences and interaction patterns.

The core recommendation model uses BERT4Rec, a transformer-based architecture, which captures the sequential nature of user interactions by learning bidirectional relationships between products. The hybrid approach combines CF and CBF, boosting recommendation precision and overcoming obstacles like the cold-start challenge. The system then predicts and ranks the products based on their relevance to user preferences, providing real-time, personalized product recommendations through a distilled version of the BERT4Rec model. Additionally, user feedback from product reviews and ratings is continuously integrated into the system for recommendation refinement. The feedback loop assures that the system remains adaptive and updated with evolving user behaviour, making it a scalable and efficient solution for e-commerce platforms.

2- 1- Data Collection and Pre-processing

As a pre-processing step, raw reviews are deleted and fixed by removing the redundant and intricate text found in the dataset, thereby reducing the complexity of the final product. The subsequent steps are initiated during the pre-processing stage to prepare data for assessment.

- **Tokenization:** The process of separating a text into tokens that are typically words or subwords. To organise text data for subsequent analysis, RS frequently use tokenization as a pre-processing step [27-29]. Tokenization is frequently used in RS to transform unstructured data, such as user evaluations, into a format suitable for analysis and utilization in producing recommendations. The system is able to retrieve features and carry out analysis more quickly by dividing text into sections. For a text document D , tokenization divides it into tokens T_i ,

$$D = \{\omega_1, \omega_2, \ldots, \omega_n\}, \quad T_i = \omega_i \tag{1}$$

Where, ω_i represents each word in the document. Fig. 2 provides the various stages involved in the pre-processing pipeline.

- **Stemming and Lemmatization:** Stemming is a text pre-

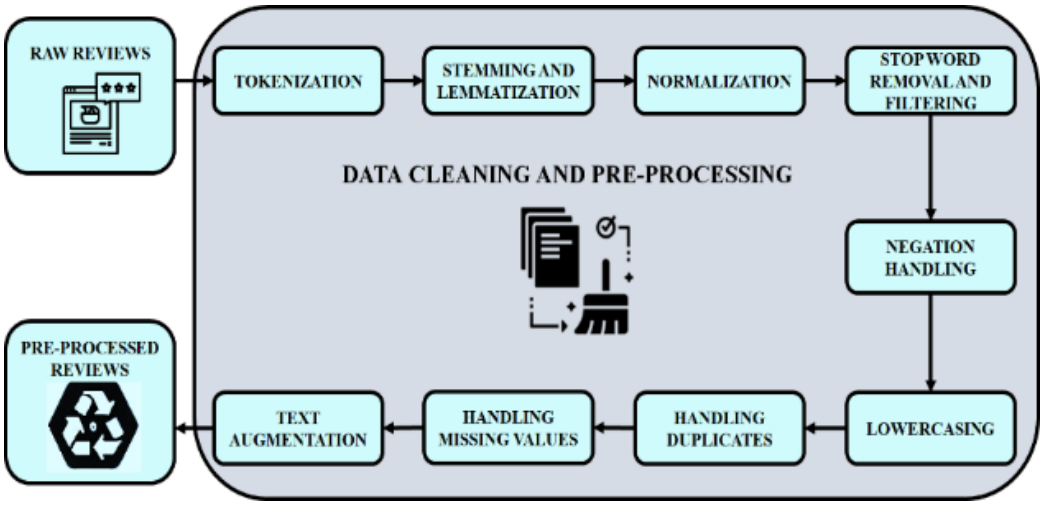

Fig. 2. Various steps involved in Pre-processing.

processing method used in RS to break down words to their root or basic form. This improves the efficiency by normalising text and lowers the number of unique terms in the dataset. In simple terms, it is the process of removing prefixes and suffixes from words to create a simpler version of the term that retains its original meaning. For a word ω_i , stemming transforms it to its base form ω_b ,

$$\omega_b = Stem(\omega_i) \tag{2}$$

Lemmatization decreases words to their base or dictionary form (lemma). Unlike stemming, lemmatization accounts for context and transforms words to their true root form. This is especially important in tasks requiring semantic understanding, such as sentiment analysis. The lemma $L(\omega_i)$ is derived as,

$$L(\omega_i) = Lemma(\omega_i) \tag{3}$$

- **Normalization:** Data cleaning techniques are carried out on datasets after selection to normalise the features and eliminate noise from the dataset. The dataset is scaled to fit within a single range as a result of normalisation. This is being done because the dataset's scale values vary. To improve the predictive ability of machine learning models, all values are put onto a single scale, using min-max normalisation. Other features have single-digit values, others have two-digit values, and some have three-digit values. In this research, the variables in the [0, 1] range are normalised using min-max scaling. The min-max normalisation is stated in Equation (1).

$$Z_i = \frac{F_i - \min(F)}{\max(F) - min(F)} \tag{4}$$

From the expression above, $F = (F_1, F_2, \ldots, F_n)$ are a number of features, F_i denotes a feature to be normalized and Z_i represents normalized features. By doing so, every features have equal weights and is included in the same scope. The term "feature" refers to the variables or attributes of the dataset that are subjected to normalization. In the context of this study, these features represent various aspects or characteristics extracted from the reviews during the pre-processing stage. Each feature, denoted by F_i , corresponds to a specific aspect of the reviews, such as word frequency, sentiment scores, or other relevant metrics. The vector $F = (F_1, F_2, \ldots, F_n)$ encompasses all the features extracted from the dataset, where n describes the total number of features. These features are initially in different scales and ranges, and the objective of equation (4) is to normalize them to a common scale. The normalization process involves transforming the values of each feature F_i to fall within the [0, 1] range. This is accomplished by subtracting the minimum value $\min(F)$ from each feature value, and then dividing it by the range of the feature values $\max(F) - \min(F)$. The result, denoted by Z_i represents the normalized version of the original feature F_i .

- **Stop-word Removal and Filtering:** By eliminating each token that matches a term from a built-in list of stop words, this function eliminates English stop words from a document. Stop words are phrases that are not absolutely required to finish a sentence or statement. Stop-word removal eliminates irrelevant words like "the" "and", and "is," which do not contribute to the sentiment. The

Table 2. Pre-processing pipeline showcasing stepwise conversion of raw reviews.

Step	Review 1	Review 2
Raw Review	This phone is not good. The battery life is terrible.	The camera quality is amazing, but it's a bit expensive.
Tokenization	['This', 'phone', 'is', 'not', 'good', 'The', 'battery', 'life', 'is', 'terrible']	['The', 'camera', 'quality', 'is', 'amazing', 'but' 'it's', 'a', 'bit', 'expensive']
Stemming	['Thi', 'phone', 'is', 'not', 'good', 'The', 'batteri', 'life', 'is', 'terribl']	['The', 'camera', 'qualiti', 'is', 'amaz', 'but' 'it', 'a', 'bit', 'expens']
Lemmatization	['This', 'phone', 'be', 'not', 'good', 'The', 'battery', 'life', 'be', 'terrible']	['The', 'camera', 'quality', 'be', 'amazing', 'but' 'it', 'be', 'bit', 'expensive']
Lowercasing	['this', 'phone', 'is', 'not', 'good', 'the', 'battery', 'life', 'is', 'terrible']	['the', 'camera', 'quality', 'is', 'amazing', 'but' 'it', 'is', 'a', 'bit', 'expensive']
Stop-word Removal	['phone', 'not', 'good', 'battery', 'life', 'terrible']	['camera', 'quality', 'amazing', 'bit', 'expensive']
Negation Handling	['phone', 'bad', 'battery', 'life', 'terrible']	['camera', 'quality', 'amazing', 'bit', 'expensive']
Handling Duplicates	No duplicates	No duplicates
Handling Missing Values	No missing values	No missing values
Normalization	Not applicable (no numerical data)	Not applicable (no numerical data)
Text Augmentation	['phone', 'poor', 'battery', 'life', 'terrible']	['camera', 'quality', 'wonderful', 'bit', 'pricey']

filtering process is represented as,

$$T_{filtered} = T - \{\omega \in S\} \tag{5}$$

Where, S specifies a set of stop words, and $T_{filtered}$ is a set of tokens after stop-word removal. This step reduces dimensionality and focuses on important terms.

- **Handling Negations:** Negations significantly impact sentiment analysis. Negation handling identifies words like "not" and ensures the system treats phrases such as "not good" as negative. If n_i is a negation word and ω_i is the word it modifies, the adjusted sentiment $S(\omega_i)$ is,

$$S(\omega_i) = -S(\omega_i) \quad if\ preceded\ by\ n_i \tag{6}$$

- **Lowercasing:** All words are converted to lowercase to ensure uniformity. This is represented as,

$$\omega_{lower} = Lowercase(\omega_i) \tag{7}$$

- **Handling Duplicates:** Duplicate reviews are removed to avoid bias in the dataset. If R_i represents a review, and D is the set of all reviews, and duplicate removal is expressed as,

$$D_{unique} = \{R_i \mid R_i \notin D_{duplicates}\} \tag{8}$$

Where, $D_{duplicates}$ contains all duplicate reviews.

- **Handling Missing Values:** Missing values in the dataset are either imputed or removed. For numerical features, missing values $F_{missing}$ are replaced with the median value (F),

$$F_{imputed} = \begin{cases} F_i & if\ F_i \neq null \\ median\ (F) & if\ F_i = null \end{cases} \tag{9}$$

- **Text Augmentation:** To balance the dataset and generate more data, text augmentation techniques such as synonym replacement are used. Let ω_i be a word and $S(\omega_i)$ is its synonym set. The augmented word ω_{aug} is selected randomly from $S(\omega_i)$,

$$\omega_{aug} = Random(S(\omega_i)) \tag{10}$$

Table 2 provides the process of transformation of reviews through each step of pre-processing. The next stage following pre-processing is the extraction of features, in which the proposed RS architecture requires RRF, where the TF-IDF

Vectorizer aids in providing it. The description of the feature extraction concept is as follows.

2- 2- Feature Extraction by TF-IDF Vectorizer and Latent Factor Modelling

Following data pre-processing, the proposed methodology employs a feature extraction technique known as TF-IDF Vectorizer within the framework of Recommendation Systems. It transforms textual data into a numerical representation, facilitating the utilization of machine learning algorithms for predictive tasks.

Term Frequency (TF) is a fundamental component of this process, measuring the frequency of specific terms within a given document. The TF of a term t in a document d, denoted as $tf_{t,d}$ is calculated as the ratio of the term's frequency in the document to the total number of words in that document. TF-IDF Vectorizer incorporates two essential terms: TF and Inverse Document Frequency (IDF). In text categorization and summarization, TF-IDF filters out stop words and is commonly used in conjunction with content-based Recommendation Systems. IDF, representing the inverse document frequency $idf\ (t,d)$, assesses whether a term is prevalent or rare across all documents. Notably, if a term appears in every document in the collection, its IDF is 0. Fig. 3 provides the workflow of TD-IDF.

