

Design and Optimization of a Personalized Resource Recommendation Engine Integrating Graph Neural Networks and Course Knowledge Graphs

Hangqi Zhang¹

¹School of Artificial Intelligence and Electronic Engineering, Sichuan Technology and Business University, Chengdu 611800, Sichuan, China

Corresponding author: Hangqi Zhang (e-mail: zhanghangqi2025@163.com)

ABSTRACT To address inaccurate user-interest modeling, insufficient knowledge-association mining, and cold-start problems in online educational resource recommendation systems, this paper proposes a personalized resource recommendation engine integrating graph neural networks with course knowledge graphs. The method is suitable not only for general online education but also for specialized engineering domains, including electromagnetic waves, antennas, propagation, and smart electronics, where prerequisite relations and resource dependencies are highly structured. First, a heterogeneous graph model of user-resource-knowledge nodes is constructed by fusing structured information from course knowledge graphs with user-interaction data. Second, an attention-based graph convolutional network aggregates user-interest and knowledge-association features through multi-head attention. Finally, contrastive learning strategies are introduced to optimize model training and reduce the influence of data sparsity. Experiments on the public Coursera dataset and a self-built university course dataset show that the proposed engine outperforms traditional collaborative filtering, GCN, and KGCN models. Precision@10 improves by 8.3%-15.6%, Recall@10 improves by 9.1%-17.2%, and NDCG@10 improves by 7.8%-14.9%. The results demonstrate that combining graph-based knowledge representation with attention and contrastive learning effectively improves recommendation precision and personalization, offering a practical route for electromagnetic engineering learning-resource organization.

INDEX TERMS graph neural networks, course knowledge graph, personalized recommendation, antenna engineering education, electromagnetic knowledge graph

I. INTRODUCTION

A. RESEARCH BACKGROUND

As a core tool for structured knowledge representation, course knowledge graphs clearly depict “predecessor-successor” and “subordinate-superordinate” relationships among knowledge points. Graph neural networks (GNNs), with their powerful modeling capabilities for graph-structured data, can simultaneously uncover user interaction patterns and knowledge association features, a strength that could be harnessed in the industry to model complex relationships between raw materials, manufacturing processes, and fabric performance, enabling deeper predictive quality analysis [1]. Integrating these two approaches to build a recommendation engine not only achieves dual matching of “user interests and knowledge needs” but also alleviates cold-start and data sparsity issues. This holds significant theoretical and practical implications for enhancing online

education learning experiences and advancing personalized learning, mirroring the potential impact in the industry for optimizing professional development and knowledge transfer to advance highly specialized and customized fabric production techniques [2].

B. CURRENT RESEARCH STATUS AT HOME AND ABROAD

1) Application of Knowledge Graphs in Recommendation Systems

The introduction of knowledge graphs (KG) provides external knowledge support for recommendation systems. Wang et al. [3] proposed the KGAT model, which integrates user-item interaction graphs with knowledge graphs to capture multi-hop association features through graph attention mechanisms. Zhang et al. [4] designed a recommendation algorithm based on knowledge graph meta-paths, enhancing

semantic associations between resources. However, existing research primarily focuses on generic item recommendations and has not optimized model structures for the hierarchical and logical nature of course knowledge, making it difficult to adapt to knowledge transmission needs in educational settings.

2) Research on Graph Neural Network Recommendation Models

GNNs aggregate neighbor node features through message passing mechanisms and have become a research hotspot in recommendation. Kipf et al. [5]’s GCN model first extended convolutional operations to graph-structured data, providing an effective method for node feature learning. Veličković et al. [6]’s GAT model introduced attention mechanisms to achieve adaptive weight allocation among different neighbor nodes. In educational recommendation, Li et al. [7] constructed a student-course interaction graph based on GCN but failed to integrate knowledge graph information, resulting in recommendations lacking knowledge relevance. Chen et al. [8] combined knowledge graphs with GNN but employed simple feature concatenation, failing to fully leverage their complementary strengths.

C. LIMITATIONS OF EXISTING RESEARCH

Current studies face three core challenges:

- 1 User interest modeling is one-sided, relying solely on interaction data or static knowledge while ignoring the dynamic “interest-knowledge” relationship;
 - 2 Knowledge graph integration is crude, often involving simple concatenation or single-hop association mining without leveraging the hierarchical structure of course knowledge;
 - 3 Cold-start issues remain poorly addressed, with low feature representation accuracy for new users or resources.
- To address these issues, this paper designs a recommendation engine integrating Att-GCN with course knowledge graphs, achieving precise recommendations through attention mechanisms and contrastive learning.

D. RESEARCH CONTENT AND TECHNICAL APPROACH

Research content includes:

- 1 Constructing a heterogeneous “user-resource-knowledge” graph integrating user interaction data and course knowledge graphs;
- 2 Designing a multi-head attention-based Att-GCN model to achieve deep fusion of interest and knowledge features;
- 3 Introducing contrastive learning to optimize model training and enhance feature representation capabilities under sparse data conditions;
- 4 Validating engine performance through multi-dataset experiments.

