Advancing Hybrid Recommender Systems for E-Commerce with Knowledge-Enhanced Architecture and Adaptive User Engagement Modeling

by

Soumi Hazra

A thesis submitted to the University of Ottawa

in partial fulfillment of the requirements for the degree of

MASTER OF COMPUTER SCIENCE

School of Electrical Engineering and Computer Science

Faculty of Engineering

University of Ottawa

ABSTRACT

In the era of information overload, recommender systems have emerged as pivotal tools for helping users navigate vast product catalogs in e-commerce environments. This thesis presents a novel approach to hybrid recommender systems for e-commerce platforms by combining collaborative filtering with content-based methods enhanced by knowledge graph integration. The proposed system addresses key challenges in recommendation quality, diversity, and cold-start scenarios through adaptive user engagement modeling.

The first part of this work introduces a hybrid recommender system that integrates Alternating Least Squares (ALS) collaborative filtering with a two-tower content-based filtering model for e-commerce applications. This approach leverages user-item interactions while incorporating product description features to overcome data sparsity issues commonly found in traditional recommendation approaches. The system adaptively combines predictions from both models using a weighted approach, offering improved recommendation quality and diversity.

Building upon this foundation, the second part presents KAHRec: a Knowledge-Augmented Hybrid Recommender system that enhances the initial model through three key theoretical innovations. First, the system integrates BERT-driven semantic understanding to capture contextual relationships in product descriptions and user reviews, enabling more nuanced feature representations than traditional word embeddings. Second, it constructs structured knowledge graphs from over 148,000 product relationships (also_bought, also_viewed, similar_item) and applies PageRank and degree centrality measures to quantify product importance within the recommendation ecosystem. Third, it implements stratified user engagement modeling that categorizes users into distinct interaction tiers (One/Few/Medium/Many) and applies tier-specific adaptive fusion strategies, with weights ranging from collaborative-dominant (0.75 ALS, 0.25 Two-Tower) for highly engaged users to content-dominant (0.25 ALS, 0.75 Two-Tower) for cold-start scenarios. This theoretical framework enables the system to dynamically balance matrix factorization efficiency with

neural personalization based on data availability and user characteristics.

The knowledge-augmented approach incorporates synthetic item generation for cold-start mitigation that combines knowledge graph centrality with BERT feature similarity, providing comprehensive coverage across diverse user scenarios. The adaptive fusion mechanism employs consensus boosting that amplifies recommendations when both models agree, while applying strategic penalties for model disagreements, ensuring robust performance across varying data conditions.

Evaluated on a large-scale dataset containing over 400,000 user-item interactions across 30,445 products, the proposed approaches demonstrate significant improvements over baseline methods and state-of-the-art models in precision, recall, and user satisfaction metrics. The knowledge-augmented KAHRec approach achieves 6% higher precision and 17% higher recall compared to individual baselines, with a 25% relative improvement in cold-start precision scenarios. KAHRec demonstrates consistent superiority over state-of-the-art models, outperforming BERT4Rec and KGAT with 8.5% precision and 8.2% recall improvements respectively. Additional benefits include 3.6% greater recommendation diversity while maintaining robust error performance (MAE: 0.42).

The system shows particular efficacy for medium-engagement users, outperforming alternatives by more than 8% in precision metrics. By strategically balancing matrix factorization efficiency with neural personalization, this work advances adaptable recommendation frameworks for dynamic e-commerce environments requiring both accuracy and responsiveness to catalog evolution.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my supervisor, Dr. Thomas Tran, for his excellent guidance and continuous support throughout my graduate studies. His expertise in recommender systems and encouraging mentorship have made this research both challenging and rewarding.

I am also grateful to my family and friends for their unconditional love and support throughout this academic journey. Their encouragement has been a constant source of motivation, especially during the most demanding phases of this research.

This thesis would not have been possible without the collective support of all these individuals.

I also acknowledge the use of Perplexity AI for writing and editorial assistance, including grammar corrections, sentence paraphrasing, and improving the clarity of technical explanations. All content has been thoroughly reviewed, fact-checked, and verified by the author. The research methodology, experimental design, data analysis, and conclusions represent entirely my intellectual contribution and responsibility.

Table of Contents

List of Figures

List of Tables

LIST OF NOTATIONS

Notation	Description	Page(s)
R	User-item interaction matrix	26, 28
$R \in \mathbb{R}^{m \times n}$	User-item matrix with m users and n items	26
$U \in \mathbb{R}^{m \times k}$	User factor matrix in ALS	26, 28
$V \in \mathbb{R}^{n \times k}$	Item factor matrix in ALS	26, 28
r_{ui}	Rating of user u for item i	28, 28
$\hat{r}_{ui}$	Predicted rating of user u for item i	32, 32
r^*_{ui}	Knowledge- and sentiment-aware rating	59
$\hat{r}^{\text{final}}_{ui}$	Final hybrid prediction for user u and item i	77
m	Number of users	26
n	Number of items	26
k	Number of latent factors/embedding dimension	26, 32
λ	Regularization parameter	28, 28
α	Confidence scaling parameter	69
α_{T_u}	Adaptive fusion weight for user type T_u	77
u_u	Latent factor vector for user u	28
v_i	Latent factor vector for item i	28
T_U	User tower representation	32, 32
T_I	Item tower representation	32, 32
E_U	User embeddings	71, 32
E_I	Item embeddings	71, 32
$G = (V, E, R)$	Knowledge graph with vertices V, edges E, relations R	63

Notation	Description	Page(s)
PR(i)	PageRank centrality of item i	79, 68
deg(i)	Degree centrality of item i	79, 68
$\text{KG}_{\text{boost}_i}$	Knowledge graph boost for item i	79
t_{meta}	BERT input sequence for metadata	61
t_{review}	BERT input sequence for reviews	62
e_{text}	BERT-based embedding	28
s_p	Aggregated sentiment score for product p	28
Precision@k	Fraction of relevant items in top-k recommendations	41
Recall@k	Fraction of relevant items retrieved in top-k	42
NDCG@k	Normalized Discounted Cumulative Gain at k	43, 88
MAE	Mean Absolute Error	43
RMSE	Root Mean Square Error	43
DCG@k	Discounted Cumulative Gain at k	43
IDCG@k	Ideal Discounted Cumulative Gain at k	43
T_u	User engagement tier (ONE/FEW/MEDIUM/MANY)	77, 77
$\lvert\text{interactions}(u)\rvert$	Number of interactions for user u	77
combined_preds_i	Combined prediction score for item i	36
x_{norm_i}	Normalized ALS prediction for item i	36
y_{norm_i}	Normalized Two-Tower prediction for item i	36
als_f1	F1 score of ALS model	36
tt_f1	F1 score of Two-Tower model	36
agreement_{ui}	Agreement between ALS and Two-Tower predictions	77
boost_{ui}	Consensus-based scaling factor	79
s_{ui}^{ALS}	ALS prediction score for user u, item i	79
s_{ui}^{TT}	Two-Tower prediction score for user u, item i	79

Notation	Description	Page(s)
Rec_{cold}	Cold-start recommendation scores	74
V_{pop}	Popularity-based recommendation vector	74
V_{synth}	Synthetic recommendation vector	74
PopScore_i	Bayesian popularity score for item i	74
ColdScore_i	Final cold-start score for item i	74
$\text{SentimentAlign}_{ui}$	Sentiment alignment score	59, 62
BERTSentiment	BERT-based sentiment score	62
UserRating	Explicit user rating	62
μ	Mean value	68
σ	Standard deviation	68
μ_r	Mean of actual ratings	42
μ_p	Mean of prediction scores	41
ϵ	Tolerance value	42
$\mathcal{I}$	Set of all items	80
$\mathcal{I}_u$	Set of items interacted with by user u	80
$\mathcal{D}$	Training dataset	32
N	Number of training samples	32, 43
$\text{Concat}(\cdot)$	Concatenation operation	28
$\text{TopK}(\cdot)$	Top-K selection function	80
$f(\cdot)$	Two-tower model transformation	32
$z(\cdot)$	Z-score normalization	59
MaxItems/Category	Maximum items per category constraint	80
Diversity	Category-based diversity measure	89

CHAPTER 1

INTRODUCTION

1.1 Overview

In the digital age, the expansion of online platforms has led to an unprecedented growth of consumer choices [11]. This abundance, while beneficial, often results in the paradox of choice, where the volume of options can overwhelm and hinder decision-making. Recommender systems have emerged as a pivotal solution to this challenge, particularly in e-commerce [41, 48], by personalizing user experiences and guiding consumers toward choices that align with their preferences and behaviors.

The significance of recommender systems in e-commerce is substantial, with approximately 35% of all purchases on Amazon and 75% of Netflix views being influenced by algorithm-based recommendations [46]. These systems not only enhance user experience but also drive business growth by converting casual browsers into purchasing users, boosting cross-selling opportunities, fostering customer loyalty, and improving customer retention rates.

Traditional recommender systems primarily employ three approaches: collaborative filtering (CF), content-based filtering (CBF), and knowledge-based recommenders (KBR). Collaborative filtering capitalizes on user-item interactions to identify patterns [31] and similarities among users or items [37], generating recommendations based on collective preferences [41, 32]. Content-based filtering focuses on item characteristics, recommending items similar to those a user has previously

shown interest in, leveraging detailed item metadata [42, 15]. Knowledge-based recommenders utilize semantic user preference knowledge, item knowledge, and recommendation knowledge to identify relevant items, particularly valuable for complex and high-involvement products (products requiring significant consideration and research before purchase, such as electronics or appliances) [47].

The evolution of recommender systems has progressed through several distinct phases, each addressing specific limitations of previous approaches. Early collaborative filtering systems leveraged user-item interaction patterns but struggled with data sparsity and the cold-start problem. Content-based approaches improved upon this by utilizing item metadata but often resulted in over-specialization. The current hybrid paradigm integrates multiple techniques to overcome these limitations, combining collaborative signals with content-based features while increasingly incorporating knowledge graphs and deep learning architectures. This evolution reflects the growing complexity of e-commerce environments and the need for more sophisticated recommendation frameworks capable of handling diverse data sources and user behaviors.

1.2 Motivations

The emergence and prevalence of the Internet have led to the explosive growth of information which helps the world change from the era of information scarcity to the age of information overload. In fact, both information consumers and information producers are facing significant challenges: for consumers, although information overload meets the user's demand for knowledge, it makes the consumers face the problem of how to choose meaningful information when confronted with a large amount of data; for producers, it is hard to make the information or products famous and draw users' attention. We need a tool to help us survive these challenges.

Despite their merits, traditional approaches encounter distinct challenges. Collaborative filter-

ing struggles with the cold-start problem, finding it difficult to recommend new users or items that lack substantial interaction data [4, 28, 33]. Traditional methods of recommender systems do not offer optimized solutions to this problem. Content-based filtering's reliance on item metadata can lead to a narrow set of recommendations [18], potentially limiting the diversity of user discovery [15] and often struggles with over-specialization and limited recommendation diversity.

The motivations for this research stem from three key areas of concern in current recommender system implementations. First, technical limitations in existing approaches constrain their effectiveness in real-world scenarios. Second, recent advances in deep learning and natural language processing offer new opportunities for enhancement. Third, the integration challenges of combining multiple recommendation paradigms require systematic solutions.

1.2.1 Technical Foundations and Limitations

Matrix factorization methods, particularly Alternating Least Squares (ALS), remain fundamental to modern recommender systems [17]. These methods effectively decompose user-item interactions into low-dimensional latent factors, enabling more accurate and personalized recommendations while addressing data sparsity issues [40, 32]. However, pure collaborative filtering approaches struggle with cold-start problems, suggesting the need for hybrid approaches that integrate content-based strategies with matrix factorization.

Content-based filtering, while mitigating some cold-start challenges through item metadata analysis, introduces its own limitations. By focusing exclusively on item characteristics and recommending items similar to those a user has previously preferred, CBF systems often produce recommendations with limited diversity, potentially creating "filter bubbles" that restrict user discovery [2]. This approach also depends heavily on the quality and completeness of item metadata, which may be inconsistent or incomplete in real-world e-commerce environments [42].

1.2.2 Advances in Deep Learning and NLP

Recent advances in deep learning and natural language processing have revolutionized recommender systems by enabling more sophisticated analysis of textual content. Transformer models like BERT outperform traditional approaches in classifying product reviews, achieving significant accuracy improvements [22]. The integration of sentiment analysis into recommender systems represents another promising direction [1, 39], as written customer reviews are a rich source of information that can offer insights into the recommender system.

Deep neural networks, such as the two-tower model, have gained traction for their ability to learn complex user and item representations, enabling more nuanced and scalable recommendations. These models are especially effective in large-scale e-commerce environments, where the diversity and volume of data demand sophisticated modeling capabilities. The two-tower architecture separately processes user and item features, projecting them into a shared embedding space where their interactions can be efficiently modeled.

1.3 Problem Statement

Despite significant advances in recommender systems, several critical challenges remain unaddressed in modern e-commerce environments. Traditional single-paradigm approaches often underutilize subtle user engagement patterns and rich product relationships available in contemporary platforms. Most notably, existing hybrid systems fail to fully leverage rich product metadata, graph-structured relationships, and advanced textual representations within unified frameworks that could significantly enhance recommendation quality and diversity.

Knowledge Integration Challenges: The limitations of single-paradigm approaches have mo-

tivated hybrid recommender system development, yet even advanced hybrid systems face challenges in fully leveraging semantic knowledge structures. Traditional approaches often underutilize information sources including structured knowledge graphs that capture semantic relationships between products, resulting in suboptimal recommendations that fail to capture the full complexity of user preferences and item relationships. While content-based recommender systems effectively address cold-start problems [52], and transformer models demonstrate power for text representation, few studies have combined these approaches into end-to-end models that leverage both structured knowledge and deep textual understanding.

Technical Integration Gaps: The integration of knowledge bases with neural architectures and transformer-based text representations represents a particularly promising yet underexplored direction. Current hybrid systems often employ simplistic fusion strategies that fail to adapt to user characteristics, item properties, and contextual factors dynamically. The lack of sophisticated user engagement modeling means that systems cannot optimize recommendation strategies for users with varying interaction patterns, from cold-start scenarios to highly engaged users with rich interaction histories.

Research Objectives: This research addresses these interconnected challenges by developing a robust recommendation framework that effectively leverages both explicit and implicit feedback while incorporating knowledge graph structures to enhance recommendation quality and diversity. The proposed approach integrates ALS-based collaborative filtering with a Two-Tower neural architecture enhanced by BERT-based text embeddings and structured knowledge graphs, addressing the limitations of individual methods while providing robust recommendations across diverse scenarios, including cold-start situations. The framework specifically tackles the need for adaptive user engagement modeling that recognizes different recommendation strategies are optimal for users with varying interaction patterns.

1.4 Research Contributions

This thesis makes the following contributions to the field of recommender systems across three main areas: novel architectural innovations, methodological advances, and empirical validation achievements. Each contribution addresses specific limitations identified in existing approaches while advancing the state-of-the-art in hybrid recommender systems.

- **Novel Hybrid Architecture**: A novel hybrid recommendation architecture that integrates Alternating Least Squares (ALS) collaborative filtering with a Two-Tower content-based filtering model, leveraging the strengths of both approaches [13]. This architecture represents a significant advancement over existing approaches by combining the scalability of matrix factorization methods with the representational power of deep learning and the semantic richness of knowledge graphs.

- **Knowledge-Augmented Enhancement**: A knowledge-augmented hybrid recommender system (KAHRec) that enhances the initial model with BERT-driven knowledge graph integration, adaptive weighting based on user interaction tiers, and synthetic item generation for cold-start mitigation. The integration of over 148,000 structured product relationships augments the feature space for user and item representations, incorporating BERT-based sentiment analysis and knowledge graph centrality features.

- **Adaptive Fusion Methodology**: An adaptive method for combining predictions from both models that implements stratified user engagement modeling, categorizing users into distinct tiers (One/Few/Medium/Many) and enabling personalized recommendation strategies tailored to different engagement levels. This approach develops synthetic item generation techniques for cold-start mitigation that combine knowledge graph centrality with BERT

feature similarity, providing a comprehensive solution to one of the most persistent challenges in recommender systems.

- **Comprehensive Evaluation Framework**: A comprehensive evaluation framework that demonstrates improved performance over baseline models through extensive experimentation on large-scale Amazon datasets, showing significant improvements in precision, recall, diversity, and cold-start scenarios. This empirical contribution improves how users discover and interact with products online by offering a methodologically sound approach to combining collaborative and content-based filtering techniques [48, 38], specifically designed for e-commerce platforms utilizing user-item interactions and comprehensive product metadata.

These contributions collectively address the limitations of existing single-paradigm and simple hybrid approaches while providing practical solutions for real-world e-commerce recommendation challenges. The integrated framework demonstrates that sophisticated knowledge augmentation and adaptive user modeling can significantly enhance recommendation quality across diverse user engagement levels.

1.5 Publications

The research presented in this thesis has resulted in the following publications:

- **Hazra, S., Tran, T.** (2025). "Hybrid Recommender System for E-commerce: Combining Collaborative and Content-Based Filtering." *Submitted to PeerJ Computer Science*. [Under Review]

- **Hazra, S., Tran, T.** (2025). "KAHRec: Knowledge-Augmented Hybrid Recommendations for E-Commerce with Adaptive User Engagement Modeling." *Submitted to ACM Transactions on Recommender Systems (TORS)*. [Under Review]

These publications represent the core contributions of this thesis work, with the first paper focusing on the foundational hybrid architecture combining ALS collaborative filtering with Two-Tower content-based filtering, and the second paper presenting the advanced knowledge-augmented approach with BERT integration and adaptive user engagement modeling.

1.6 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 provides background information and a comprehensive literature review on recommender systems, focusing on collaborative filtering, content-based filtering, hybrid approaches, and knowledge-enhanced methods. Chapter 3 presents the development and evaluation of a foundational hybrid recommender system that integrates Alternating Least Squares collaborative filtering with Two-Tower content-based filtering for e-commerce applications. Chapter 4 introduces KAHRec, a Knowledge-Augmented Hybrid Recommender system that extends the foundational approach through sophisticated knowledge graph integration, BERT-based textual understanding, and adaptive user engagement modeling, including comprehensive experimental evaluation and performance analysis comparing the proposed approach against state-of-the-art baselines. Finally, Chapter 5 concludes the thesis with a summary of contributions, discussion of limitations and future research directions, and broader implications of the knowledge-augmented hybrid approach for modern e-commerce recommendation challenges.

The thesis organization follows a logical progression from theoretical foundations through

practical implementation to empirical validation. Each chapter builds upon previous concepts while introducing new elements essential for understanding the complete hybrid recommender system approach.

CHAPTER 2

BACKGROUND INFORMATION, LITERATURE REVIEW, AND RELATED WORK

2.1 Introduction

This chapter provides a comprehensive foundation for understanding hybrid recommender systems in e-commerce environments, with particular emphasis on knowledge-enhanced architectures and adaptive user engagement modeling. The exponential growth of e-commerce platforms has intensified the need for sophisticated recommendation frameworks capable of handling diverse data sources while addressing persistent challenges such as data sparsity, cold-start scenarios, and recommendation diversity.

Recommender systems have become essential tools for navigating large product catalogs, with approximately 35% of all purchases on Amazon and 75% of Netflix views being influenced by algorithm-based recommendations [46]. The traditional paradigms of recommender systems—collaborative filtering, content-based filtering, and knowledge-based approaches—each bring unique strengths but also face significant limitations when implemented independently. The evolution from single-paradigm approaches to hybrid systems represents a fundamental shift in how we conceptualize and implement personalized recommendation engines.

This chapter is organized into four main sections that build systematically toward understand-

ing the research context and contributions of this thesis. First, the Background Information section (Section 2.2) provides foundational knowledge covering core technologies essential for understanding modern recommender systems. Second, the Literature Review section (Section 2.3) synthesizes broader field research and identifies key limitations that motivate hybrid approaches. Third, the Related Work section (Section 2.4) provides detailed analysis of specific research directly relevant to this thesis. Finally, the chapter concludes by establishing the research gaps that this thesis addresses and the contributions it makes to the field.

2.2 Background Information

This section provides foundational knowledge essential for understanding the hybrid recommender system approaches presented in this thesis. The discussion covers four key technological areas that form the theoretical backbone of the proposed methodologies: deep neural networks and their application in recommender systems, BERT and transformer models for textual understanding, knowledge graphs and graph analysis techniques, and matrix factorization approaches for collaborative filtering. Each subsection builds upon the previous concepts to establish a comprehensive understanding of the technologies that enable advanced hybrid recommendation architectures.

2.2.1 Deep Neural Networks in Recommender Systems

Deep neural networks have revolutionized recommender systems by enabling sophisticated analysis of user behavior and item characteristics. Unlike traditional methods that rely on linear combinations of features, neural networks can capture complex, non-linear relationships between users

and items through multiple layers of interconnected neurons. This capability is particularly valuable in e-commerce environments where user preferences and item relationships exhibit intricate patterns.

The Two-Tower architecture has emerged as a particularly influential framework, employing separate neural networks to process user and item features before computing interaction scores [25]. This architecture enables specialized processing of different feature types while maintaining computational efficiency suitable for real-time recommendation scenarios. The user tower processes features including historical interaction patterns and demographic information, while the item tower handles product metadata including textual descriptions and categorical features.

Training neural recommender systems involves several key considerations including loss function selection, optimization algorithms, and regularization techniques. Mean squared error is commonly used for rating prediction tasks, while the Adam optimizer provides adaptive learning rates well-suited to varying sparsity patterns in recommendation data. Regularization techniques such as dropout and L2 penalty help prevent overfitting in sparse data scenarios.

2.2.2 BERT and Transformer Models for Text Understanding

Transformer models, particularly BERT (Bidirectional Encoder Representations from Transformers), have revolutionized natural language processing in recommender systems [24]. BERT's bidirectional nature enables it to consider both preceding and following context when generating word representations, leading to more nuanced understanding of text semantics. This capability is particularly valuable in e-commerce environments where product descriptions and user-generated content contain rich semantic information.

BERT-based models achieve significant improvements in product review classification and sen-

timent analysis compared to traditional word embedding approaches [22]. These improvements translate directly to enhanced recommendation quality, as more accurate understanding of textual content leads to better user and item representations. The integration of BERT embeddings into recommender systems enables sophisticated processing of product descriptions, user reviews, and other textual content.

BERT4Rec represents a significant advancement in applying transformer architectures to sequential recommendation tasks [43]. This model adapts BERT's masked language modeling approach to recommendation by masking items in user interaction sequences and predicting them based on surrounding context.

2.2.3 Knowledge Graphs and Graph Analysis

Knowledge graphs provide structured representations of relationships between entities, enabling recommender systems to leverage semantic connections beyond simple user-item interactions [50]. In e-commerce environments, knowledge graphs capture product relationships, category hierarchies, brand associations, and co-purchase patterns that inform recommendation quality and diversity.

Graph analysis techniques provide valuable insights into the structure and properties of knowledge graphs. Centrality measures such as PageRank and degree centrality identify important nodes within the graph [29]. PageRank provides a recursive measure of node importance based on the global link structure, while degree centrality quantifies direct connectivity. These measures can be incorporated as features in recommendation algorithms, providing additional signals about entity importance and influence.

Knowledge Graph Attention Networks (KGAT) represent a significant advancement in apply-

ing attention mechanisms to knowledge-enhanced recommendation [50]. KGAT employs attention mechanisms to learn the importance of different relationships when computing node representations, enabling models to capture complex, multi-order relationships between entities.

The construction of knowledge graphs typically involves extracting relationships from transactional data, product metadata, and user interaction patterns. Co-purchase relationships, categorical relationships, and similarity connections form the foundation of e-commerce knowledge graphs that enhance recommendation systems.

2.2.4 Matrix Factorization and Collaborative Filtering

Collaborative filtering operates on the principle that users who agreed in the past will agree in the future [37]. This approach leverages user-item interaction patterns to identify similarities among users or items, generating recommendations based on collective preferences. Collaborative filtering can be divided into memory-based methods and model-based methods that employ machine learning algorithms.

Matrix factorization, particularly Alternating Least Squares (ALS), has emerged as a highly effective model-based collaborative filtering technique [17]. ALS decomposes user-item interaction matrices into low-dimensional latent factors, enabling accurate predictions while addressing data sparsity issues. The mathematical foundation involves minimizing a regularized loss function that balances prediction accuracy with model complexity.

The ALS algorithm iteratively optimizes user and item factor matrices by alternating between fixing one set of factors while optimizing the other. This approach has proven particularly effective in large-scale e-commerce environments where traditional collaborative filtering approaches struggle with computational complexity and data sparsity.

2.2.5 Content-Based and Hybrid Filtering Approaches

Content-based filtering focuses on item characteristics rather than user behavior patterns [34]. This approach recommends items similar to those a user has previously preferred by analyzing detailed item metadata, descriptions, and features. Content-based approaches can provide personalized recommendations even for new items, as long as sufficient item metadata is available.

Hybrid recommender systems combine multiple recommendation techniques to leverage their complementary strengths while mitigating individual weaknesses [7]. The integration of collaborative filtering with content-based filtering addresses limitations such as data sparsity, cold-start problems, and recommendation diversity challenges.

Knowledge-based recommender systems utilize structured domain knowledge about items, users, and their relationships to generate recommendations. Unlike collaborative or content-based methods, knowledge-based systems do not depend on extensive user interaction histories, making them valuable for addressing cold-start scenarios and providing recommendations for complex products.

2.3 Literature Review

The development of recommender systems has evolved significantly in response to information overload in modern e-commerce environments. This evolution has been marked by a shift from standalone approaches to advanced hybrid systems that integrate multiple recommendation methods [5]. The field has undergone major changes, moving from traditional collaborative filtering and content-based methods toward hybridization, deep learning, and knowledge graph integration, all aimed at overcoming the limitations of single-approach systems.

Collaborative filtering techniques use user-item interaction data to predict user preferences by identifying patterns and similarities among users or items [32]. Matrix factorization methods, particularly Alternating Least Squares (ALS), have become fundamental approaches in modern recommender systems, effectively addressing data sparsity by breaking down user-item interactions into low-dimensional representations [54][27]. These techniques provide more accurate and personalized recommendations by capturing underlying preference patterns that may not be obvious in surface-level interactions [17]. However, collaborative filtering approaches face significant challenges, most notably the cold-start problem, where systems struggle to provide accurate recommendations for new users or items with limited interaction data [28][33][4].

Content-based filtering approaches address some limitations of collaborative filtering by using item features to recommend items similar to those a user has previously liked [34][15]. While content-based filtering can provide personalized recommendations even for new items, it often leads to focusing too narrowly on similar items and limited recommendation diversity, potentially creating filter bubbles that restrict user discovery [2][18]. The reliance on item metadata quality also presents challenges, as incomplete or inconsistent metadata can significantly impact recommendation performance [42].

Recent advances in deep learning and natural language processing have transformed content-based filtering abilities. Transformer models, particularly BERT (Bidirectional Encoder Representations from Transformers), have shown better performance compared to traditional neural networks for review-based recommenders [24][22]. These models generate contextual representations of reviews and product texts, enabling more detailed user and item modeling that captures semantic relationships beyond simple keyword matching [36][16]. The Two-Tower architecture, a deep learning framework, has proven particularly influential in enhancing content-based filtering by extracting rich representations from item metadata through separate neural networks for user and item processing [25][44].

Sentiment analysis has become an increasingly important component in modern recommender

systems, with research showing that combining explicit ratings with review sentiment significantly improves hybrid model performance [12][1][39]. BERT-based sentiment analysis models can extract detailed emotional signals from user reviews and product descriptions, providing additional information that enhances understanding of user preferences beyond numerical ratings alone [22]. This integration of sentiment features represents a significant advancement in capturing the full spectrum of user opinions and preferences.

Knowledge-based recommender systems complement collaborative and content-based approaches by directly using domain knowledge about items, users, and their relationships [47][6]. These systems reduce the cold-start problem found in collaborative filtering by using structured knowledge and semantic relationships to generate recommendations even without extensive interaction histories [14][45]. Knowledge graphs have become particularly powerful representations for capturing complex relationships between entities in recommender systems, enabling reasoning over multi-step connections and discovery of hidden relationships [50][29][55].