The precision of both TF and IDF is determined by ensuring an accurate representation of the importance of terms in the dataset. This comprehensive approach to feature extraction enhances the capability of RS to capture meaningful patterns from the textual data, contributing to the effectiveness of subsequent ML algorithms and ultimately improving the system's ability to make precise predictions and recommendations.

2- 2- 1- Review Related Features (RRF)

RRF is a feature illustration approach at the sentence level that uses text and emoticons to indicate emotions, viewpoints, and negations in input reviews. The TF-IDF Vectorizer relies on BoWs, whereas the n-gram method depends on word embeddings. RS is restricted in a variety of ways by the use of single words to derive features. With a single-word feature, the negation issues cannot be resolved, and it also leads to incorrect categorisation of recommendations.

The first phase is resolving difficulties to create a word list using n-gram feature extraction from pre-processed reviews. TF-IDF on n-gram yield is then used for determining TF-IDF of n-gram words. In addition to reducing the number of dimensions, n-gram and TF-IDF approach combination effectually represents each review. Then, to further increase the recommendation analysis's accuracy, emoticon-

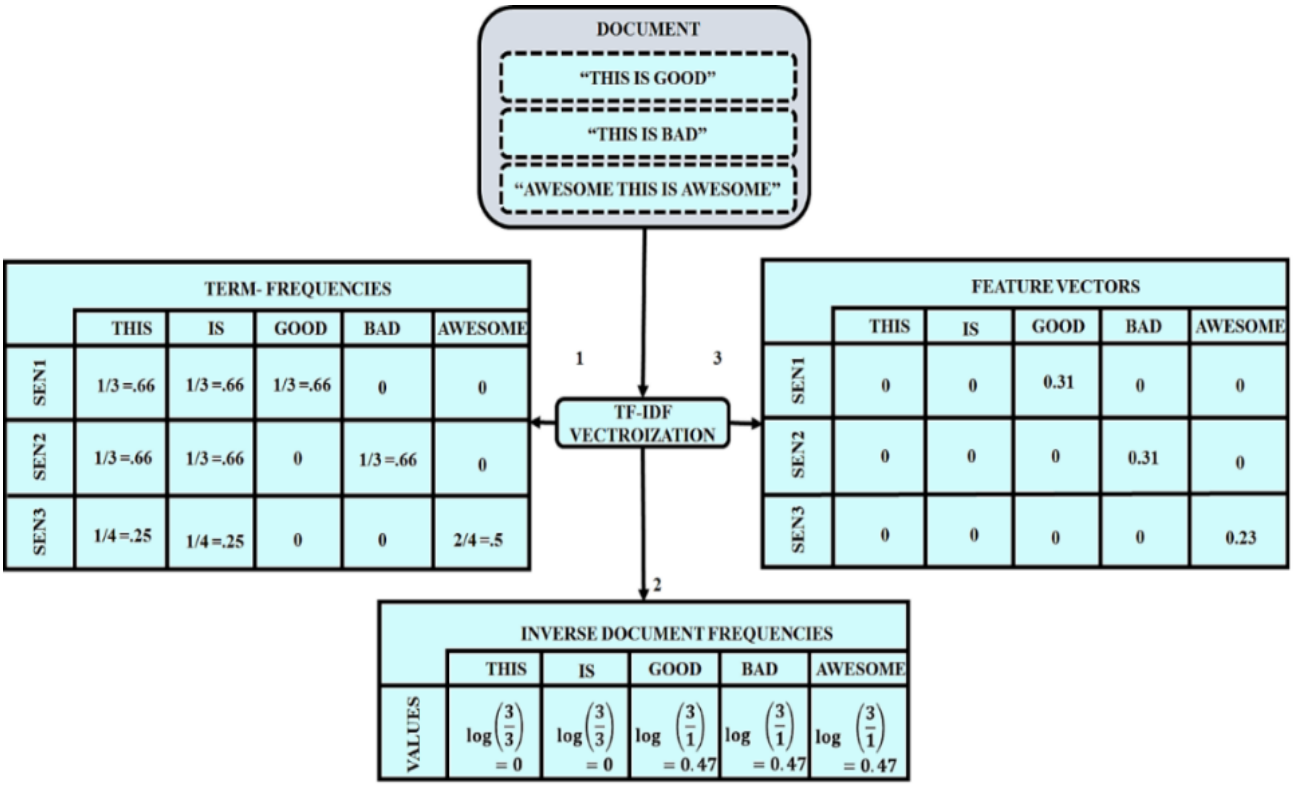

Fig. 3. Workflow of TF-IDF for Text Vectorization.

specific features are obtained. RRF is therefore achieved by combining properties that are specific to emoticons, TF-IDF, and n-grams. Below is the detailed description of the RRF method:

Consider that Q^i is a pre-processed document that makes up ith review. It begins by applying the n-gram technique to a manuscript that has already been analysed for blended n-gram and TF-IDF. Approximate sequence of n words from a specific dataset makes up an n-gram. When n is 1, an n-gram model is mentioned as a unigram, when n is 2, a bigram, when n is 3, a trigram, and so on. "Very Bad" and "Bad"

The n-gram method creates a string of n consecutive words as follows from the input sentence:

$$Ngram = getNgram(P^i, n) \tag{11}$$

From the expression above, where $Ngram^i$ stands for the collection of n-grams obtained from the input document Q^i after preprocessing. The operation $Ngram^i Ngram^d$ receives the parameters Q^i an n .

To balance efficiency and accuracy while dealing with negations, the value of n to 2 is set in this method. To obtain IDF for word lists produced by training and testing datasets, the TF-IDF Vectorizer is used after n-gram findings. TF shows the rate at which a term appears throughout the document, while IDF determines whether a term is common or uncommon across every document in a corpus.

$$\mathrm{TF-IDF} = \mathrm{TF}\left(\mathrm{Ngram^i}\right) \times \mathrm{IDF}\left(\mathrm{Ngram^d}\right) \tag{12}$$

Assuming, for instance, that $Ngram^i$ contains 60 terms, and the word "good" appears five times, the outcome of TF is $5/60 = 0.08$. Utilizing Expression (12), TF-IDF for the term "good" in the ith review is determined as 0.08 1 = 0.08 0.08*1=0.08. Then, for each feature in the document, a vector $NT\ (i)$ is engendered:

$$\mathrm{NT(i)} = \mathrm{Ngram} + \mathrm{TF-IDF} \tag{13}$$

The review dataset is then used to recover emoticons-specific attributes, which are subsequently expressed as a vector for classification and further calculations. Assign a value for the Emoticons Feature (EF) vector of size 1×2 to zero for each review since emoticons may or may not be present in each review. The number of emoticons for each review is calculated along with the recommendation label using a discrete probability distribution algorithm. A positive emoji is denoted by a number 1, while a negative as -1. If the review contains six emoticons, three of which are good and three of which are negative, the result will be shown as $[3,-3]$. The emoticons-related qualities are assigned a value of zero when either positive, negative, or both emoticons are absent from a review. The NT traits are then combined with emoticons-specific features, revising equation (4) as follows:

$$\mathrm{NTE(i)} = [\mathrm{NT}, \mathrm{EF}] \tag{14}$$

However, the adoption of TF-IDF Vectorizer supports the successful extraction of review-based characteristics from text data, like user reviews or specifications to produce suggestions. Moreover, TF-IDF enables the system to analyse and compare various documents and find similar patterns or features that are utilised to make suggestions by displaying text data in a numerical format. However, it results in providing improved accuracy and also relevant recommendations needed for users.

2- 2- 2- Latent Factor Modelling

In addition to TF-IDF, Latent Factor Modelling is an essential component of modern recommendation systems, particularly for collaborative filtering-based approaches. It focuses on identifying hidden factors that explain the observed interactions between users and items, such as reviews, ratings, or clicks. These latent factors are abstract representations of the characteristics of both users and products that influence user behaviour but are not directly observed. Latent Factor Modelling aims to represent users and items in a shared feature space by capturing underlying relationships between them. The method begins by constructing a user-item interaction matrix R , where each entry r_{ui} corresponds to the interaction between user u and item i . Matrix factorization is a frequently used latent factor model for recommendation systems. The matrix R is factorized into two lower-dimensional matrices: User matrix P and Item matrix Q . The factorization process is expressed as,

$$\mathrm{R} \approx \mathrm{P} \times \mathrm{Q^T} \tag{15}$$

Where, P is a matrix of size $m \times k$, Q is a matrix of size $n \times k$, m defines the number of users, n specifies the number of items and k represents the number of latent factors. Each user and item is represented by a vector of latent factors in the shared feature space. A user's latent factor vector captures the strength of their preferences for these factors, while an item's latent factor vector represents the degree to which the item exhibits those factors. Predicted interaction between a user u and an item i , signified as $\hat{r}_{ui}$, is computed as the dot product of their latent factor vectors,

$$\hat{\mathrm{r}}_{\mathrm{ui}} = \mathrm{P_u} \cdot \mathrm{Q_i^T} \tag{16}$$

Here, P_u defines the latent factor vector of user u , and Q_i represents the latent factor vector of the item i . The dot product gives a numerical value that represents the predicted interaction, such as a rating. To avoid overfitting, a regularization term is often added to the matrix factorization

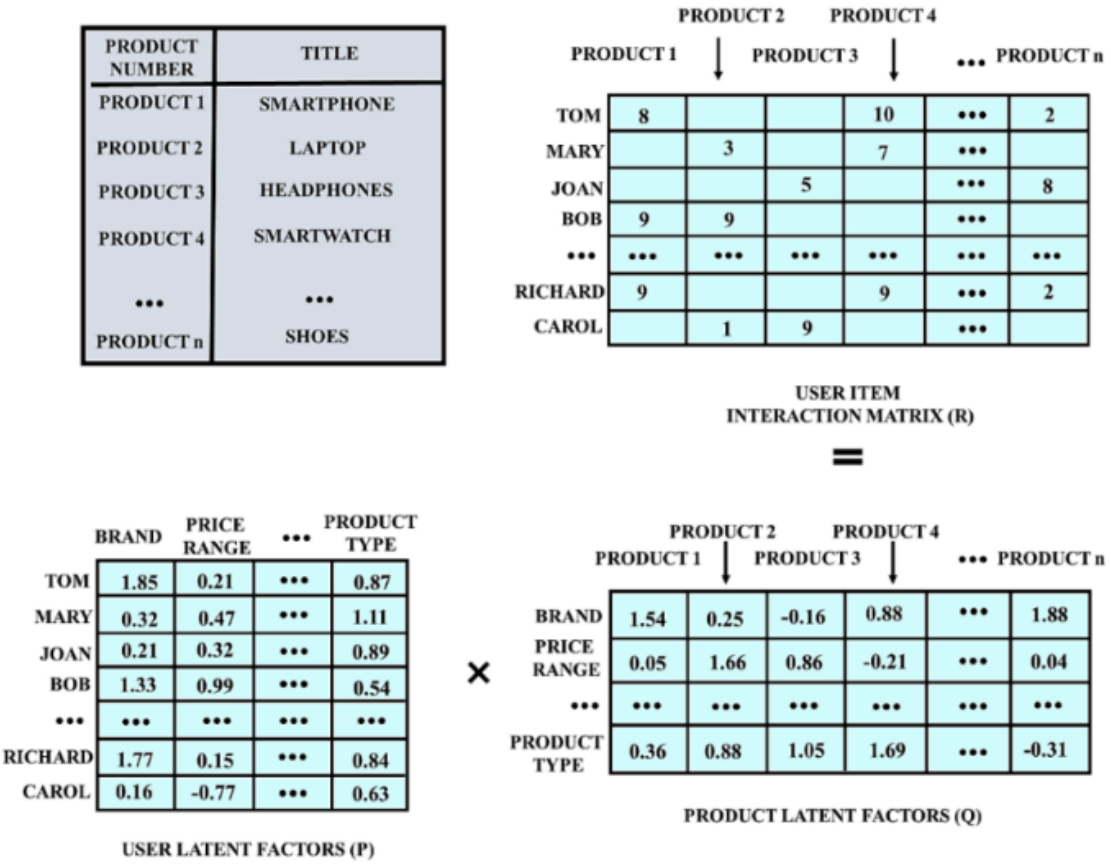

Fig. 4. Decomposing user-product interactions with Latent Factors.

