Tuning-Free LLM Can Build A Strong Recommender Under Sparse Connectivity And Knowledge Gap Via Extracting Intent

Wenqing Zheng[*] Capital One
Noah Fatsi Capital One
Daniel Barcklow Capital One
Dmitri Kalaev Capital One
Steven Yao Capital One
Owen Reinert Capital One
C. Bayan Bruss Capital One
Daniele Rosa Capital One

Abstract

Recent advances in recommendation with large language models (LLMs) often rely on either commonsense augmentation at the item-category level or implicit intent modeling on existing knowledge graphs. However, such approaches struggle to capture grounded user intents and to handle sparsity and cold-start scenarios. In this work, we present LLM-based Intent Knowledge Graph Recommender (IKGR), a novel framework that constructs an intent-centric knowledge graph where both users and items are explicitly linked to intent nodes extracted by a tuning-free, RAG-guided LLM pipeline. By grounding intents in external knowledge sources and user profiles, IKGR canonically represents *what a user seeks* and *what an item satisfies* as first-class entities. To alleviate sparsity, we further introduce a mutual-intent connectivity densification strategy, which shortens semantic paths between users and long-tail items without requiring cross-graph fusion. Finally, a lightweight GNN layer is employed on top of the intent-enhanced graph to produce recommendation signals with low latency. Extensive experiments on public and enterprise datasets demonstrate that IKGR consistently outperforms strong baselines, particularly on cold-start and long-tail slices, while remaining efficient through a fully offline LLM pipeline.

1 Introduction

Modern recommender systems are expected to reason over sparse and evolving interactions while serving highly personalized needs across vast catalogs. This challenge is acute in enterprise environments, where recommendations support internal search and knowledge discovery yet must cope with heterogeneous vocabularies, domain jargon, and long-tail content [1, 2]. Collaborative filtering and graph-based models have improved user–item representation learning [3–6], while knowledge-aware recommenders further leverage structured relations for interpretability [7, 8]. Nevertheless, their effectiveness is bounded by incomplete coverage and weak connectivity, particularly for long-tail and cold-start cases [9].

To mitigate incomplete coverage and weak connectivity, a prominent direction uses LLMs to inject commonsense relations, typically *complement* or *substitute* links then fuses them with an existing item KG (CSRec) [10]. This line is attractive because (i) it regularizes sparse graphs with priors that are broadly valid (jackets complement sweaters; lenses complement cameras), (ii) it is offline-friendly (relations are generated once and reused), and (iii) it improves item-side coverage where merchant metadata is incomplete. However, three limitations recur in practice. (1) **Granularity & intent mismatch**. Category-level commonsense smooths the space but only loosely correlates with user-specific intent. Two users who click the same *camera* page may have very different intents (e.g., *low-light astrophotography* vs. *lightweight travel kit*). Category edges cannot capture these

[*] wenqing.zheng@capitalone.com

Zheng et al., Tuning-Free LLM Can Build A Strong Recommender Under Sparse Connectivity And Knowledge Gap Via Extracting Intent. *Proceedings of the Fourth Learning on Graphs Conference (LoG 2025)*, PMLR 269, Hybrid Event, December 10–12, 2025.

fine distinctions, which limits cold-start personalization. (2) **Cross-graph fusion & alignment risk**. CSRec-style pipelines typically align an LLM-augmented commonsense graph with an existing metadata/interaction graph. This requires ontology matching, entity resolution, and confidence thresholds. Errors here introduce structural noise that is hard to debug, and different domains require repeated re-alignment [8]. (3) **Temporal & domain drift**. Off-the-shelf commonsense tends to be domain-neutral and slow-moving. Enterprise or fast-evolving domains (e.g., internal tools, niche APIs) quickly deviate; commonsense edges may become stale or irrelevant without domain grounding [11, 12].

Another line remains within a pre-existing KG and learns latent mixtures over relations to explain interactions [13–15]. This method has several strengths: it's entirely symbolic and structural, avoiding the need for LLMs; it uses relation paths to capture higher-order semantics; and it doesn't require cross-graph fusion because it operates within a single KG. However, these pre-existing KG methods have several notable gaps. First, the intent **remains latent and difficult to audit**, as it's represented as a vector or a mixture over relations rather than an explicit, human-readable node. This limits explainability and hinders downstream applications like intent analytics. Additionally, the model is **bounded by the existing KG**, meaning that if the base KG is sparse or doesn't align with how users actually express themselves, the model struggles to capture missing semantics and often overfits to popular hubs, which hurts its performance on the long tail [9]. Finally, there is a **weak tie to external knowledge**, which means issues like polysemy and synonymy in relation labels (e.g., guide, how-to, playbook) and unstandardized domain jargon persist, as the model lacks explicit text grounding or retrieval capabilities [16].

