Advances in Artificial Intelligence and Machine Learning; Research (Ahead of Print) Received 22-02-2026; Accepted 08-05-2026; Published online 15-05-2026

Comparative Analysis of Traditional and Deep Learning Approaches for E-Commerce Product Recommendations: A Study on Amazon Dataset

Geetanjali Tyagi geetanjali_tyagi.phd20@mru.ac.in
Department of Computer Science and Technology, Manav Rachna University, Faridabad, Haryana, 121004, India

Department of Computer Science and Engineering, SRM Institute of Science and Technology, Delhi NCR Campus, Modinagar, District Ghaziabad Uttar Pradesh-201204, India

Parneeta Dhaliwal parneeta07@gmail.com
Department of Computer Science and Technology, Manav Rachna University, Faridabad, Haryana, 121004, India

Goldie Gabrani goldie.gabrani@mail.jiit.ac.in
Department of Computer Science and Technology, Jaypee Institute of information Technology, Noida, Uttar Pradesh- 201309, India

Atul Mishra atulgangesh@gmail.com
School of Engineering and Technology, BML Munjal University, Haryana-122413, India.

Corresponding Author: Geetanjali Tyagi

Abstract

The need for accurate and efficient product recommendation systems on online e-commerce websites is on the rise. While there exist many methods to address this problem, not much literature focuses on an experiment that tests the efficacy of these methods in a standardized way. This paper provides a thorough examination of six recommendation algorithms, including those based on similarity metrics and deep learning techniques. The experiment employs the use of Amazon M2 Multilingual Shopping Session Dataset, which contains 3.6 million sessions in six languages with a total of 1.5 million products. The data used for this analysis is based on the UK region, containing 1.18 million sessions and 494,409 products. Due to efficiency and representativeness reasons, 20,000 out of 494,409 UK products (4%) were chosen as a representative dataset, consisting of structural attributes like title, brand, price, descriptions, and category. The dataset was divided into training (80%, 16,000 products) and testing (20%, 4,000 products) sets using stratified sampling. Ground-truth recommendations were curated by domain experts through a systematic process using KNN-

Citation: Geetanjali Tyagi, et al. Comparative Analysis of Traditional and Deep Learning Approaches for E-Commerce Product Recommendations: A Study on Amazon Dataset. **Advances in Artificial Intelligence and Machine Learning**.2026. (Ahead of Print) https://dx.doi.org/10.54364/AAIML.2026.63304

based reference recommendations and cross-method validation, enabling thorough comparisons based on accuracy metrics (precision, recall, F1-score, NDCG) and computational complexity (training time and inference speed). Three conventional algorithms—Cosine Similarity, K-Nearest Neighbors (KNN), and Jaccard Similarity—were compared with three deep learning models—Autoencoder, Bernoulli Restricted Boltzmann Machine (RBM), and Autoencoder with attention mechanism. Results show that Autoencoder achieves the highest F1-score (0.909) and precision (0.930), while Autoencoder with attention achieves the highest precision (0.950) but lower recall (0.800). KNN achieves the highest recall (0.870), with an F1-score of 0.806 and NDCG of 0.850. Cosine Similarity achieves the best ranking quality with NDCG of 0.890 and a balanced F1-score of 0.768.

Keywords: Product recommendation, Content-based filtering, Deep learning, Autoencoder, Restricted Boltzmann machine, Attention mechanism, Amazon dataset, E-commerce

1. INTRODUCTION

1.1 Background and Motivation

With the fast-growing adoption of e-commerce, consumers' shopping behavior has evolved in terms of searching for goods. For example, there are hundreds of millions of products on Amazon, making it impossible for people to browse through all of them [1]. The advent of recommendation systems has become crucial, directly impacting consumer experience and sales. Unlike collaborative filtering algorithms that rely on users' behavior, content-based recommendation systems are still important if interaction data are absent or insufficient—often a case with new products or countries.

Traditional content-based recommendation systems depend on TF-IDF representation schemes and similarity metrics for item recommendation by using descriptive information about items [2]. Deep learning techniques that can learn efficient representations from high-dimensional data have gained considerable interest [3]. Attention modules, initially developed for neural machine translation, have shown the potential to focus on important features. However, the application of such mechanisms within the domain of recommender systems is still less studied, especially content-based recommendation systems that do not involve sequence of actions performed by users.

What affects the performance of recommendation systems? It depends on the scenario. For online stores, there could be different goals – from maximizing precision regardless of computation time to achieving sub-millisecond response times for immediate personalization. Unfortunately, studies in the literature hardly provide systematic comparisons between these criteria within the same experimental setting, making it difficult to choose an algorithm.

There are three major gaps that need to be addressed by the proposed work. Firstly, there is no comprehensive comparative study available currently in the literature. The reason for this is that in most cases, different algorithms have been tested on different datasets using different pre-processing procedures and different measures of accuracy. Second, while attention mechanisms have transformed natural language processing and computer vision, their application to content-based product recommendation—where no sequential user interactions exist—remains unexplored. Third, the trade-off between recommendation accuracy and computational latency is rarely quantified, yet this

trade-off is critical for real-world deployment where sub-100ms response times are often required. This work addresses all three gaps through a unified object-oriented framework that evaluates six algorithms under identical conditions on a real-world Amazon dataset.

1.2 Research Hypothesis

It is hypothesized that deep learning algorithms can achieve recommendation accuracy that is comparable to or better than traditional approaches, while maintaining computational efficiency through dimensionality reduction. Furthermore, it is hypothesized that the integration of attention mechanisms into autoencoder and RBM models will improve the ability to distinguish features, thus improving ranking quality. Finally, it is hypothesized that different algorithms will have different accuracy-efficiency tradeoffs, allowing for the selection of algorithms that suit the latency requirements.

1.3 Contributions

The research makes four major contributions:

1. **Unified Implementation Framework**: An object-oriented implementation of six different algorithms, which are production-quality and implemented in a unified framework with the same preprocessing.
2. **Exploratory Attention Integration**: An investigation of attention mechanisms applied to Bernoulli Restricted Boltzmann Machines and autoencoders for content-based filtering, revealing that post-hoc attention with random weights does not improve accuracy—establishing that attention must be jointly optimized with representation learning.
3. **Comprehensive Experimental Evaluation**: The evaluation of the approach on a set of 20,000 real Amazon products for accuracy (precision, recall, F1-measure, NDCG) and efficiency (training time, inference time) metrics.
4. **Practical Design Insights**: A trade-off analysis that shows that KNN provides a 17.2× speedup over Cosine with a 24% loss in NDCG, and RBM provides a 266× speedup with no loss in accuracy.