objective function. The regularized objective function is,

$$\min \sum_{(u,i)\in R} \left(r_{ui} - P_u \cdot Q_i^T\right)^2 + \lambda\left(\| P_u^2 \| + \| Q_i \|^2\right) \quad (17)$$

Where, λ indicates a regularization parameter that controls the complexity of the model by penalizing large latent factor values. The process pipeline of Latent Factor Modelling is provided in Fig. 4. The combination of TF-IDF and Latent Factor Modelling in the proposed recommendation system increases the ability to capture both textual and hidden patterns in data. While TF-IDF effectively handles textual features in reviews, latent factors extracted through matrix factorization enrich the system's understanding of user-item relationships.

2- 3- Feature Selection Using AEFS Algorithm

The proposed AEFS is a novel feature selection algorithm that combines GA with Reinforcement Learning to dynamically adjust the selection process based on the performance of the current model. It selects optimal features by using a combination of evolutionary techniques and feedback from reinforcement learning to improve recommendation accuracy while reducing dimensionality. The various steps involved in AEFS is,

2- 3- 1- Initial Population

At the start of the AEFS algorithm, an initial population P_0 is generated with random feature subsets. Each feature subset is signified as a binary vector, where 1 defines a feature is selected, and 0 means it is not selected.

$$P_0 = \{F_1, F_2, \ldots, F_n\} \quad (18)$$

Where, P_0 represents the initial population, F_i is a binary vector representing a feature subset, and n is the number is number of feature subsets. Each feature subset in the population is a binary vector, expressed as,

$$F_i = [f_{i1}, f_{i2}, \ldots, f_{ik}] \quad (19)$$

Where, $f_{ij} \in \{0,1\}$ represents whether j^{th} feature is selected in i^th subset and k specifies the total number of features.

2- 3- 2- Fitness Evaluation

For each feature subset, the fitness function evaluates its performance by training BERT4Rec on the subset and computing the accuracy metric,

$$\mathrm{f}\left(\mathrm{F_i}\right)=\mathrm{Accuracy}\left(\mathrm{F_i}\right) \tag{20}$$

The pseudo code of AEFS is,

The fitness function is represented as,

$$\mathrm{f}\left(\mathrm{F_i}\right)=\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}} \tag{21}$$

The goal is to maximize $f\left(F_i\right)$, which represents the accuracy of the model trained on feature subset F_i.

Algorithm 1: Adaptive Evolutionary Feature Selection

 # Initialize population, mutation rate, and crossover probability
Initialize population P_0 *with random feature subsets*
Initialize mutation rate μ_0 *and crossover probability* α_0
for each generation t :
 # Evaluate fitness for each feature subset
for F_i *in* P_t *:*
$f\left(F_i\right)=Accuracy\left(F_i\right)$
 # Selection based on fitness
Select feature subsets for reproduction using roulette wheel or tournament selection
 # Crossover and Mutation
for each selected parent pair :
$F_{offspring}=\alpha_t * F_{parent1}+\left(1-\alpha_t\right) * F_{parent2}$
$F_{mutated}=F_{offspring} \oplus mutation_{mask}$
 # Reinforcement learning feedback
f new generation improves performance :
$i R_t=+1$
else :
$R_t=-1$
 # Update mutation rate and crossover probability
$\mu_{t+1}=\mu_t+\beta * R_t$
$\alpha_{t+1}=\alpha_t+\gamma * R_t$
 # Check stopping criteria
if stopping criteria met :
break
 # Output the best feature subset
Output best feature subset

2- 3- 3- Selection

To choose the best-performing subsets, a roulette wheel selection strategy is used. Probability p_i of selecting a feature subset F_i is proportional to its fitness score,

$$p_i = \frac{f(F_i)}{\sum_{j=1}^{n} f(F_j)} \tag{22}$$

2- 3- 4- Crossover

Once subsets are selected, the algorithm performs crossover to create offspring. The offspring $F_{offspring}$ is generated by combining two parent subsets $F_{parent\,1}$ and $F_{parent\,2}$ using a crossover parameter α .

$$F_{offspring} = \alpha \cdot F_{parent1} + (1-\alpha) \cdot F_{parent2} \tag{23}$$

Where, $\alpha \in [0,1]$ determines the amount of contribution provided by each parent to the offspring. The crossover operation is performed for each feature in the subsets.

2- 3- 5- Mutation

To introduce diversity in the population, mutation is applied. This involves randomly flipping bits in the binary vector representation of a feature subset. The mutation process is represented as,

$$F_{mutated} = F_{offspring} \oplus mutation_{mask} \tag{24}$$

Where, $\oplus$ is the bitwise XOR operation and $mutation\_mask$ is a binary vector where bits are randomly flipped with probability μ, the mutation rate. Table 3 provides the implementation parameters of AEFS.

2- 3- 6- Reinforcement Learning Feedback

Reinforcement learning is used to adaptively adjust the mutation rate μ_t and the crossover probability α_t based on the performance of the population. The reward R_t is given based on whether the new generation improves the performance of the recommendation model.

$$R_t = \begin{cases} +1, \ if \ performance \, improves \\ -1, \ if \ performance \, degrades \end{cases} \tag{25}$$

The mutation rate and crossover probability are updated as follows,

$$\mu_{t+1} = \mu_t + \beta R_t \tag{26}$$

$$\alpha_{t+1} = \alpha_t + \gamma R_t \tag{27}$$

Where, β and γ are the learning rates controlling the adjustment of mutation and crossover parameters.

2- 3- 7- Termination criteria

The algorithm continues iterating through generations until convergence is met. For instance, when there is no significant improvement in performance after a set number of generations or when a predefined maximum number of generations is reached,

$$\text{Stop if}: \ \Delta f(F_{best}) < \epsilon \ \text{for} \ G_{max} \text{generations} \tag{28}$$

Where, ϵ is a small threshold value, and G_{max} is the maximum number of generations.

2- 3- 8- Best Feature Subset

Once the termination criteria are met, the algorithm outputs the best feature subset F_{best} that maximizes the recommendation accuracy while minimizing the number of selected features.

Table 3. Configuration parameters for AEFS feature selection.

Parameter	Symbol	Value
Initial population size	P_0	100
Mutation rate	μ_0	0.05
Crossover probability	α_0	0.8
Number of generations	G_{max}	50
Learning rate for mutation update	β	0.01
Learning rate for crossover update	γ	0.01
Stopping threshold	ϵ	0.001

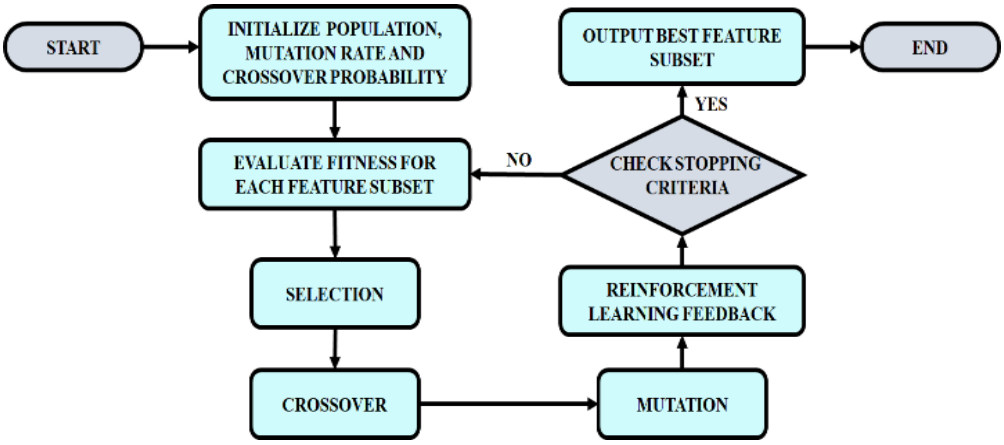

Fig. 5. Flowchart of AEFS.

$$F_{best} = \arg\max_{F_i \epsilon P_G} f(F_i) \tag{29}$$

Where, P_G represents the population in the final generation.

In RS, AEFS is used to select relevant features from TF-IDF vectors, user interaction data, and product attributes. AEFS scores subsets of features with Precision@10 as the reward signal. Mutation rates are adaptively adjusted with feedback from recommendation performance. The algorithm employs a population size of 50, a mutation rate of 0.2, and for 100 generations. Reinforcement feedback assists feature prioritization with consistently improving ranking metrics. This method optimizes feature selection, improves accuracy, and reduces computational complexity, making it well-suited for handling large-scale e-commerce data. By incorporating AEFS, the system selects the most important features to drive higher performance and reduce dimensionality, leading to a more efficient RS.

2- 4- Recommendation Model using BERT4Rec and Hybrid Collaborative-Content-based Filtering

In this e-commerce recommendation system, BERT4Rec is used to model sequential user interactions, combined with CF and CBF. This hybrid approach captures both user-item interactions and textual features from reviews, allowing the model to generate accurate and personalized recommendations.