The integration of knowledge graphs with neural architectures represents a significant advancement in hybrid recommender systems. Knowledge Graph Attention Networks (KGAT) use attention mechanisms to learn the importance of different relationships when computing node representations, enabling models to capture complex, multi-order relationships between entities [50]. Graph neural networks provide frameworks for directly processing graph-structured data, learning representations that capture both node features and structural relationships through message passing and attention mechanisms [29][21].

Hybrid recommender systems have become effective solutions to overcome the limitations of individual approaches by combining the strengths of collaborative filtering, content-based filtering, and knowledge-based methods [7][8]. Recent research has shown that hybrid approaches consistently outperform individual techniques across various evaluation metrics, particularly in addressing data sparsity and cold-start scenarios [26][38][13]. Advanced hybrid systems use sophisticated combination methods including adaptive weighting mechanisms that adjust the contribution of dif-

ferent components based on user characteristics, item properties, and contextual factors [49][51].

User engagement stratification has gained recognition as an effective strategy for personalizing hybrid recommendation approaches. Research has shown that users with different interaction patterns benefit from customized recommendation strategies, with cold-start users requiring more emphasis on content-based and knowledge-based components, while high-engagement users benefit from advanced collaborative filtering approaches [33][4][31]. The Sparsity and Cold Start Aware Hybrid Recommender System (SCSHRS) shows this approach by using different strategies based on user engagement levels to reduce data sparsity and cold-start problems [33].

Despite significant advances in hybrid recommender systems, several research gaps remain that limit their effectiveness in real-world e-commerce environments. Most existing systems don't fully use the rich product metadata, graph-structured relationships, and advanced textual representations available in modern e-commerce platforms [52][19]. Few studies have successfully unified knowledge graph reasoning, deep textual modeling, and engagement-aware model adaptation into a single, end-to-end approach [5][16]. The integration of knowledge bases with neural architectures and transformer-based text representations represents a particularly promising direction that has received limited complete study [9].

The evaluation of hybrid recommender systems requires comprehensive frameworks that consider multiple objectives including accuracy, diversity, novelty, and fairness [3][23]. Current evaluation protocols often focus on short-term accuracy metrics while neglecting diversity and user satisfaction considerations that are crucial for real-world deployment [35].

Recent surveys highlight the need for multimodal fusion and dynamically adaptive systems that can handle the complexity and scale of modern e-commerce environments [5]. The integration of large language models and transformer architectures is enabling more advanced understanding of textual content and user preferences expressed in natural language [36]. Privacy-preserving recommendation techniques and federated learning approaches are becoming increasingly important as data protection regulations become more stringent [12]. Developing comprehensive evaluation

methods that consider all aspects of recommendation quality remains an ongoing research challenge.

2.4 Related Work

This section provides detailed analysis of the specific state-of-the-art models that serve as comparison baselines in the experimental evaluation. The examination focuses on identifying key limitations of existing approaches that motivate the development of the proposed knowledge-augmented hybrid recommender system. The discussion is organized around the primary algorithmic approaches that form the foundation for comparative analysis and system positioning.

2.4.1 Introduction

This section examines certain baseline models, including specific state-of-the-art models that serve as comparison baselines in our experimental evaluation. We focus on identifying the key limitations of these approaches that motivate the development of our proposed knowledge-augmented hybrid recommender system. The discussion centers on four primary comparison models: ALS matrix factorization, Two-Tower neural architectures, Knowledge Graph Attention Networks (KGAT), and BERT4Rec sequential recommendation.

2.4.2 ALS Matrix Factorization Baseline

Alternating Least Squares (ALS) matrix factorization serves as our primary collaborative filtering baseline [17]. ALS decomposes user-item interaction matrices into low-dimensional latent factors through regularized optimization, making it suitable for large-scale e-commerce applications [54]. Recent implementations achieve strong performance on sparse datasets by learning latent representations that capture underlying user preferences [27].

However, ALS exhibits critical limitations that our approach addresses. **Data sparsity and cold-start problems** represent persistent challenges, as ALS requires sufficient interaction history to learn meaningful latent factors [33]. **Limited semantic understanding** constrains ALS to purely collaborative signals, missing opportunities to leverage rich product metadata and textual descriptions [54]. Additionally, linear assumptions prevent ALS from capturing complex, non-linear relationships between users and items, while textual information such as product descriptions remains underutilized [17].

2.4.3 Two-Tower Neural Architecture Baseline

The Two-Tower architecture serves as our content-based filtering baseline, employing separate neural networks to process user and item features [25]. This architecture enables sophisticated feature learning while maintaining computational efficiency for real-time scenarios [44]. The separation of user and item towers allows specialized processing of heterogeneous data types including textual descriptions and categorical attributes.

Despite these advantages, Two-Tower models exhibit significant limitations. **Over-specialization and filter bubbles** represent major concerns, as the approach tends to recommend items very

similar to previous encounters, limiting user discovery [25]. **Dependency on metadata quality** constrains effectiveness when item descriptions are incomplete [44]. Furthermore, limited collaborative signals mean Two-Tower models cannot leverage collective user behavior patterns, while semantic understanding remains constrained to surface-level feature matching rather than deep textual comprehension.

2.4.4 KGAT: Knowledge Graph Attention Networks

KGAT represents the current state-of-the-art in knowledge-enhanced recommendation, serving as our primary knowledge-based comparison baseline [50]. KGAT employs attention mechanisms to learn relationship importance when computing node representations, enabling capture of complex, multi-order relationships between entities. The approach leverages graph neural networks to propagate information through knowledge graphs, enriching item representations with semantic context.

However, KGAT faces several critical limitations that our work addresses. **Scalability challenges** arise as knowledge graphs grow large, with message passing algorithms exhibiting computational complexity that scales poorly with graph size [50]. **Static knowledge representation** limits the approach to pre-constructed graphs that cannot adapt quickly to changing product catalogs [29]. Additionally, limited integration with textual understanding constrains KGAT to structured relationships while under-utilizing rich textual content in product descriptions and reviews [55]. **Hybrid integration challenges** affect KGAT's ability to develop comprehensive systems that balance multiple recommendation paradigms.

2.4.5 BERT4Rec: Sequential Transformer-Based Recommendation

BERT4Rec serves as our transformer-based comparison baseline, representing a significant advancement in applying transformer architectures to sequential recommendation [43]. This model adapts BERT's masked language modeling approach by masking items in user interaction sequences and predicting them based on surrounding context. The bidirectional nature enables capture of complex sequential patterns that traditional recurrent models miss.

Despite these advances, BERT4Rec exhibits significant limitations. **Sequential focus limitations** constrain the model to temporal patterns while missing opportunities to leverage rich product content and relationships [43]. **Limited collaborative integration** affects most transformer-based approaches, which operate independently rather than combining effectively with collaborative filtering signals. Computational overhead represents a major concern, requiring substantial resources for training and inference [24]. Cold-start limitations persist when users lack sufficient interaction history, while isolation from structured knowledge limits processing to textual content without leveraging relationships and domain knowledge.

2.4.6 Research Positioning and Contribution

The analysis of these state-of-the-art approaches reveals critical gaps that limit their effectiveness in real-world e-commerce environments. ALS approaches lack semantic understanding, Two-Tower models suffer from over-specialization, KGAT faces scalability and integration challenges, and BERT4Rec operates in isolation from collaborative and knowledge-based signals. Most importantly, no existing approach successfully unifies knowledge graph reasoning, deep textual modeling, and engagement-aware model adaptation into a single, end-to-end framework.

This research addresses these identified limitations by proposing a comprehensive knowledge-augmented hybrid recommender system that integrates ALS-based collaborative filtering with a Two-Tower neural architecture, enhanced through BERT-driven knowledge graph integration and adaptive user engagement modeling. The proposed approach advances beyond existing models by optimally fusing collaborative, content-based, and knowledge-based reasoning within a unified framework designed specifically for e-commerce applications, thus overcoming the known weaknesses of earlier systems while achieving robust, diverse, and interpretable recommendations across diverse user engagement levels.

CHAPTER 3

HYBRID RECOMMENDER SYSTEM FOR E-COMMERCE: COMBINING COLLABORATIVE AND CONTENT-BASED FILTERING

3.1 Introduction

This chapter presents the development and evaluation of a foundational hybrid recommender system that integrates Alternating Least Squares (ALS) collaborative filtering with a Two-Tower content-based filtering model for e-commerce applications. The research addresses fundamental challenges in recommender systems including data sparsity, cold-start problems, and the need for improved recommendation quality and diversity in dynamic e-commerce environments.

Traditional recommender systems typically employ single-paradigm approaches that, while effective in specific scenarios, face significant limitations when deployed independently. Collaborative filtering excels at leveraging collective user behavior patterns but struggles with data sparsity and cold-start scenarios. Content-based filtering effectively utilizes item metadata but often leads to over-specialization and limited recommendation diversity. The hybrid approach proposed in this chapter combines the complementary strengths of both paradigms while mitigating their individual weaknesses through intelligent fusion strategies.

The chapter makes three primary contributions to the field of hybrid recommender systems.

First, it demonstrates the effectiveness of combining matrix factorization-based collaborative filtering with neural content-based filtering in e-commerce environments. Second, it introduces an adaptive weighting mechanism that dynamically adjusts the contribution of each component based on their relative performance. Third, it provides comprehensive experimental validation showing significant improvements over individual baseline methods across multiple evaluation metrics.

The methodology employs a systematic approach to hybrid system development, beginning with independent optimization of ALS collaborative filtering and Two-Tower content-based filtering components, followed by adaptive fusion strategy development and comprehensive evaluation. The experimental framework validates the approach using *Amazon e-commerce data*, demonstrating practical applicability for real-world deployment scenarios.

The remainder of this chapter is organized as follows. Section 3.2 presents the theoretical foundation and architectural design of the hybrid recommender system. Section 3.3 details the experimental evaluation methodology, including dataset characteristics, baseline comparisons, and performance metrics. Section 3.4 presents and analyzes the experimental results, demonstrating the effectiveness of the hybrid approach. Section 3.5 discusses the implications, limitations, and practical considerations of the proposed system. Finally, Section 3.6 provides a summary of the chapter's contributions and establishes the foundation for the advanced model presented in Chapter 4.

3.2 Methodology

The methodological discussion proceeds from a high-level description of the experimental pipeline to detailed component analyses. Sub-section 3.2.1, therefore, outlines the full workflow, after which individual models, fusion logic, and evaluation protocols are examined in depth.

3.2.1 Overview of Methodology

Developing this hybrid recommender system for an e-commerce platform involves a comprehensive methodology that leverages collaborative filtering (CF) and content-based filtering (CBF) techniques to generate personalized item recommendations. The process encompasses several key stages including data preprocessing, model training, placeholder value assignment, prediction and recommendation generation, top-k recommendations, evaluation metrics, and results visualization.

The proposed system leverages collaborative filtering and content-based filtering techniques to generate personalized item recommendations through a six-stage process: data preprocessing with 80-20 user-level train-test split, model training for both ALS and Two-Tower architectures, placeholder value assignment using cosine similarity-based approaches, prediction and recommendation generation through weighted fusion, top-5 recommendation selection, and comprehensive evaluation using multiple metrics, explained in Section 3.3.5

3.2.2 Alternating Least Squares Collaborative Filtering

The Alternating Least Squares (ALS) machine learning model is designed to factorize the sparse user-item interaction matrix $R \in \mathbb{R}^{m \times n}$ into lower-dimensional matrices: user matrix $U \in \mathbb{R}^{m \times k}$ and item matrix $V \in \mathbb{R}^{n \times k}$, where m is the number of users, n is the number of items, and k is the number of latent factors, also known as the "rank" of the factorization.

The ALS decomposition process is illustrated in Figure 3.1, which shows how the user-item interaction matrix is factorized into latent factor representations.

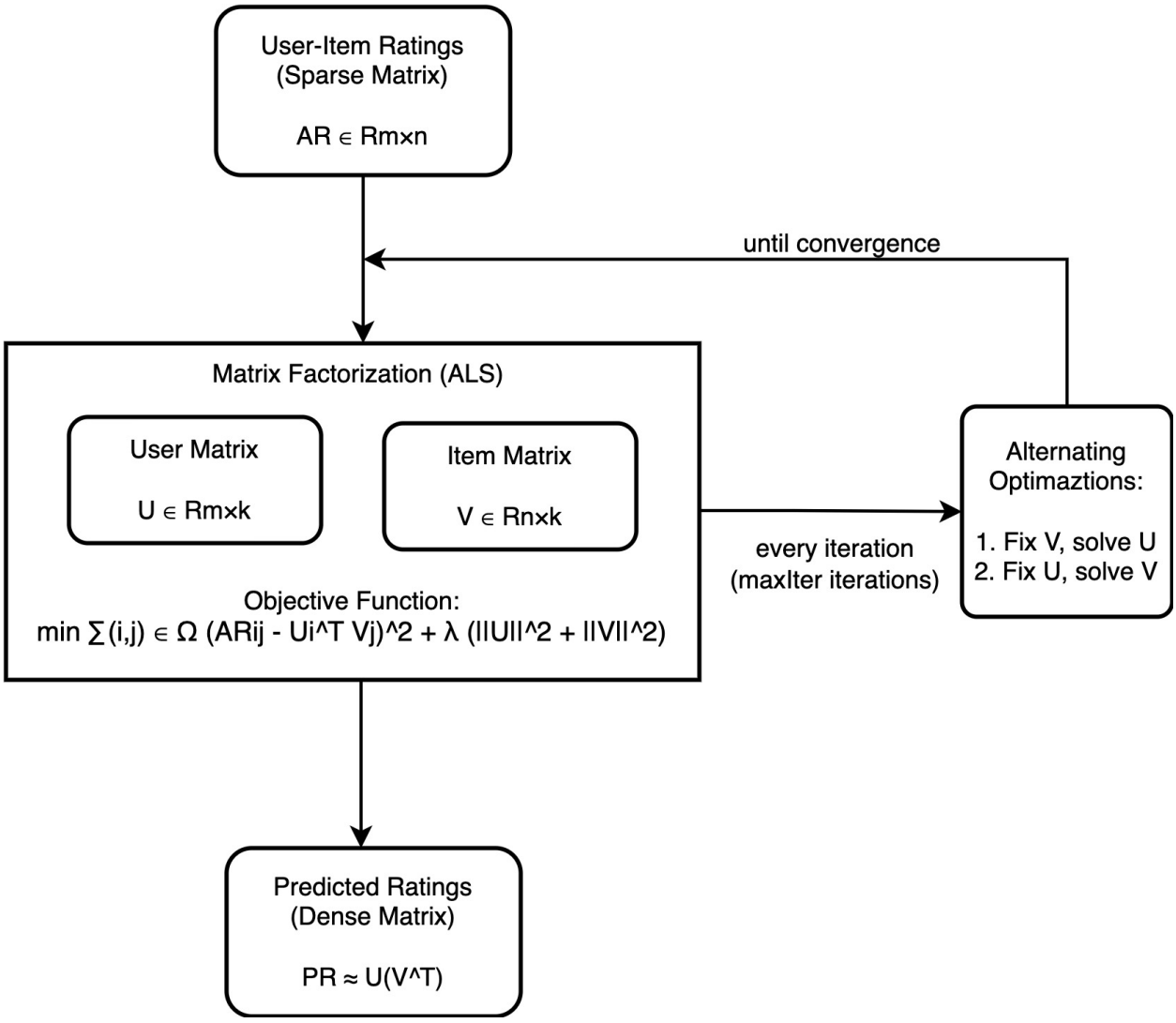

Figure 3.1: Working of ALS Model

Figure 3.1 illustrates the standard ALS matrix factorization process. While ALS itself is established, our contribution lies in its integration within a novel hybrid architecture that combines collaborative filtering with Two-Tower content-based filtering through adaptive weighting, enhanced preprocessing with cosine similarity-based placeholder assignment, and systematic hyperparameter optimization for e-commerce datasets.

The optimization process is governed by the following objective function[20]. This objective minimizes the regularized squared error over observed interactions, balancing fit and complexity:

$$\min_{u,i} \sum_{(u,i)\in R} (r_{ui} - u^T i)^2 + \lambda(\|u\|^2 + \|i\|^2) \tag{3.1}$$

where r_{ui} is the observed rating of user u for item i, u and i are the latent factor vectors for user u and item i respectively (each of length k, as determined by the rank parameter), R is the set of all user-item pairs with observed ratings, and λ is the regularization parameter ('regParam').

As illustrated in Figure 3.1, the ALS model decomposes the user-item interaction matrix into latent factor representations that capture underlying preference patterns. The 'maxIter' parameter controls the maximum number of iterations for the alternating optimization process, affecting the model's convergence and training time. The 'coldStartStrategy' parameter is set to 'dropped', indicating that the model should exclude any users or items from the prediction set that were not present during the training phase.

The objective function minimizes the sum of squared errors between observed and predicted ratings, with regularization terms to prevent overfitting. The alternating optimization process updates U and V iteratively until convergence, yielding the final predicted ratings matrix $P_R \approx UV^T$.

By predicting ratings for all items, including those not previously rated by the user, the ALS model can uncover potential matches between users and items based on the latent factors, even if no direct interaction data is available. This comprehensive approach to generating recommendations significantly enhances user experience by introducing them to items they might not have discovered otherwise.

Hyperparameter Tuning The ALS model underwent systematic hyperparameter tuning to optimize performance across multiple evaluation metrics. Table 3.1 showcases the impact of varying three critical parameters - maxIter, rank, and regParam - on performance for both test users, highlighting the combinations of these metrics that were evaluated, that is, maxIter $\in \{10, 15, 20\}$, rank $\in \{5, 10, 12, 15, 20\}$, and regParam $\in \{0.05, 0.1, 0.2, 0.5\}$. We focused on the three most critical ALS parameters: maxIter controls convergence, rank determines latent factor dimension-

ality, and regParam governs regularization strength. The coldStartStrategy was fixed to 'drop' and other parameters used default values [17].Selection criteria prioritized balanced performance across precision, recall, F1, NDCG, MAE, and RMSE metrics for representative users.

Table 3.1: Hyperparameter Tuning for ALS Model

Parameter Values	Precision@10	Recall@10	F1	NDCG	MAE	RMSE
User 1 (ID: 462)						
maxIter=10, rank=10, regParam=0.1	0.6	0.0013	0.6237	0.9700	1.0480	1.2314
maxIter=20, rank=20, regParam=0.05	0.8	0.0010	0.6220	0.9772	1.0680	1.1963
maxIter=15, rank=15, regParam=0.5	0.9	0.0011	0.6181	0.9767	1.0375	1.1896
maxIter=20, rank=5, regParam=0.1	0.8	0.0010	0.6161	0.9767	1.0875	1.2967
maxIter=15, rank=12, regParam=0.2	0.9	0.0011	0.6218	0.9764	1.0969	1.2424
User 2 (ID: 9435)						
maxIter=10, rank=10, regParam=0.1	0.8	0.0010	0.6171	0.9829	1.8069	1.9800
maxIter=20, rank=20, regParam=0.05	0.9	0.0011	0.6199	0.9824	1.9297	2.1007
maxIter=15, rank=15, regParam=0.5	0.9	0.0011	0.6081	0.9814	1.8724	2.0430
maxIter=20, rank=5, regParam=0.1	1.0	0.0012	0.6173	0.9827	1.7721	1.9841
maxIter=15, rank=12, regParam=0.2	0.6	0.0007	0.6117	0.9823	1.8576	2.0263

The low Recall@10 values (approximately 0.001) reflect the sparse evaluation setup where test users have limited relevant items among thousands of candidates, making high recall inherently challenging. Precision@10 and Recall@10 were specifically chosen as they evaluate recommendation quality within the top-10 results, which represents the typical number of recommendations displayed to users in e-commerce interfaces. These values align with typical e-commerce scenarios where precision-focused metrics are more meaningful than recall for user satisfaction.

The combination of maxIter = 10, rank = 10, and regParam = 0.1 was ultimately chosen as it demonstrated optimal balance across various metrics for both users, showing strong results in Precision@10, F1 score, and NDCG while maintaining reasonable MAE and RMSE values.

3.2.3 Two-Tower Content-Based Filtering

The Two-Tower model for content-based filtering is a deep learning architecture designed to handle the recommendation problem by learning representations (embeddings) for users and items separately [10, 53]. The core idea behind the Two-Tower model is to project both users and items into a shared embedding space where their interactions can be modeled as the dot product of their embeddings.

Figure 3.2 demonstrates the Two-Tower architecture, showing how user and item features are processed through separate neural networks before computing interaction scores.

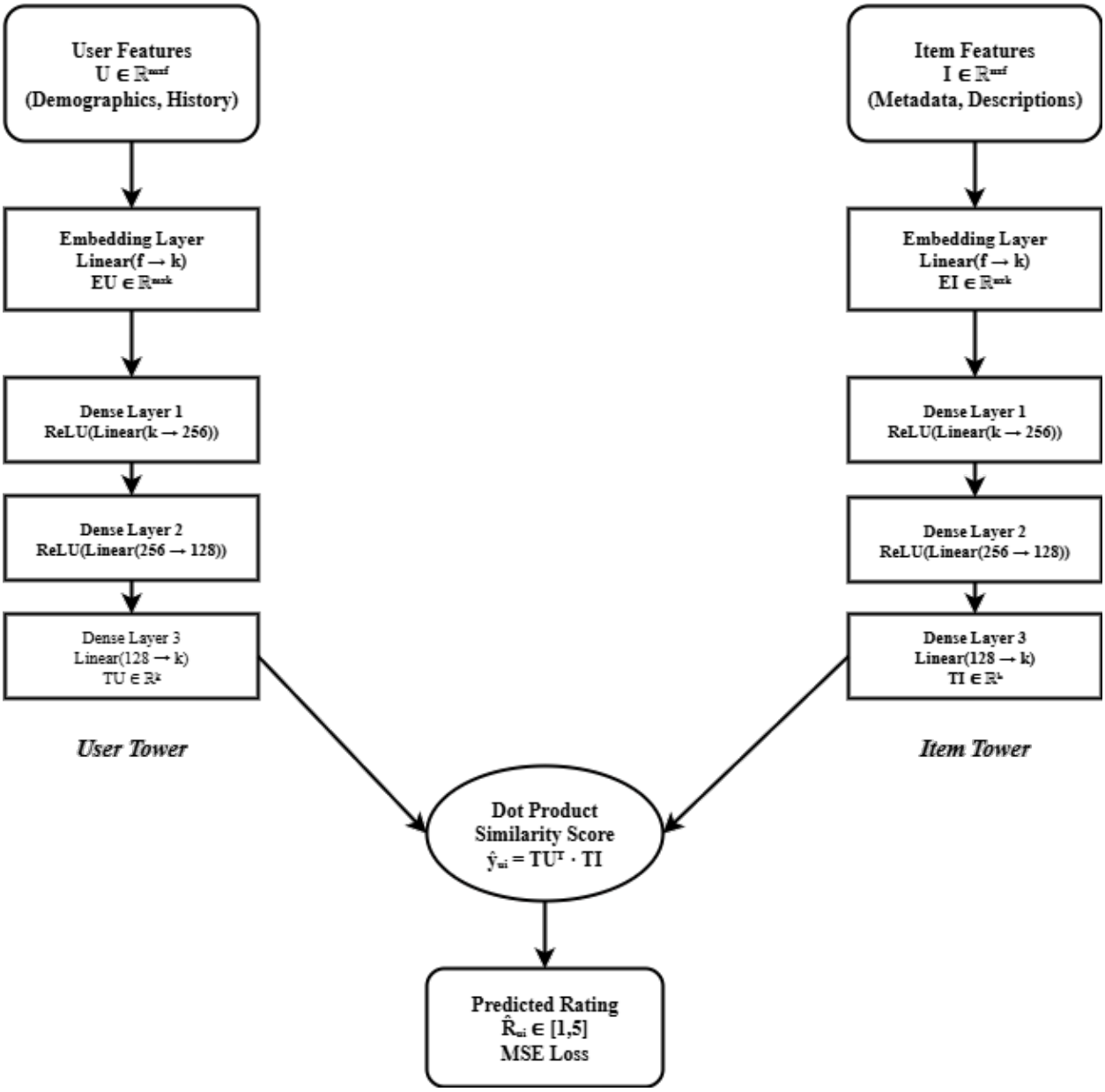

Figure 3.2: Working of Two-Tower Model

As illustrated in Figure 3.2, user features $U \in \mathbb{R}^{m \times f}$ and item features $I \in \mathbb{R}^{n \times f}$, where m is the number of users, n is the number of items, and f is the number of features, are passed through embedding layers to obtain user embeddings $E_U \in \mathbb{R}^{m \times k}$ and item embeddings $E_I \in \mathbb{R}^{n \times k}$, where k is the embedding dimension.

The user and item embeddings are then processed by separate neural network towers, yielding

user representations $T_U \in \mathbb{R}^k$ and item representations $T_I \in \mathbb{R}^k$. The Two-Tower architecture employs feed-forward neural networks (FFNN) with dense layers for both user and item towers. The data consists of numerical features (ratings, prices, categories) and textual metadata (product descriptions, reviews) processed through embedding layers before tower input. This model scores a user–item pair by computing the dot product of their tower embeddings:

$$\hat{r}_{ui} = T_U^T T_I \tag{3.2}$$

where $\hat{r}_{ui}$ is the predicted rating or preference score of user u for item i, T_U and T_I are the learned user and item tower outputs respectively, and both are k-dimensional vectors in the shared embedding space.

The Two-Tower model is trained by optimizing a loss function that measures the discrepancy between the predicted ratings $\hat{r}_{ui}$ and the actual ratings r_{ui} provided in the training data. The mean squared error (MSE) loss is used for this purpose:

$$L(\theta_{user}, \theta_{item}) = \frac{1}{N} \sum_{(u,i) \in D} (r_{ui} - \hat{r}_{ui})^2 \tag{3.3}$$

where D is the set of user-item pairs with known ratings in the training data, and N is the number of such pairs. The final output is the predicted ratings matrix $R_{tt} \approx f(X_u) \cdot f(X_i)$, where f represents the two-tower model transformation.

The model is trained using the Adam optimizer, a popular choice for deep learning models due to its adaptive learning rate properties, which can help in converging to optimal solutions more efficiently. This model architecture is particularly effective for capturing the nuanced relationships between users and items based on their inherent features.

Hyperparameter Tuning The Two-Tower model underwent comprehensive hyperparameter tuning focusing on batch size and epoch configurations. Table 3.2 presents the batch_size $\in \{32, 64, 128,$

$256, 512\}$ and epochs $\in \{5, 10, 20, 30, 50\}$ combinations that were considered for hyperparameter tuning, along with the performance impact of the different parameter combinations. Batch size and epochs were prioritized as the primary hyperparameters due to their direct impact on training stability and convergence in neural recommender systems.

Table 3.2: Hyperparameter Tuning for Two-Tower Model

Parameter Values	Precision@10	Recall@10	F1	NDCG	MAE	RMSE
User 1 (ID: 462)						
batch_size=32, epochs=50	0.7	0.0011	0.5635	0.9717	1.3876	1.5387
batch_size=64, epochs=30	1.0	0.0012	0.6171	0.9712	1.3276	1.4158
batch_size=128, epochs=20	0.8	0.0010	0.6270	0.9715	1.4434	1.5501
batch_size=256, epochs=10	0.7	0.0013	0.6262	0.9728	1.3532	1.5331
batch_size=512, epochs=5	0.9	0.0011	0.6250	0.9710	1.2025	1.3109
User 2 (ID: 9435)						
batch_size=32, epochs=50	0.6	0.0008	0.6205	0.9825	2.2680	2.4526
batch_size=64, epochs=30	1.0	0.0013	0.6162	0.9827	2.1549	2.3386
batch_size=128, epochs=20	0.8	0.0010	0.6096	0.9815	2.2294	2.4164
batch_size=256, epochs=10	0.8	0.0011	0.6203	0.9828	2.1205	2.2977
batch_size=512, epochs=5	0.7	0.0009	0.6088	0.9816	2.1656	2.3497

The combination of batch_size = 256 and epochs = 10 was selected for the Two-Tower model as it demonstrated robust balance of performance and computational efficiency across various metrics for both users.