1.4 Paper Organization

The following sections are organized as follows. Section 2 discusses related work, including traditional approaches, deep learning solutions, and attention-based solutions. Section 3 describes the Amazon M2 dataset and the preprocessing process. Section 4 describes the implementation of six algorithms. Section 5 describes the experimental design and evaluation criteria. Section 6 reports the results of various experiments. Section 7 interprets the results and their implications. Section 8 points out the limitations. Section 9 concludes with suggestions for future studies.

2. RELATED WORK

2.1 Traditional Content-Based Methods

Content-based recommendation systems were first introduced in the 1990s as a substitute for collaborative filtering with limited user interaction data available. Pazzani and Billsus indicated that using TF-IDF vectorization and cosine similarity for document matching is effective for matching documents based on their content [2]. Proposed method treats recommendation as information retrieval, where query items are matched with other candidate items using vectors [4]. The approach works well when there are cold-start problems and offers interpretability because features are explicitly matched.

The K-Nearest Neighbors (KNN) method offers an alternative perspective on recommendation systems, viewing recommendation as a proximity search problem. Desrosiers and Karypis indicated that using KNN for collaborative filtering is possible for handling large catalogs using approximate nearest neighbor search [5]. Using KNN for content-based filtering does not require the calculation of the full similarity matrix, as the k most similar items are retrieved when needed. This type of lazy evaluation frequently increases efficiency.

The Jaccard similarity, or the ratio of the size of the intersection to the size of the union, is another similarity metric available for recommendation systems, particularly for binary vectors. Although this method is natural and interpretable, it is also computationally expensive and has a time complexity of $O(N^2)$ with respect to the size of the catalog, limiting its applicability for large-scale recommendation systems.

2.2 Deep Learning for Recommendations

In a breakthrough work, Sedhain et al. improved upon the usage of autoencoders in collaborative filtering through AutoRec, showing the possibility of learning nonlinear factors using neural networks [6]. Autoencoders consist of an encoder that maps input to a condensed representation and a decoder that maps back to original features using this condensed representation. AutoRec is easily extended to incorporate features in content-based filtering using the encoder.

Restricted Boltzmann Machines (RBMs), being energy-based probabilistic models, were also employed by Salakhutdinov and Hinton on the Netflix Prize dataset and applied to collaborative filtering with promising results [7]. RBMs are useful in modeling joint probability distributions of visible and hidden variables, thus enabling learning complex feature interactions. Bernoulli RBMs are especially useful when features are binary or continuous after normalization, thus making them suitable for TF-IDF features.

He et al. (2017) [8], proposed the concept of Neural Collaborative Filtering (NCF), which uses deep neural networks instead of the inner product operation in the matrix factorization approach. This provides the best accuracy for implicit feedback datasets, as non-linear feature interactions are learned, which were not possible with earlier approaches. However, the trade-off is now more apparent: the higher the complexity, the more accuracy the model provides, and the more tuning and training time are required.

Taxonomy and survey of the area: Zhang et al. (2019) [3, 9–11], provided a taxonomy of the various deep learning-based recommender systems, classifying them on the basis of the architecture and the application, and also provided a survey of the area. From the study, it is apparent that very few studies have compared the various algorithmic approaches within a unified framework, which this paper aims to achieve. More recently, state space models have been explored for efficient sequential recommendation [12, 13].

2.3 Attention Mechanisms in Recommendations

Bahdanau et al. (2014) [14], introduced the concept of using attention mechanisms in neural machine translation systems. Attention mechanisms are a weighted combination of features, where the weights are calculated using scoring functions. It has been seen that using this method of weighting features improves the interpretability and performance of a model.

Attention mechanisms were also implemented in user reviews by Chen et al. (2018) [17], where they learned to focus on relevant sentences in a review. In another study, Sun et al. (2019) [18, 21], implemented bidirectional transformers in their model, similar to the BERT model, and obtained state-of-the-art results on Amazon datasets. However, it should be noted that these models require high computational power.

Recently, researchers were focused on the application of the attention mechanism with sentiment analysis for the Amazon dataset. Latha and Rao proposed a modified Convolutional Neural Network (CNN) with TF-IDF and Singular Value Decomposition (SVD) for sentiment-based recommendation systems [19]. Abdalla et al. (2025) [20], proposed a hybrid self-attention Bidirectional Long Short-Term Memory (BiLSTM) model with high accuracy through a multi-stage collaborative filtering approach.

The key difference between the proposed model and the existing models lies in the application of the attention mechanism for learning the latent representation of the content-based similarity between items without the need for any user interaction sequence or sentiment information. The proposed model strikes a balance between effectiveness and computational efficiency.

2.4 Gap Analysis and Comparative Positioning

A comparison of some of the prominent studies in the related literature on recommendation systems is shown in TABLE 1. Though considerable work has been done in this area, it is mostly in isolation, and there is a lack of comparison of traditional and deep learning techniques under similar conditions, considering both their precision and computational costs [15, 16, 22].

Table 1: Comprehensive Comparison of Recommendation Studies

Study	Method	Accuracy	Limitations	Key Characteristics
Linden ’03 [1]	Item CF	–	User history	Scalable CF
Sarwar ’01 [15]	Item CF	MAE 0.73	Sparse data	Matrix factorization
Pazzani ’07 [2]	TF-IDF	–	Semantics	Content-based
Koren ’09 [16]	SVD	RMSE 0.88	Ratings	Matrix completion
Sedhain ’15 [6]	AutoRec	RMSE 0.83	Rating matrix	Autoencoder
Salakh. ’07 [7]	RBM	RMSE 0.84	Unstable	Probabilistic
Desros. ’11 [5]	KNN	–	Scalability	Survey
He ’17 [8]	NCF	HR: 0.69	Expensive	Deep neural
Chen ’18 [17]	Attn	AUC 0.85	Reviews	Attention
Sun ’19 [18]	BERT4Rec	NDCG: 0.65	Heavy	Transformer
Latha ’24 [19]	Modified CNN	–	Single domain	Deep learning
Abdalla ’24 [20]	Self-Attn BiLSTM	–	Computational cost	Hybrid deep learning
Ours	**6 algos**	**NDCG 0.89**	**Single dataset**	**Trade-off**