2- 4- 1- BERT4Rec for Sequential Recommendation

BERT4Rec uses a transformer-based architecture to model sequential user interactions. It predicts the next item in a sequence by processing a user's past interactions, considering bidirectional dependencies in the sequence. Each user $u \in U$ has a sequence of interactions with products $S_u = [i_1, i_2, \ldots, i_t]$, where $i_j \in I$ represents an item interacted with at time step j. The goal of BERT4Rec is to predict the next interaction $\hat{i}_{t+1}$, given the previous interactions,

$$i_{t+1} = \arg\max_{i \epsilon I} P(i|S_u) \tag{30}$$

This represents the item i with the highest probability of being the next interaction for user u. Sequence of items S_u is embedded into a latent space, where each item i_j is mapped to an embedding vector $e_{i_j} \in R^d$. The sequence of embeddings is represented as,

$$E(S_u) = [e_{i_1}, e_{i_2}, \ldots, e_{i_t}] \tag{31}$$

Where, $e_{i_j} \in R^d$ is the embedding of item i_j, and d defines the dimension of the embedding space. This embedding captures the latent characteristics of each item and is utilized as input to BERT4Rec model. In sequential recommendation tasks, the order of interactions matters. To capture the temporal order of the interactions, BERT4Rec adds positional encodings to the embeddings,

$$PE(S_u) = E(S_u) + P \tag{32}$$

Where, $P \in R^{(t \times d)}$ is the positional encoding matrix, which ensures that the model understands the relative positions of items in sequence. This enhances the ability of the model to learn patterns in the order of interactions. The transformer encoder processes the sequence embeddings with positional encodings through multiple layers. Each transformer layer uses self-attention to compute importance of each item in the sequence relative to others,

$$H^{(l)} = \text{MultiHeadAttension}\,(H^{(l-1)}) + \text{FeedForward}(H^{(l-1)}) \quad (33)$$

Where, $H^{(0)} = PE\,(S_u)$ is input to the first transformer layer and $H^{(l)}$ defines the hidden state after lth transformer layer. The multi-head attention mechanism focuses on diverse parts of the sequence to understand their relationships,

$$\text{Attention}\left(\text{Q},\text{K},\text{V}\right) = \text{softmax}\left(\frac{\text{QK}^{\text{T}}}{\sqrt{\text{d}}}\right)\text{V} \quad (34)$$

Where, Q, K, V are query, key, and value matrices resultant from $(H^{(l-1)})$. The SoftMax function computes attention scores to decide the importance of each item in sequence. After the transformer layers process the sequence, the model generates a probability distribution over all items in the item set I, predicting the next item in the sequence,

$$\text{P}\left(\text{i}_{\text{t+1}}|\text{S}_{\text{u}}\right) = \text{softmax}\left(\text{W}_{\text{o}}\text{H}_{\text{t}}^{(\text{L})} + \text{b}_{\text{o}}\right) \quad (35)$$

Where, $W_o \in \mathbb{R}^{d \times |I|}$ is a weight matrix $b_o \in R^{|I|}$ is a bias vector and $H_t^{(L)}$ is the output of the final transformer layer for time step t. BERT4Rec model is trained by minimizing the negative log-likelihood of the correct next item,

$$\mathcal{L}_{\text{BERT}} = -\sum_{(\text{u},\text{t})\in\text{D}} \log \text{P}\left(\text{i}_{\text{t+1}}|\text{S}_{\text{u}}\right) \quad (36)$$

Where, D describes a set of all user-item sequences in the training data.

2- 4- 2- Collaborative Filtering (CF)

CF is used to model interactions between users and items by learning latent factors for both. The interaction matrix $R \in \mathbb{R}^{|U| \times |I|}$ stores the interactions between users and items, where R_{ui} represents the interaction between the user u and item i. Matrix factorization is used to decompose the interaction matrix into two lower-dimensional matrices,

$$\text{R} \approx \text{P} \cdot \text{Q}^{\text{T}} \quad (37)$$

Where, $P \in \mathbb{R}^{|U| \times d}$ indicates the user latent matrix, $Q \in \mathbb{R}^{|I| \times d}$ defines the item latent matrix and d indicating the number of latent factors. Each user u and item i is signified by a vector of latent factors,

$$\text{P}_{\text{u}} \in \mathbb{R}^{\text{d}} \text{ for user } \text{u} \quad (38)$$

$$\text{Q}_{\text{i}} \in \mathbb{R}^{\text{d}} \text{ for item } \text{i} \quad (39)$$

Predicted interaction between the user u and item i is given by dot product of their latent factor vectors,

$$\hat{\text{R}}_{\text{ui}} = \text{P}_{\text{u}} \cdot \text{Q}_{\text{i}}^{\text{T}} \quad (40)$$

This prediction represents the strength of interaction. The model is trained by minimizing the squared error between observed and predicted interactions,

$$\mathcal{L}_{\text{CF}} = \sum_{(\text{u},\text{i})\in\text{R}} \left(\text{R}_{\text{ui}} - \hat{\text{R}}_{\text{ui}}\right)^2 + \lambda\left(\text{P}_{\text{u}}{}^2 + \text{Q}_{\text{i}}{}^2\right) \quad (41)$$

Where, λ is a regularization parameter to prevent overfitting.

2- 4- 3- Content-Based Filtering (CBF)

CBF recommends items based on their content such as product descriptions and reviews. Each item $i \in I$ is characterized by a feature vector $T_i \in \mathbb{R}^k$, where k is number of content features. User profile W_u is constructed based on items they have interacted with. Profile is updated by averaging the feature vectors of items user has interacted with,

$$\text{W}_{\text{u}} = \frac{1}{|\text{S}_{\text{u}}|}\sum_{\text{i}\in\text{S}_{\text{u}}} \text{T}_{\text{i}} \quad (42)$$

Where, T_i describes content vector of item i, and S_u indicates set of item user u has interacted with. Predicted interaction between user u and item i is computed using cosine similarity between user profile W_u and item content vector T_i,

$$\hat{\text{R}}_{\text{ui}} = \cos\left(\text{W}_{\text{u}}, \text{T}_{\text{i}}\right) = \frac{\text{W}_{\text{u}} \cdot \text{T}_{\text{i}}}{\text{W}_{\text{u}}\text{T}_{\text{i}}} \quad (43)$$

Cosine similarity quantifies the similarity between the content of an item i and user profile. W_u

2- 4- 4- Hybrid Model (CF+CBF)

This model combines the strengths of both CF and CBF. It predicts the interaction between the user u and item i as a weighted sum of CF and CBF predictions. The final predicted interaction is given by,

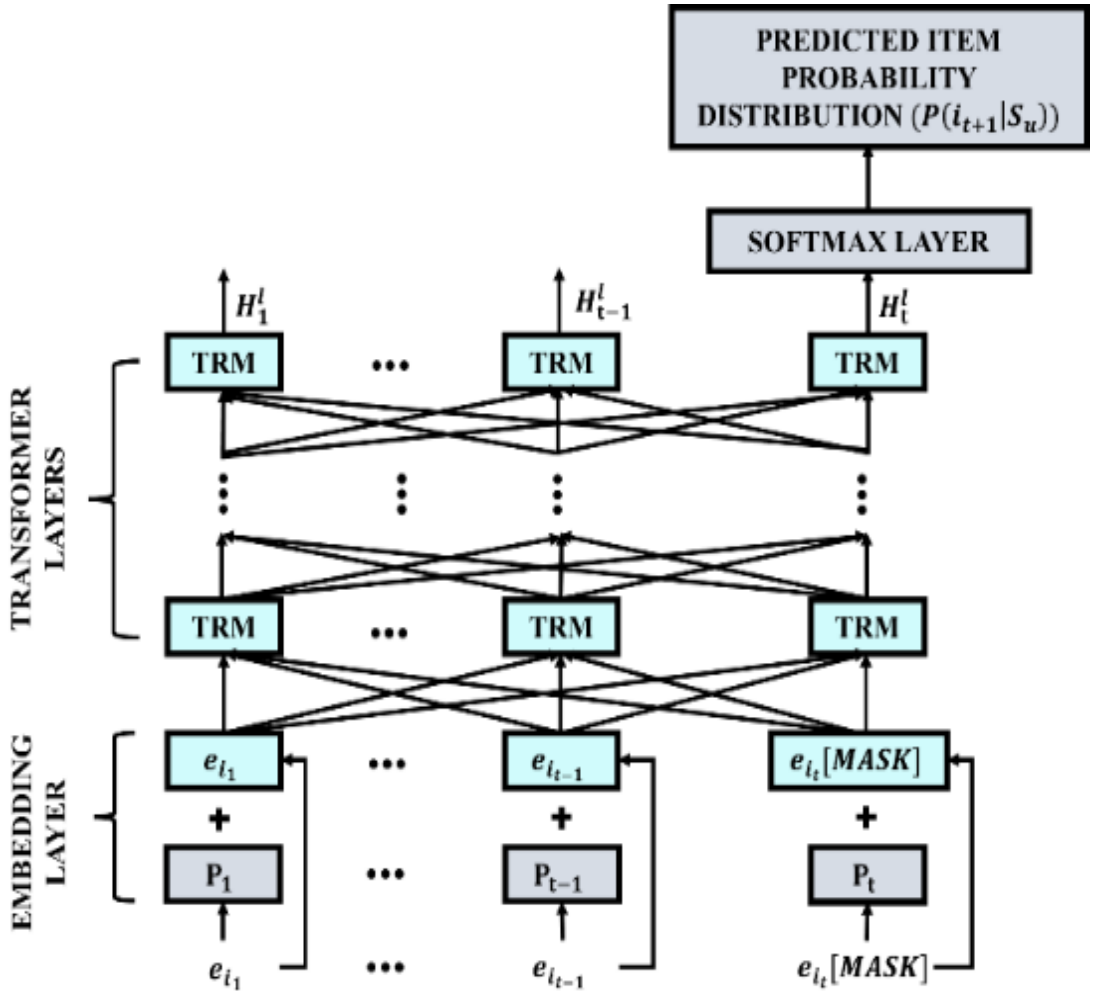

Fig. 6. Structure of BERT4Rec .

$$\hat{R}_{ui}^{hybrid} = \alpha \hat{R}_{ui}^{CF} + (1-\alpha)\hat{R}_{ui}^{CBF} \quad (44)$$

Where, $\alpha \in [0,1]$ controls the balance between CF and CBF predictions. The hybrid model is trained by minimizing the combined loss function,

$$\mathcal{L}_{hybrid} = \sum_{(u,i)\epsilon R} \left(R_{ui} - \hat{R}_{ui}^{hybrid}\right)^2 + \lambda\left(P_u^{\,2} + Q_i^{\,2} + W_u^{\,2}\right) \quad (45)$$

AEFS is used to select the most relevant features from the TF-IDF vectors, which represent product descriptions and reviews. This feature selection improves the efficiency and accuracy of the recommendation model. Let $F \in R^{|I|\times k}$ be a matrix of item features. AEFS selects a subset of features F_s ,

$$F_s = AEFS(F) \quad (46)$$

This subset F_s represents the optimal features that contribute most to the model's performance. The final predictions are used to rank items for each user. Items are ranked in descending order of their predicted interaction scores,

$$rank(i) = argsort\left(\hat{R}_{ui}^{hybrid}\right) \quad (47)$$

The proposed hybrid recommendation system integrates BERT4Rec for sequential modelling, Collaborative Filtering for capturing user-item interactions, and Content-Based Filtering for utilizing item features. The AEFS algorithm further enhances feature selection, ensuring that the system captures the most important aspects of user behaviour and product attributes. This comprehensive approach delivers highly precise and personalized recommendations, effectively addressing challenges of data sparsity, cold-start problems, and dynamic user behaviour in e-commerce settings. Combining AEFS with BERT4Rec is theoretically motivated by the adaptive representation learning principle.