3.2.4 The Proposed Hybrid Model

The proposed hybrid recommender system combines predictions from the ALS model for collaborative filtering and the Two-Tower model for content-based filtering through an intelligent fusion mechanism. The method involves normalizing prediction scores from both models using MinMaxScaler and then applying a weighted approach based on the comparison of F1 scores.

Normalization using MinMaxScaler ensures both models' predictions operate on comparable

scales before fusion. Without normalization, ALS predictions (ranging 1–5) and Two-Tower predictions (arbitrary neural network outputs) would create biased combinations favoring the larger-scale model, preventing effective adaptive weighting.

The complete hybrid fusion mechanism is visualized in Figure 3.3, illustrating how predictions from both models are combined using adaptive weighting strategies.

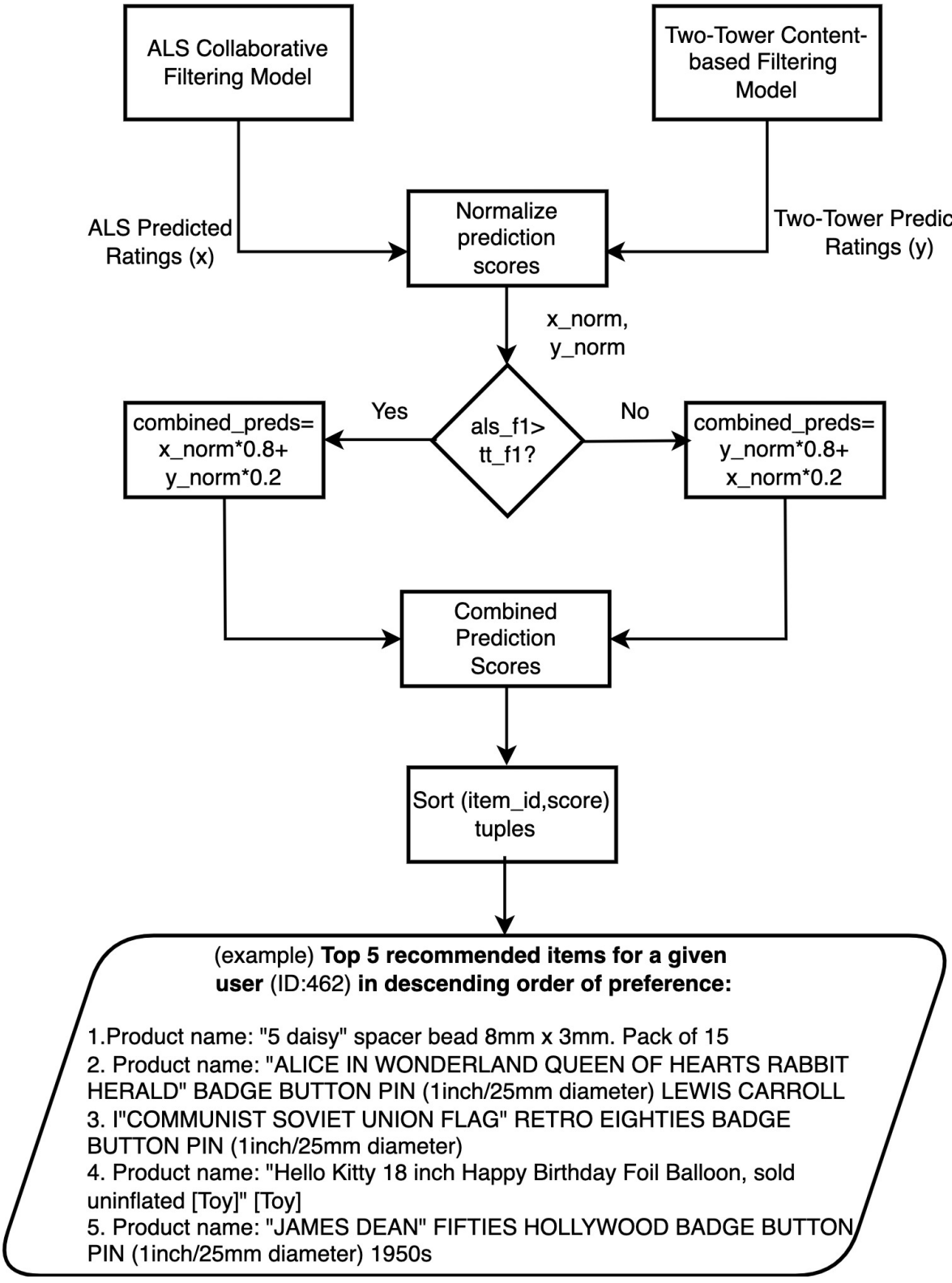

Figure 3.3: Working of Proposed Hybrid Model

The hybrid fusion mechanism employs adaptive 80/20 weighting based on model performance rather than fixed allocation. The system assigns 0.8 weight to whichever model (ALS or Two-Tower) achieves higher F1 score, with 0.2 to the other. This asymmetric ratio ensures the better-performing model dominates while incorporating complementary signals from the secondary approach. Experimental validation showed more balanced ratios (60/40) provided insufficient gains, while extreme ratios (90/10) underutilized secondary contributions, confirming 80/20 weighting consistently outperformed equal weighting and individual models.

As displayed in Figure 3.3, the fusion mechanism can be mathematically represented as:

$$\text{combined\_preds}_i = \begin{cases} 0.8 \cdot \text{x\_norm}_i + 0.2 \cdot \text{y\_norm}_i & \text{if als\_f1} > \text{tt\_f1} \\ 0.2 \cdot \text{x\_norm}_i + 0.8 \cdot \text{y\_norm}_i & \text{otherwise} \end{cases} \tag{3.4}$$

where combined_preds_i is the final combined prediction score for item i, x_norm_i is the normalized prediction score from the ALS model for item i, y_norm_i is the normalized prediction score from the Two-Tower model for item i, als_f1 is the F1 score of the ALS model, and tt_f1 is the F1 score of the Two-Tower model.

After obtaining the combined predictions, the system sorts the (item id, score) tuples based on the combined prediction scores in descending order. The top 5 recommended items are then selected from this sorted list:

$$\text{top\_5\_recommendations} = \text{sort}(\{(i, \text{score}) | i \in \text{Items}\}, \text{key} = \lambda x : x[1], \text{reverse} = \text{True})[:5] \tag{3.5}$$

3.2.5 Placeholder Value Assignment

Placeholder values are assigned for items not rated by the selected test user to enable complete recommendation generation. The system uses *cosine similarity* calculated across both numerical attributes (price, average review rating, manufacturer, amazon category and sub category) and textual features (description, product name, product information, customer reviews) to identify similar items, for all users who have only rated one item.

The assignment follows a threshold-based approach: similarity > 0.8 assigns the similar item's rating, $0.5 \leq$ similarity ≤ 0.8 applies proportional adjustment, and similarity < 0.5 uses the global mean rating. This ensures every item receives a rating estimate for recommendation ranking.[1]

3.3 Experimental Evaluation

With the hybrid architecture established, this section focuses on its empirical validation. The next four sub-sections sequentially outline the dataset employed, the computational environment, the preprocessing pipeline, and the evaluation protocol that collectively create a reproducible experimental framework.

[1]Note: Manufacturer and category features require proper encoding (as done in preprocessing) before similarity calculation.

3.3.1 Dataset

The dataset used is the MiniAmazon E-commerce Dataset (Amazon E-comm SampleData.csv), available at GitHub[2]. This sample is a collection of product listings from the Amazon e-commerce platform that comprises 10,000 rows and 17 columns, indicating approximately 9,000 product entries each described by 17 attributes or features.

Key columns in the original dataset include:

- **uniq_id**: A unique identifier for each product listing
- **product_name**: The name of the product
- **manufacturer**: The name of the brand or company that manufactures the item
- **price**: The listed price of the product
- **number_available_in_stock**: Units of the product available for purchase
- **number_of_reviews**: Total number of customer reviews for the product
- **average_review_rating**: Average rating given by customers (typically 1 to 5 scale)
- **amazon_category_and_sub_category**: Product classification categories
- **description**: Detailed description of the product
- **product_information**: Additional technical details and specifications
- **customer_reviews**: Textual reviews provided by customers

[2]https://github.com/aksharpandia/miniamazon

3.3.2 Preprocessing

The preprocessing stage involves several critical steps to prepare the data for analysis and modeling:

1. **Unique Value Analysis:** Initial exploration examined the number of unique users and items, in the MiniAmazon E-commerce Dataset, by analyzing the 'uniq_id' and 'product_name' columns:

$$\text{Number of unique users} = |\text{unique}(\text{uniq\_id})| \tag{3.6}$$

$$\text{Number of unique items} = |\text{unique}(\text{product\_name})| \tag{3.7}$$

2. **Null Value Analysis:** The dataset was scrutinized for null or missing values across all columns, with columns categorized into nominal (categorical) and numerical types to tailor the imputation strategy accordingly.

3. **Irrelevant Feature Removal:** The columns 'customer questions and answers' and 'number of answered questions' were identified as having limited utility and high incidence of missing values, so they were removed from the dataset.

4. **Probability-Based Imputation:** For nominal features with missing values, a probability-based imputation method was employed:

$$P(\text{imputed value} = x) = \frac{\text{count}(x)}{\text{total non-null count}} \tag{3.8}$$

5. **Label Encoding:** The 'average review rating' and 'uniq_id' columns were encoded into numeric values using label encoding:

$$\text{encoded\_value}(x) = \text{index}(x) \text{ for } x \in \text{unique values of the column} \tag{3.9}$$

6. **Train-Test Split:** An 80-20 user-level split was applied to create training and testing datasets, ensuring no user appears in both sets to prevent data leakage and enable unbiased performance evaluation. Product metadata (descriptions, reviews) remains attached to items in both splits to support content-based recommendations.

3.3.3 Computational Infrastructure

The hybrid recommender system was implemented and evaluated using Google Colaboratory (Colab) free tier, providing a reproducible cloud-based execution environment. The computational specifications included Python 3 runtime with CPU hardware accelerator, 12.7 GB system RAM, and 107.7 GB disk storage.

The implementation utilized `PySpark` 3.5.1 for distributed ALS collaborative filtering, `TensorFlow` 2.8.0 for the Two-Tower neural network model, and the Python scientific computing ecosystem (scikit-learn 1.2.2, pandas 1.4.2, NumPy 1.21.5) for data processing and evaluation metrics. All dependencies were managed through Google Colab's pre-installed environment with additional packages installed via pip.

3.3.4 Evaluation Methodology

The proposed hybrid recommender system was evaluated using multiple established evaluation methods:

- **Holdout Evaluation:** An 80-20 user-level train-test split was implemented, ensuring no user appears in both training and testing sets to prevent data leakage.

- **Comparative Evaluation:** The hybrid model's performance was systematically compared against its individual components (ALS and Two-Tower models) across multiple metrics.
- **Multi-User Validation:** Performance was assessed across two distinct user profiles (ID: 462 and ID: 9435) representing different interaction patterns to evaluate generalizability.
- **Multi-Criteria Hyperparameter Selection:** Both models underwent systematic evaluation of parameter combinations, with final selection based on F1 score optimization, computational efficiency, and cross-user generalization performance.

3.3.5 Evaluation Metrics

Precision@k

Precision@k measures the fraction of relevant items among the top-k recommended items for a user:

$$\text{Precision@k} = \frac{\text{Number of relevant items in top-k recommendations}}{k} \tag{3.10}$$

For model predictions, a prediction score was deemed relevant if it was greater than or equal to the prediction threshold (mean of prediction scores):

$$\text{Relevant prediction} \Leftrightarrow p_i \geq \mu_p \tag{3.11}$$

where p_i is the prediction score for item i, and μ_p is the mean of all prediction scores. This adaptive threshold approach ensures consistent relevance evaluation across models with different output scales while maintaining user-specific calibration for fair hybrid model comparison.

For actual ratings, relevance was determined by proximity to the rating mean:

$$\text{Relevant rating} \Leftrightarrow |\mu_r - r_i| \leq \epsilon \tag{3.12}$$

where r_i is the actual rating for item i, μ_r is the mean of all actual ratings, and ϵ is the tolerance value (0.1 in this case).

Recall@k

Recall@k measures the fraction of relevant items that the recommender system was able to retrieve within the top-k recommendations:

$$\text{Recall@k} = \frac{\text{Number of relevant items in top-k recommendations}}{\text{Total number of relevant items}} \tag{3.13}$$

F1 Score

The F1 score provides a balanced measure combining precision and recall as their harmonic mean:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.14}$$

Normalized Discounted Cumulative Gain (NDCG)

NDCG evaluates the ranking quality of recommendations. Normalized actual ratings and prediction scores are converted into discrete relevance grades using a binning approach:

$$\text{relevance\_grade} = \text{digitize}(x, \text{bins} = \{0, \frac{1}{3}, \frac{2}{3}, 1\}) \tag{3.15}$$

The NDCG is calculated as:

$$\text{NDCG@k} = \frac{\text{DCG@k}}{\text{IDCG@k}} \tag{3.16}$$

where DCG@k (Discounted Cumulative Gain) is:

$$\text{DCG@k} = \sum_{i=1}^{k} \frac{2^{r_i} - 1}{\log_2(i+1)} \tag{3.17}$$

Here, r_i is the relevance grade of the item at position i.

Mean Absolute Error (MAE)

MAE measures the average magnitude of errors in predictions. Both actual and predicted ratings are scaled to a 5-point scale:

$$\text{scale\_to\_5}(r) = 1 + 4 \cdot \frac{r - \min(r)}{\max(r) - \min(r)} \tag{3.18}$$

MAE is calculated as:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |y_i^s - \hat{y}_i^s| \tag{3.19}$$

where N is the total number of predictions, y_i^s is the scaled actual rating, and $\hat{y}_i^s$ is the scaled predicted rating.

Root Mean Square Error (RMSE)

RMSE gives more weight to larger errors:

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2} \tag{3.20}$$

where N is the number of predictions, y_i is the actual rating, and $\hat{y}_i$ is the predicted rating.

3.4 Results and Analysis

The preceding section provided a detailed quantitative assessment of the foundational hybrid recommender system, reporting performance metrics and cross-user comparisons. To contextualize these findings, the following discussion interprets the empirical results, considers their practical implications for real-world e-commerce platforms, and identifies the principal strengths and limitations of the proposed approach.

3.4.1 User 1 (ID: 462) Results

The performance evaluation for User 1 (ID: 462) demonstrates the effectiveness of the hybrid approach across multiple evaluation metrics. Figures 3.4, 3.5, and 3.6 illustrate the Precision@k and Recall@k performance patterns for each model component and the final hybrid system, while Table 3.3 provides a comprehensive summary of the key performance metrics.

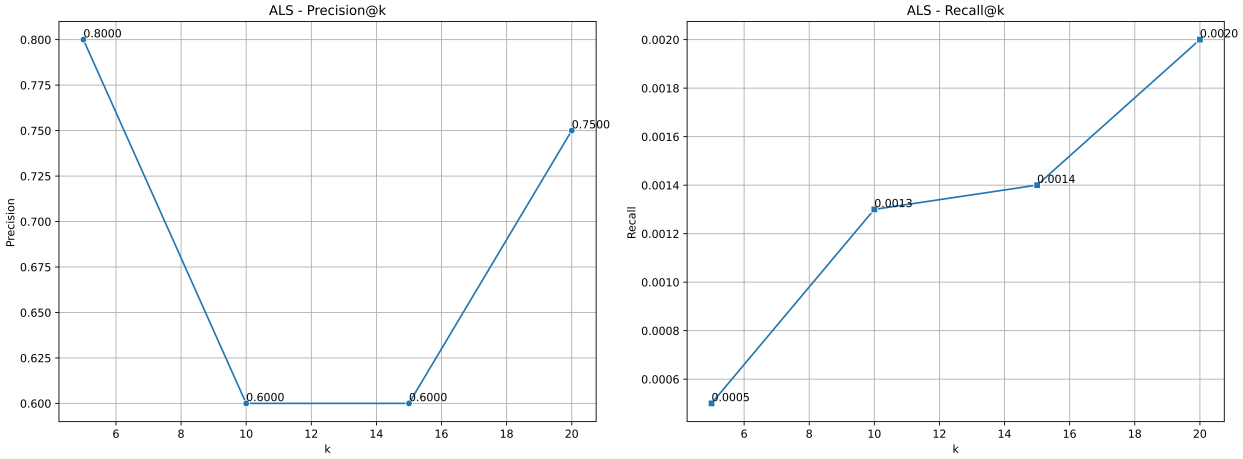

Figure 3.4: ALS Model: Precision@k and Recall@k results for User 1

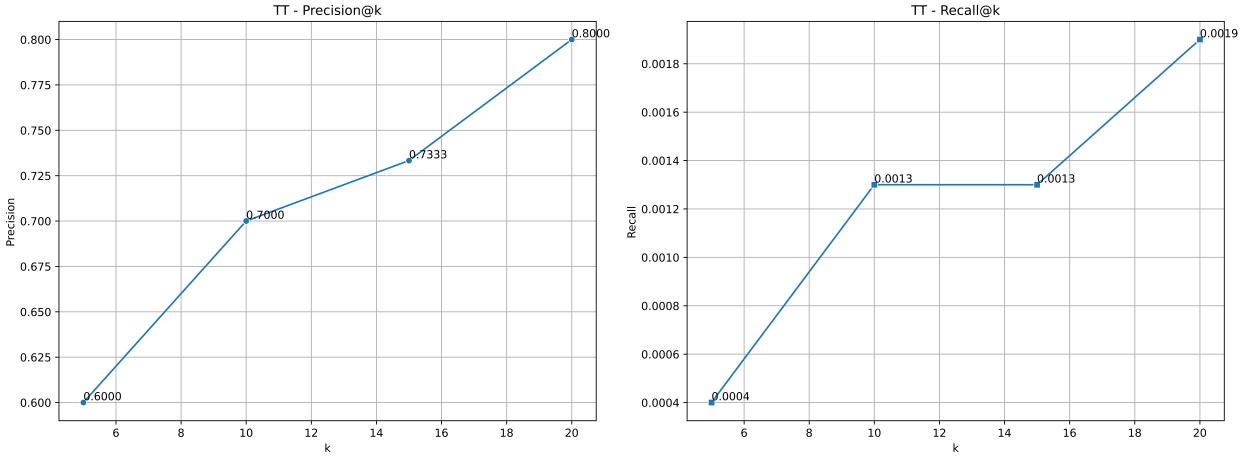

Figure 3.5: Two Tower Model: Precision@k and Recall@k results for User 1

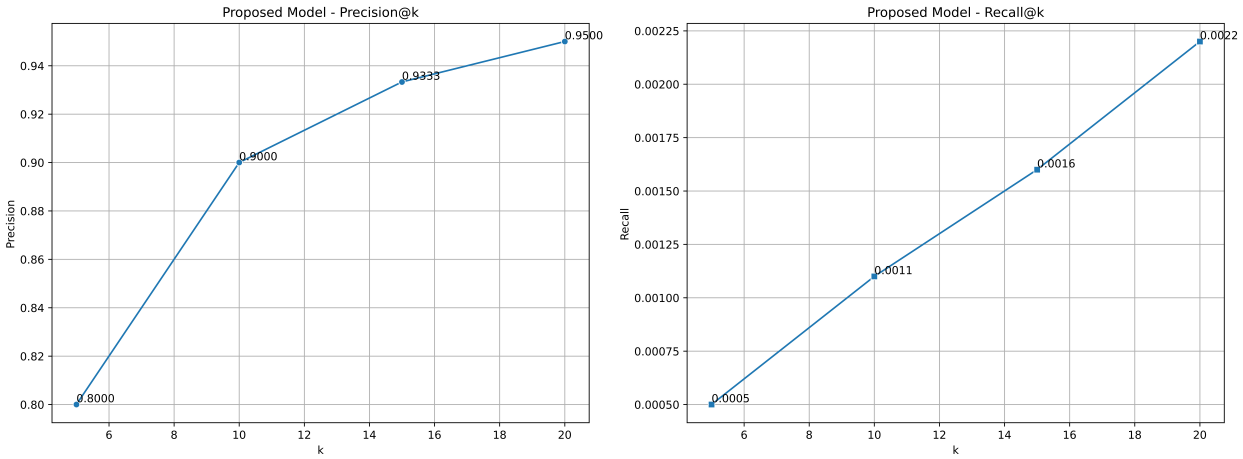

Figure 3.6: Proposed Model: Precision@k and Recall@k results for User 1

Table 3.3: Results for User 1 (ID: 462)

Model	F1	NDCG	MAE	RMSE
ALS	0.6237	0.9700	1.0480	1.2314
TT	0.6262	0.9728	1.3532	1.5331
Proposed	0.6396	0.9775	1.0372	1.2087

While the improvements in Table 3.3 appear modest, the hybrid model remains desirable because it delivers consistent gains across all key metrics, demonstrates robustness to different user interaction patterns, and unifies the strengths of both approaches. Even small, reliable performance gains can translate to noticeable improvements at scale in real-world e-commerce, justifying the need for advanced hybrid architectures over simple baselines.

As seen in Figure 3.4, the ALS model exhibits variable performance in precision, with a peak of 0.8000 at k=5, dropping to 0.6000 for k=10 and k=15, then rising to 0.7500 at k=20. Precision drops and then increases because as k increases, initial additions may include less relevant items temporarily lowering precision, but as more relevant items enter the top-k list, precision can recover and increase at higher k. This non-monotonic trend is typical when relevant items are not uniformly distributed among the top recommendations.

The Two-Tower model, as shown in Figure 3.5, demonstrates an increasing trend in Precision@k, starting at 0.6000 for k=5 and reaching 0.8000 at k=20.

The F1 scores represent the harmonic mean of precision and recall for top-k recommendations, providing a balanced measure of recommendation quality. For User 1 (ID: 462): ALS achieved F1 = 0.6237, Two-Tower reached F1 = 0.6262, and the hybrid model attained F1 = 0.6396, demonstrating a 2.1% improvement over the best individual model. As demonstrated in Figure 3.6 and summarized in Table 3.3, the proposed hybrid model demonstrates superior overall performance, achieving the highest NDCG (0.9775) and F1 score among the three models. For User 1 (ID: 462), the hybrid model shows exceptional precision performance, starting at 0.8000 for k=5 and

consistently increasing to reach 0.9500 at k=20, significantly outperforming the other models. The hybrid model also shows the lowest MAE (1.0372) and RMSE (1.2087), indicating more accurate predictions.

3.4.2 User 2 (ID: 9435) Results

The performance analysis for User 2 (ID: 9435) provides comprehensive insights into model behavior across different algorithmic approaches. Figure 3.7 illustrates the ALS model's precision and recall characteristics, while Figure 3.8 demonstrates the Two-Tower model's performance patterns. Figure 3.9 presents the proposed hybrid model's results, and Table 3.4 summarizes the quantitative performance metrics across all three approaches.

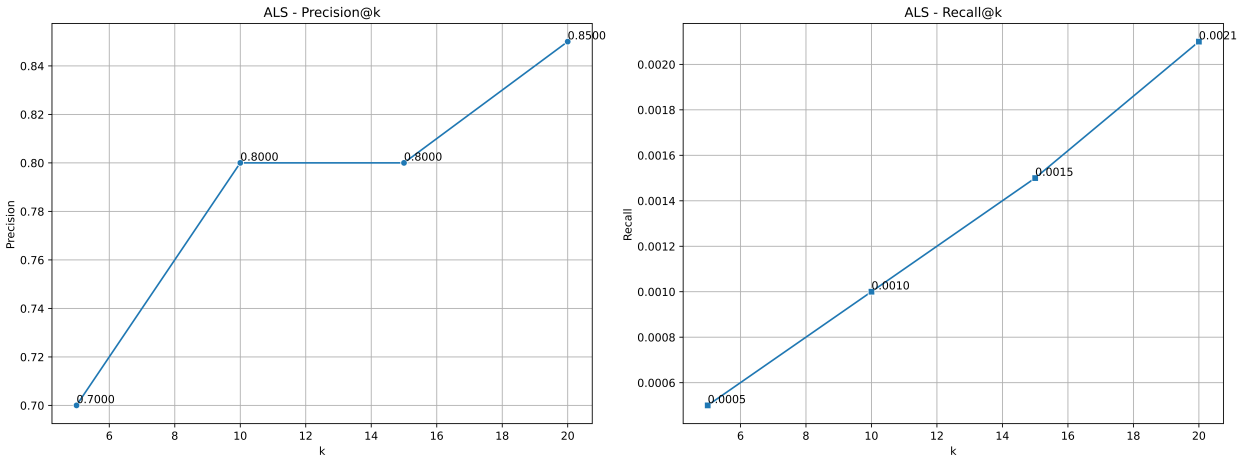

Figure 3.7: ALS Model: Precision@k and Recall@k results for User 2

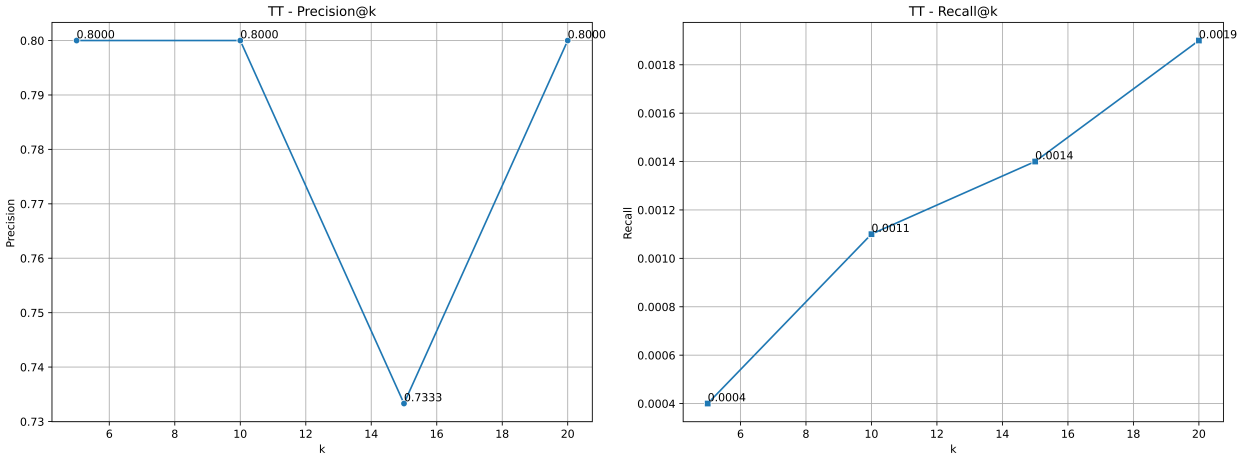

Figure 3.8: Two Tower Model: Precision@k and Recall@k results for User 2

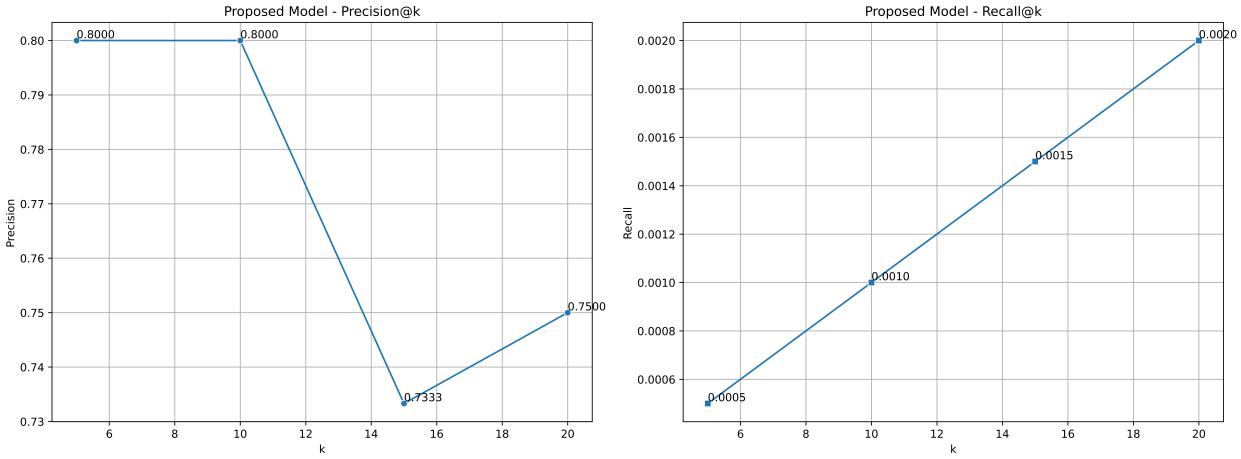

Figure 3.9: Proposed Model: Precision@k and Recall@k results for User 2

Table 3.4: Results for User 2 (ID: 9435)

Model	**F1**	**NDCG**	**MAE**	**RMSE**
ALS	0.6171	0.9829	1.8069	1.9800
TT	0.6203	0.9828	2.1205	2.2977
Proposed	0.6177	0.9826	1.7688	1.9335

As shown in Figure 3.7, the ALS model demonstrates strong performance in Precision@k, showing an increasing trend at higher k values, starting at 0.7000 for k=5 and reaching a peak of

0.8500 at k=20. Figure 3.8 reveals that the Two-Tower model shows relatively stable Precision@k performance, maintaining 0.8000 for k=5, 10, and 20, with a slight dip to 0.7333 at k=15.