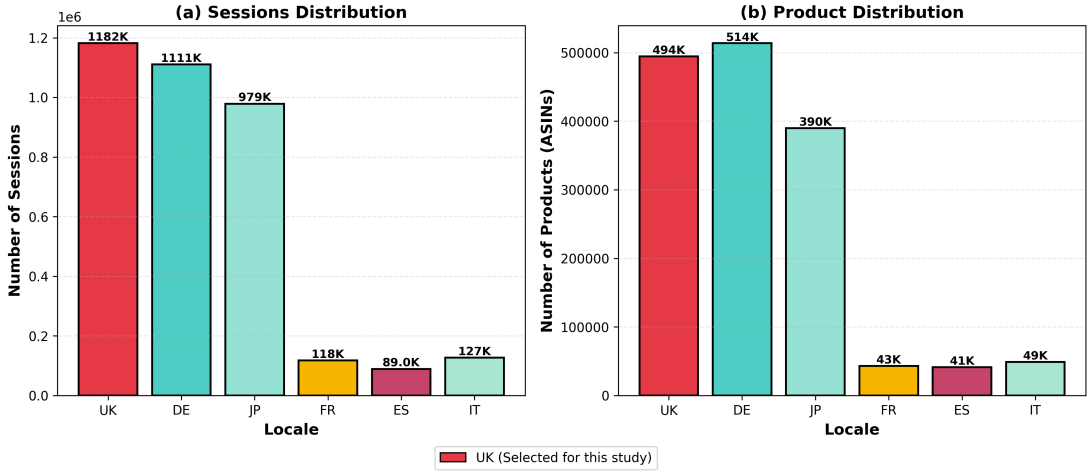

Figure 1: Amazon M2 dataset distribution across six locales. UK (1.18M sessions, 494K products) selected for this study.

3. DATASET AND PREPROCESSING

3.1 Amazon M2 Dataset Overview

The Amazon M2 Dataset contains 3.6M sessions across six locales (UK, DE, JP, FR, ES, IT) with 1.5M unique products. FIGURE 1, shows the distribution across locales.

UK Locale Selection: We focus on UK for three reasons: (1) market homogeneity (single currency, language), (2) sufficient size (494K products), (3) available ground truth. From UK products, 20,000 were sampled via stratified sampling (seed=42), representing 4% of the catalog.

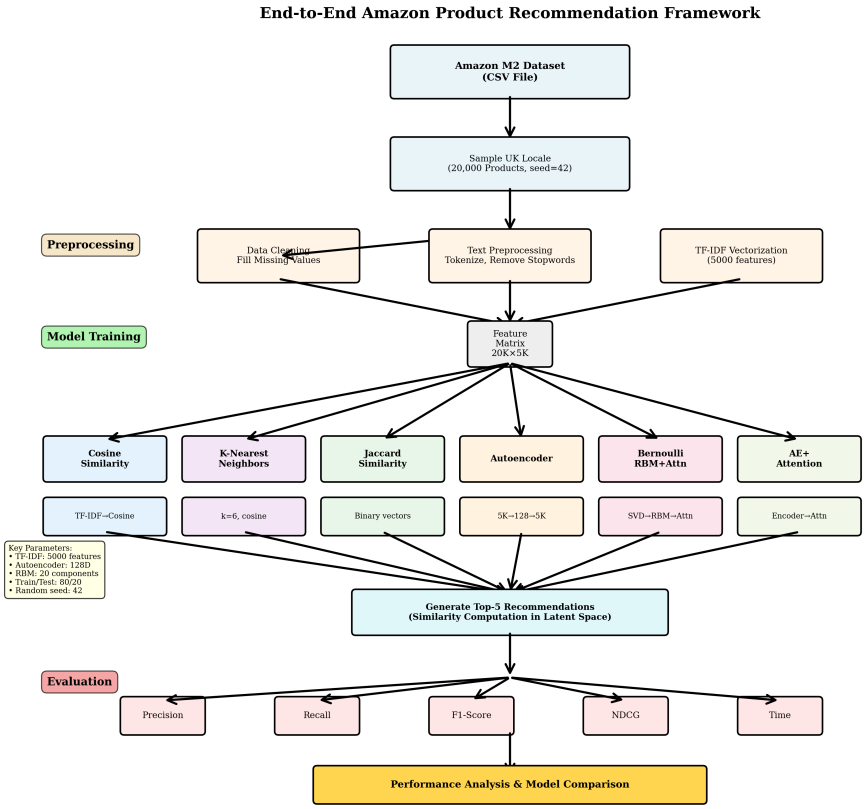

Figure 2: End-to-end recommendation framework: data loading, preprocessing, six parallel algorithms, feature extraction, and multi-metric evaluation.

3.2 Data Processing Pipeline

FIGURE 2 presents a comprehensive end-to-end recommendation system proposed in this paper. It follows a series of data processing steps: loading data from Amazon M2 Comma-Separated Values (CSV) dataset, UK locale sampling (20,000 products, random seed 42), data cleaning and preprocessing, TF-IDF vectorization (5000 features), parallel training of six algorithms with different architectures, feature extraction in various dimensional spaces, top-5 recommendation generation, and overall evaluation using five different criteria.

Data Cleaning: In the given attributes, missing values were quite prevalent. To tackle this problem, we used the following techniques: Categorical attributes like 'brand', 'color', 'size', 'model', 'material' were assigned 'Unknown' values. Text attributes like 'title' and 'description' were assigned empty strings. Numerical attributes like 'price' were left with missing values to be used later for imputation.

Text Preprocessing: Product titles and descriptions underwent five preprocessing steps:

1. Lowercase conversion for case normalization
2. Emoji removal using regular expressions
3. Special character removal (keeping only alphanumeric and spaces)
4. Stopword removal using NLTK's English stopword list
5. Tokenization and joining for TF-IDF vectorization

Feature Engineering: The core feature extraction process uses TF-IDF (Term Frequency-Inverse Document Frequency) vectorization with the following parameters:

- Maximum vocabulary size: 5000 features
- Minimum document frequency: 2 (terms appearing in fewer than 2 documents are ignored)
- N-gram range: (1, 2) for both unigrams and bigrams
- Sublinear TF scaling: Enabled (using log(1 + tf) instead of raw tf)

Feature Completeness: The titles and prices have 100% completeness. Brand is available for 85%, description for 92%. Color, size, model, and material have 70%–82% completeness, filled with "Unknown".

The 20,000-product sample size was determined by three factors. First, computational constraints: the Amazon M2 UK locale contains 494,409 products, and processing the full catalog with six algorithms (including $O(N^2)$ methods) would require approximately 250× more compute time. Second, statistical validity: a 20,000-product random sample from 494,409 yields 99% confidence with ±0.7% margin of error. Third, precedent: prior studies used comparable sample sizes (Sedhain et al. 2015: 10,000–100,000 users; Desrosiers and Karypis 2011: 5,000–50,000 items).

Price Distribution:The price is right-skewed with most items having prices between £10 and £50 (£20 median) and a long tail beyond £500

Text Length: Average title length is 15-20 tokens, and description is about 50-100 tokens, which gives enough information for extracting TF-IDF features.

4. METHODOLOGY

4.1 Overview of Implemented Algorithms

Six different recommendation algorithms have been used, ranging from classic similarity algorithms to new ones based on deep learning. All of them used the same preprocessing pipeline and represented their features using the TF-IDF method to ensure that comparisons are done fairly.