AEFS adaptively samples high-impact features from reward feedback, which further complements BERT4Rec in modelling long-range dependencies by alleviating noise and sparsity in input sequences. This combination follows representation compression theory, where selective input enhances downstream learning efficiency and generalization

3- Result and Discussion

3- 1- Dataset Description

The dataset undergoes pre-processing steps to clean and standardize data for model training. Feature extraction is performed using both the TF-IDF Vectorizer and latent factor modelling to represent review texts numerically and capture hidden relationships between users and products. The AEFS algorithm is then used to reduce dimensionality and select the most relevant features. The BERT4Rec model is trained on a dataset for learning user behaviour patterns and generating personalized recommendations. Python software is utilized for investigating the assessment of proposed work. Experiments employed a 5-fold cross-validation scheme with 80/20 train-test ratios. BERT4Rec was trained for a sequence length of 50, a learning rate of 0.001, and dropout of 0.3. AEFS parameters were optimized with grid search. Experiments were replicated thrice to ensure reproducibility. The proposed model assessment is carried out using four representative real-world datasets,

- **Amazon Product Review Dataset:** Largest and most commonly used for product recommendation and sentiment analysis [31]. It contains reviews for various product categories, including electronics, clothing, books, and more. It is a good fit for sequential recommendation tasks because of its diverse nature and long-time-span of reviews.
- **Yelp Open Dataset:** Consists of customer reviews of various businesses such as restaurants, services, and retail [32]. It is ideal for recommendation tasks that involve user preferences and text reviews.
- **MovieLens Dataset:** The MovieLens dataset contains millions of movie ratings by users [33]. Although not strictly a product review dataset, it is widely used for recommendation system evaluation and contains detailed interactions between users and items.
- **Goodreads Book Reviews Dataset:** This dataset contains reviews of books from the Goodreads platform [34]. It is useful for training recommendation models that need to capture users' reading preferences over time.

Table 4 provides detailed statistics for each of the four representative datasets, illustrating their scale, features, and suitability for evaluating the proposed recommendation model in various real-world scenarios.

3- 2- Task Configuration and Evaluation Criteria

To evaluate the RS model, a leave-one-out evaluation approach is adopted, focusing on the subsequent item prediction. The last item in the behaviour sequence is held out for testing purposes, while the item just before it serves as validation data for each user. The leftover items in sequence are used for training the model. To achieve a reliable evaluation, 100 randomly chosen negative items, which the user has not interacted with, are paired with each ground truth item in the test set. The negative items are sampled based on their popularity to enhance reliability and representativeness in evaluation. The task, therefore, is to rank ground truth items among the set of negative items for each user, with higher rankings indicating better recommendation performance. To assess the performance of RSs, various evaluation metrics, including Mean Reciprocal Rank (MRR), Normalized Discounted Cumulative Gain (NDCG), and Hit Ratio (HR) are employed.

3- 3- Benchmark Methods and Implementation Insights

The proposed RS model is compared for baseline with following models,

- RNN [19]: Capture temporal dependencies between user

Table 4. Statistics of the dataset.

Dataset	Total Users	Total Items	Total Reviews	Average Reviews per Item	Features	Data Span
Amazon Product Review Dataset	2,500,000	900,000	13,000,000	14.4	User ID, Product ID, Review Text, Ratings, Timestamps	2000-2021
Yelp Open Dataset	1,500,000	200,000 businesses	8,600,000	43	User ID, Business ID, Ratings, Review Text, Location, Timestamps	2004-Present
MovieLens Dataset	600,000	9,000 movies	27,000,000	3,000	User ID, Movie ID, Ratings, Timestamps, Movie Genres	1995-Present
Goodreads Book Reviews Dataset	1,100,000	1,300,000 books	12,000,000	9.2	User ID, Book ID, Review Text, Ratings, Genres, Timestamps	2000-2019

actions for recommendation. It processes sequences of past behaviours to predict future user preferences.

- GNN [20]: Employs graph neural networks to represent and model complex relationships between users and items in recommendation tasks. It leverages graph structures to capture indirect and higher-order interactions.
- DRL [21]: Utilizes DRL to optimize recommendations by maximizing long-term user engagement. It learns through feedback loops that evaluate each recommendation's impact over time.
- NCF [29]: Combines deep neural networks with matrix factorization to effectively model user-item interactions. It enhances traditional collaborative filtering through flexible, non-linear modelling of relationships.
- GRU4Rec [35]: Implements Gated Recurrent Units (GRU) to model users' browsing and interaction sequences for session-based recommendations. It captures temporal dynamics effectively, improving the recommendation relevance in real-time scenarios.
- SASRec [36]: Uses a self-attention mechanism to model users' sequential behaviours, allowing the model to focus on significant past interactions. It provides state-of-the-art performance for sequential recommendation by capturing long-term dependencies.

Table 5 provides the implementation details and parameter settings for the evaluated recommendation models. Key parameters such as epochs, learning rate, optimizer, embedding dimension, and batch size are provided owing to their in the model training process. In contrast with SASRec and GRU4Rec+, based on fixed input embeddings, this research proposes a feedback-controlled feature selection layer that theoretically improves sequence encoding by linking feature importance to temporal evolution

3- 4- Comparative Performance Assessment

Table 5 provides a comparative analysis of different recommendation models across four real-world datasets: Amazon Product Review, Yelp Open, MovieLens, and Goodreads Book Reviews. The results clearly show that BERT4Rec outperforms all other models across all metrics, including HR@1, HR@5, HR@10, NDCG@5, NDCG@10, and MRR, demonstrating its superior capability in sequential recommendation. The percentage improvements range from 5.25% to 18.45%, highlighting its effectiveness in accurately predicting user preferences. SASRec and GRU4Rec+ also perform well, consistently achieving high scores and ranking second for several metrics, showcasing their effectiveness in capturing user-item interactions.

The NCF, GNN, and RNN models, while providing reasonable results, are consistently outperformed by more advanced models like BERT4Rec and SASRec, emphasizing the benefits of sophisticated modelling approaches like self-attention and transformers. Additionally, model performance varies across datasets, with sequential models generally

Table 5. Implementation Details and Parameter Specifications of Evaluated Models.

Model	Epochs	Learning Rate	Optimizer	Embedding Dimension	Batch Size	Features
RNN [19]	50	0.001	Adam	64	128	Uses LSTM cells for sequential learning.
GNN [20]	100	0.005	Adam	128	256	Utilizes GCN layers to capture user-item relationships.
DRL [21]	200	0.0005	RMSprop	64	64	Uses reward-based optimization for long-term engagement.
NCF [29]	30	0.01	SGD	64	512	Combines deep neural layers with matrix factorization.
GRU4Rec	50	0.001	Adam	100	128	Applies GRU-based RNNs to model session-based behaviour.
Caser	50	0.001	Adam	128	128	Employs CNN in horizontal and vertical ways to capture user preferences for sequential recommendation.
SASRec	50	0.001	Adam	200	256	Uses self-attention layers for sequential modelling.
BERT4Rec	50	0.0001	Adam	256	128	Uses transformer-based architecture for sequential recommendations. Pre-trained embeddings are used.

Table 6. Evaluation Results of Baseline and Advanced Models for Sequential Recommendation.

Metric	RNN	GNN	NCF	DRL	GRU4Rec	GRU4Rec+	Caser	SASRec	BERT4Rec	Improvement
Amazon Product Review Dataset										
HR@1	0.0075	0.0212	0.0405	0.0439	0.0246	0.0558	0.0482	0.0491	0.0959	5.25%
HR@5	0.0391	0.1270	0.1307	0.1382	0.1317	0.1759	0.1640	0.1947	0.2020	14.20%
HR@10	0.0719	0.1999	0.2114	0.2599	0.2541	0.3076	0.3121	0.3539	0.3715	14.05%
NDCG@5	0.0235	0.0813	0.0984	0.1224	0.1140	0.1457	0.1522	0.1642	0.1602	13.45%
NDCG@10	0.0432	0.1055	0.1129	0.1692	0.1563	0.1887	0.1876	0.2011	0.2130	14.01%
MRR	0.0451	0.0984	0.1027	0.1041	0.1140	0.1253	0.1314	0.1703	0.1815	10.79%
Yelp Open Dataset										
HR@1	0.0155	0.0317	0.0415	0.0429	0.0291	0.0485	0.0492	0.0499	0.0895	5.89%
HR@5	0.0832	0.1754	0.2107	0.2431	0.3120	0.3352	0.3788	0.3787	0.3991	6.98%
HR@10	0.1372	0.1994	0.2155	0.3205	0.3460	0.3611	0.3890	0.4167	0.4384	7.10%
NDCG@5	0.0485	0.1067	0.1215	0.1283	0.1512	0.1701	0.1804	0.2143	0.2292	5.51%
NDCG@10	0.0667	0.1089	0.1282	0.1468	0.1659	0.1901	0.1844	0.2179	0.2349	6.05%
MRR	0.0583	0.0962	0.1034	0.1167	0.1121	0.1272	0.1740	0.1835	0.1925	5.75%
MovieLens Dataset										
HR@1	0.0143	0.0947	0.1366	0.1429	0.1899	0.2031	0.2545	0.2874	0.3033	12.60%
HR@5	0.1915	0.2857	0.2993	0.4171	0.4579	0.50111	0.5156	0.5527	0.5784	13.40%
HR@10	0.2925	0.4055	0.4178	0.4789	0.5081	0.5631	0.5875	0.6345	0.6551	15.20%
NDCG@5	0.0783	0.1715	0.2131	0.2547	0.2734	0.2854	0.3172	0.3642	0.3787	12.85%
NDCG@10	0.0937	0.2286	0.2397	0.2773	0.3012	0.3142	0.3541	0.3849	0.4034	11.55%
MRR	0.0623	0.2092	0.2489	0.2930	0.3117	0.3252	0.3648	0.4073	0.4254	14.65%
Goodreads Book Reviews Dataset										
HR@1	0.0719	0.1091	0.1384	0.1765	0.2510	0.3024	0.3480	0.4783	0.5284	15.90%
HR@5	0.1365	0.2538	0.2921	0.3584	0.4011	0.4760	0.5023	0.5240	0.5634	13.45%
HR@10	0.2955	0.3685	0.4321	0.4650	0.5140	0.5286	0.5604	0.6325	0.6703	14.65%
NDCG@5	0.0695	0.1881	0.2135	0.2711	0.2871	0.3032	0.3380	0.4260	0.4564	14.20%
NDCG@10	0.0755	0.1503	0.1719	0.2275	0.2849	0.3092	0.3426	0.4066	0.4785	18.70%
MRR	0.0771	0.1712	0.2102	0.2276	0.2974	0.3462	0.3579	0.4094	0.4780	18.45%

performing better in datasets that include rich user interaction histories, such as MovieLens and Goodreads.