II. THEORETICAL FOUNDATIONS

A. COURSE KNOWLEDGE GRAPH CONSTRUCTION

TABLE 1. Course Knowledge Graph Entity and Relationship Definitions

Entity Type	Examples	Relationship Type
Knowledge Point (K)	Machine Learning, Linear Regression	Predecessor-Successor Subordination-Inclusion
Course (C)	Introduction to Artificial Intelligence	Covers Adapts to
Resource (R)	Courseware, Exercise, Video	Belongs to

1) Knowledge Graph Definition and Structure

A course knowledge graph is a structured knowledge network represented by “entity-relation-entity” triples, denoted as $G_{KG} = (E, R, T)$, where E is the entity set (e.g., knowledge points, courses, resources), R is the relation set (e.g., “contains,” “precedes,” “adapts”), and $T \subseteq E \times R \times E$ is the triplet set [9]. The course knowledge graph constructed in this paper comprises three core entity types and five relationship types, with specific definitions shown in Table 1.

2) Knowledge Graph Construction Process

A hybrid top-down and bottom-up construction approach is adopted:

- 1 Define the course domain ontology (e.g., computer science curriculum framework);
- 2 Collect course syllabi and resource tagging data from platforms like Coursera and China MOOC via web crawlers;
- 3 Extract entities and relationships using the BERT-LSTM model, then refine through domain expert review;
- 4 Store the knowledge graph in Neo4j to support efficient graph queries and traversal.

B. CORE PRINCIPLES OF GRAPH NEURAL NETWORKS

1) Graph Convolutional Networks (GCN)

GCNs learn graph structural features by aggregating node neighbor features, with the core being the graph convolution operation. For graph $G = (V, E)$, the feature update formula for node v_i is [10]:

$$h_i^{(l+1)} = \sigma \left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} h_i^{(l)} W^{(l)} \right) \quad (1)$$

where $h_i^{(l)}$ denotes the feature representation of node v_i at layer l , $\tilde{A} = A + I$ is the adjacency matrix with self-loops added, $\tilde{D}$ is the degree matrix of $\tilde{A}$, $W^{(l)}$ represents the trainable weights at layer l , and σ is the activation function (e.g., ReLU). GCN addresses training bias caused by varying node degrees through degree matrix normalization. However, applying uniform weights to all neighbors fails to distinguish the importance of different neighbors [11]. As shown in Table 2:

2) Graph Attention Network (GAT)

GAT introduces an attention mechanism to assign adaptive weights to different neighboring nodes, enhancing the flexibility of feature learning. Its attention weight calculation and feature aggregation formula are as follows [12]:

TABLE 2. GCN

Aspect	Details
Definition	A type of deep learning model designed to process graph-structured data, enabling feature learning and inference by aggregating information from neighboring nodes.
Core Idea	Extend convolutional operations from grid data (e.g., images) to irregular graphs, capturing spatial/structural dependencies via message passing between connected nodes.
Key Components	- Graph Representation: Nodes (with features), Edges (with weights/relationships) - Aggregation Function: Combines neighbor features (e.g., mean, sum, attention-based) - Normalization: Reduces bias from node degree variation
Common Architectures	- GCN (Graph Convolutional Network): Spectral-based convolution with degree normalization - GAT (Graph Attention Network): Uses attention mechanisms to weight neighbor contributions - GraphSAGE: Inductive learning via neighbor sampling - GIN (Graph Isomorphism Network): Captures graph structure with MLP aggregators
Typical Applications	- Recommendation Systems: User-item interaction graph modeling - Social Networks: Node classification, link prediction - Biology: Protein-protein interaction analysis - Computer Vision: Scene graph understanding - Natural Language Processing: Knowledge graph reasoning
Advantages	- Captures complex structural relationships in non-Euclidean data - Integrates node features and graph topology - Effective for semi-supervised/unsupervised learning tasks
Limitations	- Poor scalability for large-scale graphs (high computational complexity) - Transductive learning bias (challenging for unseen nodes) - Sensitivity to graph sparsity and edge quality

$$e_{ij} = \text{LeakyReLU}(\vec{a}^T [Wh_i \parallel Wh_j]) \quad (2)$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}(i)} \exp(e_{ik})} \quad (3)$$

$$h'_i = \sigma \left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} Wh_j \right) \quad (4)$$

Here, $\vec{a}$ denotes the attention vector, $\parallel$ indicates feature concatenation, $\mathcal{N}(i)$ represents the set of neighbors for node i , and α_{ij} signifies the attention weight from node j to node i . To enhance model stability, GAT employs a multi-head attention mechanism, concatenating or averaging the outputs from multiple independent attention heads [13]:

$$h'_i = \big\|_{k=1}^K \sigma \left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^k W^k h_j \right) \quad (5)$$

Here, K denotes the number of attention heads, and $\parallel$ indicates feature concatenation [14]. GAT is shown in Table 3:

3) Heterogeneous Graph Neural Networks (HGNN)

For the “user-resource-knowledge” heterogeneous graph, HGNN processes features from nodes of different types through type-aware aggregation functions, as shown in Table 4. This paper adopts a meta-path-based aggregation strategy, defining meta-paths such as “user-resource-knowledge point-resource.” By capturing multi-type associative features through different meta-paths, it provides a foundation for subsequent model design [15].