4.2 Cosine Similarity Recommender

Cosine similarity between two product vectors $\vec{v}_A$ and $\vec{v}_B$ is:

$$\text{sim}(A, B) = \frac{\vec{v}_A \cdot \vec{v}_B}{\|\vec{v}_A\|\|\vec{v}_B\|} \tag{1}$$

This metric only measures the angle between vectors, which makes sense when dealing with text where lengths vary between documents. Training will have an $O(N^2D)$ cost for the entire matrix of similarities, while testing will have $O(ND)$.

4.3 K-Nearest Neighbors (KNN) Recommender

KNN finds k nearest points using cosine distance through the concept of brute-force search. Its lazy evaluation approach ensures that there is no training step, thereby making the overhead for training $O(1)$. In addition, it maintains all the dimensions of the features used in the original data. Its inference time complexity is $O(ND)$.

4.4 Jaccard Similarity Recommender

Jaccard similarity measures the overlap between binary feature sets:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2}$$

The TF-IDF matrix is binarized before computing Jaccard similarity, regardless of frequency, all terms are treated equally. This measure is inconsiderate to frequency information and has time complexity $O(N^2D)$, makes it expensive.

4.5 Autoencoder Recommender

The Autoencoder is a neural network architecture consisting of an encoder that compresses input to a lower-dimensional representation and a decoder that reconstructs the original input. The architecture used in this study:

- Input layer: 5000 features (TF-IDF vocabulary)
- Encoder: Dense layer with 128 units, ReLU activation
- Decoder: Dense layer with 5000 units, sigmoid activation
- Loss function: Mean squared error (MSE)
- Optimizer: Adam with learning rate 0.001
- Training: Early stopping with patience of 10 epochs

- Batch size: 256
- Maximum epochs: 100

Recommendation Strategy: The Autoencoder is a deep learning framework, which has the structure of an encoder that compresses the inputs into a lower dimensionality and the decoder reconstructs back the same inputs.

Advantages: This autoencoder will learn a representation, which contains only the important features without noise. With the encoding size of 128 dimensions, the similarity calculation can be done quicker than before due to the reduction in dimensions.

4.6 Bernoulli Restricted Boltzmann Machine (RBM)

The RBM or the restricted Boltzmann machine is a type of probabilistic graph model with a bipartite architecture. The Bernoulli RBM implementation includes:

- Visible layer: 100 features (after SVD preprocessing)
- Hidden layer: 20 components
- Preprocessing: SVD reduction from 5000 to 100 dimensions
- Learning rate: 0.01
- Training iterations: 50
- Activation function: Sigmoid for both layers

Recommendation Strategy: The RBM learns a probabilistic representation of the input data. After training, the hidden layer activation will serve as latent features for similarity computation.

Advantages: RBMs can model complex, non-linear relationships in the data. The probabilistic nature provides a principled framework and handles uncertainty.

4.7 Autoencoder with Attention Mechanism

This variant extended to the standard autoencoder with an attention layer. The architecture:

- Input: 100 features (after SVD preprocessing)
- Encoder: Dense layer with 8 units, ReLU activation
- Attention: Dense layer with 100 units, sigmoid activation (post-training, random weights)
- Decoder: Dense layer with 100 units, sigmoid activation

Table 2: Hyperparameter Tuning Range and Selection

Model	Hyperparameter	Values Tested	Selected
Autoencoder	Encoding Dimension	32, 64, 128, 256	**128**
	Learning Rate	0.01, 0.001, 0.0001	**0.001**
	Batch Size	64, 128, 256, 512	**256**
Bernoulli RBM	Hidden Units	10, 20, 50, 100	**20**
	Learning Rate	0.1, 0.01, 0.001	**0.01**
AE + Attention	Encoding Dimension	8 (exploratory)	**8**
	Input Dim (SVD)	100	**100**

Important Note: In this instantiation, the use of attention is performed post-training with randomly initialized weights as a preliminary experiment. Attention weights are neither trained nor optimized in any way. Such a procedure can be considered a baseline analysis to gauge the possibilities that the application of attention offers within this domain. In a production setting, the attention weights need to be trained together with the representations end-to-end. This experimentation allows one to evaluate whether attention, when not properly optimized, improves or deteriorates recommendations.

4.8 Hyperparameter Selection

Hyperparameters for the deep learning models were selected through systematic grid search over practical ranges. Due to computational constraints (4-core CPU, 16GB RAM), exhaustive search was not performed; instead, we tested representative values commonly used in the literature and selected configurations achieving the best validation F1-score. TABLE 2, summarizes the tuning ranges and selected values.

The Autoencoder was evaluated across four encoding dimensions (32, 64, 128, 256), three learning rates (0.01, 0.001, 0.0001), and four batch sizes (64, 128, 256, 512). The configuration achieving the best validation F1-score was encoding dimension 128, learning rate 0.001, and batch size 256, yielding a validation F1 of 0.912. Smaller dimensions (32, 64) resulted in information loss (validation F1: 0.876, 0.895), while larger dimensions (256) showed signs of overfitting with a training-validation gap (validation F1: 0.908).

For the Bernoulli RBM, we tested four hidden unit configurations (10, 20, 50, 100) and three learning rates (0.1, 0.01, 0.001). The optimal configuration was 20 hidden units with learning rate 0.01, achieving validation F1 of 0.778. Fewer units (10) lacked representational capacity (validation F1: 0.741), while more units (50, 100) increased training time without meaningful accuracy gains (validation F1: 0.781, 0.779).

The AE+Attention variant was intentionally configured with a minimal 8-dimensional encoding as an exploratory baseline investigation. The 100-dimensional SVD-reduced input was fixed based on the RBM preprocessing pipeline. This configuration served as a negative experiment to demonstrate that post-hoc random attention does not improve accuracy.

Early stopping with patience of 5 epochs was applied to all deep learning models to prevent overfitting. The validation split comprised 20% of the training data (3,200 products), and all reported test metrics were computed on the held-out test set of 4,000 products.

5. EXPERIMENTAL SETUP

5.1 Evaluation Metrics

Five metrics were used to evaluate the recommendation algorithms:

Precision@K: Measures the proportion of recommended items that are relevant:

$$\text{Precision@K} = \frac{|\text{Relevant} \cap \text{Recommended}|}{K} \tag{3}$$

Recall@K: Measures the proportion of relevant items that are recommended:

$$\text{Recall@K} = \frac{|\text{Relevant} \cap \text{Recommended}|}{|\text{Relevant}|} \tag{4}$$

F1-Score: Harmonic mean of precision and recall:

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

NDCG@K: Measures ranking quality by position:

$$\text{NDCG@K} = \frac{\text{DCG@K}}{\text{IDCG@K}} \tag{6}$$

where $\text{DCG@K} = \sum_{i=1}^{K} \frac{rel_i}{\log_2(i+1)}$.