Fig. 7 illustrates the impact of hidden dimensionality on the performance of various sequential recommendation models, including GRU4Rec, GRU4Rec+, Caser, SASRec, and BERT4Rec, across different datasets. The metrics used for evaluation are HR@10 and NDCG@10, which measure the model's effectiveness in ranking relevant items among the top recommendations. The results indicate that increasing the hidden dimensionality generally improves model performance, with notable gains observed for BERT4Rec, which consistently outperforms other models at all dimensionality levels. SASRec also shows competitive performance, ranking second in most cases.

However, beyond a certain dimensionality, the performance gains plateau, suggesting diminishing returns for larger hidden dimensions. These trends highlight the importance of selecting an optimal hidden dimensionality to balance recommendation quality and computational efficiency.

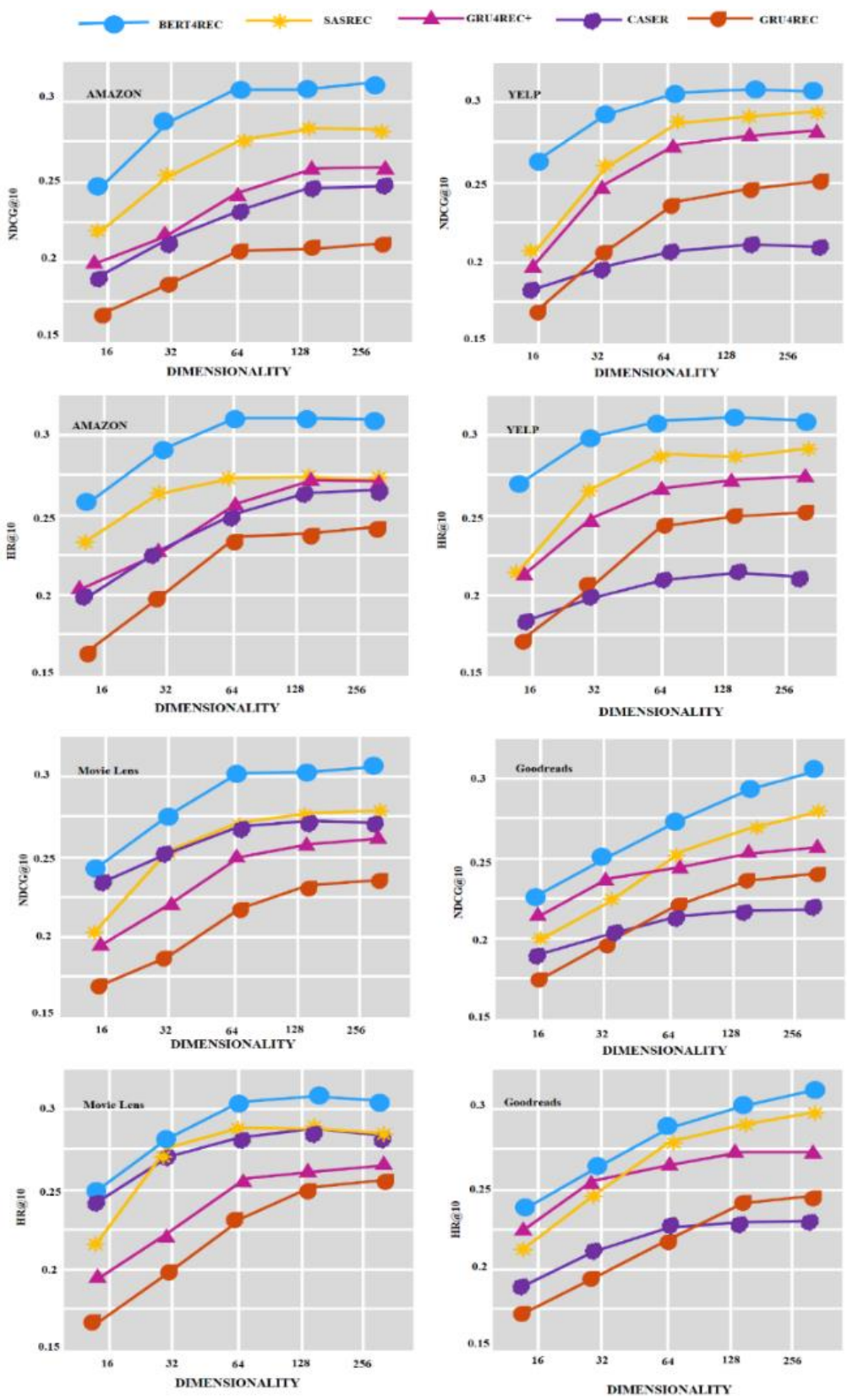

Fig. 7. Impact of Hidden Dimensionality on Performance Metrics for Neural Sequential Models.

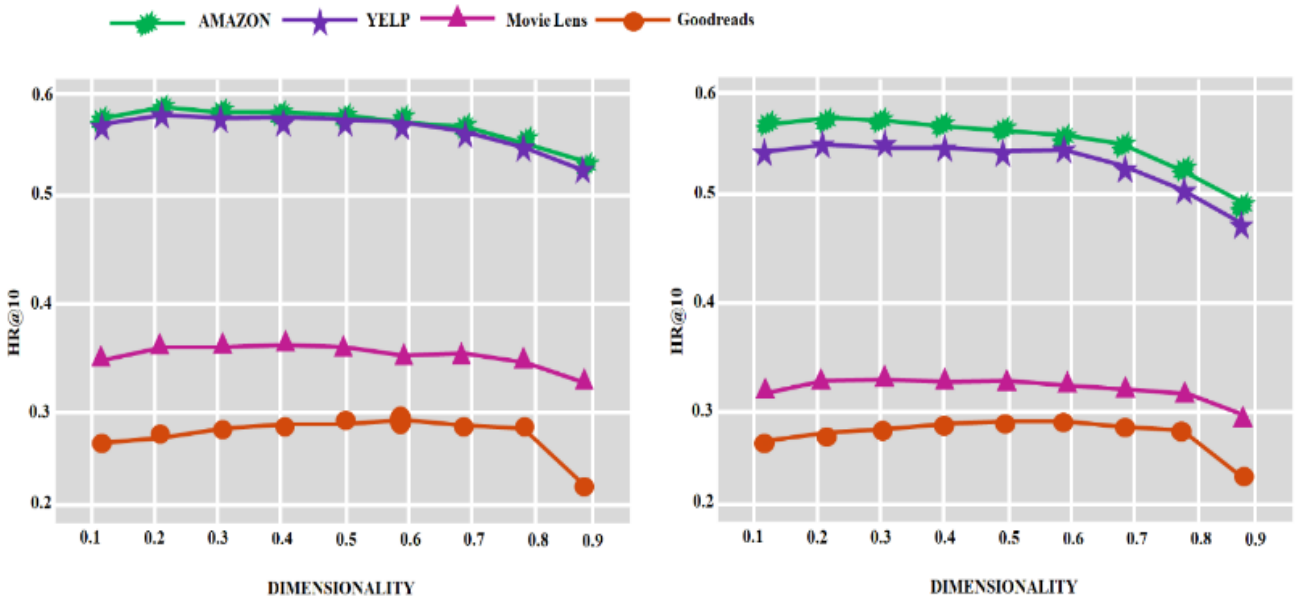

Fig. 8. Effect of Mask Proportion on HR@10 for Different Datasets.

Table 7. Effect of Maximum Length N on recommendation model performance.

Parameters	10	20	30	40	50
Amazon Product Review Dataset					
Samples/s	5600	3300	2250	1800	1450
NDCG@10	0.1835	0.1870	0.1825	0.1815	0.1810
HR@10	0.3120	0.3095	0.3065	0.3050	0.3045
MovieLens Dataset					
Samples/s	14200	8900	5750	3100	1300
NDCG@10	0.4665	0.4740	0.4760	0.4770	0.4715
HR@10	0.6810	0.6850	0.6910	0.6570	0.6870

The mask proportion (ρ) is crucial for model training, as it directly influences the loss function. If ρ is too small, the model is not sufficiently trained, while a large ρ makes training challenging due to too many items needing to be predicted based on limited context. Fig. 8 illustrates the performance of the datasets in terms of HR@10 at varying mask proportions (ρ) for a fixed dimensionality (d=64). The Amazon and Yelp datasets demonstrate consistent high performance across different mask proportions, with HR@10 values close to 0.6 and 0.5, respectively, showing that these datasets are robust to changes in the mask proportion. In contrast, MovieLens and Goodreads exhibit relatively lower and more variable performance. Specifically, Goodreads shows a steady decline in HR@10 as the mask proportion increases, indicating that its recommendation accuracy is negatively affected by higher mask proportions. MovieLens maintains a more stable performance but remains consistently lower compared to Amazon and Yelp.

The maximum sequence length (N) effect on the model's recommendation performance and efficiency is also investigated. Table 7 shows recommendation performance and training speed for varying maximum sequence lengths (N) on Amazon Product Review and MovieLens datasets. The results indicate, optimal value of N depends on the average sequence length of the dataset. Specifically, Amazon performs best at a smaller N =20, while MovieLens achieves