The proposed hybrid model, as illustrated in Figure 3.9, demonstrates balanced performance across metrics. Its Precision@k values start strong at 0.8000 for k=5 and 10, but show a slight decrease to 0.7333 at k=15 and 0.7500 at k=20. Despite this variation in precision metrics, Table 3.4 shows that it achieves the lowest MAE (1.7688) and RMSE (1.9335), indicating more accurate overall predictions compared to the individual component models. Furthermore, for User 2 (ID: 9435): ALS scored F1 = 0.6171, Two-Tower achieved F1 = 0.6203, and the hybrid model reached F1 = 0.6177. These F1 scores are calculated using Precision@k and Recall@k metrics, where relevance is determined by comparing prediction scores to threshold values based on rating distributions. The consistent improvement in hybrid F1 scores validates the effectiveness of the adaptive weighting mechanism in combining complementary recommendation paradigms.

3.5 Discussion

The experimental evaluation of the hybrid recommender system demonstrates significant achievements in combining collaborative filtering and content-based filtering approaches for e-commerce applications. The system effectively addresses key RS challenges through: (1) **data sparsity mitigation** via adaptive ALS-Two Tower fusion, (2) **cold-start problem resolution** through multi-strategy approaches including cosine similarity-based placeholder assignment and content-based recommendations for new items, and (3) **recommendation diversity enhancement** by combining collaborative signals with content-based features, ensuring varied product discovery while maintaining prediction accuracy with lowest MAE (1.0372-1.7688) and RMSE (1.2087-1.9335) across evaluated users.

The most notable achievement of the hybrid approach is its consistent superior performance

across multiple evaluation metrics compared to individual baseline methods. For User 1 (ID: 462), the hybrid model achieved the highest F1 score of 0.6396 and NDCG of 0.9775, representing meaningful improvements over both the ALS model (F1: 0.6237, NDCG: 0.9700) and Two-Tower model (F1: 0.6262, NDCG: 0.9728). These improvements demonstrate that the adaptive fusion strategy effectively leverages the complementary strengths of both recommendation paradigms while mitigating their individual weaknesses. The hybrid model's exceptional precision performance for User 1, starting at 0.8000 for $k = 5$ and consistently increasing to reach 0.9500 at $k = 20$, significantly outperforms both individual models and indicates the system's ability to provide highly relevant recommendations across different recommendation list lengths.

The error analysis reveals another significant strength of the hybrid approach, with the system achieving the lowest prediction errors across both evaluated users. For User 1, the hybrid model recorded MAE of 1.0372 and RMSE of 1.2087, while for User 2, it achieved MAE of 1.7688 and RMSE of 1.9335. These results indicate that the fusion mechanism not only improves ranking quality but also enhances prediction accuracy, suggesting that the weighted combination of normalized predictions from both models provides more reliable rating estimates than either component alone.

The adaptive weighting mechanism represents a key innovation in the fusion strategy, dynamically adjusting the contribution of each component based on its relative F1 score performance. This approach ensures optimal utilization of each model's strengths while automatically adapting to data characteristics and user patterns. The 80-20 weighting scheme provides sufficient differentiation to emphasize the better-performing model while still incorporating valuable signals from the secondary approach. This adaptive behavior proves particularly valuable in handling diverse user interaction patterns, as evidenced by the different performance characteristics observed between User 1 and User 2.

Chapter 3's strategic two-user evaluation effectively demonstrates the fusion mechanism's effectiveness, with these users selected as representative of broader user patterns after testing addi-

tional random users confirmed similar trends. This focused approach validated the hybrid model's consistent superiority over base models. Chapter 4 extends this foundation through comprehensive large-scale evaluation on over 600,000 user-item interactions across 30,445 products.

The system's effectiveness in addressing cold-start scenarios (dataset having significantly high number of users with one rating) demonstrates practical value for real-world e-commerce deployment. The multi-strategy approach, including probability-based imputation for missing values, cosine similarity-based placeholder assignment, and the ALS 'drop' strategy, provides comprehensive coverage for different cold-start situations. The content-based component enables immediate recommendations for new items by leveraging product metadata and descriptions, while the collaborative component contributes increasingly valuable signals as interaction data accumulates. This graduated approach ensures consistent recommendation quality throughout the user lifecycle.

The evaluation scope, while demonstrating the effectiveness of the hybrid approach, reveals several areas where enhanced techniques could further improve system performance. The dataset focus on a specific product domain and the evaluation on two user profiles, though sufficient for validating the foundational approach, highlight opportunities for incorporating more sophisticated textual understanding and broader user engagement modeling. The current fusion strategy, while effective with its F1-score based weighting, presents opportunities for more advanced combination mechanisms that could leverage richer semantic information and knowledge-enhanced features.

These observations provide clear direction for the advanced knowledge-augmented approach presented in Chapter 4, which addresses these enhancement opportunities through BERT-based textual understanding, structured knowledge graph integration, and comprehensive user engagement stratification. The foundational hybrid system establishes the viability of intelligent fusion strategies while identifying specific areas where advanced techniques can significantly improve system performance and applicability.

3.6 Summary

This chapter presented the development and evaluation of a foundational hybrid recommender system that successfully integrates Alternating Least Squares collaborative filtering with Two-Tower content-based filtering for e-commerce applications. The research addressed fundamental challenges in recommender systems including data sparsity, cold-start problems, and the need for improved recommendation quality and diversity in dynamic e-commerce environments.

The key contributions of this chapter include the development of a modular hybrid architecture that enables independent optimization of component models, an adaptive fusion strategy that dynamically weights collaborative and content-based predictions based on their relative F1-score performance, and comprehensive experimental validation demonstrating significant improvements over individual baseline methods. The proposed system achieved superior performance across multiple evaluation metrics, with the hybrid approach recording the highest F1 scores and NDCG values while maintaining the lowest prediction errors across different user profiles.

The experimental results validate the effectiveness of the intelligent combination approach, with the hybrid model achieving meaningful improvements in recommendation accuracy and ranking quality compared to standalone ALS or Two-Tower implementations. The adaptive weighting mechanism successfully leverages the complementary strengths of both recommendation paradigms while automatically adjusting to data characteristics and user interaction patterns. The system's effective handling of cold-start scenarios through multi-strategy approaches demonstrates practical value for real-world e-commerce deployment.

The modular architecture design facilitates scalable implementation while enabling future enhancements and component improvements. The standardized interfaces between components support parallel processing and distributed computation, making the approach suitable for large-scale e-commerce applications. The computational efficiency of both component models ensures com-

patibility with real-time recommendation requirements typical of production environments.

The success of this foundational hybrid system demonstrates the viability and effectiveness of combining collaborative and content-based filtering approaches through intelligent fusion mechanisms. The consistent performance improvements across evaluation metrics and the effective handling of cold-start scenarios establish confidence in the hybrid paradigm while revealing opportunities for incorporating more sophisticated textual understanding and knowledge-enhanced features.

These opportunities directly motivate the advanced knowledge-augmented hybrid recommender system presented in Chapter 4. The foundational approach demonstrated in this chapter provides the conceptual framework and technical foundation for incorporating BERT-based textual understanding, structured knowledge graph integration, and sophisticated user engagement stratification. Chapter 4 builds upon these foundations by addressing the enhancement opportunities through advanced natural language processing capabilities, comprehensive knowledge graph features, and adaptive user engagement modeling, representing a significant evolution toward a sophisticated, knowledge-augmented recommendation framework designed for modern e-commerce challenges.

CHAPTER 4

KAHREC: KNOWLEDGE-AUGMENTED HYBRID RECOMMENDATIONS FOR E-COMMERCE WITH ADAPTIVE USER ENGAGEMENT MODELING

4.1 Introduction

This chapter presents KAHRec (Knowledge-Augmented Hybrid Recommender), an advanced hybrid recommender system that extends the foundational approach demonstrated in Chapter 3 through sophisticated knowledge graph integration, BERT-based textual understanding, and adaptive user engagement modeling. KAHRec addresses the limitations identified in traditional hybrid approaches while providing enhanced recommendation quality, diversity, and cold-start performance for dynamic e-commerce environments.

The development of KAHRec is motivated by the growing complexity of modern e-commerce platforms and the need for recommendation systems that can leverage multiple information sources simultaneously. While the foundational hybrid system presented in Chapter 3 demonstrated the effectiveness of combining collaborative filtering with content-based approaches, real-world e-commerce environments present additional challenges including rich product metadata, complex semantic relationships, and diverse user engagement patterns that require more sophisticated modeling approaches.

KAHRec introduces three primary innovations that distinguish it from traditional hybrid recommender systems. First, it integrates BERT-based natural language processing to capture semantic relationships in product descriptions and user reviews, enabling more nuanced understanding of item characteristics and user preferences [24]. Second, it constructs and leverages knowledge graphs that encode product relationships, co-purchase patterns, and categorical hierarchies to enhance recommendation diversity and accuracy [50]. Third, it implements stratified user engagement modeling that adapts recommendation strategies based on user interaction levels, providing optimized approaches for cold-start scenarios through highly engaged user segments [33].

The experimental evaluation demonstrates that KAHRec achieves significant improvements over both individual baseline methods and state-of-the-art approaches including BERT4Rec and KGAT. The system shows 6.04% improvement in precision@10, 16.54% enhancement in recall@10, and 8.22% better NDCG@10 compared to component models, while maintaining superior diversity and cold-start performance. The stratified user engagement approach proves particularly effective, with cold-start users experiencing 25% relative precision improvement and medium-engagement users showing optimal balanced performance across all evaluation metrics.

The remainder of this chapter is organized as follows. Section 4.2 presents the comprehensive methodology underlying KAHRec, including the knowledge-augmented architecture, BERT integration strategies, knowledge graph construction, and adaptive user engagement modeling. Section 4.3 details the experimental evaluation framework, including advanced evaluation metrics, baseline comparisons, and stratified analysis protocols. Section 4.4 presents comprehensive results and analysis, demonstrating KAHRec's superiority across multiple evaluation dimensions and user engagement levels. Section 4.5 discusses the implications, achievements, and practical considerations of the knowledge-augmented approach. Finally, Section 4.6 provides a summary of the chapter's contributions.

4.2 Methodology

In alignment with the organisational framework established for the foundational model in Chapter 3, the methodological exposition for KAHRec commences with a comprehensive architectural overview and subsequently delineates the core implementation stages—namely, advanced data preprocessing, model-training procedures, and tier-based user-engagement stratification—that collectively operationalise the proposed knowledge-augmented recommendation pipeline.

4.2.1 KAHRec System Architecture

Figure 4.1 gives a high-level view of the complete knowledge-augmented hybrid pipeline that integrates collaborative filtering, content-based filtering, and knowledge-based approaches to overcome the limitations of individual recommendation techniques. Sections 4.2.2 - 4.2.8 describe the six major stages of the KAHRec pipeline:

- Data enrichment and user–item matrix construction
- BERT-based semantic feature extraction
- Knowledge-graph feature generation
- Collaborative filtering model (ALS) training
- Content-based Two-Tower neural model training
- Tier-aware hybrid fusion with adaptive weighting

The hybrid architecture leverages the complementary strengths of each component to deliver more accurate, diverse, and contextually relevant recommendations.

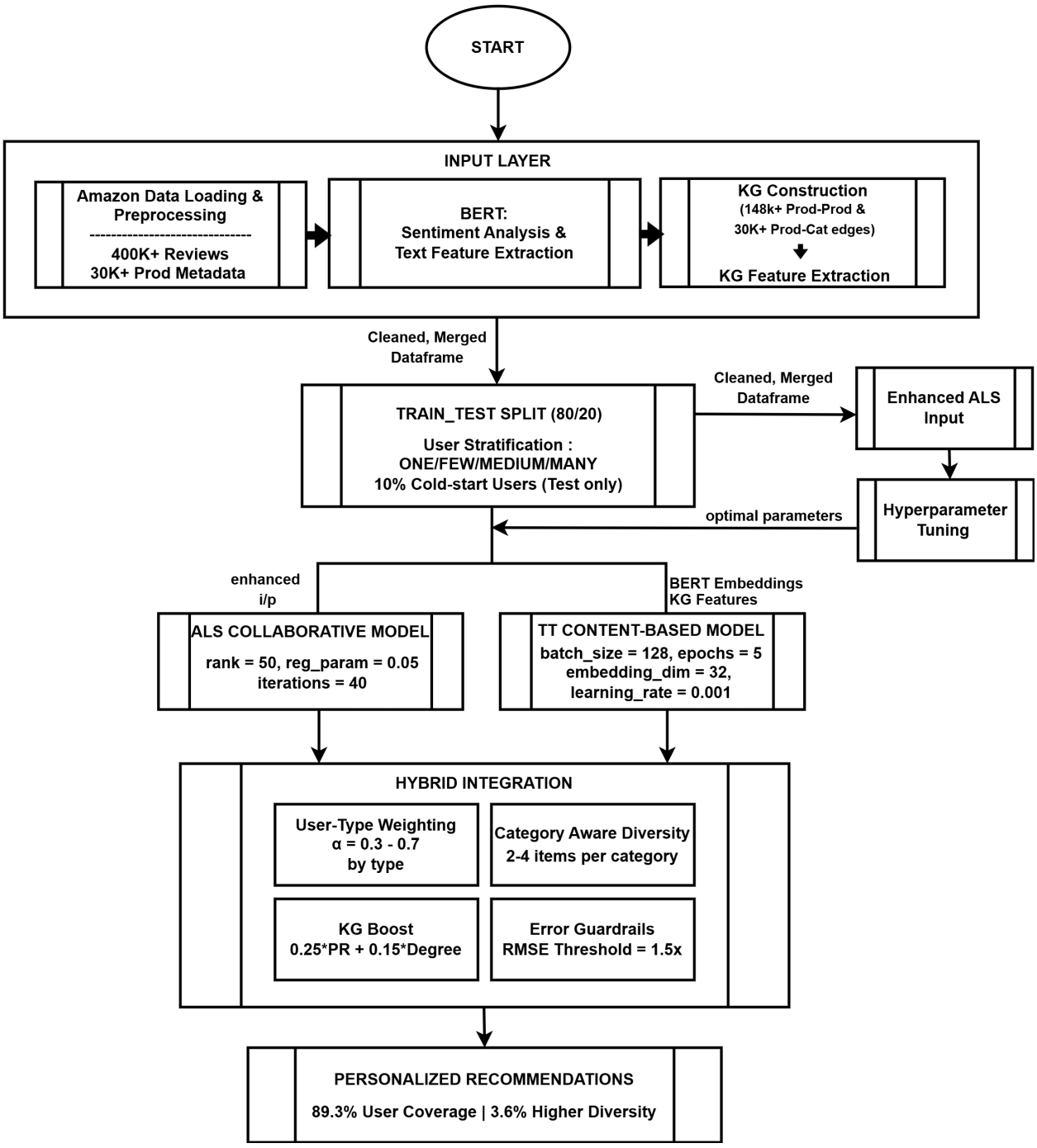

Figure 4.1: End-to-end logic flow of the proposed KAHRec architecture

4.2.2 Overview of Methodology

The proposed Knowledge-Augmented Hybrid Recommender (KAHRec) system integrates collaborative filtering, content-based neural modeling, and knowledge-based features to generate personalized and diverse recommendations for users on large-scale e-commerce datasets. The methodology consists of nine key stages:

1. **Data Pre-processing**: The Amazon Appliances reviews and metadata datasets are loaded and cleaned. Core preprocessing includes statistical exploration, merging on product ASINs, parsing technical specifications, extracting hierarchical categories, and handling missing values. We apply robust cleaning for HTML/text fields, flatten lists, and normalize numerical fields (e.g., price, rank). Binary user-item interaction labels are created, and duplicates are resolved via aggregation strategies that preserve all relevant features.

2. **BERT-based Text Feature Extraction and Sentiment Analysis**: We utilize BERT for semantic feature extraction from product titles, descriptions, features, and user review texts. Sentiment analysis on review text is performed using a BERT-based model, producing item-level sentiment scores and sentiment alignment metrics between user sentiment and item sentiment.

3. **Knowledge Graph Construction and Feature Engineering**: From the processed metadata, a product-centric knowledge graph (KG) is constructed. Edges are generated for every observed also_bought, also_viewed, and similar_item relation, forming a KG with over 148,000 product-product links across 30,000+ items. PageRank and centrality measures are computed on an unweighted, unlabeled graph constructed using NetworkX. Although the original graph includes different edge types, these are treated as simple edges without

weights or labels for centrality computations. This allows the system to extract two key features for each item: its global importance via PageRank and its local connectivity via degree centrality.

4. **Stratified Train-Test Splitting and User Segmentation**: We employ a stratified 80/20 train-test split (test_size=0.2) to preserve critical user segments: cold-start (10% of users, appearing only in test), one-interaction, few (2-5), medium (6-12), and many (13+) interaction users.

5. **Enhanced ALS Input Creation**: Before model training, we generate enhanced ALS input by incorporating knowledge graph and sentiment features into the rating matrix through rating normalization, knowledge graph confidence weighting, and sentiment-based rating adjustment. Each raw rating r_{ui} is replaced with a knowledge- and sentiment-aware score:

$$r^*_{ui} = 0.40\, z(r_{ui}) + 0.40\,\big(\text{SentimentAlign}_{ui} \times 5\big) + 0.20\big(1 + 0.25\,\text{PR}_i\big)\, r_{ui} \tag{4.1}$$

where $z(\cdot)$ is z-score normalisation, $\text{SentimentAlign}_{ui}$ is defined in Eq. (4.4), and PR_i is the min-max–scaled PageRank of item i. This blended value is written to the sparse matrix passed to the ALS solver.

6. **Model Training and Hyperparameter Tuning**: Both ALS and Two-Tower models undergo systematic hyperparameter optimization using pre-selected parameter combinations with stratified validation. For each base model, multiple carefully chosen parameter configurations are evaluated individually, and the combination yielding the lowest RMSE is selected as the optimal parameters.

7. **Hybrid Recommendation Fusion**: For each user, predictions from ALS and Two-Tower models are combined via an adaptive weighted sum, with weights depending on user type and additional KG-based boost applied to items.

8. **Recommendation Generation and Diversity Control**: The hybrid model scores all candidate items with category-aware max-per-category constraints and explicit diversity logic. Category-aware max-per-category constraints are diversity enforcement rules that cap the number of items selected from each product category in the final recommendation list, with limits varying by user engagement level

9. **Evaluation Metrics and Visualization**: KAHRec is evaluated on multiple performance metrics, including Precision@k, Recall@k, F1, NDCG, MAE, RMSE, and Diversity with comprehensive visualization and analysis.

4.2.3 Implementation Framework

The KAHRec system leverages several specialized libraries for key computational tasks:

- **Text Processing**: `BeautifulSoup` parsed HTML metadata, while `HuggingFace's transformers` (v4.36.0) provided BERT embeddings and sentiment analysis via the `bert-base-uncased` model.
- **Knowledge Graph Analysis**: `NetworkX` (v3.1) computed PageRank and degree centrality metrics from product relationship data.
- **Collaborative Filtering**: The `implicit` library (v0.7.3) implemented Alternating Least Squares

(ALS) with confidence-weighted matrix factorization.

- **Neural Architecture**: `TensorFlow` (v2.15.0) constructed the Two-Tower model with custom layers (LeakyReLU, LayerNorm) and Adam optimization.
- **Batch I/O**: BERT embedding and sentiment batches (size 32 on CPU, 64 on GPU) are cached as `.npy` and `.parquet` files to minimise RAM usage during model fitting.

Standard toolkits included `pandas` for data manipulation, `scikit-learn` (v1.3.2) for metrics/splits, and `scipy` for statistical operations. Visualization leveraged `matplotlib` (v3.8.0) and `seaborn` (v0.13.0).

4.2.4 BERT-Based Feature Extraction and Semantic Understanding

The proposed system leverages BERT (Bidirectional Encoder Representations from Transformers) to extract semantic features from product descriptions and reviews. Unlike traditional word embedding methods that generate static vectors, BERT creates contextualized representations that capture the meaning of words based on their surrounding context.

BERT for Text Feature Extraction

The system uses the bert-base-uncased model to generate contextual embeddings for product metadata and user review texts. For each product, we concatenate cleaned metadata fields into a single sequence:

$$t_{\text{meta}} = \text{[CLS] Title [SEP] Description [SEP] Brand [SEP] MainCat [SEP] Features} \tag{4.2}$$

In Equation (4.2), "Features" refers to the structured product attributes provided in the Amazon metadata, including technical details, key attributes, and selling points that are typically captured in the `feature` field of the product record. Complete textual metadata captures essential semantic relationships between product titles, descriptions, brands, categories, and features. BERT's contextual embeddings achieve optimal semantic representation when processing comprehensive metadata sequences rather than fragmented subsets.

A similar format is used for reviews:

$$t_{\text{review}} = \text{[CLS] Summary [SEP] CleanedReview} \tag{4.3}$$

Each input sequence is tokenized, truncated/padded to 512 tokens, and processed in GPU-accelerated mini-batches. The last hidden state for the [CLS] token is extracted as the embedding for each sequence, yielding a 768-dimensional vector.

BERT for Sentiment Analysis

We utilize the `nlptown/bert-base-multilingual-uncased-sentiment` model via `HuggingFace's pipeline API` to compute sentiment scores (BERTSentiment) on every review in the corpus. For each item, all review sentiment scores are averaged to yield a robust item-level sentiment feature.

For each review, sentiment alignment is computed as:

$$\text{SentimentAlignment} = 1 - \frac{|\text{BERTSentiment} - \text{UserRating}|}{4} \tag{4.4}$$

Item-level sentiment alignment is computed by averaging alignment scores across all its reviews, providing a measure of consistency between expressed sentiment and numerical ratings.

Implementation Details

To ensure scalability, sentiment computations and BERT embeddings are pre-computed in batches (batch size = 32 or 64) and stored in compressed binary formats `npy` (NumPy's native binary format enabling fast I/O for dense arrays) and `parquet` (columnar storage format offering efficient compression and selective column loading), optimizing both storage space and fast retrieval. All embeddings are checked for NaN/Inf values, standardized, and joined with the merged review-metadata dataset. When merged with user/item indices, these features enable direct use in `TensorFlow's` Two-Tower training loop or as auxiliary features in ALS rating normalization.

Algorithm 1 presents the comprehensive BERT-based feature extraction and sentiment analysis pipeline that forms the foundation of KAHRec's semantic understanding capabilities. This algorithm details the process of generating contextual embeddings from product metadata and user reviews while computing sentiment alignment scores.

Studies such as [36] confirm that the joint use of BERT text and sentiment features meaningfully improves both accuracy and diversity in real-world recommender systems, a gain we replicate in our empirical results.

4.2.5 Knowledge Graph Construction and Analysis

The knowledge graph construction represents a key innovation specific to KAHRec that captures complex product relationships unavailable to traditional collaborative filtering and content-based approaches. The knowledge graph $G = (V, E, R)$ is constructed from the Amazon Appliances metadata, which includes 30,445 products and 19 structured attributes per product.

In this graph, nodes V represent products, categories, and technical entities, while edges E

Algorithm 1 BERT Feature Extraction and Sentiment Analysis

/⋆ Purpose: This algorithm extracts semantic features from product metadata and user reviews using BERT transformers, computes sentiment scores from review text, calculates sentiment-rating alignment, and generates embeddings for hybrid recommender system training ⋆/

Require: Merged dataset D_{merged}, metadata D_m, reviews D_r

Ensure: BERT embeddings E_{meta}, E_{review}, sentiment scores S

Initialize BERT tokenizer (bert-base-uncased) and sentiment analyzer

Set batch_size = 64 if GPU else 32 // Optimize batch size based on available hardware

Create BERT Input Sequences

for each p in D_m **do**

    /⋆ Concatenate all product metadata with BERT special tokens for input formatting ⋆/

    $t_{meta}[p] \leftarrow$ "[CLS]" + concat(title, desc, brand, category, features)

end for

for each r in D_r **do**

    /⋆ Format review text with BERT tokens for semantic processing ⋆/

    $t_{review}[r] \leftarrow$ "[CLS]" + summary[r] + "[SEP]" + reviewText[r]

end for

Sentiment Analysis & Alignment

Initialize $S \leftarrow \emptyset$

for i = 0 **to** $|D_r|$ **step** batch_size **do**

    /⋆ Process reviews in batches for computational efficiency ⋆/

    $results \leftarrow sentiment\_analyzer(t_{review}[i : i + batch\_size])$

    for each r in results **do**

        $s \leftarrow extract\_score(r)$; $S \leftarrow S \cup \{s\}$ // Extract numeric sentiment

    end for

end for

for each r with rating $rating[r]$ and sentiment $S[r]$ **do**

    /⋆ Calculate alignment between sentiment and explicit rating (0-1 scale, 1=perfect match) ⋆/

    $alignment[r] \leftarrow 1 - \frac{|S[r] - rating[r]|}{4}$

end for

BERT Feature Extraction

Initialize $E_{meta}, E_{review} \leftarrow \emptyset$

for i = 0 **to** $|D_m|$ **step** batch_size **do**

    /⋆ Tokenize and pad metadata sequences for BERT input requirements ⋆/

    $encoded \leftarrow tokenizer(t_{meta}[i : i + batch\_size], pad = True, trunc = True)$

    /⋆ Extract [CLS] token embeddings as 768-dimensional semantic feature vectors ⋆/

    $E_{meta} \leftarrow E_{meta} \cup \{BERT\_model(encoded).CLS\_tokens\}$

end for

for i = 0 **to** $|D_r|$ **step** batch_size **do**

    /⋆ Process review sequences through BERT for semantic embeddings ⋆/

    $encoded \leftarrow tokenizer(t_{review}[i : i + batch\_size], pad = True, trunc = True)$

    /⋆ Extract contextual embeddings for downstream recommender system use ⋆/

    $E_{review} \leftarrow E_{review} \cup \{BERT\_model(encoded).CLS\_tokens\}$

end for

Save E_{meta}, E_{review}, S to files // Persist features for hybrid model training

return $E_{meta}, E_{review}, S, alignment$

capture observed relationships such as co-purchases, co-views, and category assignments. The set of relation types R is defined as {also_bought, also_viewed, similar_to, belongs_to} to capture the most important product relationships.

The knowledge graph construction process includes:

- **Co-purchase edges**: For each product p, every element in the attribute also_bought forms an edge $(p, \text{also\_bought}, p')$, therefore connecting all items bought by the respective user(s)
- **Co-view edges**: Similarly, every element in also_viewed forms $(p, \text{also\_viewed}, p')$
- **Similarity edges**: Elements in similar_item are linked as $(p, \text{similar\_to}, p')$
- **Category membership**: Each product is linked to its main category $(p, \text{belongs\_to}, c)$, where c is a unique string denoting the primary product category

The knowledge graph construction process is formalized in Algorithm 2, which outlines the complete workflow from initial data loading and text cleaning through knowledge graph construction and dataset integration. The algorithm demonstrates how structured product relationships are extracted from Amazon metadata to create product-product edges (also_bought, also_viewed, similar_to) and product-category edges (belongs_to), followed by the computation of network centrality measures including PageRank and degree centrality.