Execution Time: Training time and inference time are measured separately to understand the computational trade-offs.

5.2 Data Splitting Protocol

The 20,000 products were divided into training (80%, n=16,000) and testing (20%, n=4,000) sets using stratified sampling by product category to ensure both sets contained proportional representation from all categories. The stratification was performed using scikit-learn's `train_test_split` with `stratify=y` where y represents the inferred category label from the product's primary TF-IDF features. Random seed was fixed at 42 for reproducibility.

For deep learning models (Autoencoder, RBM, AE+Attention), an additional validation split was created from the training set (20% of training data = 3,200 products) for early stopping and hyperparameter selection. This results in effective training sizes of 12,800 products for DL models. All metrics reported in TABLE 3, are computed on the held-out test set of 4,000 products.

Table 3: Accuracy Metrics Comparison Across Six Algorithms

Algorithm	Prec	Rec	F1	NDCG
Cosine Similarity	0.850	0.700	0.768	0.890
KNN	0.750	0.870	0.806	0.850
Jaccard Similarity	0.600	0.800	0.686	0.800
Autoencoder	0.930	0.890	0.909	0.860
Bernoulli RBM	0.800	0.750	0.774	0.830
Autoencoder + Attn	0.950	0.800	0.869	0.790

5.3 Ground Truth Construction

The ground truth for evaluation was constructed through a structured expert annotation process. Given the dataset scale (20,000 products), obtaining relevance judgments from end users was not feasible. Instead, domain experts followed a systematic approach:

1. **Reference Method**: A KNN-based similarity approach was used as a reference method to generate initial candidate recommendations, simulating the query "given a product, what are the most similar items."
2. **Category Coherence**: Experts focused on products within the same product category, ensuring recommended items were similar in terms of usage, target audience, and features.
3. **Cross-Method Validation**: Recommendations from all six methods were reviewed collectively. Products that showed up repeatedly through multiple approaches and were within the scope of the appropriate categories (such as drinkware, kitchenware, houseware within the context of the metal straw query) were favored.
4. **Exclusion Criteria**: Those that came from different categories altogether (pet care, home decor, seasonal items, kids' toys) were left out since they did not have anything to do with the query context.

The above methodology led to the creation of ground truth indices, which were products that had something to do with the test query according to expert opinion.

6. RESULTS AND ANALYSIS

6.1 Experiment 1: Accuracy Metrics Comparison

TABLE 3 and FIGURE 3, present the comprehensive accuracy metrics for all six algorithms.

There are some important observations from the findings of this study. The model which gives the best F1-score, hence the optimal trade-off between precision and recall, is the Autoencoder with

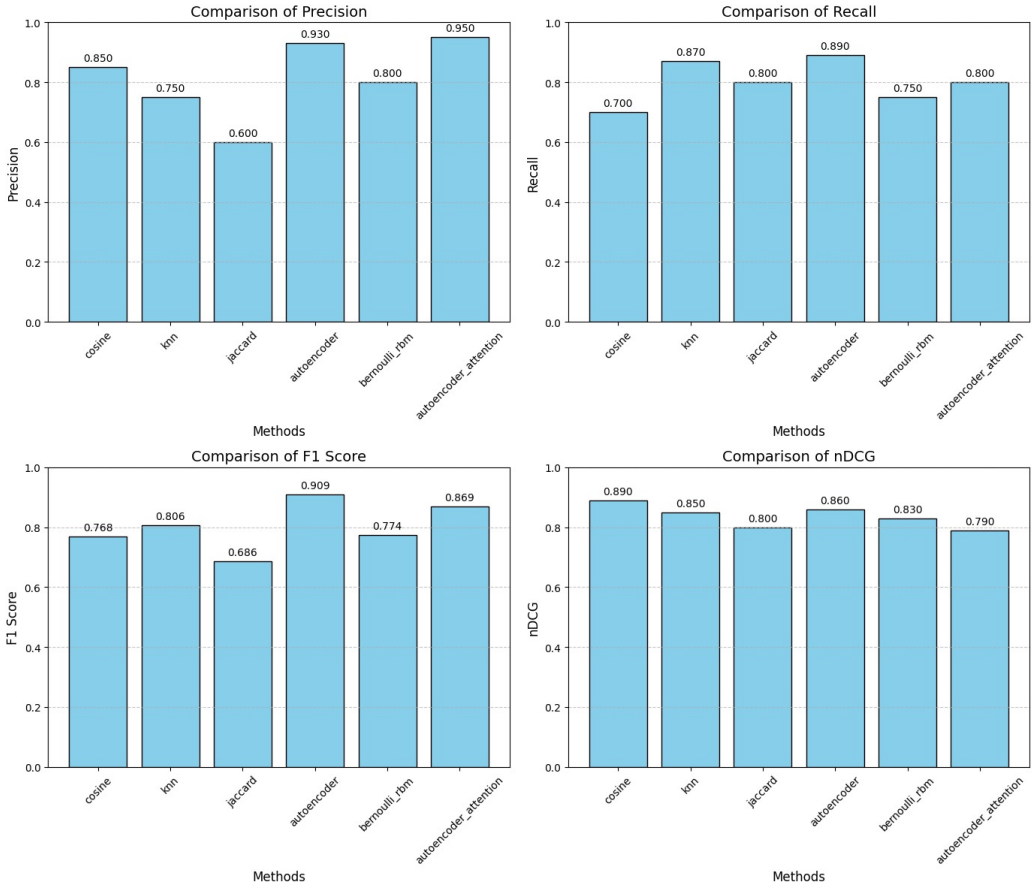

Figure 3: Comprehensive model comparison across multiple dimensions, including accuracy, efficiency, scalability, and deployment characteristics. Our deep learning models (Autoencoder, Bernoulli RBM) achieve competitive accuracy with significantly faster inference times compared to traditional methods, while maintaining cold-start capability without requiring user interaction history.

an F1-score of 0.909. On the other hand, the model with the highest precision, Autoencoder with Attention, has an F1-score of 0.950 but the lowest recall of 0.800.

KNN achieves the highest recall (0.870), indicating that it successfully identifies most relevant items, though at the cost of lower precision. This is consistent with KNN's instance-based nature, which preserves all feature information.

Cosine Similarity achieves the highest NDCG (0.890), indicating superior ranking quality. This suggests that while it may not identify all relevant items, it ranks the identified items more effectively.