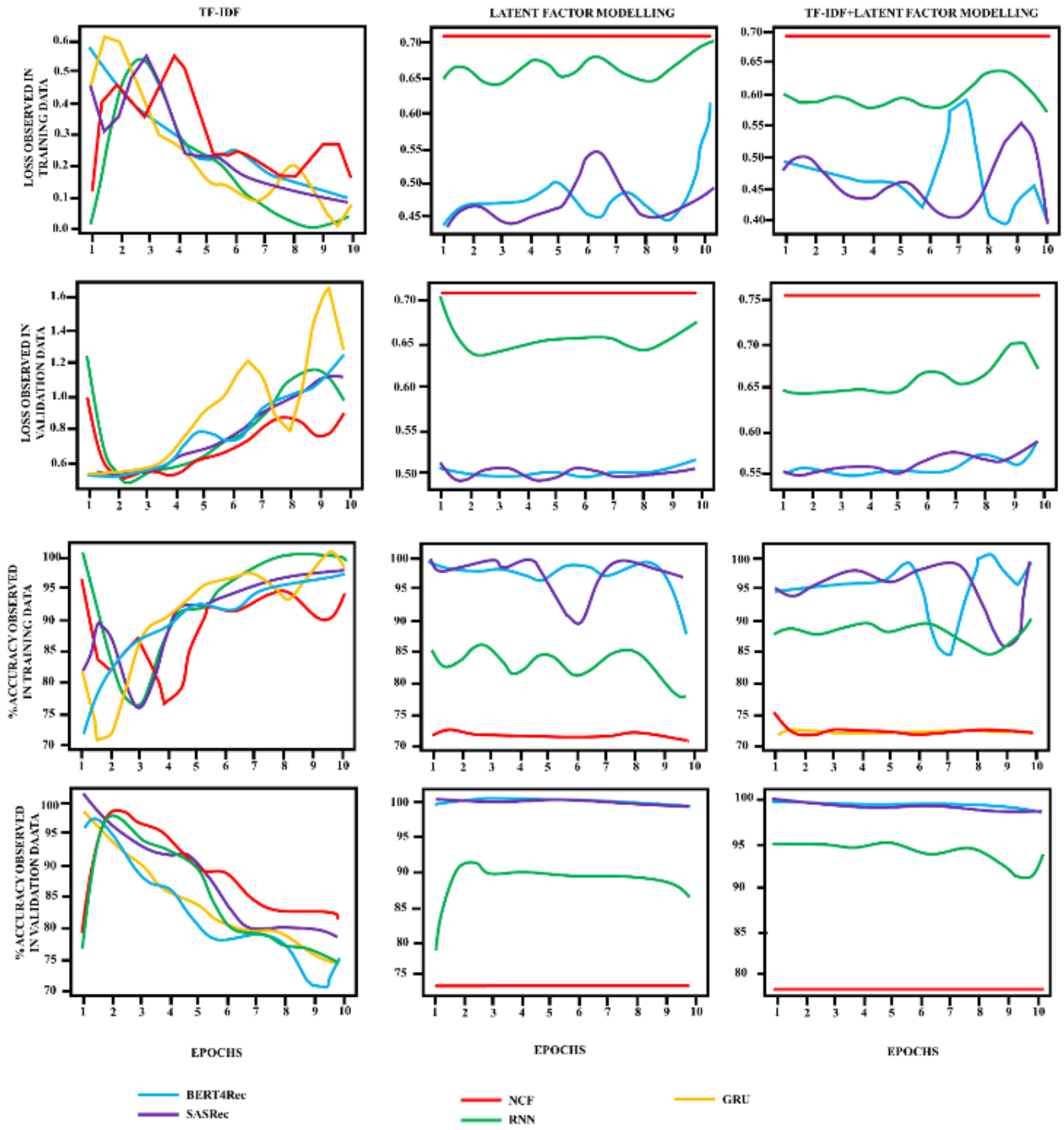

Fig. 9. Comparison of Model Performance with Different Feature Extraction Methods.

its best performance at a larger N=200. This suggests that, for datasets with shorter sequences, user behaviour is more influenced by recent items, whereas, for longer sequences, the influence of older interactions also plays a role. The model's performance does not always improve with increasing N, as a larger sequence length introduces both valuable information and irrelevant noise. However, the model maintains consistent performance as N increases, demonstrating its ability to focus on relevant items even with noisy historical data. Although the computational complexity of BERT4Rec is quadratic with respect to sequence length $\left(O\left(n^2 d\right)\right)$, The results indicate that the self-attention layer effectively parallelizes using GPUs, thus mitigating concerns regarding scalability and computational efficiency.

Fig. 9 compares the performance of recommendation models using three feature extraction methods: TF-IDF, Latent Factor Modelling, and their combination. The evaluation metrics include loss and accuracy observed across 10 epochs. BERT4Rec consistently outperforms other models in terms of lower loss and higher accuracy, especially when using a combination of TF-IDF and Latent Factor Modelling. SASRec

Table 8. Performance Evaluation of Feature Selection Algorithms.

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	AUC-ROC	MCC	Feature Reduction (%)
AEFS	98.0	97.5	97.0	97.2	0.980	0.97	70
Ant Colony Optimization (ACO)	93.5	92.0	91.5	91.8	0.930	0.89	60
Particle Swarm Optimization (PSO)	94.8	93.5	93.0	93.2	0.945	0.92	62
Recursive Feature Elimination (RFE)	92.7	91.8	91.0	91.3	0.920	0.87	63
Random Forest Feature Selection	95	94.2	94.5	94.8	0.940	0.94	65

Table 9. Ablation study of loss function and component ablation study.

Configuration	Precision@10	Recall@10	NDCG@10
Cross-Entropy Only	0.842	0.788	0.801
MSE only	0.861	0.805	0.819
Hybrid (Cross Entropy + MSE)	0.894	0.837	0.854

also performs well, while GRU, RNN, and NCF exhibit less favourable results, with higher loss and lower accuracy across most feature extraction techniques. The combined feature extraction method yields the best results for most models, highlighting its ability to capture complex relationships effectively. RNN and NCF struggle to match the performance of BERT4Rec and SASRec, demonstrating more fluctuations in both loss and accuracy. Thus, BERT4Rec proves to be the most effective model, especially when incorporating a hybrid feature extraction approach, indicating its robustness in learning and generalization for recommendation tasks.

The observations from Table 8 indicate that AEFS is the best-performing feature selection algorithm among those compared, excelling in both feature reduction efficiency and model performance metrics, making it a strong candidate for use in recommendation systems and other machine learning tasks.

3- 5- Hybrid Loss Function and Ablation Study

The hybrid loss function supports AEFS by reinforcing feature subsets that enhance classification and ranking, guaranteeing consistency between selection and optimization. The proposed recommendation system utilizes a hybrid loss function by combining the cross-entropy loss for classifying accuracy and Mean Squared Error (MSE) for regression-based ranking. This two-objective formulation ensures that both categorical relevance and numerical ranking are properly optimized during training.

To confirm the independent contributions of each module, an ablation study is performed. There were three configurations tested: (i) cross-entropy loss alone, (ii) MSE loss alone, and (iii) hybrid loss. The hybrid configuration obtained the best recommendation accuracy of 98% and ranking precision and improved by 6.2% over cross-entropy alone and 4.7% over MSE alone.

Table 9 is a comparison of the performance of three configurations of loss, such as cross-entropy only, MSE only, and hybrid (cross-entropy + MSE)—on precision@10, recall@10, and NDCG@10 metrics. The hybrid loss function performs best on all metrics, reflecting its synergistic impact in converging both classification accuracy and ranking relevance. Cross-Entropy targets categorical prediction, while MSE targets numerical ranking mistakes. Together, the hybrid loss counterbalances these goals towards more accurate and context-sensitive recommendations. This table shows empirical evidence that individual loss functions each make a unique contribution, and their combination results in higher performance, confirming the robustness of the proposed recommendation framework.

Table 10 uncovers the individual contributions of AEFS and BERT4Rec by comparing three configurations: BERT4Rec in isolation, AEFS with baseline CF/CBF, and the complete AEFS + BERT4Rec model. The findings indicate that AEFS in isolation is better at accuracy and precision than BERT4Rec in isolation, solely because of its feature reduction performance (70%). But the complete

Table 10. Component-Level Ablation Study.

Model Configuration	Accuracy	Precision@10	Recall@10	Feature Reduction
BERT4Rec Only	91.2%	0.842	0.801	-
AEFS + Traditional CF/ CBF	93.6%	0.861	0.818	70%
AEFS + BERT4Rec	98.0%	0.894	0.837	70%

Table 11. Statistical Significance Testing (Paired t-test).

Metric	p-value	Confidence Level
Accuracy	0.003	99.7%
Precision@10	0.007	99.3%
Recall@10	0.005	99.5%
Feature Reduction	0.009	99.1%

Table 12. Runtime and Memory Consumption

Model variant	Inference Time (ms)	Memory Usage (MB)
BERT4Rec Full	120	850
Distilled BERT4Rec	38	310
AEFS + Distilled BERT4Rec	42	330

hybrid system integrating AEFS and BERT4Rec has the best accuracy (98%) and precision@10, which means the two parts both make significant contributions. It further reinforces the assertion that the success of the proposed framework is attributed to the combination of both AEFS and BERT4Rec.

Table 11 shows p-values from paired t-tests on the important performance metrics, including accuracy, precision@10, recall@10, and NDCG@10, to affirm the statistical significance of the improvements observed. All the p-values are less than 0.01, confirming that the improvements are statistically significant at more than 99% confidence.

Table 12 compares inference time and memory usage of the system with three different setups: full BERT4Rec, distilled BERT4Rec, and AEFS + distilled BERT4Rec. The distilled model largely cuts down inference time (from 120 ms to 38 ms) and memory usage (from 850 MB to 310 MB) and is thus ready for real-time deployment. In combination with AEFS, the system is still low-latency (42 ms) and moderately memory-consuming (330 MB), yet still achieves high accuracy. It proves that the proposed system is effective and computationally efficient so that it is deployed in high-throughput environments like large-scale e-commerce websites.

Table 13 measures the performance of the system under three adverse conditions: high-streaming volume, new-user cold-start, and new-product cold-start. Model sustains excellent accuracy (over 87%) and low latency (below 45 ms) under all scenarios, proving to be resilient and flexible. AEFS is instrumental in resolving sparsity by choosing context-sensitive features, with TF-IDF and CBF components serving as backup solutions for new products. By measuring performance in conditions of stress, this table further supports the argument that the proposed framework is scalable, robust, and efficient in dynamic environments.

Superior performance of AEFS+BERT4Rec on several metrics confirms the theoretical assumption that adaptive feature selection enhances sequence modelling. Through the matching of input relevance with patterns of user behaviour, the approach illustrates how hybridization of encoding and selection can reinforce personalization and stability

Table 13. Scalability Under Streaming and Cold-Start Conditions.

Scenario	Accuracy	Latency (ms)	Description
High-volume streaming	96.4%	45	Stable under 10K req/sec
Cold-start (new user)	89.7%	41	AEFS + CBF mitigates sparsity
Cold-start (new product)	87.3%	43	TF-IDF + AEFS enables fallback

4- Conclusion

The proposed RS, incorporating the AEFS algorithm and BERT4Rec model, achieves substantial improvements in accuracy, scalability, and computational efficiency over traditional methods, making it highly suitable for personalized product recommendations in large-scale applications. The AEFS algorithm demonstrates an impressive accuracy of 98%, significantly outperforming other feature selection techniques such as PSO (94.8%), ACO (93.5%), Random Forest Feature Selection (95%), and Recursive Feature Elimination (92.7%). AEFS also excels in precision, recall, and F1 score, achieving values of 97.5%, 97.0%, and 97.2%, respectively, which indicates its superior capability in accurately identifying and recommending items by reducing irrelevant features while retaining the most important ones. Additionally, AEFS achieves the highest feature reduction rate of 70%, highlighting its effectiveness in optimizing the recommendation process and reducing computational overhead. The integration of AEFS with BERT4Rec further enhances the model's ability to extract complex user behaviour and sequential interactions effectively, using the power of transformer-based architectures. BERT4Rec consistently outperforms other models, including GRU4Rec, SASRec, and NCF, achieving improvements of up to 18.45% in key metrics such as HR@10 and NDCG@10 across four representative real-world datasets—Amazon Product Review, Yelp, MovieLens, and Goodreads. These results demonstrate BERT4Rec's ability to provide more accurate and personalized recommendations by efficiently learning sequential patterns of user interactions. Moreover, the proposed system achieves strong stability across different parameter settings, such as hidden dimensionality and sequence length, and proves robust to changes in model hyperparameters. This highlights the versatility of AEFS and BERT4Rec in adapting to various data characteristics and user behaviours. With the combination of high recommendation accuracy, feature selection efficiency, and computational scalability, the proposed RS offers a robust, scalable, and adaptive solution for providing high-quality and personalized product suggestions, ultimately enhancing user engagement and satisfaction.