A third family places the LLM directly in the loop — either to synthesize interactions (augment clicks/purchases) or to produce rankings end-to-end [17–20]. This approach offers benefits such as broader semantic coverage and simplified modeling with fewer bespoke modules. However, the trade-offs are substantial. Inference-time LLM calls introduce significant **latency and cost**, complicating service-level agreements and A/B testing at scale, especially for multi-stage recommenders [12]. Furthermore, synthetic interactions often lead to **distribution shift and popularity bias**, as they over-represent head items and reflect the LLM's generic priors rather than the platform's actual demand patterns; even with loss calibration, realistic long-tail fidelity is difficult to guarantee [20]. Finally, **governance and reproducibility** are challenging due to frequent model or LoRA updates that change behavior and varying privacy filtering across deployments, making it difficult to maintain offline-online consistency and respond to incidents in enterprise environments [11].

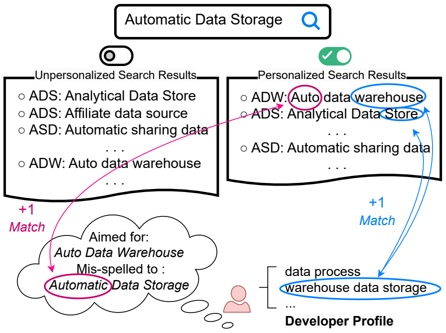
Automatic Data Storage
Unpersonalized Search Results
○ ADS: Analytical Data Store
○ ADS: Affiliate data source
○ ASD: Automatic sharing data
. . .
○ ADW: Auto data warehouse
. . .
Personalized Search Results
○ ADW: Auto data warehouse
○ ADS: Analytical Data Store
. . .
○ ASD: Automatic sharing data
. . .
+1
Match
+1
Match
Aimed for:
Auto Data Warehouse
Mis-spelled to :
Automatic Data Storage
data process
warehouse data storage
...
Developer Profile

Figure 1: In enterprise search, the queries could contain special terminologies and acronyms, where traditional search engines or personalized rerankers fail to capture the real intents under such knowledge gap. IKGR addresses the challenge via injecting fine-grained understanding to textual features.

Summarizing the pros and cons of various existing methods, we distill three core challenges to LLM and KG based recommenders.

(i) User intent extraction is an effective way to clarify and densify the KG, but this step needs to be more grounded. User profiles, queries, and enterprise documents are messy: synonyms, abbreviations, internal codenames, polysemy, and multi-lingual snippets all coexist. Extracting *what a user seeks* and *what an item satisfies* requires more than NER; it needs disambiguation (MLP = multilayer perceptron vs. marketing launch plan), normalization to a stable vocabulary, and aggregation across sources. Pure LLM prompts help but are prone to hallucination and label drift across batches. Without external knowledge grounding (retrieval of glossaries, wikis, policy pages) and schema-aware canonicalization, intent nodes will be noisy, redundant, and unmaintainable [11, 16]. This suggests the approach to elevate intent to first-class nodes extracted and normalized via RAG-guided LLMs.

(ii) The extracted intent needs to be more directly aligned with the existing user and items. The goal is to shorten semantic paths by adding edges that reflect shared or similar intents (user ↔ intent, item ↔ intent), thereby enabling information to flow even when user–item links are missing.

This step is better achieved by leveraging the extracted intents to densify a single graph, rather than through a separate cross-graph alignment stage. If densification requires merging a separate LLM commonsense graph with an interaction/metadata graph, we re-introduce entity resolution and ontology mismatch—exactly the failure modes that cause silent errors in production [8, 10].

(iii) The system should be efficient, stable, and scalable for deployment. An efficient system should keep all heavy LLM work offline. That means handling batch extraction, incremental refresh, and backfill for new content/users; caching retrieval results; and defining compatible online components (e.g., a small GNN layer) [12]. To avoid synthetic-data drift [20], improvements should also come from structural connectivity and explicit intents, not from generating pseudo-interactions.