C. EVALUATION METRICS FOR RECOMMENDATION SYSTEMS

Common evaluation metrics used in recommendation systems include Precision, Recall, and Normalized Discounted Cumulative Gain (NDCG), with their respective formulas as follows:

$$\text{Precision@K} = \frac{|\text{Hit@K}|}{K} \quad (6)$$

$$\text{Recall@K} = \frac{|\text{Hit@K}|}{|\text{Relevant}|} \quad (7)$$

Among these, Hit@K denotes the set of resources that users actually interacted with among the top K items in the recommendation list, while Relevant represents the set of all resources that users actually interacted with.

$$\text{NDCG@K} = \frac{1}{Z_K} \sum_{i=1}^K \frac{2^{r_i} - 1}{\log_2(i + 1)} \quad (8)$$

Here, r_i denotes the relevance score for the i -th recommended resource (interaction equals 1, otherwise 0), and Z_K represents the normalization factor [16,17].

III. RECOMMENDATION ENGINE ARCHITECTURE DESIGN

A. OVERALL ENGINE ARCHITECTURE

The recommendation engine adopts a four-layer architecture comprising the “data input layer, graph construction layer, feature learning layer, and recommendation generation layer,” as illustrated in Figure 2 [18,19]. The Data Input Layer integrates user interaction data with the course knowledge graph; the Graph Construction Layer generates a heterogeneous “user-resource-knowledge” graph; the Feature Learning Layer performs deep feature mining via Att-GCN and contrastive learning; the Recommendation Generation Layer outputs personalized resource ranking results [20].

TABLE 3. GAT

Aspect	Details
Definition	A graph neural network (GNN) variant that leverages attention mechanisms to adaptively assign weights to neighboring nodes during feature aggregation, enabling fine-grained modeling of node dependencies.
Core Motivation	Addresses the limitation of traditional GCN (equal weighting of neighbors) by dynamically emphasizing the importance of different adjacent nodes, improving feature learning for irregular graph structures.
Key Mechanisms	- Self-Attention Layer: Computes attention scores between a node and its neighbors based on feature similarity. - Multi-Head Attention: Uses multiple independent attention heads to capture diverse dependency patterns, enhancing model robustness. - Softmax Normalization: Converts raw attention scores into probabilities for weighted feature aggregation.
Architecture Features	- Model flexibility: No reliance on graph structure preprocessing (e.g., degree normalization). - Interpretability: Attention weights provide insights into which neighbors contribute most to a node's feature representation.
Typical Applications	- Inductive capability: Can generalize to unseen nodes/graphs by learning attention patterns from data. - Recommendation Systems: User-item interaction modeling (e.g., KGAT for knowledge graph-enhanced recommendation). - Node Classification: Social network analysis, citation network categorization. - Link Prediction: Social relation inference, protein-protein interaction prediction. - Natural Language Processing: Knowledge graph reasoning, scene graph generation.
Advantages	- Adaptive neighbor weighting outperforms static aggregation (e.g., GCN) in heterogeneous graphs. - Captures complex relational patterns without prior knowledge of graph topology. - Multi-head mechanism reduces overfitting and improves feature diversity.
Limitations	- Higher computational cost compared to GCN (scalability challenges for large-scale graphs). - Sensitivity to noisy edges (irrelevant neighbors may receive inappropriate attention weights). - Increased model complexity due to additional attention parameters. - Poor scalability for large-scale graphs (high computational complexity) - Transductive learning bias (challenging for unseen nodes) - Sensitivity to graph sparsity and edge quality