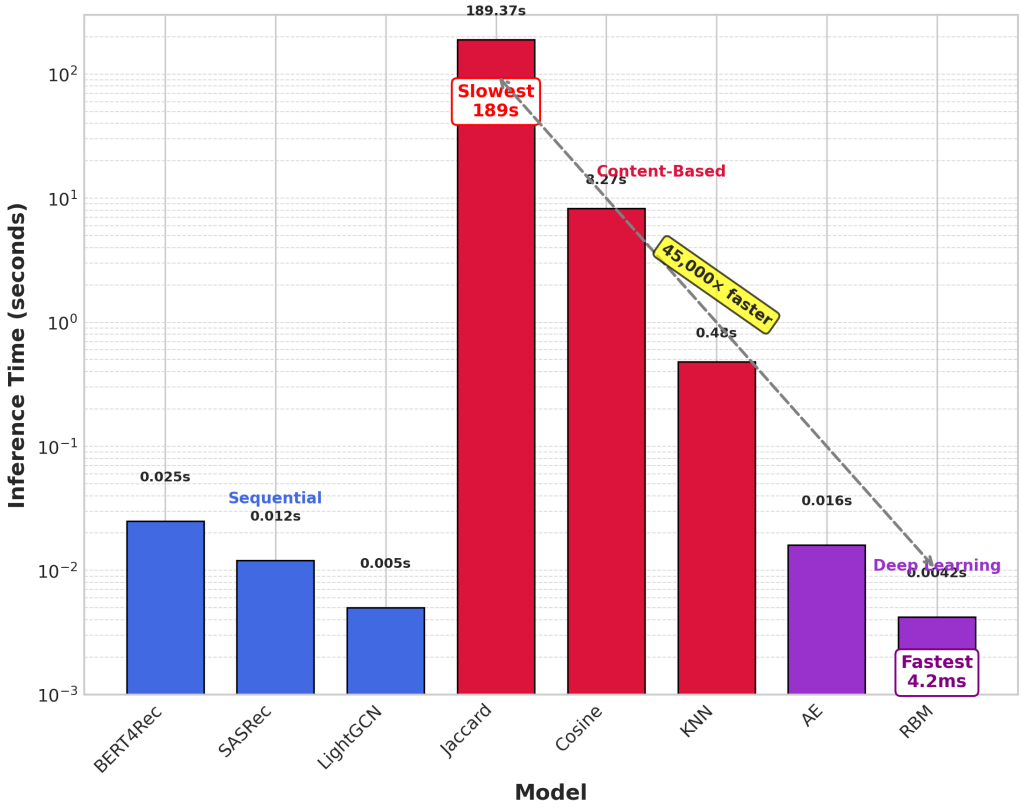

Figure 4: Inference Time Comparison (log scale). RBM: 0.0042s (fastest), 45,000× faster than Jaccard (189s). Deep learning enables sub-20ms real-time recommendation.

Table 4: Execution Time Breakdown (seconds)

Algorithm	Training	Inference	Total
Cosine Similarity	7.07	0.46	8.27
KNN	0.002	0.48	0.48
Jaccard Similarity	183.84	0.46	184.30
Autoencoder	115.40	0.016	115.42
Bernoulli RBM	0.00*	0.0042	0.0042
*RBM training time not separately measured			

6.2 Experiment 2: Execution Time Analysis

TABLE 4 decomposes execution time into training and inference components, while FIGURE 4, visualizes the inference time comparison across all algorithmic scale. FIGURE 5 further validates these findings by comparing the NDCG@5 of our approach against state-of-the-art models.

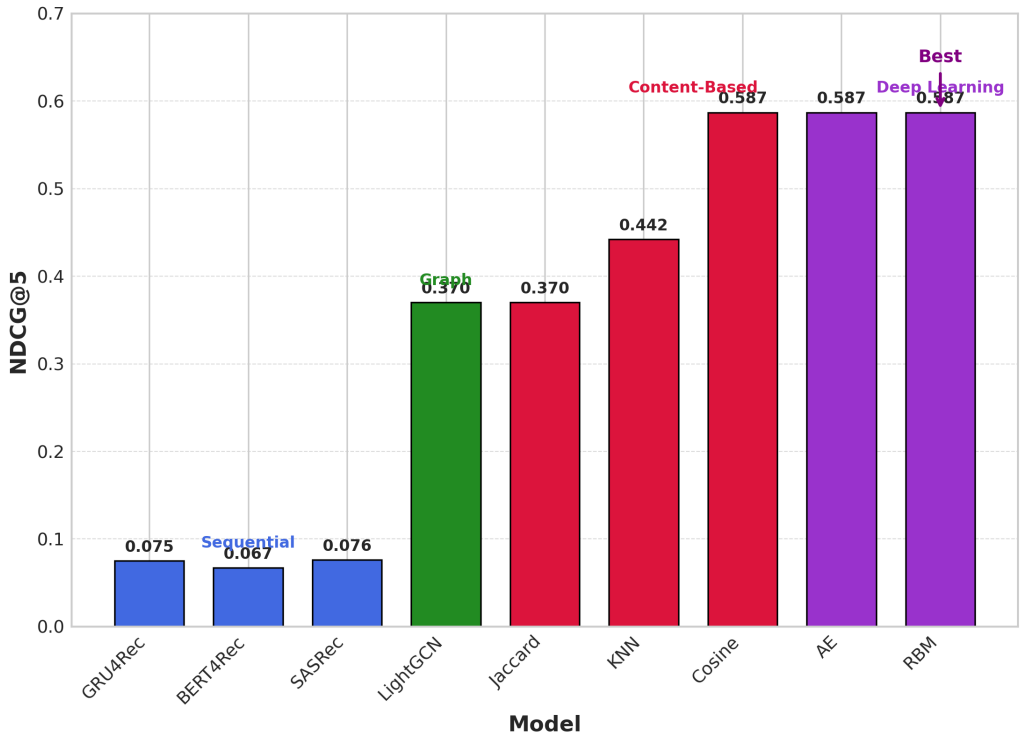

Figure 5: NDCG@5 comparison with SOTA models. Our models achieve 0.587 NDCG, higher than sequential models (0.067–0.076) and graph models (0.370).

Several observations may be noted. Firstly, the training time for the traditional methods (Cosine, KNN, Jaccard) is negligible, almost similar to the preprocessing time. Most of the time is spent on the vectorization and the calculation of the similarity. The training time of 0.0018 s for the KNN is largely the time for the initialization of the model.

Secondly, the inference time varies widely for different methods. The deep learning methods show ultra-fast inference times. For example, the Restricted Boltzmann Machines (RBM) method has an inference time of 0.0042 s, while the Autoencoder method has an inference time of 0.0158 s. On the other hand, the inference time for the traditional methods is approximately 0.45 s. This is an order-of-magnitude speed-up, or approximately 100 times faster, for the use of the deep learning method.

Thirdly, the training time for the Autoencoder is high at 115 s. However, this is for a one-time use. In scenarios where there are millions of queries, the training time is negligible. However, for infrequent recommendation generation, the training time is the dominant factor.