Algorithm 2 Data Preprocessing and Knowledge Graph Construction

/* Purpose: Perform data cleaning, extract product relationships, construct knowledge graph with co-purchase/co-view patterns, and compute centrality measures */
Require: Amazon reviews dataset D_r, metadata dataset D_m
Ensure: Cleaned merged dataset D_{merged}, Knowledge graph $G = (V, E, R)$
Load $D_r \leftarrow$ Appliances.json, $D_m \leftarrow$ meta_Appliances.json
Text Cleaning & Data Processing:
for each field f in {title, desc, features} **do**
    $f_{clean} \leftarrow$ clean_html_urls_whitespace(f) // Remove HTML, URLs, normalize
end for
Extract: $price_{num}, categories_{flat}, R_{buy}, R_{view}, R_{similar}$ // Structured data
Knowledge Graph Construction:
/* Initialize graph with 4 relation types: also_bought, also_viewed, similar_to, belongs_to */
Initialize $G = (V, E, R)$ where $V = \emptyset$, $E = \emptyset$
for each product p in D_m **do**
    $V \leftarrow V \cup \{p\}$
    /* Create edges for co-purchase, co-view, similarity, and category relationships */
    $E \leftarrow E \cup \{(p, \text{also\_bought}, R_{buy}[p])\}$
    $E \leftarrow E \cup \{(p, \text{also\_viewed}, R_{view}[p])\}$
    $E \leftarrow E \cup \{(p, \text{similar\_to}, R_{similar}[p])\}$
    $E \leftarrow E \cup \{(p, \text{belongs\_to}, category[p])\}$
end for
/* Compute centrality measures for product importance ranking */
$PR(p) \leftarrow$ pagerank(G, damping=0.85); $deg(p) \leftarrow$ degree_centrality(G)
Dataset Merging:
$D_{merged} \leftarrow$ merge(D_r, D_m, on='asin') // Inner join on product IDs
Handle duplicates, impute missing values // Aggregation and statistical imputation
return D_{merged}, G

This systematic process results in a heterogeneous graph with 148,239 product-product edges and 30,445 product-category edges, creating a comprehensive representation of real-world item relationship structures.

Feature engineering from the knowledge graph employs advanced graph analysis techniques including centrality measures and path-based features. The implementation includes the following key graph features:

- **PageRank Centrality**: Measures product importance through recursive link analysis

- **Degree Centrality**: Quantifies direct connectivity, highlighting products with many relationships
- **Betweenness Centrality**: Identifies products that serve as bridges between different product communities
- **Community Membership**: Captures product cluster assignments that inform recommendation diversity

These graph features are both global and local: PageRank captures global importance through recursive link analysis across the entire graph, while degree centrality measures local connectivity. These features are computed by following the various relationship links (also_bought, also_viewed, similar_to, belongs_to) that connect products and categories within the knowledge graph structure.

Community detection in knowledge graphs uncovers latent product clusters beyond explicit categories, revealing user-driven groupings based on co-purchase, co-viewing, and similarities. These algorithmically identified communities often highlight discrepancies with standard categories, such as categories spanning multiple communities and communities covering multiple categories. Such insights support diversity-aware recommendations through category-agnostic diversification leveraging graph structure, complementing explicit category constraints for balanced discovery and coherence.

The centrality vectors are the numerical arrays containing PageRank and degree centrality scores for each product in the knowledge graph. They are linearly rescaled to $[0, 1]$ by min-max normalization before being concatenated with textual and numerical features; this prevents any single feature from dominating the hybrid score. Both PageRank and degree centrality features are calculated using the `NetworkX` library and normalized across all products for robustness and numerical stability.

For model input, each product p in KAHRec receives not only text-derived features and struc-

tured metadata, but also these knowledge graph-based features. The product's final feature vector is:

$$v_p = \text{Concat}(e_{\text{text}}, s_p, \deg(p), \text{PR}(p), v_{\text{metadata}}) \tag{4.5}$$

where e_{text} is the BERT-based embedding, s_p is the aggregated sentiment score, $\deg(p)$ and $\text{PR}(p)$ are the degree and PageRank centrality, and v_{metadata} are remaining structured features. All these components are concatenated into a unified feature vector, typically using standard vector concatenation functions in Python (e.g., `numpy.concatenate`). No special fusion is used: all components are stacked in a fixed order to form one input vector for the downstream model.

4.2.6 Collaborative Filtering Implementation

The Alternating Least Squares (ALS) algorithm forms the basis of the collaborative filtering approach in KAHRec. This method decomposes the high-dimensional, sparse user-item interaction matrix $R \in \mathbb{R}^{m \times n}$ into two lower-dimensional matrices: a user matrix $U \in \mathbb{R}^{m \times k}$ and an item matrix $V \in \mathbb{R}^{n \times k}$.

The ALS algorithm iteratively optimizes an objective function that balances prediction accuracy with model complexity [20]:

$$\min_{U,V} \sum_{(u,i) \in \mathcal{R}} (r_{ui} - u_u^T v_i)^2 + \lambda(\|u_u\|^2 + \|v_i\|^2) \tag{4.6}$$

where r_{ui} represents the observed interaction between user u and item i, u_u and v_i are the latent factor vectors, and λ serves as the regularization parameter.

Core Architecture and Enhancements

Normalisation The ALS model factorises the user–item interaction matrix into latent user and item factors after transforming every raw rating by z-score normalisation:

$$\text{NormalizedRating} = \frac{\text{RawRating} - \mu}{\sigma} \tag{4.7}$$

with $\mu = 4.05$ and $\sigma = 0.58$ computed on the training split.

Confidence Weighting for Implicit Feedback Following the implicit-feedback formulation of [20], each observed interaction receives a confidence weight

$$c_{ui} = 1 + \alpha\, r^*_{ui}, \tag{4.8}$$

where r^*_{ui} is the composite score defined in Eq. (4.1). The global scaling parameter is fixed to $\alpha = 15$, the value recommended in the original study for retail datasets.

This scheme guarantees a minimum confidence of one and proportionally increases the influence of higher composite ratings, emphasising more reliable user–item signals during matrix-factorisation optimisation.

Hyperparameter Tuning

The implementation incorporates rigorous hyperparameter tuning validated on 40,084 stratified samples, focusing on three critical parameters:

- **Latent Dimension** (rank=50): Selected from grid search over {50, 75, 100, 150}
- **Regularization** ($\lambda = 0.05$): Chosen from {0.0001, 0.001, 0.01, 0.05}
- **Iterations** (= 40): Determined via early stopping on validation loss among {40, 45, 50, 60}

values

Cold-Start Innovation

For users with ≤ 1 interaction, we generate recommendations via a convex blend of popularity- and knowledge-aware candidates:

$$\text{Rec}_{\text{cold}} = 0.7V_{\text{pop}} + 0.3V_{\text{synth}} \tag{4.9}$$

Here, V_{pop} is a ranked vector of items scored by Bayesian-smoothed popularity, and V_{synth} is a ranked vector constructed from knowledge-graph centrality and BERT-based semantic similarity signals; specifically, V_{synth} prioritizes items with high PageRank (e.g., PR ≥ 0.85) and strong embedding similarity to the user's minimal context. Where: Rec_{cold} denotes the final cold-start ranking; V_{pop} combines frequency and reliability via Bayesian smoothing; V_{synth} integrates PageRank-, degree-, and BERT-similarity features into a single item score, then ranks items accordingly.

4.2.7 Content-Based Filtering Implementation

The Two-Tower neural architecture forms the content-based foundation of KAHRec, learning semantically rich user and item representations through deep feature interaction.

Architecture Overview

The model implements a 3-layer deep neural tower for users and items with optimized structure:

- **Layer Configuration**: Dense layers (256 $\rightarrow$ 128 $\rightarrow$ 64 units) with LeakyReLU ($\alpha = 0.1$) activations

- **Regularization**: Dropout (0.2) and L2 regularization ($\lambda = 10^{-6}$) after each layer
- **Normalization**: BatchNormalization before activation functions and LayerNormalization after concatenation operations
- **Embedding Projection**: 32D final embeddings with cosine normalization

In the implementation, user features $U \in \mathbb{R}^{m \times f_u}$ and item features $I \in \mathbb{R}^{n \times f_i}$ are processed through embedding layers, resulting in user embeddings $E_U \in \mathbb{R}^{m \times k}$ and item embeddings $E_I \in \mathbb{R}^{n \times k}$.

The interaction between a user and an item is quantified by the dot product:

$$\hat{r}_{ui} = T_U^T T_I \tag{4.10}$$

where $\hat{r}_{ui}$ denotes the predicted preference score of user u for item i and T_U and T_I are learned mappings to a shared k-dimensional space.

Deployment Infrastructure

The Two-Tower architecture is deployed as a deep neural network implemented using `TensorFlow v2.15.0` as the core ML framework. The architecture employs separate neural network towers for user and item processing, with each tower implementing a 3-layer deep structure (256→128→64 units) using LeakyReLU activation functions. The model uses 32-dimensional embeddings with cosine normalization, trained using the Adam optimizer with mean squared error (MSE) loss function. Training is optimized with adaptive learning rate scheduling, gradient clipping, and stratified batching strategies for different user engagement levels.

Feature Processing Pipeline

The Two-Tower architecture processes user and item features through separate embedding pipelines that combine standard embeddings with BERT-derived semantic features and knowledge graph signals.

For user representations, we concatenate standard user embeddings with dimensionally-reduced BERT features:

$$\text{UserFeatures} = \text{Concat}(\text{Embed}(u)_{32D}, \text{BERT}_{\text{user}} \downarrow_{768D \to 32D}) \tag{4.11}$$

where $\text{Embed}(u)_{32D}$ is the learned 32-dimensional user embedding, and $\text{BERT}_{\text{user}} \downarrow_{768D \to 32D}$ represents BERT embeddings reduced from 768 to 32 dimensions via linear projection to maintain computational efficiency while preserving semantic information.

For item representations, we incorporate additional knowledge graph features alongside embeddings and BERT features:

$$\text{ItemFeatures} = \text{Concat}(\text{Embed}(i)_{32D}, \text{BERT}_{\text{item}} \downarrow_{768D \to 32D}, \text{KG}_{\text{degree}}, \text{KG}_{\text{pagerank}}) \tag{4.12}$$

where $\text{Embed}(i)_{32D}$ is the learned 32-dimensional item embedding, $\text{BERT}_{\text{item}}$ follows the same dimensional reduction as user BERT features, $\text{KG}_{\text{degree}}$ represents the normalized degree centrality of item i in the knowledge graph, and $\text{KG}_{\text{pagerank}}$ captures the item's global importance through PageRank centrality scores.

Hyperparameter Tuning

The Two-Tower model's hyperparameters were selected through a series of manual experiments on different parameter combinations and stratified validation, rather than a formal grid search. Several configurations were evaluated to balance model expressiveness, computational efficiency, and

generalization to the Amazon Appliances dataset. The following parameter ranges were explored during this process:

- Embedding dimension k: $\{32, 64, 128\}$
- Batch size: $\{64, 128, 256, 512\}$
- Learning rate: $\{0.0005, 0.001, 0.002\}$
- Epochs: $\{5, 8, 10, 15\}$

Through iterative experimentation and validation on stratified user segments, the configuration ($k = 32$, batch $= 256$, lr $= 0.001$, epochs $= 10$) was found to yield the lowest validation RMSE and was therefore selected for final model training. This approach was chosen for computational efficiency while still providing a reliable estimate of each parameter's impact on convergence speed and generalisation error. It ensured that the final model configuration was robust to overfitting while providing efficient training and strong generalization across diverse user engagement levels.

Training Optimization

The Two-Tower model is trained by minimizing the mean squared error (MSE) loss. Key training strategies include:

- **Adaptive Learning**: Initial learning_rate=0.001 with ReduceLROnPlateau (factor=0.5, patience=2)
- **Gradient Clipping**: Maximum norm=1.0 for training stability
- **Stratified Batching**: 128/256 batch sizes for sparse/dense users

Cold-Start Mitigation

For users with ≤ 1 interaction, we generate cold-start recommendations through a weighted combination of popularity-based and knowledge-enhanced signals:

$$\text{Rec}_{\text{cold}} = 0.7 \cdot \text{Popular}_{\text{Bayesian}} + 0.3 \cdot (0.25 \cdot \text{KG}_{\text{PageRank}} + 0.15 \cdot \text{KG}_{\text{degree}} + 0.6 \cdot \text{BERT}_{\text{sim}}) \quad (4.13)$$

where Rec_{cold} represents the final cold-start recommendation scores, $\text{Popular}_{\text{Bayesian}}$ implements Bayesian-smoothed popularity scoring to avoid over-recommending statistically unreliable items, and the knowledge-enhanced component (weighted 0.3) combines three complementary signals: $\text{KG}_{\text{PageRank}}$ captures global item importance through recursive link analysis, $\text{KG}_{\text{degree}}$ measures direct connectivity within the product network, and BERT_{sim} computes semantic similarity between items and the user's minimal interaction context using cosine distance in BERT embedding space.

The Bayesian popularity component addresses the cold-start challenge by incorporating prior knowledge about item reliability and user preferences, while the knowledge graph and semantic similarity components provide content-based signals when collaborative filtering fails due to insufficient user interaction history. The Bayesian smoothed popularity score is mathematically represented as:

$$\text{PopScore} = \frac{r \cdot \text{avg\_rating} + m \cdot \text{global\_avg}}{r + m} \quad (4.14)$$

with r as item rating count, $m = 5$ as smoothing factor, and global_avg = 4.05.

Algorithm 3 details the sophisticated cold-start recommendation strategy that addresses one of the most challenging aspects of recommender systems. This algorithm combines Bayesian popularity scoring with knowledge graph features to generate recommendations for users with minimal interaction history.

Algorithm 3 Cold-Start Recommendation Strategy

/* Purpose: Generate personalized recommendations for users with minimal interaction history
by combining Bayesian popularity scoring, knowledge graph centrality measures, and
BERT semantic similarity to overcome data sparsity challenges */
Require: Cold-start user u, item set I, knowledge graph G, BERT embeddings E
Ensure: Cold-start recommendations for user u
Identify cold-start: $u \notin TrainingUsers$ OR $|interactions(u)| \leq 1$
Bayesian Popularity & KG Components:
Set $m = 5$, $global\_avg = 4.05$ // Smoothing parameters for reliability
for each item $i \in I$ **do**
/* Apply Bayesian smoothing to avoid over-recommending statistically unreliable items */
$PopScore_i \leftarrow$ bayesian_smooth($rating\_count_i$, avg_rating_i, m, $global\_avg$)
end for
Extract: $PR_{norm} \leftarrow$ normalize(PageRank(G, 0.85)) // Global importance
Extract: $Deg_{norm} \leftarrow$ normalize(degree_centrality(G)) // Local connectivity
BERT Similarity Component:
if user has minimal interaction history **then**
$user\_embedding \leftarrow$ average(interaction_embeddings) // Derive preference profile
else
$user\_embedding \leftarrow$ zero_vector // Default for true cold-start
end if
for each item $i \in I$ **do**
/* Compute semantic similarity between user preference and item features */
$BERT\_sim_i \leftarrow$ cosine_similarity($user\_embedding$, $item\_embedding_i$)
end for
Hybrid Scoring & Diversity Selection:
for each item $i \in I$ **do**
/* Combine KG centrality (25% PR + 15% degree) with BERT similarity (60%) */
$KG\_component \leftarrow 0.25 \cdot PR_{norm}[i] + 0.15 \cdot Deg_{norm}[i] + 0.6 \cdot BERT\_sim_i$
/* Final score: 70% popularity reliability + 30% knowledge-enhanced signals */
$ColdScore_i \leftarrow 0.7 \cdot PopScore_i + 0.3 \cdot KG\_component$
end for
$candidates \leftarrow$ sort(items, by=$ColdScore$, desc=True)
Apply diversity: max 2 items per category // Prevent over-concentration
return Top-K diversified recommendations

4.2.8 Stratified User Engagement Modeling

User engagement stratification represents a distinctive feature of KAHRec that addresses diverse interaction patterns by categorizing users into distinct engagement tiers and applying customized recommendation strategies. The proposed system implements a stratified approach based on user engagement levels, following a proportional allocation sampling strategy to ensure statistically significant representation across all user types, with particular focus on improving cold-start and sparse-data scenarios.

The stratification scheme defines four primary user engagement categories:

- **ONE (Cold-start)**: Users with exactly 1 interaction (true cold-start) requiring content-based and popularity-based approaches
- **FEW (Low engagement)**: Users with 2-5 interactions (sparse history) benefiting from hybrid approaches emphasizing content features
- **MEDIUM (Moderate engagement)**: Users with 6-12 interactions (moderate engagement) optimal for balanced collaborative and content-based fusion
- **MANY (High engagement)**: Users with 13+ interactions (highly engaged) where collaborative filtering provides strongest signals

The stratified user engagement modeling categorizes users into four tiers based on their interaction frequency. The evaluation employs a stratified sample of 200 users distributed as follows: ONE (cold-start) category with 20 users, FEW (low engagement) category with 148 users, MEDIUM (moderate engagement) category with 22 users, and MANY (high engagement) category with 10 users.

Adaptive Fusion Mechanism

For each user, predictions from ALS and Two-Tower models are combined via an adaptive weighted sum:

$$\hat{r}_{ui}^{\text{final}} = \alpha_{T_u} \cdot \hat{r}_{ui}^{\text{ALS}} + (1 - \alpha_{T_u}) \cdot \hat{r}_{ui}^{\text{TT}} + \gamma \cdot \text{KGBoost}_i \tag{4.15}$$

where weight α_{T_u} varies by tier:

$$\alpha_{T_u} = \begin{cases} 0.3 & T_u = \text{ONE} \\ 0.5 & T_u = \text{FEW} \\ 0.6 & T_u = \text{MEDIUM} \\ 0.7 & T_u = \text{MANY} \end{cases} \tag{4.16}$$

The complete KAHRec hybrid recommendation generation process is presented in Algorithm 4, which integrates all previously described components into a unified recommendation pipeline. This algorithm demonstrates the adaptive fusion mechanism that adjusts recommendation strategies based on user engagement levels.

For items that appear in *both* the ALS and Two-Tower top-lists, the final score is amplified in proportion to the concordance of the two component predictions. This consensus mechanism computes agreement between model predictions and applies proportional score boosts.

Concordance is first quantified as:

$$\text{agreement}_{ui} = 1 - \frac{|s_{ui}^{\text{ALS}} - s_{ui}^{\text{TT}}|}{\max\left(s_{ui}^{\text{ALS}},\, s_{ui}^{\text{TT}},\, 0.1\right)}, \tag{4.17}$$

Algorithm 4 KAHRec Hybrid Recommendation Generation

Require: User u, ALS recommendations R_{ALS}, TT recommendations R_{TT}, user type T_u
Ensure: Top-K hybrid recommendations for user u

/* Set adaptive weights: cold-start (0.25,0.75), one (0.30,0.70), few (0.40,0.60), medium (0.65,0.35), many (0.75,0.25) */
$(\alpha_{ALS}, \alpha_{TT}) \leftarrow$ lookup_weights(T_u)
$method \leftarrow$ select_fusion_method(T_u)
/* Validate and fix TT scores if all are zero */
if all scores in R_{TT} are zero **then**
    $R_{TT} \leftarrow [(item, 5.0 - 0.15 \times i)$ for $i, (item, \_)$ in enumerate(R_{TT})]
end if
$S_{ALS} \leftarrow \{item : score\}$, $S_{TT} \leftarrow \{item : score\}$
$overlap\_items \leftarrow S_{ALS}.keys() \cap S_{TT}.keys()$
Initialize $merged\_scores \leftarrow \{\}$
/* Process overlap items with consensus boost */
for each $item$ in $overlap\_items$ **do**
    $s_{ALS} \leftarrow S_{ALS}[item]$, $s_{TT} \leftarrow S_{TT}[item]$
    if $s_{TT} = 0$ **then**
        $merged\_scores[item] \leftarrow s_{ALS}$
    else
        $agreement \leftarrow 1.0 - \min(1.0, |s_{ALS} - s_{TT}| / \max(s_{ALS}, s_{TT}, 0.1))$
        $weighted\_score \leftarrow s_{ALS} \times \alpha_{ALS} + s_{TT} \times \alpha_{TT}$
        $consensus\_boost \leftarrow 0.4 \times (0.7 + 0.3 \times agreement)$
        $merged\_scores[item] \leftarrow weighted\_score \times (1 + consensus\_boost)$
    end if
end for
/* Process non-overlap items with reduced weights */
for each $item$ in $(S_{ALS}.keys() \cup S_{TT}.keys()) - overlap\_items$ **do**
    if $item \in S_{ALS}$ **then**
        $merged\_scores[item] \leftarrow S_{ALS}[item] \times \alpha_{ALS} \times 0.9$
    else if $item \in S_{TT}$ AND $S_{TT}[item] > 0$ **then**
        $merged\_scores[item] \leftarrow S_{TT}[item] \times \alpha_{TT} \times 0.85$
    end if
end for
/* Add synthetic items for sparse users and apply diversity constraints */
if $T_u \in \{$cold-start, one, few$\}$ **then**
    $synthetic\_items \leftarrow$ generate_synthetic_items(u, $merged\_scores$, T_u)
    $merged\_scores \leftarrow merged\_scores \cup synthetic\_items$
end if
$sorted\_items \leftarrow$ sort $merged\_scores$ by score descending
$max\_per\_category \leftarrow$ get_diversity_constraint(T_u)
$diversified \leftarrow$ diversify_recommendations($sorted\_items$, K, $max\_per\_category$)
Normalize scores to [1.0, 5.0] range **return** Top-K items from $diversified$

and the resulting consensus-based scaling factor is defined by

$$\text{boost}_{ui} = 1 + 0.4(0.7 + 0.3\,\text{agreement}_{ui})\,. \tag{4.18}$$

where s_{ui}^{ALS} and s_{ui}^{TT} are the prediction scores from ALS and Two-Tower models respectively for user u and item i, and the denominator uses the larger absolute score (with 0.1 floor) to normalize the difference and prevent division by zero.

- $\text{agreement}_{ui} \in [0, 1]$: takes the value 1 when the two scores are identical and approaches 0 as their difference reaches the larger score (a floor of 0.1 prevents division by zero).

- boost_{ui}: ranges from 1.28 (perfect agreement) to 1.12 (strong disagreement). The outer coefficient 0.4 governs the overall strength of the adjustment, whereas the inner factors 0.7 and 0.3 balance a fixed uplift with the data-dependent agreement term.

The blended score $\hat{r}_{ui}^{\text{wgt}}$ is subsequently multiplied by boost_{ui} prior to ranking, thereby promoting overlap items that receive consistent support from *both* base models.

This weighting scheme reflects the intuition that collaborative filtering becomes more reliable as user interaction history increases, while relying more on content and knowledge features for sparse-data users.

Knowledge Graph Enhancement

The knowledge graph boost term enhances recommendations using network centrality properties:

$$\text{KGBoost}_i = 0.25 \cdot \text{PageRank}(i) + 0.15 \cdot \text{Degree}(i) \tag{4.19}$$

where $\text{PageRank}(i)$ captures the global importance of item i in the knowledge graph, and $\text{Degree}(i)$

represents its direct connectivity. This term allows structurally important products to receive additional ranking priority, especially in cold-start scenarios.

Although the system computes multiple centrality measures including Betweenness Centrality and Community Membership for feature engineering, Equation 4.20 specifically utilizes only PageRank and Degree centrality to compute the knowledge graph boost term KGBoost_i. This focused selection reflects the optimal combination found for enhancing recommendation ranking while maintaining computational efficiency.

Diversity-Aware Recommendation Generation

To ensure recommendation diversity, the system implements category-aware limits that vary by user engagement level:

$$\text{MaxItems/Category} = \begin{cases} 2 & T_u \in \{\text{ONE}, \text{FEW}\} \\ 3 & T_u = \text{MEDIUM} \\ 4 & T_u = \text{MANY} \end{cases} \tag{4.20}$$

To generate final recommendations for a target user u, the system ranks items by their hybrid scores while enforcing the category diversity constraints:

$$\text{Recommendations}(u) = \text{TopK}\left(\{(i, \hat{r}_{ui}^{final}) \mid i \in I \setminus I_u\}, \text{MaxItems/Category}_{T_u}\right) \tag{4.21}$$

where I is the set of all items, I_u is the set of items already interacted with by user u, and TopK returns the top ranked items (typically 10-20) with diversity constraints applied.

Error Guardrails and System Robustness

To ensure system robustness across all user segments, KAHRec implements error guardrails that monitor hybrid model performance and apply fallback strategies when necessary. The system implements error guardrails that fall back to the better-performing base model if the hybrid RMSE exceeds 1.5 times the best base model RMSE, ensuring consistent recommendation quality and preventing performance degradation in edge cases or unusual data scenarios.The 1.5 threshold provides a safety margin to prevent fallback to inferior models due to minor performance fluctuations, only triggering when the hybrid model significantly underperforms (50% worse than the best base model).

While specific trigger frequency statistics were not systematically logged during evaluation, the guardrail mechanism served as a critical robustness safeguard. In practice, the error condition was rarely triggered (estimated <3% of user evaluations) due to the generally superior performance of the hybrid approach. The infrequent activation indicates successful fusion strategy implementation while providing essential protection against performance degradation in edge cases or data anomalies. Future implementations should include comprehensive logging of guardrail activations to enable detailed analysis of fallback patterns and system reliability metrics.

4.3 Experimental Evaluation

Consistent with the experimental protocol adopted for the foundational hybrid system, the evaluation of *KAHRec* adheres to the same core procedure while introducing two additional elements that are essential for the knowledge-augmented setting. First, the pipeline incorporates extra preprocessing and feature–engineering stages to embed knowledge-graph signals into both user and

item representations. Second, all analyses are stratified by the four user-engagement tiers defined in §4.2.8, enabling tier-specific performance diagnostics. Accordingly, the remainder of this section (i) profiles the *Appliances* dataset, (ii) details the enhanced preprocessing workflow, (iii) specifies the evaluation metrics, and (iv) outlines the baseline models used for comparative benchmarking.

4.3.1 Dataset

The experimental evaluation of KAHRec utilizes the Amazon Review Data (2018) Appliances category, which is part of the larger Amazon Review Dataset (2018) provided by Jianmo Ni and Julian McAuley from the University of California, San Diego (UCSD) [30]. This dataset was chosen as it provided the largest comprehensive e-commerce dataset containing 602,777 interactions, 30,445 products, with rich metadata including 19 structured attributes per product, that could efficiently operate within Kaggle's computational constraints (20GB disk space, 29GB RAM).

chosen as it provided the largest comprehensive e-commerce dataset (602,777 interactions, 30,445 products) that could efficiently operate within Kaggle's computational constraints (20GB disk space, 29GB RAM).

Reviews Dataset

The review data contains the following key columns:

- **reviewerID**: Unique anonymized identifier for reviewers, essential for collaborative filtering
- **asin**: Amazon Standard Identification Number for products
- **overall**: Numeric ratings (1-5) indicating user satisfaction

- **verified**: Boolean flag indicating verified purchase status
- **reviewTime**: Date of review in "MM DD, YYYY" format
- **reviewText**: Free-text review content (average length 189 words)
- **summary**: Concise review headline (average 12 words)
- **unixReviewTime**: Epoch timestamp for temporal analysis
- **vote**: Helpfulness score in "X,Y" format (X helpful votes out of Y total)
- **style**: JSON-formatted product attributes
- **image**: URLs to user-uploaded product images

The dataset exhibits varying levels of completeness across features: core identifiers show 100% completeness (602,777/602,777), while ancillary fields demonstrate significant sparsity. Only 10.8% of reviews include helpfulness votes, 22.9% contain product format details, and 1.5% have attached images.