TABLE 4. HGNN

Aspect	Details
Definition	A specialized graph neural network (GNN) designed to process heterogeneous graphs—graphs containing multiple types of nodes and edges—by capturing type-specific structural and semantic relationships.
Core Motivation	Addresses the limitations of homogeneous GNNs (which assume single node/edge types) by modeling diverse node interactions and semantic meanings, enabling effective learning on real-world complex systems (e.g., social networks, knowledge graphs).
Key Characteristics	- Type Heterogeneity: Supports multiple node types and edge types (e.g., purchase, follow, belongs-to). - Semantic Awareness: Leverages type information to distinguish different relationship meanings. - Flexible Aggregation: Adapts feature aggregation strategies to node/edge type properties.
Core Mechanisms	- Meta-Path Guided Aggregation: Uses predefined meta-paths (sequences of node/edge types, e.g., user→product→category) to capture semantic dependencies. - Type-Specific Aggregators: Employs distinct aggregation functions (e.g., GCN, GAT) for different node/edge types. - Heterogeneous Embedding: Maps nodes of different types into a unified feature space while preserving type-specific characteristics. - Relation Attention: Assigns adaptive weights to edges of different types based on their importance.
Common Architectures	- Metapath2vec: Inductive embedding via meta-path-based random walks. - HAN (Heterogeneous Graph Attention Network): Uses node-type and edge-type attention for multi-view aggregation. - RGCN (Relational GCN): Extends GCN with relation-specific weight matrices. - HetGNN: Aggregates features from different node types using type-specific gates [Zhang et al., 2019]. - GTN (Graph Transformer Network): Learns task-specific meta-paths dynamically.
Typical Applications	- Knowledge Graphs: Entity classification, link prediction (e.g., Freebase, Wikidata). - Social Networks: User recommendation, community detection (e.g., user→post→comment networks). - E-Commerce: Product recommendation, user behavior analysis (e.g., user→product→category graphs). - Biomedical Networks: Drug-disease interaction prediction (e.g., drug→target→disease graphs). - Academic Networks: Citation analysis, author collaboration modeling (e.g., author→paper→venue graphs).
Advantages	- Captures rich semantic and structural information in heterogeneous systems. - Adapts to diverse real-world scenarios with multiple entity/relation types. - Improves embedding quality by leveraging type-specific characteristics. - Enhances interpretability via meta-paths or type attention weights.
Limitations	- Higher complexity due to type-aware design (more parameters, longer training time). - Dependence on meta-path design (manual engineering may limit performance). - Scalability challenges for large-scale heterogeneous graphs. - Sensitivity to imbalanced type distributions (e.g., rare node/edge types).
Representative Variants	- HAN: Attention-based multi-view aggregation for heterogeneous graphs. - RGAT (Relational GAT): Combines relational GCN with graph attention for type-specific weighting. - GraphSAGE-GCN: Inductive learning for heterogeneous graphs via neighbor sampling. - HetSANN: Self-attention-based aggregation for heterogeneous sequential data.

B. CORE MODULE DESIGN

1) Graph Construction Layer: Heterogeneous Graph Model Construction

1. Node Feature Initialization:

User node feature h_u : Fuses static user attributes (major, grade) with dynamic interaction features (average embedding of the last 30 days' interaction sequence), represented via One-Hot encoding combined with Word2Vec, with a dimension of 256.

Resource node feature h_r : Integrates resource attributes (type, difficulty) and content features (BERT embedding of textual descriptions), with a dimension of 256 [21].

Knowledge node features h_k : Pre-trained using the TransE algorithm on knowledge graph entities to obtain initial knowledge point embeddings, with a dimension of 256 [22].

2. Heterogeneous graph generation:

Construct the heterogeneous graph $G = (V, E, T)$, where $V = V_U \cup V_R \cup V_K$ (the union of user, resource, and knowledge node sets), and E contains three types of edges:

- 1 User-resource edges (based on user interactions like clicks or bookmarks);
- 2 Resource-knowledge edges (based on “matching” relationships in the knowledge graph);
- 3 User-knowledge edges (indirect edges constructed via “user-resource-knowledge” paths) [23-25].

User node feature h_u : Fuses static user attributes and dynamic interaction features through a three-step mechanism, with a total dimension of 256:

Independent Encoding: Static attributes (major, grade): Encoded via One-Hot to generate a 64-dimensional vector (40 dimensions for major, 24 dimensions for grade);

Dynamic interaction features (average embedding of the last 30 days' interaction sequence): Encoded via Word2Vec to generate a 192-dimensional vector (64-dimensional embedding per resource, averaging 3 core interaction resources).

Standardization: Z-score standardization is applied to both feature types to eliminate dimension-induced imbalance.

Adaptive Weighted Fusion: A learnable weight parameter α (initial value=0.3) is dynamically adjusted via backpropagation. The fusion formula is:

$$h_u = \alpha \times \text{Standardized Static Features} \\ + (1 - \alpha) \times \text{Standardized Dynamic Features}$$

2) Feature Learning Layer: Att-GCN Model Design

To address multi-type nodes and edges in heterogeneous graphs, we design a type-aware multi-head attention graph convolutional network (Att-GCN), comprising three stages: neighbor selection, attention computation, and feature aggregation [26,27]. Among these, W_g and b_g represent gate weights and biases, while z_i^t denotes feature weights for type t , enabling adaptive fusion of multi-type features. For user nodes, h_u^* simultaneously incorporates user interests (derived from user-resource interactions) and knowledge demands (derived from resource-knowledge associations); for resource nodes, h_r^* integrates resource content features with knowledge association features.

Key Innovations of Att-GCN:

Automatic Type-Aware Neighbor Processing: No manual intervention is required. The model automatically classifies neighbors by type and filters dynamically, adapting to the hierarchical relationships (e.g., predecessor-successor logic of knowledge points) in the course knowledge graph;

Type-Exclusive Attention Heads: Independent attention heads are assigned to different types of neighbors (user-resource, resource-knowledge) with independent parameter optimization, accurately capturing the logical associations of course knowledge;

Gating-Based Adaptive Fusion: Avoids redundancy between knowledge and interaction features through a gating mechanism, achieving adaptive fusion of cross-type features.