In response to these needs, we propose LLM-based Intent Knowledge Graph Recommender (IKGR), an intent-centric knowledge graph framework. First, we instantiate intent nodes and link users and items to them via a tuning-free, RAG-guided LLM extraction pipeline, grounded in external knowledge [11, 16]. Second, we introduce mutual-intent connectivity densification, which structurally improves graph connectivity for sparse and cold-start regimes without relying on cross-graph fusion [10, 18]. Finally, a lightweight GNN layer learns over the intent-enhanced graph to produce low-latency recommendations, fully decoupled from online LLM inference [12, 17]. Empirically, IKGR consistently outperforms strong baselines on public and enterprise datasets, especially on long-tail slices, while providing interpretable pathways via explicit intents. The contributions of IKGR can be summarized as follows.

- IKGR introduces an intent-centric KG construction approach that turns user and item intents into first-class nodes. Using a tuning-free, RAG-guided LLM extractor, it canonically links users and items to intents with high precision.
- IKGR involves an intent connectivity densification step, which shortens semantic paths between users and long-tail items, improving sparsity and cold-start performance without cross-graph fusion.
- IKGR's pipeline is offline and low-latency online by construction, and consistently outperforms state-of-the-art KG or LLM baselines on public and enterprise datasets, with notable improvements on long-tail and cold-start slices.

2 Related Works

LLM and KG based Recommenders. Recent responses to sparsity and cold start fall into four camps. (1) LLM-assisted item-side augmentation injects category/attribute-level commonsense (e.g., complement/substitute) and fuses it into an item KG (CSRec) [10]. This regularizes sparse graphs with broadly valid priors and is offline-friendly, yet it struggles to align with user-specific intent (two *camera* clicks can imply very different needs), requires brittle cross-graph ontology/entity alignment, and drifts in fast-moving or enterprise domains without domain grounding [8, 11, 12]. (2) KG-based implicit-intent modeling (KGIN) stays inside a single KG and explains interactions via latent mixtures over relations [13]. It avoids fusion and can exploit multi-hop paths, but the *intent* remains a non-auditable latent vector, performance is bounded by the coverage of the base KG (hurting the long tail), and weak ties to text/external knowledge leave synonymy/polysemy unresolved [9, 16]. (3) LLM-as-Recommender / synthetic interactions broaden semantic coverage or even replace ranking end-to-end [17, 18, 20], but introduce inference-time cost and tail latency, amplify popularity bias in generated data, and complicate governance and reproducibility under frequent model updates [11, 12]. (4) Text/multimodal node enrichment improves representations by attaching reviews/UGC/features to nodes, yet leaves knowledge ungraphized—no new edges/nodes for message passing or structural densification [21, 22]. Taken together, current trends either inject coarse item-side priors, keep intent latent and hard to use beyond ranking, or pay online-LLM costs and stability penalties.

Recommender Approaches Discovering Intent. Recent works explore intent modeling by leveraging external signals such as search queries and knowledge graphs. UDITSR [23] jointly models search and recommendation by utilizing explicit search queries to infer implicit demand intents, while a dual-intent translation mechanism captures relationships between inherent intent, demand intent, and item interactions. Alternatively, knowledge graph-based methods, such as KGIN [24], refine intent discovery by representing user-item interactions through fine-grained relational paths, improving interpretability and recommendation quality. These approaches demonstrate that integrating external intent signals and structured relational modeling enhances intent-aware recommendations.

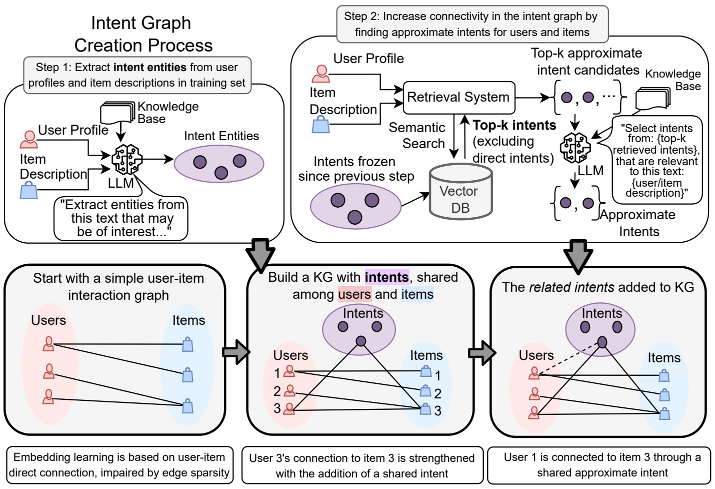
Intent Graph
Creation Process
Step 1: Extract **intent entities** from user profiles and item descriptions in training set
Knowledge Base
User Profile
Item Description
LLM
Intent Entities
"Extract entities from this text that may be of interest..."
Step 2: Increase connectivity in the intent graph by finding approximate intents for users and items
User Profile
Item Description
Retrieval System
Top-k approximate intent candidates
Knowledge Base
Semantic Search
Top-k intents (excluding direct intents)
Intents frozen since previous step
Vector DB
"Select intents from: {top-k retrieved intents}, that are relevant to this text: {user/item description}"
LLM
Approximate Intents
Start with a simple user-item interaction graph
Users
Items
Build a KG with **intents**, shared among users and items
Intents
Users
Items
1
2
3
1
2
3
The *related intents* added to KG
Intents
Users
Items
Embedding learning is based on user-item direct connection, impaired by edge sparsity
User 3's connection to item 3 is strengthened with the addition of a shared intent
User 1 is connected to item 3 through a shared approximate intent