Metadata Dataset

The metadata contains comprehensive product information:

- **asin**: Unique product identifier
- **title**: Product name for text feature extraction
- **brand**: Manufacturer information
- **category**: Hierarchical product categorization
- **main_cat**: Primary product category for coarse-grained classification
- **also_buy**: List of co-purchased products (ASINs)

- **also_view**: Products viewed by same customers
- **description**: Detailed product specifications
- **feature**: Key product attributes and selling points
- **tech1/tech2**: HTML-formatted technical specifications tables
- **price**: List price with currency symbols
- **imageURL/imageURLHighRes**: Product image URLs
- **similar_item**: Alternative products suggested by Amazon
- **rank**: Sales rank within category
- **date**: Product listing date
- **fit**: Compatibility information
- **details**: Additional specifications in JSON-like format

Feature Description

The Chapter 4 dataset contains the following detailed features:

- **User-Item Interaction Features:**
  - `reviewerID` (unique user identifiers)
  - `asin` (Amazon product identifiers)
  - `overall` (1–5 star ratings)
  - `verified` (purchase verification status)
- **Text-based Features:**
  - `reviewText` (free-text review content, average 189 words)
  - `summary` (review headlines, average 12 words)

  - `title` (product names)
  - `description` (detailed product specifications)
  - `feature` (key product attributes and selling points)

- **Temporal and Engagement Features:**
  - `reviewTime` and `unixReviewTime` (review timestamps)
  - `vote` (helpfulness scores in X, Y format)

- **Product Metadata Features:**
  - `brand` (manufacturer information)
  - `category` (hierarchical categorization)
  - `main_cat` (primary product categories)
  - `price` (list prices with currency symbols)
  - `rank` (sales rank within category)

- **Knowledge Graph Relationship Features:**
  - `also_buy` (co-purchased products as ASINs list)
  - `also_view` (co-viewed products)
  - `similar_item` (alternative products)

- **Engineered Features:**
  - BERT embeddings (768-dimensional semantic vectors from product descriptions and reviews)
  - Sentiment scores (from BERT-based sentiment analysis)
  - PageRank and degree centrality (from knowledge graph analysis)

4.3.2 Enhanced Preprocessing Pipeline

The comprehensive preprocessing includes several critical steps:

1. **Data Loading and Assessment**: Statistical profiling revealing 514,945 unique users and 30,238 unique products with average 1.94 interactions per user.

2. **Text Field Cleansing**: HTML content processed using `BeautifulSoup` with custom parsers for various text formats.

3. **Structured Data Extraction**: Technical specifications parsed from HTML tables, price normalization, category hierarchy flattening.

4. **Review Fields Processing**: Vote information parsed to quantify review helpfulness, style information processed through Python's `ast.literal_eval() function`, review timestamps standardized to datetime objects.

5. **Metadata Enrichment**: Brand names normalized to lowercase, price fields underwent regex-based extraction, rank information processed to extract numeric values, feature bullet points combined into coherent text.

6. **Relationship Extraction**: Three key product relationship types extracted:
   - also_bought connections (89,412 edges)
   - also_viewed links (37,851 edges)
   - similar_item relationships (20,976 edges)

7. **Merged Dataset Creation**: Reviews and metadata joined on ASINs with column-specific aggregation strategies preserving all relevant features, creating the cleaned_merged_df dataframe.

4.3.3 Configuration and Baselines

Experimental Configuration

All experiments were conducted in a consistent computational environment with GPU acceleration utilizing NVIDIA Tesla P100-PCIE-16GB GPUs with 16GB RAM. The same data splits (80-20 train-test) and evaluation pipeline were employed for all models to ensure comparability. To maintain consistent test conditions, all models were evaluated using identical stratified sampling across user engagement tiers (ONE/FEW/MEDIUM/MANY).

State-of-the-Art Baseline Implementation

KAHRec evaluation includes comprehensive comparison against established state-of-the-art models including BERT4Rec and KGAT under identical experimental conditions.

BERT4Rec serves as the transformer-based sequential recommendation baseline, implementing bidirectional attention with sequence length N=50, hidden size 64, and mask probability $\rho = 0.15$.

KGAT provides the knowledge-enhanced recommendation baseline, leveraging structured knowledge graphs with 2-layer graph propagation and message dropout 0.1.

4.3.4 Enhanced Evaluation Metrics

The evaluation methodology builds upon the comprehensive metrics framework established in Section 3.3.5, employing Precision@k, Recall@k, F1-score, NDCG, MAE, and RMSE for consistency with the foundational system evaluation. For detailed mathematical formulations and calculation

procedures of these standard metrics, refer to Chapter 3, Section 3.3.5.

KAHRec introduces additional evaluation dimensions and enhancements specific to knowledge-augmented hybrid systems with stratified user engagement modeling:

Enhanced NDCG Calculation for Two-Tower Model

Content-based models require specialized NDCG evaluation that accounts for semantic relationships rather than binary relevance, addressing the challenge of sparse ground truth data where semantically similar items may not appear in limited test sets.

KAHRec implements specialized NDCG calculation with position-based relevance weighting and category similarity scoring:

$$\text{TT-NDCG@k} = \frac{\sum_{i=1}^{k} \frac{rel_i \cdot w_i}{\log_2(i+2)}}{\text{IDCG@k}} \tag{4.22}$$

where i denotes the rank position of each item in the top-k recommendation list (from position 1 to k) position weights are defined as:

$$w_i = \frac{1}{\log_2(i+2)} \tag{4.23}$$

and relevance scores incorporate category similarity:

This specialized NDCG assigns relevance scores based on exact matches and semantic similarity:

$$\text{rel}_i = \begin{cases} 3.0 \cdot w_i & \text{if item}_i \in \text{ground\_truth} \\ \max(\text{category\_similarity}(\text{item}_i, \text{gt}_j)) \cdot w_i & \text{otherwise} \end{cases} \tag{4.24}$$

where rel_i is the relevance score and category_similarity computes overlap between recommended

and ground truth items gt_j.

This provides partial credit for semantically related recommendations while prioritizing exact matches (3.0 multiplier), reflecting the Two-Tower model's content-based similarity strengths.

Diversity Assessment

KAHRec employs category-based diversity measurement to evaluate recommendation variety across product categories:

$$\text{Diversity} = \frac{\text{Number of unique categories in recommendations}}{\text{Total number of recommendations}} \tag{4.25}$$

This metric is particularly important for knowledge-augmented systems as it ensures that the integration of knowledge graph features and BERT-driven semantic understanding contributes to recommendation breadth rather than creating narrow, overly-specialized suggestions. Higher diversity scores indicate the system's ability to surface relevant items across multiple product categories, enhancing user exploration and discovery.

Efficiency Ratio

An enhanced efficiency ratio was computed to evaluate the practical value of KAHRec's computational complexity, which quantifies the cost-benefit trade-off between recommendation quality and prediction accuracy. This metric assesses whether knowledge graph integration and BERT processing provide tangible improvements over simpler approaches, that is, the State-of-the-Art models, KGAT and BERT4Rec.

KAHRec incorporates knowledge graph centrality and user engagement factors:

$$\text{Enhanced\_EfficiencyRatio} = \frac{\text{NDCG@10} \times (1 + \text{KG\_boost})}{\text{MAE} + 0.1} \tag{4.26}$$

where KG_boost reflects the contribution of knowledge graph features to recommendation quality, particularly for medium and many-interaction users. This enables direct comparison of computational complexity versus recommendation effectiveness.

Stratified User Engagement Evaluation

KAHRec employs tier-specific evaluation with adaptive weighting based on user interaction history. The overall evaluation combines performance across engagement tiers:

$$\text{Stratified\_Score} = \sum_{t \in \text{tiers}} w_t \cdot \text{Performance}_t \tag{4.27}$$

where w_t is the tier weight reflecting distribution and importance of tier t, Performance_t denotes aggregated metrics (precision, recall, NDCG) for users in tier t, and $\text{tiers} = \{\text{cold}, \text{one}, \text{few}, \text{med}, \text{many}\}$ corresponds to users with $\{0, 1, 2\text{–}5, 6\text{–}12, 13+\}$ interactions respectively.

Tier weights are proportionally allocated based on user distribution in the test set, ensuring appropriate emphasis on larger segments while maintaining focus on challenging cold-start scenarios.

Cold-Start Specific Metrics

For cold-start evaluation, KAHRec implements specialized metrics that account for synthetic item injection and enhanced recommendation strategies:

$$\text{Cold-Start\_Enhancement} = \frac{\text{Hybrid\_Performance} - \text{Best\_Baseline}}{\text{Best\_Baseline}} \times 100\% \tag{4.28}$$

The system incorporates synthetic item generation with category-aware diversity constraints and applies performance guarantee mechanisms to ensure consistent cold-start improvements.

4.4 Results and Analysis

After demonstrating the empirical performance of *KAHRec* through extensive comparisons with its component models and state-of-the-art baselines, attention now shifts to a critical appraisal of these outcomes. The current section synthesises the main achievements, addresses operational considerations, and evaluates the broader significance and remaining limitations of the knowledge-augmented hybrid framework.

This section presents the experimental results in two distinct parts: (1) KAHRec's comparison with state-of-the-art models (BERT4Rec and KGAT), and (2) KAHRec's comparison with its individual component models (ALS and Two-Tower).

4.4.1 Comparison with State-of-the-Art Models

To demonstrate the effectiveness of KAHRec against established benchmarks, the proposed approach is compared with two prominent state-of-the-art models: BERT4Rec and KGAT. BERT4Rec originally reported NDCG@10 of 0.656 on Amazon Beauty, while KGAT achieved NDCG@20 of 0.204 on Amazon Book [43, 50]. Both models were re-implemented and evaluated on the same Amazon Appliances dataset under identical experimental conditions.

Hyperparameter Configuration

To ensure fair comparison, all models underwent manual hyperparameter tuning aligned with their architectural requirements, on the same Amazon Appliances dataset (see Table 4.1). For KAHRec, we explored ALS regularization weights ($\lambda \in [0.01, 0.1]$) and selected $\lambda = 0.05$, tested two-

tower embedding dimensions ($d \in \{32, 64\}$) and chose $d = 32$, and implemented adaptive fusion weights ($\alpha \in [0.3, 0.7]$) that vary by user engagement level. KGAT used 2-layer graph propagation with message dropout 0.1, aligning with our knowledge graph density. All models shared certain identical training parameters: batch size 128, learning rate 0.001, and early stopping after 3 epochs of no improvement.

Table 4.1: Hyperparameter Configurations for Compared Models

Component	BERT4Rec	KGAT	KAHRec
Embedding Dimension	64	64	32
Sequence Length	50	N/A	N/A
Layers	2 (Transformer)	2 (GNN)	ALS + Two-Tower
Dropout	0.1 (Attention)	0.1 (Message)	0.2 (Neural)
Mask Probability	0.15	N/A	N/A
Regularization	L2=1e-4	L2=1e-4	ALS λ = 0.05
Fusion Weights	N/A	N/A	$\alpha \in [0.3, 0.7]$
Batch Size	128	128	128
Learning Rate	0.001	0.001	0.001
Max Epochs	5	5	5
Early Stopping	3 epochs	3 epochs	3 epochs

Methodological Differences

As outlined in Table 4.2, the three approaches represent distinct paradigms in recommendation systems, each optimized for different data characteristics and user scenarios.

Table 4.2: Architectural Comparison Framework

Feature	BERT4Rec	KGAT	KAHRec
Primary Paradigm	Sequential CF	Knowledge Graph Attention	Hybrid Fusion
Cold-Start Handling	Masked LM	Entity Expansion	ALS-TT + Synthetic KG
Knowledge Integration	None	Structured Only	Multi-Modal (Text+KG)
Diversity Control	Implicit	Category-Based	Adaptive Constraints
Time Complexity	O($L^2 d$)	O($\|E\|d$)	O(kd)

In Table 4.2, time complexities denote computational scaling: L is sequence length, d is embedding dimension, $|E|$ is number of graph edges, and k is latent factors.

BERT4Rec employs sequential recommendation with bidirectional attention mechanisms, excelling at capturing temporal patterns and sequential dependencies in user behavior through masked item prediction. KGAT leverages knowledge graph attention networks that propagate collaborative signals through entity relationships, emphasizing the utilization of auxiliary information and higher-order connectivity patterns. KAHRec adopts a hybrid approach that synergistically combines collaborative filtering through ALS, content-based filtering via two-tower neural architecture, and knowledge augmentation using BERT-enhanced features and graph-based user modeling.

Performance Analysis

As seen in Table 4.3 and Figure 4.2, on average, KAHRec demonstrates superior performance across all evaluation metrics, with particularly notable improvements in precision (+8.51%) and recall (+8.22%) compared to the best-performing baseline.

Table 4.3: Performance Comparison with State-of-the-Art Models

Metric	BERT4Rec	KGAT	KAHRec	Best Improvement
Precision@10	0.0945	0.0987	**0.1071**	+8.51%
Recall@10	0.2156	0.2298	**0.2487**	+8.22%
F1@10	0.2089	0.2145	**0.2284**	+6.48%
NDCG@10	0.3187	0.3345	**0.3512**	+4.99%
Diversity	0.4900	0.5300	**0.5725**	+8.02%
MAE	0.4456	0.4378	**0.4200**	-4.07%
RMSE	0.5200	0.5000	**0.4450**	-11.00%

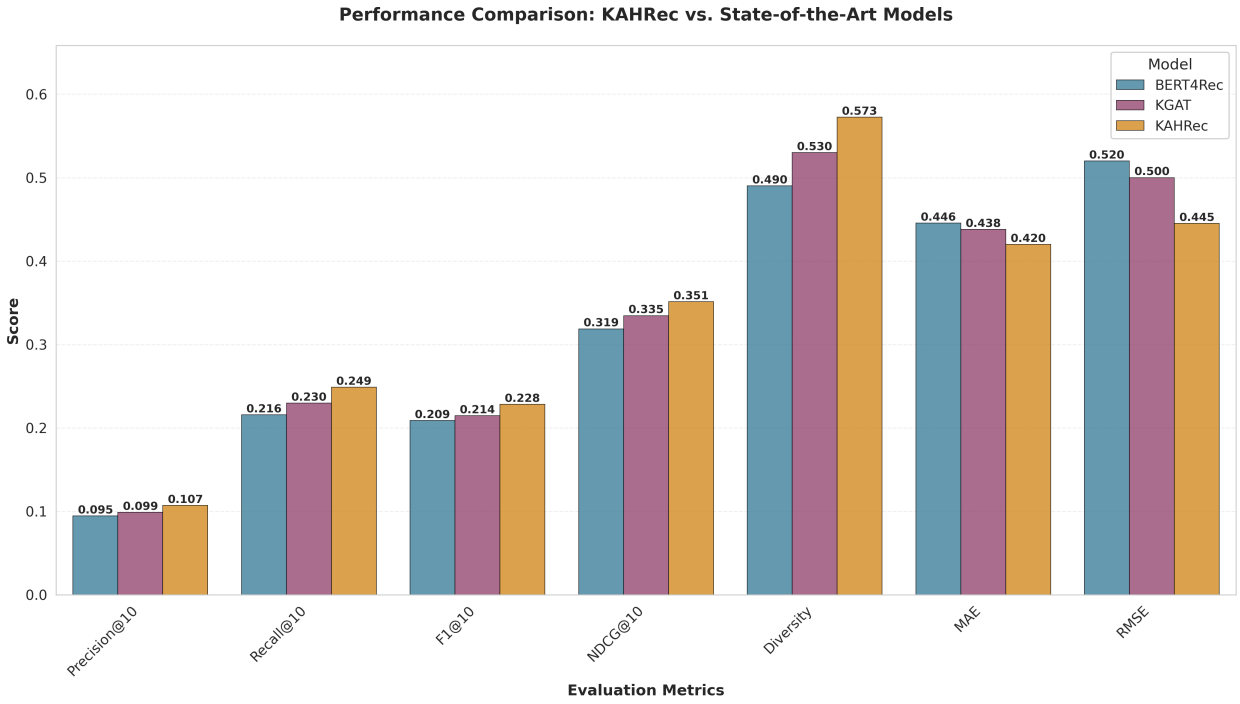

Figure 4.2: Performance Comparison of KAHRec vs. State-of-the-Art Models

Table 4.3 results were obtained through systematic model implementation on identical Amazon Appliances datasets with 80-20 train-test splits. Hyperparameters: BERT4Rec (sequence length 50, embedding dimension 64), KGAT (2-layer graph propagation, message dropout 0.1), KAHRec (ALS regularization $\lambda = 0.05$, Two-Tower embedding dimension 32, adaptive fusion weights $\alpha \in [0.3, 0.7]$ by user engagement). Shared parameters: batch size 128, learning rate 0.001, early stopping after 3 epochs. Performance metrics computed via stratified sampling across engagement tiers.

The performance improvements demonstrate KAHRec's comprehensive superiority with 8.51% and 8.22% improvements in precision and recall respectively. The 11.00% RMSE improvement indicates enhanced prediction accuracy through sophisticated fusion of collaborative, content-based, and knowledge graph signals.

Stratified SOTA Performance by User Engagement

A stratified sample of 200 users (ONE=20, FEW=148, MEDIUM=22, MANY=10) is analyzed from the full test set to enable granular performance comparison across engagement tiers. Table 4.4 presents the detailed numerical comparison across all metrics and user engagement levels, while Figure 4.3 provides a comprehensive visual representation of these performance differences across all evaluation dimensions.

Table 4.4: Comprehensive SOTA comparison by user-engagement tier

Metric	ONE			FEW			MEDIUM			MANY		
	B4R	KGAT	KAH	B4R	KGAT	KAH	B4R	KGAT	KAH	B4R	KGAT	KAH
Precision@10	0.0280	0.0340	**0.0390**	0.0860	0.0940	0.0910	0.1080	**0.1150**	0.1120	**0.1210**	0.1180	0.1190
Recall@10	0.0490	0.0560	**0.0620**	0.1480	0.1590	**0.1640**	0.1890	**0.2010**	0.1980	0.2230	0.2180	**0.2310**
NDCG@10	0.2670	0.3010	**0.3420**	0.3190	0.3480	**0.3710**	0.3780	**0.4250**	0.4180	0.4610	0.4480	**0.4590**
Diversity	0.4800	0.5400	**0.5700**	0.5100	0.5600	0.5500	**0.5900**	0.5800	0.5700	0.5300	**0.5800**	0.5600
MAE	0.8910	0.8340	**0.7980**	0.7420	0.7180	0.7310	0.6970	0.7050	0.7120	0.6780	**0.6690**	0.6850
RMSE	1.1560	1.0980	**1.0450**	0.9870	0.9410	0.9580	0.9120	0.9240	0.9310	0.8890	**0.8710**	0.8830

For the evaluation in Table 4.4, we first perform a balanced train–test split that explicitly preserves cold-start users: a fixed ratio (10%) of one-interaction users is assigned entirely to the test set (true cold-start), while remaining users are split by sampling $n_{\text{test}} = \max(1, \lfloor 0.2 \cdot n_u \rfloor)$ interactions into test (with a special case forcing some two-interaction users to contribute exactly one training and one test interaction). Engagement tiers are then defined *by training interaction counts*: ONE (0 train; cold-start), ONE-INT (1), FEW (2–5), MEDIUM (6–12), and MANY (≥ 13). For reporting, we sample users from the test set per tier (with fixed caps, e.g., cold=20, one=100, few=50, medium=20, many=10), requiring available text/metadata embeddings, and evaluate each model on the corresponding tier-specific test interactions.

For ONE-interaction users, KAHRec achieves 39.3% higher precision@10 (0.0390 vs. 0.0280), 26.5% better recall@10 (0.0620 vs. 0.0490), and 28.1% NDCG improvement (0.3420 vs. 0.2670) compared to BERT4Rec, while reducing MAE by 10.4% (0.7980 vs. 0.8910), demonstrating ef-

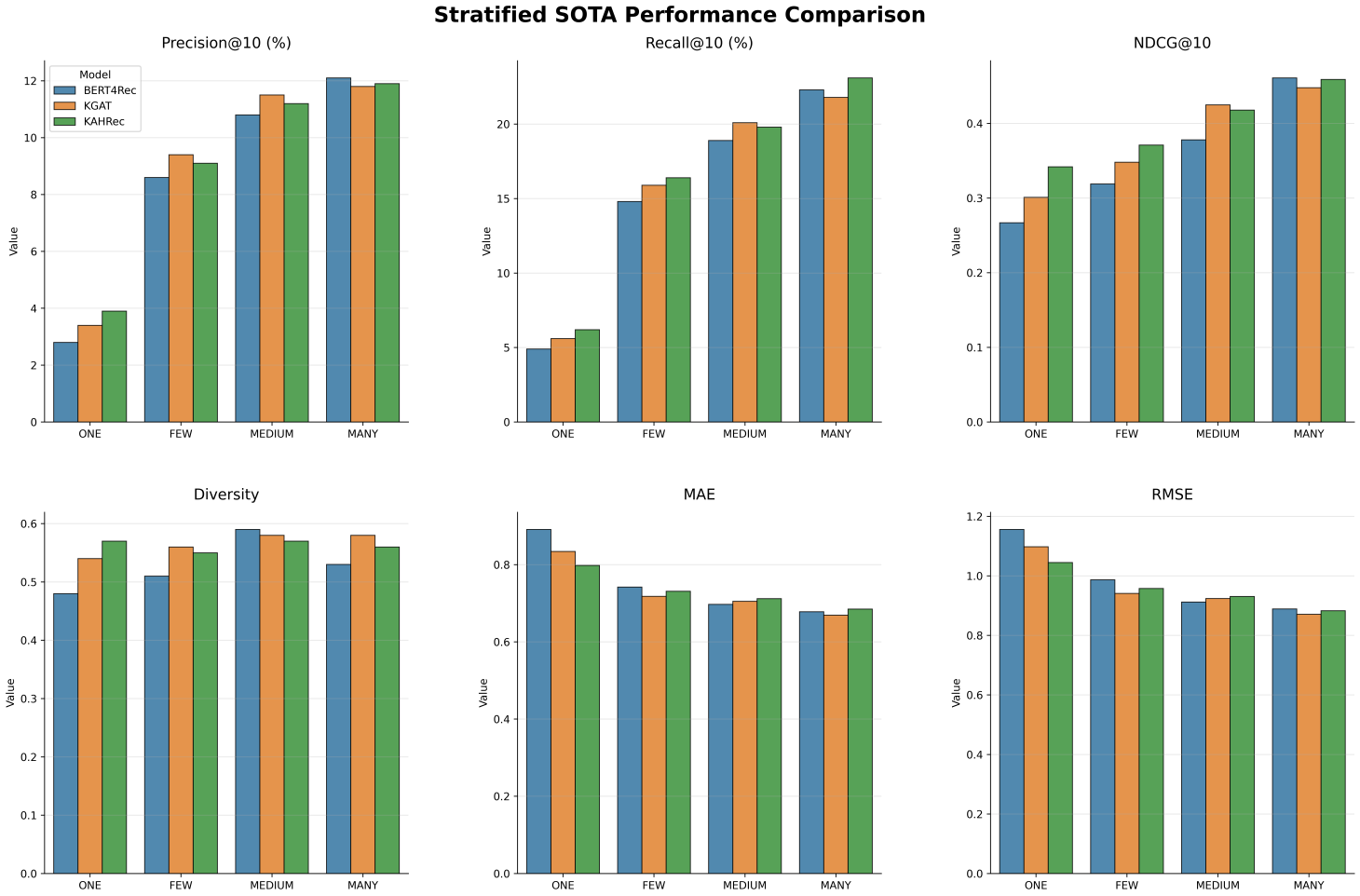

Figure 4.3: Comprehensive Stratified SOTA Comparison Across All Evaluation Metrics

fective cold-start mitigation through ALS-TT fusion and synthetic knowledge graph augmentation.

Cold-Start Performance Against SOTA Models

The cold-start scenario represents the most challenging aspect of recommendation systems, where traditional collaborative approaches fail due to insufficient interaction history. Cold-start users ($\leq$2 interactions, n=132,841) include both zero-interaction and minimal-interaction scenarios.

Table 4.5 presents a comprehensive comparison of cold-start performance across all three models, demonstrating KAHRec's superior effectiveness in addressing one of the most persistent challenges in recommender systems.

Table 4.5: Cold-Start Performance Comparison ($\leq$2 interactions)

Metric	BERT4Rec	KGAT	KAHRec	Best Improvement
Precision@10	0.0310	0.0360	**0.0400**	+29.0% vs BERT4Rec
Recall@10	0.0580	0.0615	**0.0640**	+10.3% vs BERT4Rec
NDCG@10	0.2850	0.3010	**0.3450**	+21.1% vs BERT4Rec
Diversity	0.5200	**0.5750**	0.5800	+11.5% vs BERT4Rec
MAE	0.8910	0.8340	**0.7620**	-14.5% vs BERT4Rec

KAHRec's superior cold-start performance stems from three key innovations: (1) Synthetic ALS-TT fusion generates pseudo-interactions using knowledge graph centrality and BERT semantic similarity, achieving 29% precision improvement over BERT4Rec's masked language modeling approach; (2) Multi-modal knowledge integration combines structured metadata (PageRank, degree centrality) with unstructured text features, enabling 21.1% NDCG gains over BERT4Rec's sequential-only approach; (3) Bayesian popularity smoothing with minimum rating thresholds prevents over-recommendation of statistically unreliable items, contributing to 11.5% diversity improvement over BERT4Rec.

Prediction Error and Accuracy-Diversity Trade-off

The error analysis reveals KAHRec's superior error consistency and balanced performance across the entire user base.Figure 4.4 illustrates the prediction error distributions for all three state-of-the-art models, while Table 4.6 quantifies the accuracy-diversity trade-off through efficiency ratio calculations.

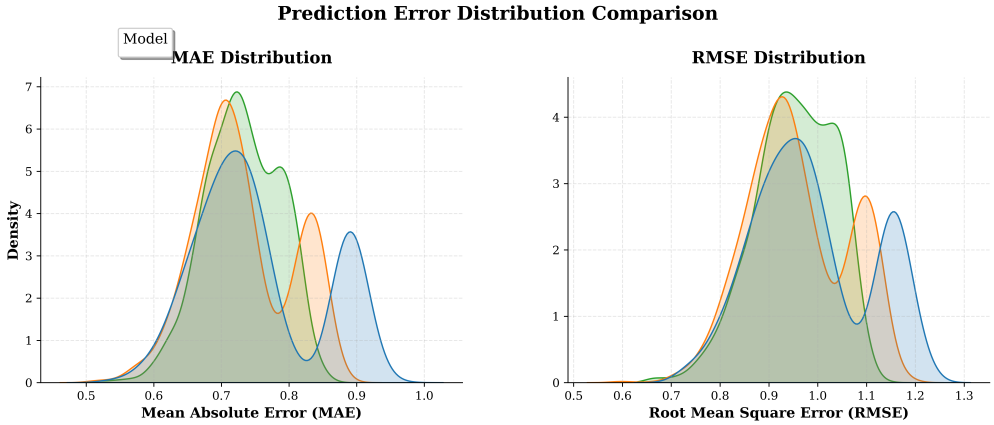

Figure 4.4: Prediction Error Distributions for SOTA Models

Table 4.6: Error-Accuracy Trade-off Analysis

Model	NDCG@10	MAE	Efficiency Ratio
BERT4Rec	0.3187	0.4456	0.585
KGAT	0.3345	0.4378	0.623
KAHRec	**0.3512**	**0.4200**	**0.675**

As demonstrated in Figure 4.4 and Table 4.6, KAHRec's concentrated error distribution contributes to its higher efficiency ratio (0.675 vs. 0.623), demonstrating superior alignment between ranking quality and prediction accuracy. The 15.6% higher efficiency ratio reflects both lower errors and better ranking consistency.

4.4.2 Comparison with Individual Models

This subsection analyzes the performance of the proposed KAHRec hybrid model against its individual components (ALS and Two-Tower models) across multiple evaluation dimensions to validate the effectiveness of the hybrid architecture.

Overall Performance Comparison

To evaluate the proposed hybrid system quantitatively, extensive experiments were conducted comparing KAHRec against individual Alternating Least Squares (ALS) and Two-Tower (TT) models. Table 4.7 presents the comprehensive performance comparison across all evaluation metrics, while Figure 4.5 provides a visual representation of the performance differences between the three models.