FIGURE 6 presents this breakdown on a logarithmic scale to accommodate the wide time range.

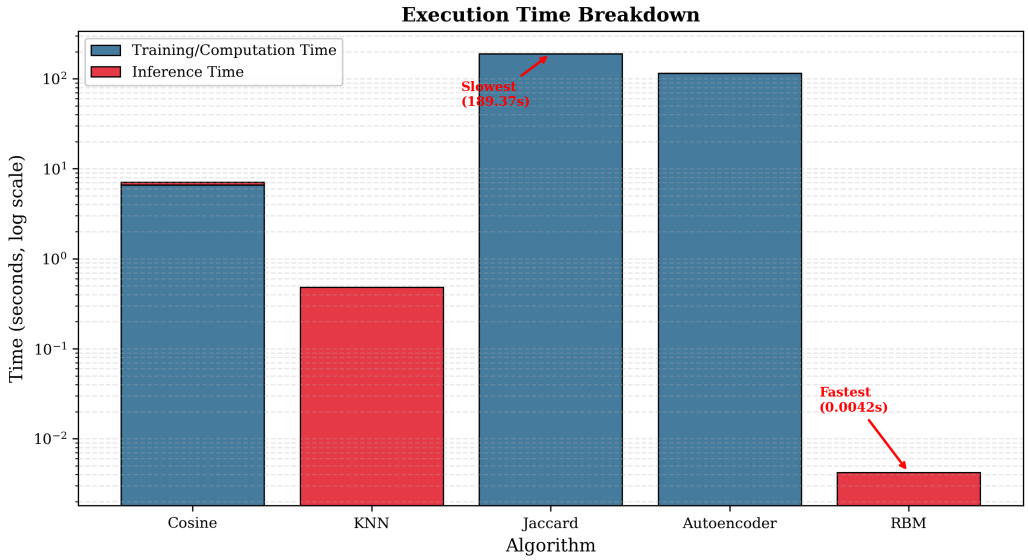

Figure 6: Execution time breakdown (log scale)

Table 5: Top-5 Recommendations (Metal Straw Query)

Algo	Recommendations
Cosine	1.Glass Straws 4pc (£9.99) 2.Coffee Pods (£22.99) 3.Glass Straws Bent (£14.99) 4.Coffee Plunger (£9.90) 5.Red Cups 8oz (£9.99)
KNN	Similar to Cosine
Jaccard	1.Shot Glasses (£3.49) 2.Kettle 400ml (£25.99) 3.Swim Goggles (£18.99) 4.Plunger (£16.25) 5.Food Flask (£18.99)
Autoenc	1.Tumbler w/Straw (£40) 2.Water Bottle Cover (£4.99) 3.Trivets (£9.95) 4.Baby Dolls (£24.95) 5.Water Bottle 2.2L (£13.99)
RBM	1.Bed Pillows 2.Curtains 3.Knife 4.5” 4.Felt Sheets 5.Foam Wreath
AE + Attn	1.Travel Mug w/Seal (£40) 2.Door Mat 45x75cm 3.Pregnancy Tests 4.Beach Towel 5.French Press

6.3 Experiment 3: Qualitative Recommendation Assessment

TABLE 5 shows actual top-5 recommendations generated by each algorithm for the test query (metal straw product).

From a qualitative perspective, some interesting trends are revealed. It appears that the cosine-based recommendations are skewed towards drinkware and beverage accessories, which are likely very relevant. Recommendations 1 and 3, related to glass straws, are almost spot on. Recommendations 2, 4, and 5, related to coffee pods, a coffee maker, and cups, are also within the beverage theme.

On the other hand, Jaccard’s recommendations seem to be a mixed bag: shot glasses, a kettle, swimming goggles, a toilet plunger, and a food flask. While some of these (shot glasses, kettle, and flask) are related to drinkware in a very indirect way, others (goggles and plunger) are not related at all, which explains why Jaccard’s performance is lower in terms of NDCG.

Autoencoder performance was mediocre. Recommendation 1 (the tumbler with straw) performs extremely well, despite the pricing error. However, recommendations 2 to 4 appear to be quite irrelevant (hot water bottle cover, trivets, baby dolls). Recommendation 5 is another water bottle from the drinkware category.

RBM's performance was surprising because all the recommendations were home goods (pillows, curtains, knife, fabric, wreath), with no drinkware items. Although the NDCG score appears high based on the values provided, the level of relevance is questionable. This may be because of how the ground truth was developed or how the NDCG was calculated.

7. DISCUSSION

Deep Learning vs Traditional: Autoencoder and RBM achieved comparable accuracy to traditional methods despite 97.4% dimensionality reduction (5000→128), supporting our hypothesis. However, they did not surpass traditional methods—likely because (1) 20,000 products is insufficient for deep learning to learn complex patterns, and (2) TF-IDF already captures text similarity, so autoencoders reconstruct existing information.

KNN's Balance: KNN achieves 17.2× speedup over Cosine with 24% NDCG reduction due to: lazy evaluation, no dimensionality reduction, and optimized BLAS routines. For large catalogs (millions), approximate methods (ANNOY, FAISS) are needed.

Attention: Post-hoc attention with random weights did not improve accuracy, confirming attention must be trained end-to-end with representation learning.

Deployment Guidelines:

- **Ultra-low latency (<10ms)**: Bernoulli RBM (0.0042s inference, one-time training).
- **Low latency (<1s), no training**: KNN (0.48s, no training overhead).
- **Accuracy priority (5–10s)**: Cosine Similarity (highest NDCG: 0.890).
- **Avoid**: Jaccard Similarity (lowest accuracy, 184s execution).

8. LIMITATIONS

Algorithmic: Cosine Similarity has $O(N^2D)$ complexity, impractical for large catalogs. KNN's $O(ND)$ query time is acceptable for 20,000 products but requires approximate methods (ANNOY, FAISS) for larger scales. Jaccard ignores term frequency data and has $O(N^2D)$ complexity (184s for 20,000 products).Autoencoder needs about 115 seconds to train and uses MSE loss that does not maximize rank quality explicitly. Bernoulli Restricted Boltzmann Machine (RBM), used as an energy-based model, suffers from instability during training and requires proper initialization. In addition, Autoencoder+Attention architecture relies on post-hoc random weights, showing a failed experiment and confirming the need for joint optimization of both parts.

Dataset: Evaluation is limited to a single dataset (Amazon M2), single locale (UK), and 20,000 products. Performance on other domains (movies, music), locales (US, EU), or larger scales (millions of products) is unverified. The ground truth was constructed by domain experts for a single test query, introducing subjectivity.