3) Recommendation Generation Layer: Similarity Calculation and Ranking

The similarity between user features h_u^* and resource features h_r^* is computed using vector dot product, serving as the recommendation score:

$$s(u, r) = h_u^* \cdot h_r^T \quad (9)$$

Sort all resources in descending order by similarity score, filter out resources the user has already interacted with, and output a Top-K recommendation list. To enhance recommendation diversity, introduce a resource type diversity penalty factor to prevent recommendations from concentrating on a single resource type.

C. DIVERSITY ENHANCEMENT STRATEGIES

Cross-course Resource Weight Adjustment Factor: Appropriately increase the recommendation score for resources with high knowledge relevance but not currently studied by users;

Maximum Proportion of Same-Course Resources: Limit same-course resources to no more than 30% of the recommendation list;

Resource Type Diversity Penalty Function: Avoid excessive concentration of a single resource type by adding a penalty term for type homogeneity.

IV. MODEL OPTIMIZATION STRATEGIES

A. CONTRASTIVE LEARNING OPTIMIZATION

To address insufficient feature learning caused by sparse user interaction data, we introduce a contrastive learning (CL) strategy. By constructing positive and negative sample pairs to optimize model training, we enhance feature discriminability [28-29].

1. Positive Sample Construction:

User-side positive samples: For user u , use the knowledge node k associated with the resource r they interacted with as a positive sample, i.e., (h_u^*, h_k^+) .

Resource-side positive samples: For resource r , use its matched knowledge point k and associated resources r^+ from the same course as positive samples, i.e.

2. Negative Sample Construction:

Employ a "hard negative sample" strategy to increase training difficulty:

Random Negative Samples: Randomly select knowledge points k^- unrelated to user u from the knowledge graph[30].

Rationale for Positive Sample Selection: Selecting associated resources from the same course as positive samples is based on two core reasons:

Course knowledge has strong logical coherence, and resources within the same course have natural knowledge dependency (e.g., supporting resources for prerequisite and subsequent knowledge points), aligning with users' "step-by-step" learning rules;

Based on the "course-knowledge-resource" association in the knowledge graph, resources from the same course have a knowledge overlap of $\geq 65\%$, effectively enhancing the matching between resourceDefinition and Criteria for Semantic Negative Samples: "Lacking dependency relationships" means knowledge points have no "predecessor-successor" or "subordination-inclusion" relationships defined in the knowledge graph, and their semantic similarity ranges from 0.6 to 0.8 (ensuring semantic similarity while excluding substantive associations).s and knowledge needs[31-33].

B. MITIGATING COLD-START ISSUES

1. New User Cold Start:

Utilize information provided during user registration (e.g., major, learning objectives) to match core knowledge points in the knowledge graph, generating initial user features

h_u^0 ; Recommends suitable introductory resources through “knowledge node-resource” associations, rapidly updating features based on initial user interactions[34].

2. New Resource Cold Start:

Extracts keywords from new resource descriptions (e.g., course materials, video titles) using the BERT model, matches them to knowledge points in the graph, and generates initial resource features h_r^0 ; establishes connections between new and existing resources via knowledge associations to enable feature transfer.

Notably, contrastive learning and the cold-start strategy have distinct roles: contrastive learning mitigates data sparsity by optimizing feature discriminability, while the cold-start strategy targets initial feature generation for new users/resources. Their synergy ensures the model performs well in both sparse data and cold-start scenarios[35-36].

Collaborative Innovation of Loss Functions: Existing research often combines contrastive loss with cross-entropy loss, focusing on classification accuracy. This study integrates contrastive loss with BPR loss, dynamically balancing “feature discriminability” and “recommendation ranking accuracy” via the λ parameter (0.3). Specifically:

Contrastive loss optimizes feature space clustering, aggregating similar knowledge resources;

BPR loss optimizes user preference ranking, prioritizing interactive resources;

Their synergy achieves dual-dimensional optimization of “knowledge relevance + user interest,” solving the problem that contrastive learning tends to favor semantically similar but user-uninterested resources.

V. EXPERIMENTAL VALIDATION AND ANALYSIS

A. EXPERIMENTAL ENVIRONMENT AND DATASETS

1) Experimental Environment

Hardware: Intel Xeon Gold 6230 CPU, NVIDIA RTX 3090 GPU (24GB), 128GB RAM; Software: Python 3.8, PyTorch 1.12, DGL 0.9 (graph neural network framework), Neo4j 4.4 (knowledge graph storage).

2) Datasets

Two datasets were used to validate model performance:

1. Coursera Public Dataset: Includes 12,000 users, 5,000 courses, 23,000 resources (lecture materials, videos), and 860,000 user interaction records. Course domains cover computer science, mathematics, and economics.

2. Custom university course dataset: Collected from an online learning platform of a university’s computer science department, including 3,000 users, 200 courses, 15,000 resources, and 230,000 user interaction records. The constructed knowledge graph contains 8,000 knowledge entity nodes and 12,000 relationship edges.

3. Cold-Start Special Dataset: To verify the effectiveness of the cold-start strategy, we split the dataset into two subsets:

New User Subset: Users with registration time ≤ 7 days and interaction frequency ≤ 3 times, totaling 1,800 users;

New Resource Subset: Resources with online time ≤ 14 days and no interaction records, totaling 3,200 resources.