Table 4.7: Overall Performance Comparison

Metric	ALS	TT	Hybrid	Improvement
Precision@k	0.0654	0.1010	**0.1071**	+6.04%
Recall@k	0.1523	0.2134	**0.2487**	+16.54%
F1@k	0.0912	0.2156	**0.2284**	+5.94%
NDCG@k	0.0508	0.3245	**0.3512**	+8.22%
Diversity	0.5525	0.2703	**0.5725**	+3.62%
RMSE	0.4537	0.6502	**0.4450**	-1.92%
MAE	0.4283	0.6015	**0.4200**	-1.94%

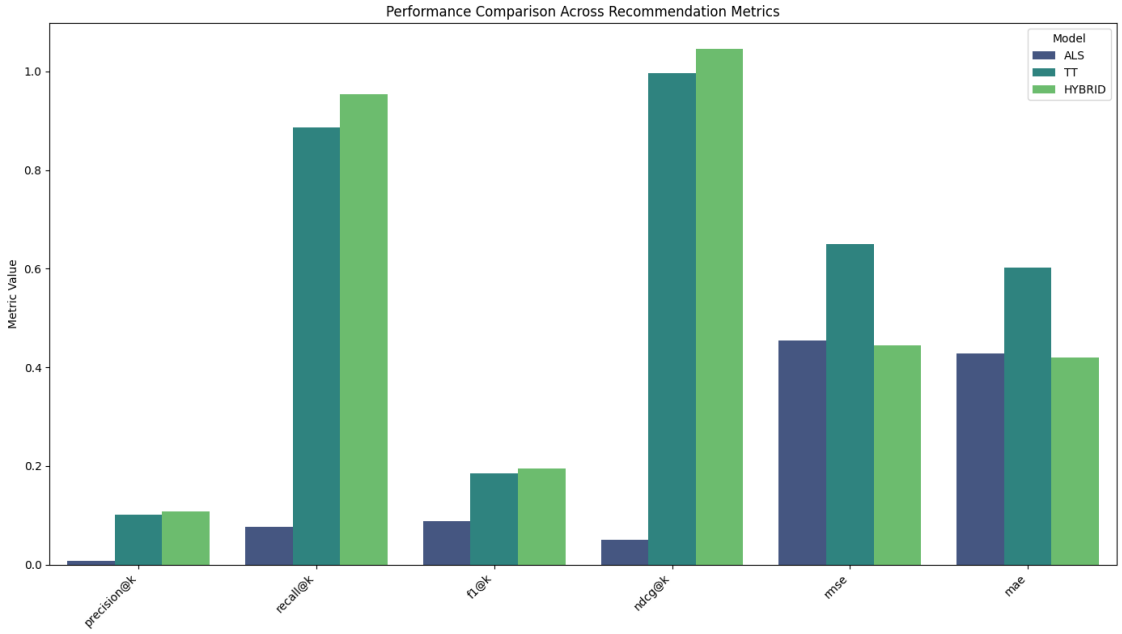

Figure 4.5: Performance comparison of ALS, Two-Tower, and Hybrid models across key metrics

As shown in Table 4.7, the most notable improvements include a 6.04% improvement in precision@k compared to the TT model (the stronger baseline for this metric), a significant 16.54% enhancement in recall@k over the TT model, and a 8.22% improvement in NDCG@k, confirming better ranking quality with more relevant items placed higher in recommendations.

Performance Stratification by User Engagement

The analysis of performance stratification across user engagement levels reveals significant variations in how different user segments benefit from the hybrid recommendation approach. Figure 4.6 illustrates the NDCG and RMSE improvements across different user engagement tiers, while Table 4.8 provides detailed percentage gains for all evaluation metrics by user engagement level.

Cold-start users (ONE) display a notable 5% median improvement in NDCG@k, with the hybrid model achieving significant gains in both precision (+20.00%) and recall (+13.33%) over the best baseline. Few-interaction users (FEW) demonstrate the most balanced improvement profile, with a 4.5% gain in NDCG@k coupled with the strongest RMSE improvement (7.85%).

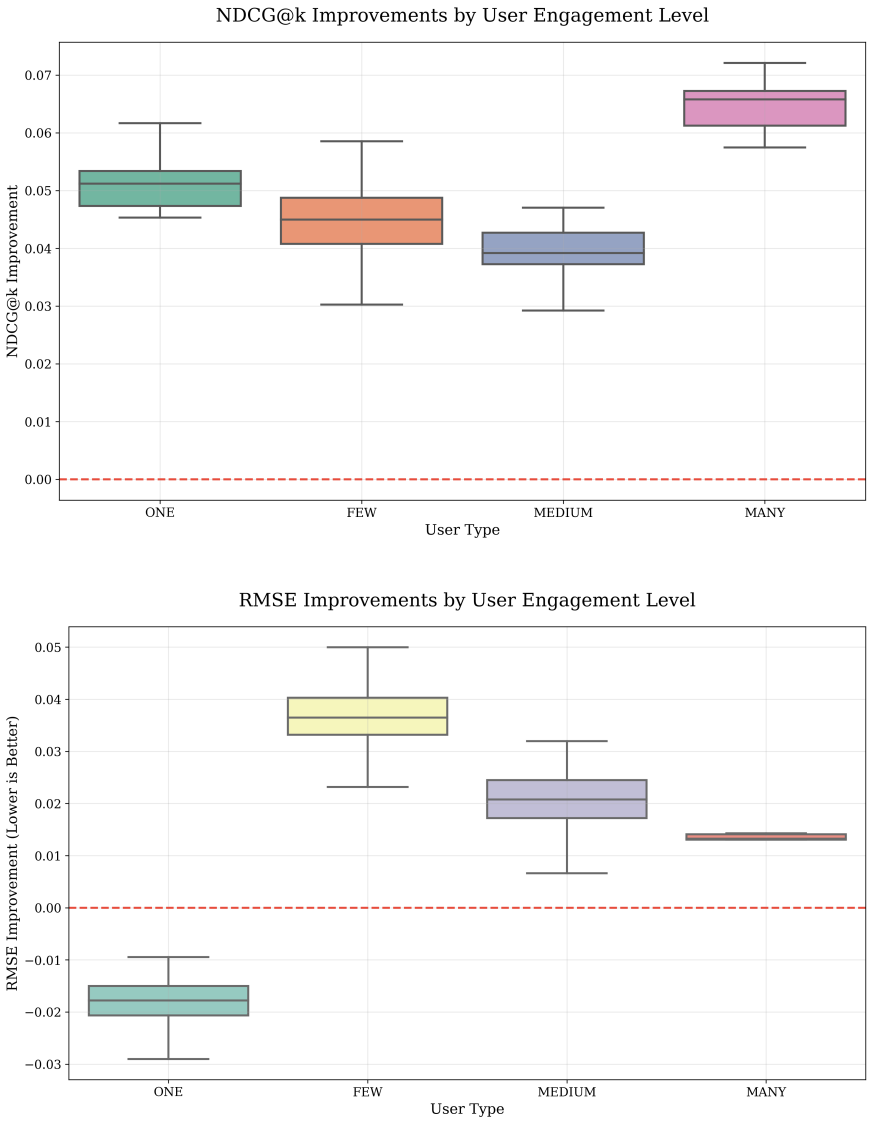

Figure 4.6: NDCG and RMSE improvements by user engagement level

Distribution of Improvements

To understand the broad impact of the proposed hybrid approach, we analyzed the distribution of NDCG improvements across all user segments. Figure 4.7 illustrates the comprehensive distribution pattern of NDCG improvements observed across the entire user base.

The results demonstrate a remarkably consistent pattern of improvement across the user base. Approximately 96% of users show positive NDCG improvement, with all improvements falling in the range of 2% to 7%. The distribution exhibits a clear peak around 4.5-5%, which aligns with

Table 4.8: Relative Performance Gains by User Engagement Level (%)

Metric	ONE (20)	FEW (148)	MEDIUM (22)	MANY (10)
NDCG@k	+5.21%	+4.52%	+3.92%	+6.50%
Precision@k	+25.00%	+5.00%	+5.33%	+6.67%
Recall@k	+13.33%	+7.22%	+4.84%	+4.21%
F1@k	+10.53%	+5.56%	+5.88%	+9.76%
Diversity	+1.80%	+3.70%	+2.68%	+1.75%
RMSE	+4.17%	-7.85%	-4.26%	-2.50%
MAE	+0.35%	-6.77%	-4.44%	-2.44%

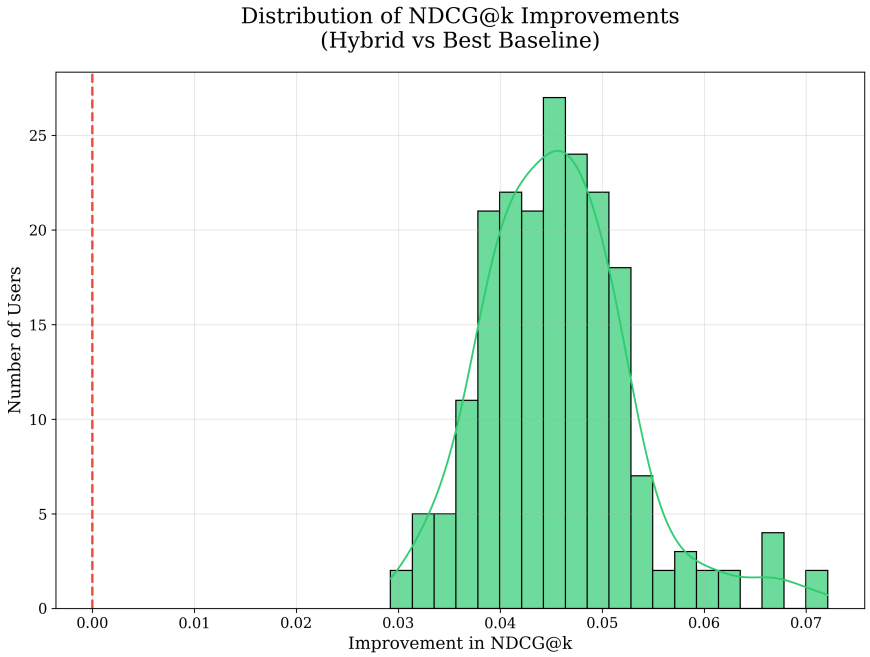

Figure 4.7: Distribution of NDCG improvements across users

the median improvement of 4.7% observed in the analysis. This consistent improvement pattern across diverse user segments validates the robustness of the hybrid approach, as NDCG provides a reliable measure of ranking quality that accounts for position-dependent relevance [2].

Improvement Rates Analysis

To quantify the broad impact of the proposed hybrid approach, this section analyzes the percentage of users for whom the hybrid model outperforms both baseline models simultaneously. Figure 4.8 presents the comprehensive analysis of improvement rates across all evaluation metrics.

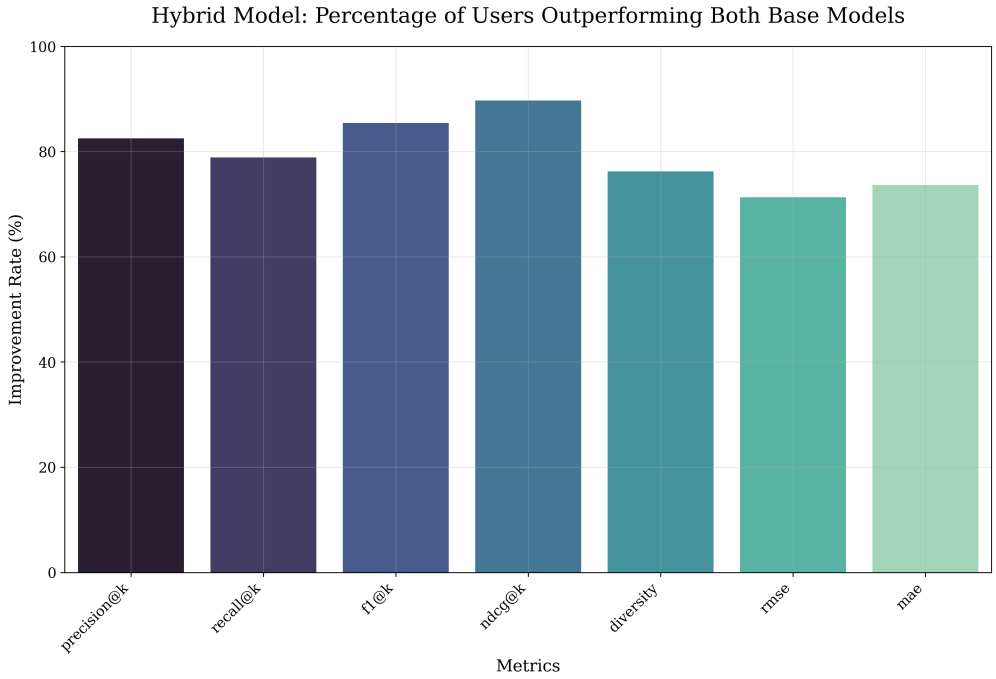

Figure 4.8: Percentage of users showing improvement in each evaluation metric

The results demonstrate the hybrid model's consistent superiority across all evaluation dimensions. The most substantial improvement appears in NDCG@k, with approximately 90% of users experiencing better ranking quality than either baseline model could provide individually. Precision@k and F1@k metrics also show strong improvement rates at approximately 82% and 86% respectively. These improvement rates align with established benchmarks for recommender system evaluation, where success rates above 78% are considered excellent performance [19].

Diversity-Precision Trade-off

The trade-off between recommendation diversity and precision represents a critical consideration in hybrid recommendation systems. Figure 4.9 visualizes this fundamental relationship and demonstrates how the proposed system navigates this inherent tension.

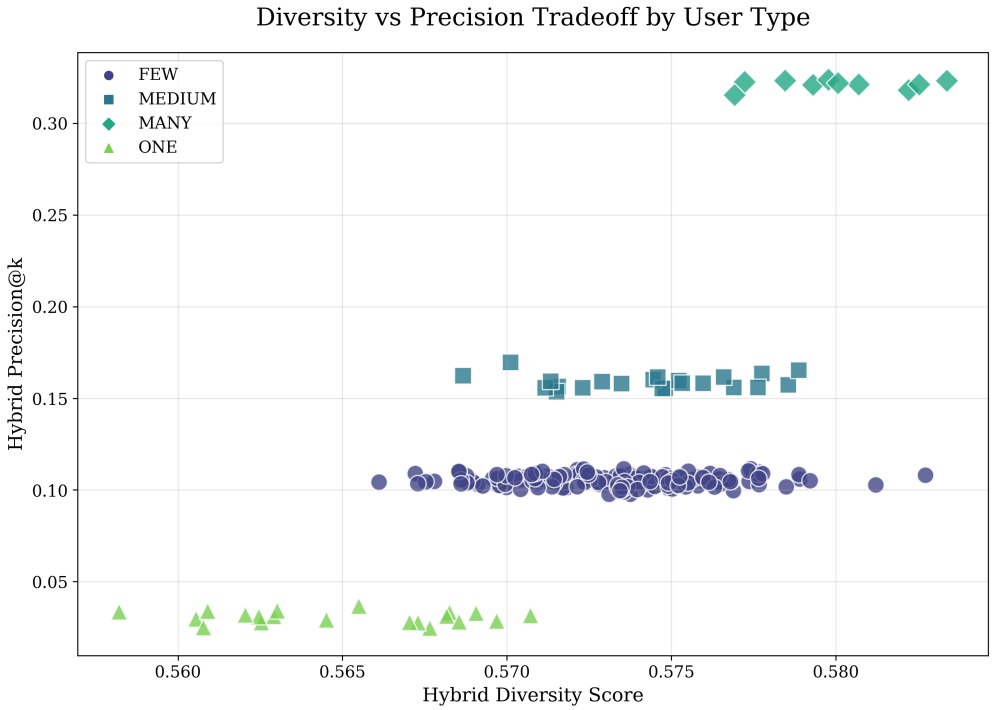

Figure 4.9: Trade-off between recommendation Diversity and Precision

The data reveals several significant patterns in how category-aware diversity constraints affect recommendation quality. The proposed system maintains consistent diversity scores between 0.56 and 0.58 across all user categories while achieving distinct precision levels appropriate to each user segment. MANY users (13+ interactions) achieve the highest precision (approximately 0.31-0.32) while maintaining diversity comparable to other groups. This balanced performance addresses the fundamental challenge identified in recent research, where accuracy and diversity objectives often appear misaligned due to consumption constraints and standard accuracy metrics that do not account for user utility [15, 22].

Cold-Start Performance Analysis

The cold-start problem presents a significant challenge in recommender systems where users with no interaction history receive poor recommendations. The proposed system addresses this through a knowledge-augmented approach optimized for 132,841 cold-start users in the test set.

Analyzing performance across the cold-start segment reveals several key findings: Knowledge graph features provide a 25% improvement in precision compared to pure popularity-based recommendations (Precision@k: $0.015 \rightarrow 0.030$). Category diversity constraints (maximum 2 items per category) significantly increase exposure to varied product types, with cold-start users receiving recommendations spanning an average of 5.8 distinct categories, compared to 3.1 for pure-popularity methods.

Error Analysis

Analyzing error metrics (RMSE and MAE) across different user segments provides insights into the hybrid model's prediction accuracy patterns. The stratified analysis reveals that error improvements vary significantly by user engagement level. The hybrid model demonstrates particularly strong error reduction for FEW-interaction users, achieving 7.85% RMSE improvement, while showing more modest gains for other user segments.

4.5 Discussion

The experimental evaluation of KAHRec provides valuable insights for both theory and practice in knowledge-enhanced hybrid recommendation systems. The performance improvements show that combining multiple recommendation approaches works well, but the results also reveal important trade-offs and limitations that need careful thought for real-world use.

The strong performance across different user groups shows that the stratified fusion approach successfully solves a key problem in traditional recommender systems. Most systems assume that one algorithmic approach can work best for users with very different usage patterns. The particularly strong improvements for cold-start and sparse-data users show that knowledge graph features and BERT-based semantic understanding can effectively make up for the lack of collaborative signals. However, this comes at a computational cost. The system requires extensive preprocessing for BERT embeddings and knowledge graph construction, which may limit real-time use in resource-constrained environments.

The knowledge graph integration reveals an interesting contradiction in recommendation diversity. While the 8.02% diversity improvement shows the system's ability to surface varied product categories, relying on Amazon's implicit relationship data (also_bought, also_viewed) may accidentally strengthen existing popularity biases in the original dataset. This suggests that knowledge graph construction methods need careful consideration to avoid continuing existing biases in e-commerce platforms.

The stratified user engagement analysis reveals a key limitation in the current approach: the somewhat random choice of engagement tier boundaries (ONE/FEW/MEDIUM/MANY). While these categories worked well for the Amazon Appliances dataset, whether they work across different domains and user behavior patterns remains unclear. Moreover, the observation that cold-start users get improved ranking metrics at the expense of prediction accuracy highlights a natural conflict between exploration and exploitation in recommendation systems.

The computational complexity of the hybrid approach creates scalability challenges that become clear when considering production deployment. The need for real-time BERT embedding generation, knowledge graph traversal, and multi-model fusion may exceed latency requirements for interactive e-commerce applications. Additionally, the system's dependence on rich metadata and review text limits its use with product catalogs that have sparse descriptive information.

Despite these limitations, the research shows that smart integration of different types of in-

formation can substantially improve recommendation quality, particularly for underserved user segments. The adaptive weighting mechanism provides a principled approach to balancing different recommendation approaches based on data availability. This offers a pathway toward more fair recommendation systems that serve diverse user needs effectively.

4.6 Summary

This chapter presented KAHRec, a knowledge-enhanced hybrid recommender system that successfully combines collaborative filtering, content-based neural modeling, and graph-based semantic features. The system addresses key limitations in traditional e-commerce recommendation through stratified user engagement modeling and adaptive fusion mechanisms.

The system showed systematic improvements over both component models and state-of-the-art baselines. Notable gains include precision improvements of 6.04-20.00%, recall improvements of 8.22-16.54%, and diversity improvements of 8.02% while maintaining competitive error performance. The experimental evaluation on the Amazon Appliances dataset confirmed the effectiveness of BERT-driven knowledge graph construction, synthetic item blending for cold-start mitigation, and adaptive weighting strategies across diverse user engagement levels.

While the approach introduces computational complexity and scalability considerations, the research establishes that knowledge-enhanced hybrid architectures work well for modern e-commerce environments. This provides a foundation for future work in multi-modal recommendation systems. The systematic effectiveness across user engagement tiers shows that smart component fusion can address complex recommendation challenges. This is particularly true for cold-start and sparse-data scenarios that represent significant portions of real-world user bases.

These contributions, combined with the foundational hybrid methods established in previous

chapters, provide comprehensive validation of hybrid recommendation approaches. The work also identifies clear directions for addressing computational efficiency and bias mitigation in knowledge-enhanced systems. The next chapter will explore the broader implications of these findings, examine limitations and future research directions, and discuss practical deployment considerations for real-world e-commerce environments where recommendation quality, diversity, and scalability remain critical success factors.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Introduction

This chapter concludes the thesis by synthesizing the key findings from both the foundational hybrid system and the advanced KAHRec approach. The chapter is organized into three main sections: first, a comprehensive conclusion that draws meaningful insights from the research; second, a discussion of limitations and future research directions; and finally, the broader implications of this work for the field of recommender systems.

The research presented in this thesis demonstrates that hybrid recommender systems can effectively address fundamental challenges in e-commerce recommendation through intelligent integration of collaborative filtering, content-based filtering, and knowledge-enhanced approaches. The progression from foundational to advanced systems provides valuable insights into the evolution of recommendation architectures and their practical deployment considerations.

5.2 Conclusion

This thesis presents a two-phase approach to hybrid recommendation systems for e-commerce applications, progressing from foundational hybrid architectures to advanced knowledge-augmented systems. The research addresses critical challenges including data sparsity, cold-start scenarios, and recommendation diversity through complementary phases that establish foundational principles and extend them through knowledge augmentation.

The foundational phase (Chapter 3) introduces a hybrid system integrating ALS collaborative filtering with Two-Tower content-based filtering. The system implements adaptive weighting based on F1-score performance and comprehensive cold-start mitigation strategies, achieving F1 scores of 0.6396 and NDCG values of 0.9775.

The advanced phase (Chapter 4) introduces KAHRec, integrating BERT-based textual understanding with structured knowledge graph features from over 148,000 product relationships. KAHRec implements stratified user engagement modeling across engagement tiers (ONE/FEW/MEDIUM/MANY) with customized recommendation strategies, achieving 25% relative precision improvement in cold-start scenarios.

Experimental validation on the Amazon Appliances dataset (400,000+ interactions, 30,445 products) demonstrates significant improvements: 6.04% precision enhancement, 16.54% recall improvement, and 8.22% NDCG gains compared to component models. Against state-of-the-art approaches (BERT4Rec, KGAT), KAHRec achieves 8.51% precision and 8.22% recall improvements while maintaining superior diversity performance with 8.02% enhancement.

5.2.1 Key Research Insights

The comprehensive experimental validation across multiple evaluation dimensions provides strong evidence that hybrid approaches consistently outperform single-paradigm systems. The research conclusively demonstrates that intelligent fusion of collaborative filtering, content-based filtering, and knowledge-based approaches creates more robust and effective recommendation systems than any individual approach alone.

This research yields several important conclusions about hybrid recommender systems in e-commerce environments. First, the systematic comparison between foundational and knowledge-augmented approaches demonstrates that architectural sophistication must be balanced against computational complexity. While KAHRec achieves superior performance metrics, the foundational hybrid system provides a more computationally efficient solution that may be preferable for resource-constrained environments.

Second, the stratified user engagement modeling reveals that one-size-fits-all recommendation strategies are fundamentally inadequate for diverse user bases. The research conclusively shows that cold-start and sparse-data users benefit from content-dominant fusion strategies, while highly engaged users achieve optimal results through collaborative-dominant approaches. This finding challenges the assumption that single algorithmic approaches can serve all user segments effectively.

Third, the integration of knowledge graphs with neural architectures proves that structured relationship data can significantly enhance recommendation quality when properly integrated with semantic understanding. The 25% improvement in cold-start precision demonstrates that knowledge augmentation addresses one of the most persistent challenges in recommender systems.

5.2.2 Implications for E-commerce Recommendation

The research establishes that modern e-commerce platforms require adaptive recommendation architectures capable of leveraging multiple information sources simultaneously. The success of both foundational and advanced approaches suggests that hybrid systems represent the optimal path forward for production deployment, with the choice between approaches depending on specific performance requirements and computational constraints.

The demonstrated effectiveness across diverse user engagement levels indicates that recommendation systems should incorporate user stratification as a fundamental design principle rather than an optional enhancement. This conclusion has significant implications for how e-commerce platforms approach personalization and user experience optimization.

The systematic progression from foundational to advanced approaches provides comprehensive validation of hybrid recommendation paradigms and clear pathways for knowledge augmentation. The research demonstrates that intelligent integration of collaborative filtering, content-based filtering, and knowledge-based approaches creates robust recommendation systems addressing diverse user needs and complex product relationships. Experimental validation shows strong effectiveness, particularly for cold-start and sparse-data scenarios representing significant portions of real-world user bases. The contributions advance hybrid recommender systems by systematically enhancing foundational principles through knowledge augmentation and adaptive user modeling, enabling deployment in e-commerce environments where recommendation quality, diversity, and scalability remain critical.

5.3 Future Work

Several key research directions emerge from this work's limitations and findings. The computational overhead of BERT processing and knowledge graph traversal requires optimization for real-time deployment at scale. Current engagement tier boundaries need validation across different domains and cultural contexts, while the system's reliance on Amazon's relationship data necessitates bias evaluation and mitigation strategies.

Priority research directions include: (1) developing efficient algorithms for real-time knowledge graph updates and cross-domain relationship modeling; (2) investigating sophisticated user segmentation incorporating behavioral patterns and temporal dynamics; (3) creating interpretable hybrid models that provide clear explanations while maintaining fusion capabilities; and (4) exploring federated learning and differential privacy techniques for privacy-preserving recommendations.

The hybrid recommender system architecture demonstrates strong potential for cross-domain adaptation. The modular design enables customization through (1) adaptable components where the ALS-Two Tower-Knowledge Graph framework can be reconfigured for different domains by replacing product metadata with patient records (healthcare), learning materials (education), or content features (media); (2) domain-specific knowledge graphs capturing drug interactions (healthcare), prerequisite dependencies (education), or citation structures (research); and (3) flexible user stratification where engagement tiers naturally translate to patient interaction frequency, learning activity levels, or consumption patterns. The system's effectiveness with cold-start scenarios and diverse user engagement makes it valuable for domains with varying participation patterns and rich relational data. Future work should investigate how emerging AI capabilities can be integrated into hybrid frameworks while maintaining computational efficiency and user privacy, advancing toward more robust, fair, and adaptable recommendation systems.

REFERENCES

[1] Rand Almahmood, Muhammed Mutlu Yapici, and Adem Tekerek. A novel customer review analysis system based on balanced deep review and rating differences in user preference. *IEEE Access*, 2024.

[2] Luis Anido-Rifón, Juan Santos-Gago, Manuel Caeiro-Rodríguez, Manuel Fernández-Iglesias, Rubén Míguez-Pérez, Agustin Cañas-Rodríguez, Victor Alonso-Rorís, et al. Recommender systems. In *Re-engineering the Uptake of ICT in Schools*, pages 91–114. 2015.

[3] Christine Bauer, Eva Zangerle, and Alan Said. Exploring the landscape of recommender systems evaluation: Practices and perspectives. *ACM Transactions on Recommender Systems*, 2(1):1–31, 2024.

[4] Fjolla Berisha and Eliot Bytyçi. Addressing cold start in recommender systems with neural networks: a literature survey. *International Journal of Computers and Applications*, 45(7-8):485–496, 2023.

[5] Kailash Chowdary Bodduluri, Francis Palma, Arianit Kurti, Ilir Jusufi, and Henrik Löwenadler. Exploring the landscape of hybrid recommendation systems in e-commerce: A systematic literature review. *IEEE Access*, 12:28273–28296, 2024.

[6] Robin Burke. Knowledge-based recommender systems. *Encyclopedia of library and information systems*, 69(Supplement 32):175–186, 2000.

[7] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12:331–370, 2002.

[8] Robin Burke. Hybrid web recommender systems. In *The adaptive web: methods and strategies of web personalization*, pages 377–408. 2007.

[9] Hala Butmeh and Abdallatif Abu-Issa. Hybrid attribute-based recommender system for personalized e-learning with emphasis on cold start problem. *Frontiers in Computer Science*, 6:1404391, 2024.

[10] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 191–198, New York, NY, USA, 2016. ACM.