Figure 2: IKGR's graph augmentation steps to build intent knowledge graph with LLM. IKGR overcomes knowledge gap via in-context learning, avoids synthetic noise by focusing on the simple node-level intent entity retrieval task, and being light weight and tuning-free.

LLM-Based Interaction Augmentation for Recommender Systems. Gen-RecSys provides a comprehensive overview of generative models in recommendation, highlighting LLM-driven natural language understanding and multimodal integration [25]. LLMRec introduces graph augmentation strategies using LLMs to enrich interaction graphs, refine side information, and denoise implicit feedback, demonstrating performance gains across benchmark datasets [20]. BLAIR further bridges language and recommendation by pretraining sentence embeddings on large-scale review data, improving item retrieval in complex natural language contexts [26]. Additionally, BERT4Rec employs bidirectional self-attention to model user behavior sequences, overcoming limitations of traditional sequential models [27]. These studies collectively showcase the potential of LLMs in augmenting interactions, refining representations, and enhancing recommendation performance.

More related works regarding data sparsity, LLM as recommender, and personalized re-rankers are detailed in Appendix C.

3 Preliminaries

We leverage this section to present the problem formulation and commonly adopted techniques across these domains, and introduce our contributions in Section 4.

3.1 Graph Formulation of Recommendation System

We consider the recommendation system where a set of registered users $\mathcal{U}$ interact over a set of items $\mathcal{I}$. Define the interactions as a graph $\mathcal{G}$, and each user u or item i is a node in the graph. The collection of user-item interactions are the edges in the graph, denoted as $\mathcal{E}$. We employ the implicit feedback protocol, where each edge of (u, i) implies the user $u \in \mathcal{U}$ consumes the item $i \in \mathcal{I}$. The goal is to learn a model that recommends the top-N items for a target user.

The goal is to learn a scoring function that is trained over some graph $\mathcal{G}'$, and predicts positive interactions among a set of (u, i) pairs at inference time. This scoring function is denoted as $g_\Theta(\{(u,i) \text{ pairs}\}; \mathcal{G}')$, parameterized by Θ. In mathematical formulation, the objective of the recommendation systems is to maximize the link prediction posterior probability of accurately predicting all interactions in the dev set:

$$\Theta^* = \arg\max_{\Theta} p(g_\Theta(\{(u,i)\}_{\text{dev}}; \mathcal{G}_{\text{train}}) = \mathcal{E}_{\text{dev}}) \tag{1}$$

where $\{(u,i)\}_{\text{dev}}$ is the dev set input user-item pairs, $\mathcal{G}_{\text{train}}$ is the graph used to train the model, $\mathcal{E}_{\text{dev}}$ is the ground truth positive edges in the dev set inputs. The formulation above means that the model maximizes the probability of accurately predicting edges on the dev set, while observing certain training graph $\mathcal{G}_{\text{train}}$.

We study a simple and effective approach to augment the graph, which adds new nodes and edges to augment the original graph $\mathcal{G}$ into a larger graph $\mathcal{G} \cup \mathcal{G}^+$. The training objective then becomes:

$$\Theta^* = \arg\max_{\Theta} p(g_\Theta(\{(u,i)\}_{\text{dev}}; \mathcal{G}_{\text{train}} \cup \mathcal{G}^+) = \mathcal{E}_{\text{dev}}) \tag{2}$$

The augmented graph $\mathcal{G}^+$ is comprised of a unified set of intent nodes and heterogeneous connections to existing graph nodes, which is visualized in Figure 2 and to be described in Section 4.1.