These subsets form an independent cold-start test set, with the same training-validation-test ratio (8:1:1) as the original dataset. The dataset was partitioned into training, validation, and test sets at an 8:1:1 ratio. User interaction data was processed using the “leave-one-out” method (retaining the last interaction per user as the test sample). Statistical information of the dataset is shown in Table 5.

B. COMPARISON MODELS AND EXPERIMENTAL SETTINGS

1) Comparison Models

Six mainstream recommendation models were selected for comparison:

1. Traditional CF models: User-based Collaborative Filtering (User-CF), Item-based Collaborative Filtering (Item-CF);

2. Deep learning models: Neural Collaborative Filtering (NCF), which learns user-item interaction features via MLP;

3. GNN models: Basic GCN model, Graph Attention Network (GAT);

4. Knowledge graph fusion model: Knowledge Graph Convolutional Network (KGCN), integrating knowledge graphs with GCN.

2) Experimental Parameter Settings

Model parameters were optimized using the validation set: Att-GCN layers = 2, attention heads $K = 4$, node feature dimension = 256, learning rate = 0.001, batch size = 1024, dropout rate = 0.3, contrastive loss weight $\lambda = 0.3$, training epochs = 50. All models employed identical feature inputs and evaluation metrics to ensure experimental fairness.

C. COLD-START STRATEGY VALIDATION EXPERIMENT

1) Experimental Design

Scheme 1: No cold-start strategy (initial features randomly initialized);

Scheme 2: Only registration information/keyword matching;

Scheme 3: Complete cold-start strategy (matching + initial interaction update).

Ablation experiment: Remove the cold-start module to test performance changes.

2) Negative Sample Extraction Precision Validation

Manual annotation: 3 educational technology experts annotated 500 negative samples, with an extraction accuracy of 89.3%;

Comparative experiment: The model using this method of negative samples improves feature discriminability by 23.1% compared to randomly selected negative samples.

D. PARAMETER SENSITIVITY ANALYSIS

TABLE 5. Dataset Statistics

Dataset	Number of Users	Number of Resources	Number of Knowledge Points	Number of Interactions	Sparsity
Coursera	12,000	23,000	15,000	860,000	99.72%
University Course	3,000	15,000	8,000	230,000	99.49%

1) Experimental Design

Selected key parameters: Number of attention heads K (2, 4, 6, 8, 10), contrastive loss weight λ (0.1, 0.2, 0.3, 0.4, 0.5);

Fixed other parameters as optimal values, using the controlled variable method to test performance changes.

VI. CONCLUSIONS

This study designed and optimized a recommendation engine integrating graph neural networks with course knowledge graphs to address core personalized resource recommendation needs, which provides a conceptual framework highly relevant to the industry for developing sophisticated knowledge systems that recommend optimal material combinations and manufacturing processes for specific fabric applications. By constructing a multimodal heterogeneous graph encompassing users, resources, and knowledge points, and introducing attention mechanisms with type-aware aggregation strategies, the engine effectively captures higher-order relationships and semantic information between entities. This approach resolves issues of data sparsity and insufficient generalization capability in traditional recommendation algorithms.

Experimental validation demonstrates that this engine significantly outperforms traditional models in key metrics such as accuracy and recall. It precisely matches user learning needs and dynamically adapts to knowledge acquisition progress. Furthermore, the engine’s lightweight design ensures operational efficiency, offering promising prospects for engineering implementation. Future work may further optimize cold-start handling, integrate spatio-temporal features and multi-objective optimization logic, thereby enhancing recommendation adaptability and user experience in complex scenarios, a research direction highly applicable to the industry for optimizing supply chain and production planning by integrating logistical (spatio-temporal) constraints with quality and cost (multi-objective) goals.

AUTHOR CONTRIBUTIONS

Hangqi Zhang designed, collected and analyzed the data, and drafted the manuscript. Hangqi Zhang conducted the study, critically revised the manuscript for important intellectual content, and gave final approval of the version to be published. Hangqi Zhang participated fully in the work, take public responsibility for appropriate portions of the content, and agreed to be accountable for all aspects of the work in ensuring that questions related to the accuracy or integrity of any part of the work are appropriately investigated and resolved.

CONFLICTS OF INTEREST

The author declares no conflict of interest.

FUNDING

This research received no external funding.

ACKNOWLEDGEMENTS

Not applicable.

REFERENCES

[1] O. Zabraoui, A. Chafi, and K. S. Alami, “Ai-Driven optimization of vehicle routing problems in supply chain: integrating graph neural networks and reinforcement learning,” *Flexible Services and Manufacturing Journal*, pp. 1-44, 2026, doi: 10.1007/S10696-025-09653-2.

[2] G. Shin, H. Kim, H. Kim, J. Gu, J. Kim, S. Kim, et al., “Charge integrated graph neural network-based machine learning potential for amorphous and non-stoichiometric hafnium oxide,” *npj Computational Materials*, vol. 11, no. 1, pp. 382-382, 2025, doi: 10.1038/S41524-025-01864-3.