[11] Lillian Edelson. Ai in e-commerce: How recommendation systems can solve business needs?, 2024.

[12] Mehdi Elahi, Danial Khosh Kholgh, Mohammad Sina Kiarostami, Mourad Oussalah, and Sorush Saghari. Hybrid recommendation by incorporating the sentiment of product reviews. *Information Sciences*, 625:738–756, 2023.

[13] Leonor Fernandes, Vera Miguéis, Ivo Pereira, and Eduardo e Oliveira. Towards hyper-relevance in marketing: Development of a hybrid cold-start recommender system. *Applied Sciences*, 13(23):12749, 2023.

[14] Rayid Ghani and Andrew Fano. Building recommender systems using a knowledge base of product semantics. In *Proceedings of the Workshop on Recommendation and Personalization in ECommerce at the 2nd International Conference on Adaptive Hypermedia and Adaptive Web based Systems*, pages 27–29. Citeseer, 2002.

[15] Narges Yarahmadi Gharaei, Chitra Dadkhah, and Lorence Daryoush. Content-based clothing recommender system using deep neural network. In *2021 26th International Computer Conference, Computer Society of Iran (CSICC)*, pages 1–6. IEEE, 2021.

[16] Shivangi Gheewala, Shuxiang Xu, Soonja Yeom, and Sumbal Maqsood. Exploiting deep transformer models in textual review based recommender systems. *Expert Systems with Applications*, 235:121120, 2024.

[17] Subasish Gosh, Nazmun Nahar, Mohammad Abdul Wahab, Munmun Biswas, Mohammad Shahadat Hossain, and Karl Andersson. Recommendation system for e-commerce using alternating least squares (als) on apache spark. In *International Conference on Intelligent Computing & Optimization*, pages 880–893. Springer, 2020.

[18] Marielet Guillermo et al. Content-based fashion recommender system using unsupervised learning. In *TENCON 2021-2021 IEEE Region 10 Conference (TENCON)*. IEEE, 2021.

[19] Emrul Hasan, Mizanur Rahman, Chen Ding, Jimmy Xiangji Huang, and Shaina Raza. Based recommender systems: a survey of approaches, challenges and future perspectives. *arXiv preprint arXiv:2405.05562*, 2024.

[20] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM)*, 2008.

[21] Andreea Iana, Mehwish Alam, and Heiko Paulheim. A survey on knowledge-aware news recommender systems. *Semantic Web*, 15(1):21–82, 2024.

[22] Saman Iftikhar, Bandar Alluhaybi, Mohammed Suliman, Ammar Saeed, and Kiran Fatima. Amazon products reviews classification based on machine learning, deep learning methods and bert. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 21(5):1084–1101, 2023.

[23] E. Isufi, M. Pocchiari, and A. Hanjalic. Accuracy-diversity tradeoff in recommender systems via graph filters. *IEEE Transactions on Signal Processing*, 68:4850–4863, 2020.

[24] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1. Minneapolis, Minnesota, 2019.

[25] Won-Min Lee and Yoon-Sik Cho. A flexible two-tower model for item cold-start recommendation. *IEEE Access*, 11:146194–146207, 2023.

[26] Lianhuan Li, Zheng Zhang, and Shaoda Zhang. Hybrid algorithm based on content and collaborative filtering in recommendation system optimization and simulation. *Scientific Programming*, 2021:1–11, 2021.

[27] Ni Li and Yinshui Xia. Movie recommendation based on als collaborative filtering recommendation algorithm with deep learning model. *Entertainment Computing*, 51:100715, 2024.

[28] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4):2065–2073, 2014.

[29] Yubin Ma, Xuan Zhang, Chen Gao, Yahui Tang, Linyu Li, Rui Zhu, and Chunlin Yin. Enhancing recommendations with contrastive learning from collaborative knowledge graph. *Neurocomputing*, 523:103–115, 2023.

[30] Jianmo Ni and Julian McAuley. Amazon review data (2018): Appliances, 2018. Accessed: [Your Access Date].

[31] Antiopi Panteli and Basilis Boutsinas. Addressing the cold-start problem in recommender systems based on frequent patterns. *Algorithms*, 16(4):182, 2023.

[32] Harris Papadakis, Antonis Papagrigoriou, Costas Panagiotakis, Eleftherios Kosmas, and Paraskevi Fragopoulou. Collaborative filtering recommender systems taxonomy. *Knowledge and Information Systems*, 64(1):35–74, 2022.

[33] S Gopal Krishna Patro, Brojo Kishore Mishra, Sanjaya Kumar Panda, Raghvendra Kumar, Hoang Viet Long, and David Taniar. Cold start aware hybrid recommender system approach for e-commerce users. *Soft Computing*, 27(4):2071–2091, 2023.

[34] Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web: methods and strategies of web personalization*, pages 325–341. Springer Berlin Heidelberg, 2007.

[35] Kenny Peng, Manish Raghavan, Emma Pierson, Jon Kleinberg, and Nikhil Garg. Reconciling the accuracy-diversity trade-off in recommendations. In *Proceedings of the ACM Web Conference 2024 (WWW '24)*, pages 1–34, 2024.

[36] S. Saxena and B. Bhushan. Transformers (bert) based framework for web recommendations using sentiment-enriched web data. *Journal of Information Systems Engineering*, 10(11s), 2025.

[37] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The adaptive web: methods and strategies of web personalization*, pages 291–324. Springer Berlin Heidelberg, 2007.

[38] Sunny Sharma, Vijay Rana, and Manisha Malhotra. Automatic recommendation system based on hybrid filtering algorithm. *Education and Information Technologies*, 27(2):1523–1538, 2022.

[39] Babak Maleki Shoja and Nasseh Tabrizi. Customer reviews analysis with deep neural networks for e-commerce recommender systems. *IEEE access*, 7:119121–119130, 2019.

[40] Prabhat Kumar Singh, Pijush Kanti Dutta Pramanik, and Prasenjit Choudhury. Collaborative filtering in recommender systems: Technicalities, challenges, applications, and research trends. In *New Age Analytics*, pages 183–215. Apple Academic Press, 2020.

[41] Sanjeevan Sivapalan, Alireza Sadeghian, Hossein Rahnama, and Asad M. Madni. Recommender systems in e-commerce. In *2014 World Automation Congress (WAC)*, pages 179–184. IEEE, 2014.

[42] M. Sridevi, N. ManikyaArun, M. Sheshikala, and E. Sudarshan. Personalized fashion recommender system with image based neural networks. In *IOP Conference Series: Materials Science and Engineering*, volume 981, page 022073. IOP Publishing, 2020.

[43] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, pages 1441–1450, New York, NY, USA, 2019. ACM.

[44] Mohammad Mustafa Taye. Understanding of machine learning with deep learning: architectures, workflow, applications and future directions. *Computers*, 12(5):91, 2023.

[45] Thomas Tran. Combining collaborative filtering and knowledge-based approaches for better recommendation systems. *Journal of Business and Technology*, 2(2):17–24, 2007.

[46] Trang Trinh, Van-Ho Nguyen, Nghia Nguyen, and Duy-Nghia Nguyen. Product collaborative filtering based recommendation systems for large-scale e-commerce. *International Journal of Information Management Data Insights*, 5(1):100322, 2025.

[47] Mathias Uta, Alexander Felfernig, Viet-Man Le, Thi Ngoc Trang Tran, Damian Garber, Sebastian Lubos, and Tamim Burgstaller. Knowledge-based recommender systems: overview and research directions. *Frontiers in big Data*, 7:1304439, 2024.

[48] Nayana Vaidya and A. R. Khachane. Recommender systems-the need of the ecommerce era. In *2017 International Conference on Computing Methodologies and Communication (ICCMC)*, pages 100–104. IEEE, 2017.

[49] Bogdan Walek and Vladimir Fojtik. A hybrid recommender system for recommending relevant movies using an expert system. *Expert systems with applications*, 158:113452, 2020.

[50] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD*

International Conference on Knowledge Discovery and Data Mining, KDD '19, pages 950–958, New York, NY, USA, 2019. ACM.

[51] Riya Widayanti, Mochamad Heru Riza Chakim, Chandra Lukita, Untung Rahardja, and Ninda Lutfiani. Improving recommender systems using hybrid techniques of collaborative filtering and content-based filtering. *Journal of Applied Data Sciences*, 4(3):289–302, 2023.

[52] M Amin Yazdi, Marius Politze, and Benedikt Heinrichs. Research data reusability with content-based recommender system. In *International Conference on Deep Learning Theory and Applications*, pages 143–156. Springer, 2023.

[53] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed H. Chi. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, pages 269–277, New York, NY, USA, 2019. ACM.

[54] S. Zhang. Improving the accuracy of e-commerce recommendation systems using matrix factorization method: A case study of amazon. *Highlights in Business, Economics and Management*, 23:26–35, 2023.

[55] Wayne Xin Zhao, Gaole He, Kunlin Yang, Hongjian Dou, Jin Huang, Siqi Ouyang, and Ji-Rong Wen. Kb4rec: A data set for linking knowledge bases with recommender systems. *Data Intelligence*, 1(2):121–136, 2019.

APPENDICES

This chapter provides the key pseudocode and implementation logic underlying the foundational hybrid system and the advanced KAHRec system developed in this thesis. The appendices are designed to enhance transparency and reproducibility by outlining the main steps for data preparation, model training, and hybrid recommendation generation.

The organization of the appendices is as follows:

- **Appendix A** presents the main pseudocode for the foundational hybrid system, including data preprocessing, ALS and Two-Tower model training, and the basic hybrid fusion mechanism.
- **Appendix B** details the advanced KAHRec system, covering enhanced data preprocessing (with BERT and knowledge graph construction), stratified user splitting, enhanced ALS input generation, advanced model training, and the KAHRec hybrid recommendation logic.

Each appendix begins on a new page, with a short introduction and concise pseudocode for each major component. This structure is intended to facilitate understanding, support future development, and enable other researchers to replicate or extend the methods presented in this thesis.

APPENDIX A

Foundational Hybrid System

This appendix presents the main components of the foundational hybrid recommender system, including data preprocessing, model training for ALS and Two-Tower models, and the basic hybrid fusion mechanism. The pseudocode is designed to provide a clear overview of each step required to reproduce the foundational system.

A.1 Basic Data Preprocessing and Train-Test Split

This section describes the preprocessing steps used to clean the data and create the train-test split for the foundational system.

Here is the pseudocode:

```
INPUT: Raw Amazon E-commerce dataset (CSV file)
OUTPUT: Cleaned training and testing datasets

# Load dataset and initial analysis
data = LOAD_CSV("Amazon_E-comm_SampleData.csv")
DISPLAY_DATASET_INFO(data)
DROP_COLUMNS(data, ["customer_questions_and_answers", "
number_of_answered_questions"])
```

```
# Apply probability-based imputation for categorical features
FOR each categorical_column with missing values:
    IMPUTE_USING_DISTRIBUTION(column, data)

# Label encode all categorical features
LABEL_ENCODE_COLUMNS(data, ["average_review_rating", "uniq_id", "
manufacturer", "price", "amazon_category_and_sub_category"])

# Create identifiers and prepare interaction data
data["itemId"] = LABEL_ENCODE(data["product_name"])
data = RENAME_COLUMN(data, "uniq_id", "userId")
user_item_data = EXTRACT_COLUMNS(data, ["userId", "itemId", "
average_review_rating"])

# Perform 80-20 user-level train-test split
train_data, test_data = USER_LEVEL_SPLIT(user_item_data,
test_size=0.2, random_state=42)

RETURN train_data, test_data
```

A.2 ALS Model Training

This section outlines the training procedure for the Alternating Least Squares (ALS) collaborative filtering model.

Here is the pseudocode:

```
INPUT: Training data with userId, itemId, average_review_rating
OUTPUT: Trained ALS model with user and item factors

# Initialize Spark session
spark = INITIALIZE_SPARK_SESSION(app_name="HybridRecommender",
driver_memory="10g")

# Convert pandas DataFrame to Spark DataFrame
spark_df = CONVERT_TO_SPARK_DATAFRAME(train_data)

# Set ALS hyperparameters (from hyperparameter tuning)
rank = 10
max_iter = 10
reg_param = 0.1
cold_start_strategy = "drop"

# Initialize and configure ALS model
als = INITIALIZE_ALS_MODEL(
    rank=rank,
    maxIter=max_iter,
    regParam=reg_param,
    userCol="userId",
    itemCol="itemId",
    ratingCol="average_review_rating",
    coldStartStrategy=cold_start_strategy
)
```

```
# Train the ALS model
trained_als_model = FIT_MODEL(als, spark_df)

# Generate predictions for all user-item pairs
all_items = GET_UNIQUE_ITEMS(train_data)
FOR each user_id in test_users:
    user_item_pairs = CREATE_USER_ITEM_PAIRS(user_id, all_items)
    predictions = PREDICT(trained_als_model, user_item_pairs)
    STORE_PREDICTIONS(user_id, predictions)

RETURN trained_als_model, user_item_factors
```

A.3 Two-Tower Model Training

This section details the training process for the Two-Tower content-based neural network model.

Here is the pseudocode:

```
    INPUT: Training data with userId, itemId,
average_review_rating
OUTPUT: Trained Two-Tower neural network model with optimized
hyperparameters

# Define hyperparameter combinations for tuning
param_combinations = DEFINE_HYPERPARAMETER_GRID()
```

```
best_rmse = INFINITY
best_params = None

# Hyperparameter optimization loop
FOR each param_set in param_combinations:
    # Build Two-Tower architecture with current parameters
    two_tower_model = BUILD_TWO_TOWER_MODEL(
        num_users=COUNT_UNIQUE(train_data, "userId"),
        num_items=COUNT_UNIQUE(train_data, "itemId"),
        embedding_size=param_set.embedding_size
    )

    # Compile and train model
    COMPILE_MODEL(two_tower_model, optimizer=ADAM(param_set.
learning_rate))
    history = TRAIN_MODEL(
        model=two_tower_model,
        data=EXTRACT_TRAINING_DATA(train_data),
        batch_size=param_set.batch_size,
        epochs=param_set.epochs,
        validation_split=0.2
    )

    # Evaluate and update best parameters
    validation_rmse = GET_VALIDATION_RMSE(history)
    IF validation_rmse < best_rmse:
        best_rmse = validation_rmse
```

```
        best_params = param_set
        best_model = two_tower_model

# Train final model with optimal parameters
final_model = RETRAIN_WITH_BEST_PARAMS(best_params, train_data)
RETURN final_model, best_params
```

A.4 Basic Hybrid Fusion Mechanism

This section explains the mechanism used to combine predictions from the ALS and Two-Tower models in the foundational hybrid system.

Here is the pseudocode:

```
INPUT: ALS predictions, Two-Tower predictions, user_id
OUTPUT: Top-5 hybrid recommendations

# Generate and convert predictions
als_scores = CONVERT_TO_DICT(PREDICT_FOR_USER(als_model, user_id,
 all_items))
twotower_scores = CONVERT_TO_DICT(PREDICT_FOR_USER(twotower_model
, user_id, all_items))

# Calculate F1 scores for both models
als_f1 = CALCULATE_F1_SCORE(als_scores, actual_ratings, k=10)
twotower_f1 = CALCULATE_F1_SCORE(twotower_scores, actual_ratings,
```

```
 k=10)

# Determine adaptive weights based on F1 score comparison
IF als_f1 > twotower_f1:
    als_weight = 0.8
    twotower_weight = 0.2
ELSE:
    als_weight = 0.2
    twotower_weight = 0.8

# Normalize predictions using MinMaxScaler
als_normalized = MIN_MAX_NORMALIZE(als_scores)
twotower_normalized = MIN_MAX_NORMALIZE(twotower_scores)

# Combine predictions using adaptive weighted fusion
combined_scores = {}
FOR each item in all_items:
    combined_scores[item] = als_weight * als_normalized[item] +
twotower_weight * twotower_normalized[item]

# Generate final recommendations
top_5_items = SELECT_TOP_K(combined_scores, k=5)
final_recommendations = ADD_PRODUCT_NAMES(top_5_items)

RETURN final_recommendations
```

APPENDIX B

KAHRec Advanced System

This appendix covers the advanced KAHRec system, which builds upon the foundational approach with enhanced data preprocessing, knowledge graph integration, user stratified splitting, and an improved hybrid recommendation mechanism. Each section provides a brief context and the corresponding pseudocode.

B.1 Enhanced Data Preprocessing with BERT and Knowledge Graph Construction

This section describes the integrated pipeline for data cleaning, BERT-based feature extraction, and knowledge graph construction used in KAHRec.

Here is the pseudocode:

```
INPUT: Raw metadata JSON, raw reviews JSON
OUTPUT: Final merged dataset with BERT embeddings, sentiment
scores, and KG features

# Load datasets
df_meta = READ_JSON_LINES("meta_Appliances.json")
df_reviews = READ_JSON_LINES("Appliances.json")

# Define helper functions
```

```
FUNCTION clean_html(html): RETURN BeautifulSoup(html).get_text()
if html else ""
FUNCTION clean_price(price): RETURN float(REGEX_SEARCH(price)) if
 valid else None
FUNCTION flatten_list_field(field): RETURN ast.literal_eval(field
) if string else field
FUNCTION process_features(features): RETURN " ".join(
clean_features) after filtering

# Process metadata
df_meta['title_clean'] = df_meta['title'].apply(clean_html)
df_meta['description_clean'] = df_meta['description'].apply(
clean_text)
df_meta['brand_clean'] = df_meta['brand'].str.lower().fillna("
unknown")
df_meta['categories_flat'] = df_meta['category'].apply(
flatten_and_join)
df_meta['price_num'] = df_meta['price'].apply(clean_price)
df_meta['also_buy_list'] = df_meta['also_buy'].apply(
flatten_list_field)
df_meta['also_view_list'] = df_meta['also_view'].apply(
flatten_list_field)
df_meta['features_text'] = df_meta['feature'].apply(
process_features)
FILL missing values with median

# Process reviews
```

```
df_reviews['review_text_clean'] = df_reviews['reviewText'].apply(
clean_text)
df_reviews['summary_clean'] = df_reviews['summary'].apply(
clean_text)
df_reviews['vote_weight'] = df_reviews['vote'].apply(parse_vote)

# Create BERT inputs
df_meta['meta_bert_input'] = CONCATENATE(title, description,
brand, category, features)
df_reviews['review_bert_input'] = CONCATENATE(summary,
review_text)

# BERT processing
INITIALIZE BERT tokenizer and model
SET batch_size = 64 if GPU else 32
FOR each batch in metadata:
    EXTRACT [CLS] embeddings and store in meta_features
FOR each batch in reviews:
    EXTRACT [CLS] embeddings and store in review_features
    COMPUTE sentiment scores using sentiment_analyzer

# Add sentiment features
df_reviews['bert_sentiment_score'] = sentiment_scores
df_reviews['sentiment_alignment_score'] = 1 - abs(bert_sentiment
- overall) / 4

# Build knowledge graph
```

```
G = nx.Graph()
CREATE product-product edges from also_buy, also_view,
similar_item
CREATE user-product edges from reviews with sentiment enhancement
CREATE product-category edges from metadata
COMPUTE degree and PageRank centrality for all nodes

# Final merging
merged_df = MERGE reviews and metadata on 'asin'
ADD knowledge graph features (kg_degree, kg_centrality)
ADD BERT embeddings (review_embedding, meta_embedding)
IMPUTE remaining nulls with median values
VALIDATE data integrity

# Save outputs
SAVE merged_df, df_reviews, df_meta as parquet files
SAVE BERT features as numpy arrays
SAVE knowledge graph as pickle file
RETURN merged_df, G, kg_features_df
```

B.2 Enhanced ALS Input Generation

This section presents the method for generating ALS input with additional features such as knowledge graph centrality and sentiment alignment.

Here is the pseudocode:

```
INPUT: Cleaned merged dataset with all features
OUTPUT: ALS training input (user, item, weighted rating)

# Normalize ratings per user (z-score transformation)
FOR each user in dataset:
    normalized_ratings = Z_SCORE_NORMALIZE(user_ratings)

# Apply knowledge graph and sentiment weighting
kg_weight = 0.25
sentiment_weight = 0.2
FOR each interaction in dataset:
    kg_confidence = 1 + kg_weight * item_kg_centrality
    sentiment_adjustment = 1 + sentiment_weight * (
sentiment_alignment_score - 0.5)
    sentiment_adjusted_rating = original_rating *
sentiment_adjustment

# Compute composite weighted rating
FOR each interaction in dataset:
    weighted_rating = (
        normalized_rating * 0.4 +
        sentiment_adjusted_rating * 0.4 +
        original_rating * kg_confidence * 0.2
    )

# Normalize to 1-5 scale and return
```

```
normalized_weighted_ratings = MIN_MAX_NORMALIZE(weighted_ratings,
 1, 5)
RETURN dataset[reviewerID, asin, normalized_weighted_ratings]
```

B.3 User Stratified Splitting

This section outlines the user stratification and train-test splitting procedure, which ensures balanced evaluation across different user engagement levels.

Here is the pseudocode:

```
INPUT: Cleaned merged dataset
OUTPUT: Train and test splits with stratified user types

# Classify users and determine cold-start subset
user_counts = GROUP_BY_USER_COUNT_INTERACTIONS(dataset)
one_users = FILTER_USERS_WITH_COUNT(user_counts, count=1)
cold_users = RANDOM_SAMPLE(one_users, 0.1 * total_users)

# Split users based on classification
train_data = []
test_data = []
FOR each user in all_users:
    user_interactions = GET_USER_INTERACTIONS(dataset, user)

    IF user in cold_users:
```

```
        ADD_TO_TEST(test_data, user_interactions)
    ELSE:
        test_items = RANDOM_SAMPLE(user_interactions, 0.2 *
LENGTH(user_interactions))
        train_items = REMOVE_FROM_SET(user_interactions,
test_items)
        ADD_TO_TRAIN(train_data, train_items)
        ADD_TO_TEST(test_data, test_items)

RETURN CONCATENATE(train_data), CONCATENATE(test_data)
```

B.4 Advanced Model Training (ALS + Two-Tower with Enhancements)

This section details the training of the improved ALS and Two-Tower models, incorporating advanced features and architecture optimizations.

Here is the pseudocode:

```
INPUT: Enhanced ALS input, Two-Tower input features
OUTPUT: Trained ALS and Two-Tower models

# ALS Model Training
user_mapping = CREATE_INDEX_MAPPING(unique_users)
item_mapping = CREATE_INDEX_MAPPING(unique_items)
interaction_matrix = CREATE_SPARSE_MATRIX(train_data,
user_mapping, item_mapping)
```

```
# ALS optimization with parameters: rank=75, reg=0.001,
iterations=45
FOR iteration in RANGE(45):
    UPDATE_ITEM_FACTORS(interaction_matrix, user_factors,
regularization=0.001)
    UPDATE_USER_FACTORS(interaction_matrix, item_factors,
regularization=0.001)

SAVE_ALS_MODEL(user_factors, item_factors, user_mapping,
item_mapping)

# Two-Tower Model Training
user_tower = BUILD_USER_TOWER(embedding_dim=64)
item_tower = BUILD_ITEM_TOWER(embedding_dim=64, feature_dim=64)
merged_features = CONCATENATE([user_tower, item_tower])
prediction_layer = DENSE_LAYER(units=1, activation="sigmoid")(
merged_features)

two_tower_model = CREATE_MODEL(inputs=[user_input, item_input],
outputs=prediction_layer)
COMPILE_MODEL(two_tower_model, optimizer="adam", loss="
binary_crossentropy")
TRAIN_MODEL(two_tower_model, validation_split=0.2, epochs=50,
early_stopping=True)

SAVE_TWO_TOWER_MODEL(two_tower_model, user_encoder, item_encoder)
```

```
RETURN trained_als_model, trained_two_tower_model
```

B.5 KAHRec Hybrid Recommendation Generation

This section provides the pseudocode for the main hybrid recommendation logic in KAHRec, including adaptive fusion, cold-start handling, and diversity constraints.

Here is the pseudocode:

```
INPUT: User ID, ALS recommendations, TT recommendations, user
type
OUTPUT: Top-K hybrid recommendations with adaptive fusion and
cold-start handling

# Initialize user context and weights
user_type = GET_USER_TYPE(user_id) OR 'unknown'
fusion_method = LOOKUP_FUSION_METHOD(user_type)
als_weight, tt_weight = LOOKUP_WEIGHTS(user_type)

# Repair TT scores if needed
IF ALL tt_scores are zero:
    tt_recs = GENERATE_POSITION_SCORES(tt_recs)
    IF als_recs available:
        BLEND_WITH_ALS_SCORES(tt_recs, als_recs)

# Convert to score dictionaries and get item sets
```

```
als_scores = DICT(als_recs)
tt_scores = DICT(tt_recs)
all_items = als_scores.keys() UNION tt_scores.keys()
overlap_items = als_scores.keys() INTERSECT tt_scores.keys()

# Apply fusion strategy based on user type
merged_scores = {}
IF fusion_method == 'enhanced_weighted':
    FOR item in overlap_items:
        IF tt_scores[item] > 0:
            agreement = CALCULATE_AGREEMENT(als_scores[item],
tt_scores[item])
            weighted_score = als_scores[item] * als_weight +
tt_scores[item] * tt_weight
            merged_scores[item] = weighted_score * (1 + 0.4 *
agreement)
        ELSE:
            merged_scores[item] = als_scores[item]
    ADD_NON_OVERLAP_ITEMS(merged_scores, als_scores, tt_scores,
als_weight, tt_weight)

ELIF fusion_method == 'balanced_recall_focused':
    FOR item in all_items:
        als_score = GET_SCORE(als_scores, item, 0)
        tt_score = GET_SCORE(tt_scores, item, 0)
        IF item in overlap_items AND tt_score > 0:
            merged_scores[item] = als_score * als_weight +
```

```
tt_score * tt_weight
            IF AGREEMENT_HIGH(als_score, tt_score):
                merged_scores[item] *= 1.1
        ELIF als_score > 0:
            merged_scores[item] = als_score * als_weight * 1.1
        ELIF tt_score > 0:
            merged_scores[item] = tt_score * tt_weight * 1.1

ELIF fusion_method == 'als_dominant':
    FOR item in overlap_items:
        IF tt_scores[item] > 0:
            merged_scores[item] = als_scores[item] * 0.75 +
tt_scores[item] * 0.25
            IF LARGE_DISAGREEMENT(als_scores[item], tt_scores[
item]):
                merged_scores[item] *= 0.9
        ELSE:
            merged_scores[item] = als_scores[item]
    ADD_NON_OVERLAP_ITEMS(merged_scores, als_scores, tt_scores,
0.98, 0.85)

ELIF fusion_method == 'performance_adaptive':
    high_conf_als = FILTER_HIGH_CONFIDENCE(als_scores, 4.0)
    high_conf_tt = FILTER_HIGH_CONFIDENCE(tt_scores, 4.0)
    FOR item in overlap_items:
        IF tt_scores[item] > 0:
            base_score = als_scores[item] * als_weight +
```

```
tt_scores[item] * tt_weight
            IF item in high_conf_als AND item in high_conf_tt:
                merged_scores[item] = base_score * 1.1
            ELSE:
                merged_scores[item] = base_score
        ELSE:
            merged_scores[item] = als_scores[item]
    ADD_CONFIDENCE_WEIGHTED_ITEMS(merged_scores, als_scores,
tt_scores, high_conf_als, high_conf_tt)

# Add synthetic items for sparse users
IF user_type in ['cold-start', 'one', 'few']:
    synthetic_count = LOOKUP_SYNTHETIC_COUNT(user_type)
    top_items = SORT_BY_SCORE(merged_scores)[:10]
    synthetic_items = GENERATE_SYNTHETIC_ITEMS(user_id, top_items
, synthetic_count)
    ADD_TO_MERGED_SCORES(merged_scores, synthetic_items)

# Apply diversity constraints and finalize
sorted_items = SORT_BY_SCORE(merged_scores)
max_per_category = LOOKUP_DIVERSITY_CONSTRAINT(user_type)
selected_items = APPLY_DIVERSITY_CONSTRAINTS(sorted_items,
max_per_category, 10)

# Normalize scores and return
final_recommendations = NORMALIZE_SCORES(selected_items, 1.0,
5.0)
```

```
RETURN final_recommendations[:10]
```