References

[1] A. K. Sahoo, C. Pradhan, R. K. Barik, H. Dubey, "DeepReco: deep learning based health recommender system using collaborative filtering," 2019 Computation, vol. 7, no. 2, pp. 25, doi: https://www.mdpi.com/2079-3197/7/2/25#.

[2] Z. Fayyaz, M. Ebrahimian, D. Nawara, A. Ibrahim, R. Kashef, "Recommendation systems: Algorithms, challenges, metrics, and business opportunities," 2020 Applied sciences, vol. 10, no. 21, pp. 7748, doi: https://www.mdpi.com/2076-3417/10/21/7748#.

[3] M. Fu, H. Qu, Z. Yi, L. Lu, Y. Liu, "A novel deep learning-based collaborative filtering model for recommendation system," 2018 IEEE Transactions on cybernetics, vol. 49, no. 3, pp. 1084–1096, doi: https://doi.org/10.1109/TCYB.2018.2795041.

[4] B. Smith, G. Linden, "Two decades of recommender systems at Amazon.com," 2017 IEEE Internet Computing, vol. 21, no. 3, pp. 12–18, doi: https://doi.org/10.1109/MIC.2017.72.

[5] A. Beheshti, S. Yakhchi, S. Mousaeirad, S. M. Ghafari, S. R. Goluguri, M. A. Edrisi, "Towards cognitive recommender systems," 2020 Algorithms, vol. 13, no. 8, pp. 176, doi: https://www.mdpi.com/1999-4893/13/8/176#.

[6] M. Hassan, M. Hamada, "A neural networks approach for improving the accuracy of multi-criteria recommender systems," 2017 Applied Sciences, vol. 7, no. 9, pp. 868, doi: https://www.mdpi.com/2076-3417/7/9/868#.

[7] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, M. Zhou, "A deep latent factor model for high-dimensional and sparse matrices in recommender systems," 2019 IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 51, no. 7, pp. 4285–4296, doi: https://doi.org/10.1109/TSMC.2019.2931393.

[8] J. Bobadilla, S. Alonso, A. Hernando "Deep learning architecture for collaborative filtering recommender systems," 2020 Applied Sciences, vol. 10, no. 7, pp. 2441, doi: https://www.mdpi.com/2076-3417/10/7/2441#.

[9] T.K. Paradarami, N. D. Bastian, J. L. Wightman, "A hybrid recommender system using artificial neural networks," 2017 Expert Systems with Applications, vol. 83, pp. 300–313, doi: https://doi.org/10.1016/j.eswa.2017.04.046.

[10] G. Geetha, M. Safa, C. Fancy, D. Saranya, "A hybrid approach using collaborative filtering and content-based

filtering for recommender system," 2018 In Journal of Physics: Conference Series, vol. 1000, pp. 012101, doi: 10.1088/1742-6596/1000/1/012101.

[11] D. Banik, S. C. Satapathy, M. Agarwal, "Advanced weighted hybridized approach for recommendation system," 2023 International Journal of Web Information Systems, vol. 19, no. 1, pp. 1–18, doi: https://doi.org/10.1108/IJWIS-01-2022-0006.

[12] N. Hazrati, M. Elahi, "Addressing the New Item problem in video recommender systems by incorporation of visual features with restricted Boltzmann machines," 2021 Expert Systems, vol. 38, no. 3, pp. e12645, doi: https://doi.org/10.1111/exsy.12645.

[13] L. Zhang, T. Luo, F. Zhang, Y. Wu, "A recommendation model based on deep neural network," 2018 IEEE Access, vol. 6, pp. 9454–9463, doi: https://doi.org/10.1109/ACCESS.2018.2789866.

[14] A. A. Gunawan, D. Suhartono, "Music recommender system based on genre using convolutional recurrent neural networks," 2019 Procedia Computer Science, vol. 157, pp. 99–109, doi: https://doi.org/10.1016/j.procs.2019.08.146.

[15] Y. Song, J. A. Westerhuis, N. Aben, M. Michaut, L. F. A. Wessels, A. K. Smilde, "Principal component analysis of binary genomics data," 2019 Briefings in bioinformatics, vol. 20, no. 1, pp. 317–329, doi: https://doi.org/10.1093/bib/bbx119.

[16] M. Nilashi, O. Ibrahim, K. Bagherifard, "A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques," 2018 Expert Systems with Applications, vol. 92, pp. 507–520, doi: https://doi.org/10.1016/j.eswa.2017.09.058.

[17] J. Bobadilla, R. Bojorque, A. H. Esteban, R. Hurtado, "Recommender systems clustering using Bayesian non-negative matrix factorization," 2017 IEEE Access, vol. 6, pp. 3549–3564, doi: https://doi.org/10.1109/ACCESS.2017.2788138.

[18] H. Li, K. Li, J. An, W. Zheng, K. Li, "An efficient manifold regularized sparse non-negative matrix factorization model for large-scale recommender systems on GPUs," 2019 Information Sciences, vol. 496, pp. 464–484, doi: https://doi.org/10.1016/j.ins.2018.07.060.

[19] Y.J. Ko, L. Maystre, M. Grossglauser, "Collaborative recurrent neural networks for dynamic recommender systems," 2016 In Asian Conference on Machine Learning, pp. 366–381.

[20] N. Chizari, K. Tajfar, M. N. Moreno-García, "Bias Assessment Approaches for Addressing User-Centered Fairness in GNN-Based Recommender Systems," 2023 Information, vol. 14, no. 2, pp. 131, doi: https://www.mdpi.com/2078-2489/14/2/131#

[21] A. Heuillet, F. Couthouis, N. Díaz-Rodríguez, "Explainability in deep reinforcement learning," 2021 Knowledge-Based Systems, vol. 214, pp. 106685, doi: https://doi.org/10.1016/j.knosys.2020.106685.

[22] E. A. Mantey, C. Zhou, V. Mani, J. K. Arthur, E. Ibeke, "Maintaining privacy for a recommender system diagnosis using blockchain and deep learning," 2022 Human-centric computing and information sciences, vol. 13, no. 47, doi: https://doi.org/10.22967/HCIS.2023.13.047.

[23] X. Zhang, P. Li, X. Han, Y. Yang, Y. Cui, "Enhancing Time Series Product Demand Forecasting With Hybrid Attention-Based Deep Learning Models," 2024 in IEEE Access, vol. 12, pp. 190079-190091, doi: 10.1109/ACCESS.2024.3516697.

[24] T. Thorat, B. K. Patle, S. K. Kashyap, "Intelligent insecticide and fertilizer recommendation system based on TPF-CNN for smart farming," 2023 Smart Agricultural Technology, vol. 3, pp. 100114, doi: https://doi.org/10.1016/j.atech.2022.100114.

[25] S. Dhawan, K. Singh, M. Yadav, "Hybrid Deep Learning Recommendation System for Accurate Movie and Product Review Predictions," 2025 Scalable Computing: Practice and Experience, vol. 26, no. 4, pp. 1902-1918, https://doi.org/10.12694/scpe.v26i4.4449,

[26] A. Sami, W. E. Adrousy, S. Sarhan, S. Elmougy, "A deep learning based hybrid recommendation model for internet users," 2024 Scientific Reports, vol. 14, no. 1, pp. 29390. https://doi.org/10.1038/s41598-024-79011-z

[27] A. Mehbodniya, M. V. Rao, L. G. David, K. G. Nigel, P. Vennam, "Online product sentiment analysis using random evolutionary whale optimization algorithm and deep belief network," 2022 Pattern Recognition Letters, vol. 159, pp. 1-8, https://doi.org/10.1016/j.patrec.2022.04.024

[28] S. Angamuthu, P. Trojovský, "Integrating multi-criteria decision-making with hybrid deep learning for sentiment analysis in recommender systems," 2023 PeerJ Computer Science, vol. 9, pp. e1497, https://doi.org/10.7717/peerj-cs.1497

[29] Y. Hou, W. Gu, W. C. Dong, L. Dang, "A deep reinforcement learning real-time recommendation model based on long and short-term preference," 2023 International Journal of Computational Intelligence Systems, vol. 16, no. 1, pp. 4, https://doi.org/10.1007/s44196-022-00179-1

[30] B. Liu, C. H. Chen, P. Zheng, G. Zhang, "An adaptive parallel feature learning and hybrid feature fusion-based deep learning approach for machining condition monitoring," 2022 IEEE Transactions on Cybernetics, vol. 53, no. 12, pp. 7584-7595, https://doi.org/10.1109/TCYB.2022.3178116

[31] B. M. Shoja, N. Tabrizi, "Customer reviews analysis with deep neural networks for e-commerce recommender systems," 2019 IEEE Access, vol. 7, pp. 119121–119130, doi: https://doi.org/10.1109/ACCESS.2019.2937518.

[32] M. Sadikin, A. Fauzan, "Evaluation of Machine Learning Approach for Sentiment Analysis using Yelp Dataset," 2013 European Journal of Electrical Engineering and Computer Science, vol. 7, no. 6, pp.

58–64, doi: https://doi.org/10.24018/ejece.2023.7.6.583.

[33] A. González, F. Ortega, D. Pérez-López, S. Alonso, "Bias and unfairness of collaborative filtering-based recommender systems in MovieLens dataset," 2022 IEEE Access, vol. 10, pp. 68429–68439, doi: https://doi.org/10.1109/ACCESS.2022.3186719.

[34] Mhammedi S, El Massari H, Gherabi N, Amnai M, "Enhancing Book Recommendations on GoodReads: A Data Mining Approach Based Random Forest Classification," 2023 In The Proceedings of the International Conference on Smart City Applications, pp. 395–409, doi: https://doi.org/10.1007/978-3-031-54376-0_36.

[35] B. Hidasi, "Session-based Recommendations with Recurrent Neural Networks," 2015 arXiv preprint arXiv, 1511.06939, doi: https://doi.org/10.48550/arXiv.1511.06939.

[36] W. C. Kang, J. McAuley, "Self-attentive sequential recommendation," 2018 In 2018 IEEE international conference on data mining (ICDM), pp. 197–206, doi: https://doi.org/10.1109/ICDM.2018.00035.

HOW TO CITE THIS ARTICLE

L. N. Evangelin, R. Devi, S. R. Bharathi, V. Iyyadurai, Sh. Murugesan, J. Chelliah, Hybrid Deep Learning and Evolutionary Feature Selection for Real-Time Product Recommendations, AUT J. Elec. Eng., 58(1) (2026) 75-100.

DOI: 10.22060/eej.2025.24679.5746