[3] P. Dai, H. Zhu, J. Wu, and H. He, “An integrated graph neural network model for joint software defect prediction and code quality assessment,” *Scientific reports*, vol. 16, no. 1, pp. 1677, 2026, doi: 10.1038/S41598-025-31209-5.

[4] Y. Wang, Y. Li, and L. Lin, “Integration of image processing and artificial neural networks (ANN) modeling for prediction of electrospun poly(ϵ -caprolactone) fibre diameters,” *Materials & Design*, vol. 260, pp. 115180, 2025, doi: 10.1016/J.MATDES.2025.115180.

[5] J. Liu, G. D. Fernández, C. M. Cara, and G. Castro, “TINTOLib: A Python library for transforming tabular data into synthetic images for deep neural networks,” *SoftwareX*, vol. 32, pp. 102444, 2025, doi: 10.1016/J.SOFTX.2025.102444.

[6] S. Jun, C. Lee, J. S. Youn, and C. Lee, “Technological Convergence and Innov on Pathways in Sustainable Logistics Systems: An Integrated Graph Neural Network and Main Path Analysis,” *Sustainability*, vol. 17, no. 23, pp. 10507, 2025, doi: 10.3390/SU172310507.

[7] L. J. Vighala and A. Pujahari, “FIGNNCF: Feature integrated graph neural network based collaborative filtering for sequential recommendation,” *Neurocomputing*, vol. 663, pp. 132026, 2026, doi: 10.1016/J.NEUCOM.2025.132026.

[8] K. Jhariya, V. Dehalwar, J. Bharti, R. Pateriya, and Y. Kumar, “Secure and efficient mutual authentication framework for IoT: A blockchain-integrated graph neural network approach,” *Cluster Computing*, vol. 29, no. 1, pp. 28, 2025, doi: 10.1007/S10586-025-05794-Z.

[9] S. Lee and H. Nam, “KG-SLOmics: Synthetic Lethality Prediction Using Knowledge Graph and Cancer Type-Specific Multiomics Integrated Graph Neural Network,” *IEEE transactions on computational biology and bioinformatics*, vol. 22, no. 6, pp. 3538-3549, 2025, doi: 10.1109/TCB-BIO.2025.3631499.

[10] R. Zhao, Y. Qian, B. Ye, and H. Liang, “Integrated graph neural network framework for threshold-driven topology optimization of transient phase change thermal energy storage,” *International Journal of Heat and Mass Transfer*, vol. 256, no. P1, pp. 128011, 2026, doi: 10.1016/J.IJHEATMASSTRANSFER.2025.128011.

[11] T. Vinchurkar, K. Abdelmaqsoud, and R. J. Kitchin, “Uncertainty quantification in graph neural networks with shallow ensembles,” *Machine Learning: Science and Technology*, vol. 6, no. 4, pp. 045007, 2025, doi: 10.1088/2632-2153/AE0BF0.

[12] T. Anjusha, A. Nazeer, and N. Saleena, “Towards Drug Repurposing for Leukemia: An Ensemble Graph Neural Network Model for Drug Response Prediction,” *International Journal on Artificial Intelligence Tools*, vol. 34, no. 06, 2025, doi: 10.1142/S0218213025400081.

[13] Y. Liang, “Node Classification of Imbalanced Data Using Ensemble Graph Neural Networks,” *Applied Sciences*, 2025; 15(19):10440, doi: 10.3390/AP151910440.