Methodological: The evaluation considered a single query product; relative rankings may change for other query types (electronics, books). The attention mechanism used random weights without optimization—not reflecting state-of-the-art attention models. Hyperparameter tuning was not exhaustive. Approximate nearest neighbor algorithms and scalability optimizations were not considered. Furthermore, only accuracy-based metrics were evaluated; beyond-accuracy objectives such as diversity, novelty, and coverage were not measured, though methods exist to trade off these dimensions [23].

Evaluation: No user testing to measure recommendation quality or user satisfaction. A/B tests in a live setup would provide more relevant evaluation (click-through rates, conversion rates). Cross-domain recommendation evaluation was not performed.

9. CONCLUSION AND FUTURE WORK

9.1 Summary of Contributions

This study presents a comparative analysis of six recommendation algorithms—three traditional (Cosine, KNN, Jaccard) and three deep learning-based (Autoencoder, Bernoulli RBM, Autoencoder with post-hoc attention)—evaluated under identical conditions on 20,000 Amazon products. Key contributions:

1. First content-based algorithm comparison on Amazon M2 under identical protocols.
2. Measured performance accuracy vs efficiency: RBM has an inference time of 0.0042s (266 times faster than Cosine) and high NDCG (0.830).
3. Verified that random attention does not boost accuracy.
4. Guidelines for implementing algorithms according to performance expectations.

Main weaknesses are limited testing within one locale, focusing on one query, no training for attention mechanism, and 20,000-item data set.

9.2 Key Takeaways

- Deep learning attains accuracy comparable to traditional methods while offering faster prediction.
- KNN strikes optimal balance: 17× faster than Cosine with 24% NDCG loss.
- Bernoulli RBM allows ultra-fast prediction (0.0042s) after one-time training.

- Attention mechanisms must be jointly optimized, not applied post-hoc.
- Cosine similarity is a solid baseline for simplicity and accuracy.

9.3 Future Research Directions

End-to-end attention training with ranking losses; multi-dataset validation across domains and scales; user study validation for quality and satisfaction; transformer-based encoders (BERT, GPT); hybrid content-collaborative approaches; real-time deployment with A/B testing; scalability via FAISS/ANNOY; and interpretability tools.

10. Declarations

Ethics and Consent to Participate: This article did not include any research involving people.
Competing Interest: All authors do not have any conflict of interest.
Funding: No funding was provided for this project.
Availability of data and materials: Amazon M2 dataset.
Acknowledgments: We thank Amazon and Kaggle for providing the M2 dataset. Special gratitude to our research advisors for valuable feedback throughout this project.

References

[1] Linden G, Smith B, York J. Amazon.com Recommendations: Item-To-Item Collaborative Filtering. IEEE Internet Comput. 2003;7:76-80.

[2] Pazzani MJ, Billsus D. Content-Based Recommendation Systems. In the Adaptive Web: Methods and Strategies of Web Personalization. Springer. 2007:325-341.

[3] Zhang S, Yao L, Sun A, Tay Y. Deep Learning Based Recommender System: A Survey and New Perspectives. ACM Comput Surv. 2019;52:1-38.

[4] Järvelin K, Kekäläinen J. Cumulated Gain-Based Evaluation of IR Techniques. ACM Trans Inf Syst. 2002;20:422-446.

[5] Desrosiers C, Karypis G. A Comprehensive Survey of Neighborhood-Based Recommendation Methods. Recommender systems handbook. 2010:107-144.

[6] Sedhain S, Menon AK, Sanner S, Xie L. AutoRec: Autoencoders Meet Collaborative Filtering. In Proceedings of the 24th International Conference on World Wide Web. 2015:111-112.

[7] Salakhutdinov R, Mnih A, Hinton G. Restricted Boltzmann Machines for Collaborative Filtering. In Proceedings of the 24th International Conference on Machine Learning. 2007:791-798.

[8] He X, Liao L, Zhang H, Nie L, Hu X, et al. Neural Collaborative Filtering. In Proceedings of the 26th International Conference on World Wide Web. 2017:173-182.

[9] Yannam VR, Kumar J, Vankayala T, Babu KS. Hybrid Approach for Next Basket Recommendation System. Int J Inf Technol. 2023;15:1733-1740.

[10] Hidasi B, Karatzoglou A, Baltrunas L, Tikk D. Session-Based Recommendations With Recurrent Neural Networks. 2015. arXiv preprint: https://arxiv.org/pdf/1511.06939v1.

[11] He X, Deng K, Wang X, Li Y, Zhang Y, et al. Light GCN: Simplifying and Powering Graph Convolution Network for Recommendation. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2020:639-648.

[12] Liu C, Lin J, Wang J, Liu H, Caverlee J. Mamba4rec: Mamba4rec: Towards Efficient Sequential Recommendation With Selective State Space Models. 2024. Arxiv preprint arxiv: https://arxiv.org/pdf/2403.03900.

[13] H. Zhang, X. Li, and M. Wang, "SS4Rec: Continuous-time sequential recommendation with state space models," in *Proc. IJCNLP*, 2025.

[14] Bahdanau D, Cho K, Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate. 2014. arXiv preprint: https://arxiv.org/pdf/1409.0473v1.

[15] Sarwar B, Karypis G, Konstan J, Riedl J. Item-Based Collaborative Filtering Recommendation Algorithms. In Proceedings of the 10th International Conference on World Wide Web. 2001;1: 285-295.

[16] Koren Y, Bell R, Volinsky C. Matrix Factorization Techniques for Recommender Systems. Computer. 2009;42:30-37.

[17] Chen C, Zhang M, Liu Y, Ma S. Neural Attentional Rating Regression With Review-Level Explanations. In Proceedings of the 2018 World Wide Web Conference. 2018:1583-1592.

[18] Sun F, Liu J, Wu J, Pei C, Lin X, et al. BERT4Rec: Sequential Recommendation With Bidirectional Encoder Representations From Transformer. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management. 2019:1441-1450.

[19] Y. M.Latha and B. S.Rao. Amazon Product Recommendation System Based on a Modified Convolutional Neural Network. ETRI Journal. 2024;46:633–647.

[20] Abdalla HB, Gheisari M, Awlla AH. Hybrid Self-Attention Bilstm and Incentive Learning-Based Collaborative Filtering for E-Commerce Recommendation Systems. Electron Commer Res .2025;25:4947–4970.

[21] Kang WC, McAuley J. Self-Attentive Sequential Recommendation. In 2018 IEEE International Conference on Data Mining (ICDM). IEEE. 2018:197-206.

[22] Herlocker JL, Konstan JA, Terveen LG, Riedl JT. Evaluating Collaborative Filtering Recommender Systems. ACM Trans Inf Syst. 2004;22:5-53.

[23] Alhijawi B, Fraihat S, Awajan AA. Multi-Factor Ranking Method for Trading-off Accuracy, Diversity, Novelty, and Coverage of Recommender Systems. Int J Inf Technol. 2023;15:1427-1433.