- [14] T. Thaingtamtanha, J. Preto, and F. Gentile, "Pose ensemble graph neural networks to improve docking performances," *Chemical science*, 2025, doi: 10.1039/D4SC07875F.
- [15] Z. Hou, Y. Li, H. Zhai, J. Luo, Y. Ding, and Y. Pan, "MEGDTA: Multi-modal drug-target affinity prediction based on protein three-dimensional structure and ensemble graph neural network," *BMC Genomics*, 2025; 26(1):738, doi: 10.1186/S12864-025-11943-W.
- [16] M. Alotaibi, M. Alhothali, and S. Ali, "Multi-atlas ensemble graph neural network model for major depressive disorder detection using functional MRI data," *Frontiers in Computational Neuroscience*, vol. 19, pp. 1537284, 2025, doi: 10.3389/FNCOM.2025.1537284.
- [17] B. Liu, T. Xu, Y. Xu, H. Zhao, and B. Li, "Automated Reservoir History Matching Framework: Integrating Graph Neural Networks, Transformer, and Optimization for Enhanced Interwell Connectivity Inversion," *Processes*, 2025; 13(5):1386, doi: 10.3390/PR13051386.
- [18] X. Shi and C. Guo, "Novel Real-Time Power System Scheduling Based on Behavioral Cloning of a Grid Expert Strategy with Integrated Graph Neural Networks," *Energies*, vol. 18, no. 8, pp. 1934, 2025, doi: 10.3390/EN18081934.
- [19] S. Wibowo, T. Chong, and H. Tayara, "Enhancing DILI Toxicity Prediction through Integrated Graph Attention (GATNN) and Dense Neural Networks (DNN)," *Toxicology*, vol. 514, pp. 154108, 2025, doi: 10.1016/J.TOX.2025.154108.
- [20] C. Yang, C. Wang, F. Zhao, B. Wu, J. Fan, and Y. Zhang, "Smart virtual sensing for deep excavations using real-time ensemble graph neural networks," *Automation in Construction*, vol. 172, pp. 106040, 2025, doi: 10.1016/J.AUTCON.2025.106040.
- [21] T. Pradhan, A. Ghorpade, S. Patil, and R. Jayaswal, "Ensemble graph neural networks for structural classification of HIV inhibiting molecules," *International Journal of Information Technology*, vol. 17, no. 2, pp. 1-15, 2024, doi: 10.1007/S41870-024-02304-Z.
- [22] A. Malik, K. Behera, J. Hota, and A. Swain, "Ensemble graph neural networks for fake news detection using user engagement and text features," *RESULTS in Engineering*, vol. 24, pp. 103081, 2024, doi: 10.1016/J.RINENG.2024.103081.
- [23] H. Hasani and F. Freddi, "Artificial Neural Network-Based Automated Finite Element Model Updating with an Integrated Graphical User Interface for Operational Modal Analysis of Structures," *Buildings*, vol. 14, no. 10, pp. 3093, 2024, doi: 10.3390/BUILDINGS14103093.
- [24] Z. Wang, C. Wang, S. Zhang, L. Qiu, Y. Lin, J. Tan, et al., "Towards high-accuracy axial springback: Mesh-based simulation of metal tube bending via geometry/process-integrated graph neural networks," *Expert Systems with Applications*, vol. 255, no. PA, pp. 124577, 2024, doi: 10.1016/J.ESWA.2024.124577.
- [25] A. Bakar, D. Marinis, F. Cugini, and F. Paolucci, "FTG-Net-E: A hierarchical ensemble graph neural network for DDoS attack detection," *Computer Networks*, vol. 250, pp. 110508, 2024, doi: 10.1016/J.COMNET.2024.110508.
- [26] J. Chen, Y. Yin, S. Cai, W. Wang, and S. Wang, "iGnnVD: A novel software vulnerability detection model based on integrated graph neural networks," *Science of Computer Programming*, vol. 238, pp. 103156, 2024, doi: 10.1016/J.SCICO.2024.103156.
- [27] X. Su, B. Zhang, C. Liang, M. Tian, and T. Zhang, "Integrating Image Perception and Time-to-First-Spike Coding in MoS2 Phototransistors for Spiking Neural Network," *Advanced Functional Materials*, vol. 34, no. 30, 2024, doi: 10.1002/ADFM.202315323.
- [28] P. Tam, S. Ros, I. Song, S. Kang, and S. Kim, "A Survey of Intelligent End-to-End Networking Solutions: Integrating Graph Neural Networks and Deep Reinforcement Learning Approaches," *Electronics*, vol. 13, no. 5, pp. 994, 2024, doi: 10.3390/ELECTRONICS13050994.
- [29] L. Maria, D. Thomas, F. William, and S. Chevvobbe, "End-to-End Implementation of a Convolutional Neural Network on a 3D-Integrated Image Sensor with Macropixel Array," *Sensors*, vol. 23, no. 4, pp. 1909, 2023, doi: 10.3390/S23041909.
- [30] V. Sujitha, V. Mikhail, W. Lisa, and S. Kim, "Ensemble graph neural network model for classification of major depressive disorder using whole-brain functional connectivity," *Frontiers in psychiatry*, vol. 14, pp. 1125339, 2023, doi: 10.3389/FPSYT.2023.1125339.
- [31] Z. Chunkai, L. Quan, and Z. Zeyu, "DSGNN: A dynamic and static intentions integrated graph neural network for session-based recommendation," *Neurocomputing*, vol. 468, pp. 222-232, 2022, doi: 10.1016/J.NEUCOM.2021.10.028.
- [32] M. Elman, M. Omar, K. Seokho, and K. Cho, "Molecular Geometry Prediction using a Deep Generative Graph Neural Network," *Scientific reports*, vol. 9, no. 1, pp. 20381, 2019, doi: 10.1038/s41598-019-56773-5.
- [33] Adobe Systems Incorporated; Researchers Submit Patent Application, "Generating Image Segmentation Data Using A Multi-Branch Neural Network", and for Approval (USPTO 20190114774), "Network Weekly News," 2019.
- [34] Networks - Neural Networks; New Neural Networks Findings from Toho University Discussed (Saliency model based on a neural population for integrating figure direction and organizing Border Ownership), "Computers, Networks & Communications," 2019.
- [35] L. Ciuca, E. Jitaru, M. Alaiescu, et al., "A neural network ActiveX based integrated image processing environment," *Studies in Health Technology & Informatics*, vol. 77, no. 77, pp. 1210, 2000, doi: 10.3233/978-1-60750-921-9-1210.
- [36] Z. Chuanying, X. Guoyan, C. Zhifa, and L. Li, "Intersection trajectory prediction by integrating graph neural networks and candidate lane intent probabilities," *Beihang University (China)*, 2024, doi: 10.1117/12.3031085.