3.2 Knowledge Graph Convolution Layer

A Knowledge Graph (KG) is a directed graph composed of *subject-property-object* triple facts. Each triplet (e_h, e_t, r) denotes a relationship r from head entity e_h to tail entity e_t.

Similar to KGCN [28], we employ a knowledge graph convolution layer to capture structural proximity among entities in a knowledge graph. The model learns node embeddings $\boldsymbol{E} \in R^{N \times d}$ and relation embeddings $\boldsymbol{R} \in R^{T_R \times d}$, where T_R is number of relation types, N is number of nodes, and d is the embedding dimensionality. Denote the input features for some node v as $\mathbf{v}$, the set of entities directly connected to v as $\mathcal{S}_v$, then the output embedding is the summation-aggregated neighborhood entity and relation embeddings:

$$\mathbf{v}^{\text{out}} = \sigma\left(\mathbf{W} \cdot [\mathbf{v} + \text{softmax}(\boldsymbol{R}[\mathcal{S}_v]\boldsymbol{E}[\mathcal{S}_v]^T)\boldsymbol{E}[\mathcal{S}_v]] + \mathbf{b}\right) \tag{3}$$

where $\boldsymbol{R}[\mathcal{S}_v]$ and $\boldsymbol{E}[\mathcal{S}_v]$ are the relationship embedding vector set and entity embedding vector set, each containing $|\mathcal{S}_v|$ of d-dimensional vectors. $\mathbf{W}$ and $\mathbf{b}$ are transformation weights and a bias term, respectively, and σ is the activation function.

4 Methodology

The key components of IKGR involve grounded user/item intent extraction with RAG, KG densification and a GNN prediction layer.

4.1 Intents Extraction with RAG

Despite significant progress in incorporating side information into recommendation systems, introducing low-quality side information may even undermine what little signal we can glean from sparse interactions. To address this challenge, the proposed IKGR focuses on an effective and simple node-level graph augmentation. We leverage an LLM to build a knowledge graph with an additional type of node: the interaction intent entities. The intent nodes are linked to the existing graph via two types of edges: exact intent $\mathcal{E}_E$ and related intent $\mathcal{E}_R$.

In order to address challenges posed by sparse user behavior scenarios, we leverage the LLM to perform data augmentation. Unlike existing LLM recommendation methods that directly synthesize user-item interactions [20], we take a more conservative approach, tasking the LLM with a simpler, more reliable role to minimize noise in the generated outputs. This approach leverages the LLM's pre-trained common sense knowledge for accurate distillation.

We note that a forward call to the LLM with appropriate instructions is able to extract an intent entity set. For example, for item node i, we denote the set of intent entities as Ω_i, $\Omega_i = LLM($*###Item Description: {Item i description} Return a list of entities mentioned in the Item Description that the user may have intents to interact with*$)$.

4.2 Factual Knowledge Access for Grounded Extraction

The intent extraction step assumes a high quality *Item Description* paragraph describing the intent. However, the *Item Description* can often be poorly-formulated in the datasets, leading to a gap in properly extracting intent. (i) In the case of enterprise search, the existence of private enterprise knowledge makes it difficult for the LLM to fully understand the context, and (ii) in the open sourced datasets case, only the item names are available without appropriate description - though details of these items are often available via online search or baked into the LLM, so that an agent with web access is able to augment it easily [29].

We bridge this gap by feeding the intent extraction module with enterprise private knowledge or open-world knowledge. Specifically, for our adapted enterprise use-case, user profiles and item descriptions often contain abbreviations and domain-specific concepts unknown to LLMs. We curate a knowledge base of key-value pairs, where keys are the domain-specific abbreviations/concepts, and values are their explanations. We identify any such terminology present in the user profile or item description and append their corresponding explanations to the prompt under a $\#\#\#Concept\ Explanation$ tag. This enriches the context for the LLM, enabling better understanding without the need for fine-tuning. For open source datasets, the item name is first expanded into a paragraph summary by an LLM agent before it feeds into the entity extraction module.

4.3 Intent Connectivity Enrichment via RAG

The connectivity of the extracted intents could follow a very long-tail distribution, meaning that the majority of intent nodes might only link to few user/item nodes. To mitigate these challenges, a two-round process is used to extract and densify the intents KG. In the first round, a simple prompt template is used to generate specific entities, linking users and items to intent nodes. However, this initial graph may be sparsely connected, with many intent nodes linked to only a few users/items. To address this, a second round enriches connectivity by linking additional user/item nodes to existing intent nodes, avoiding the computationally expensive alternative of grouping similar intents ($\mathcal{O}(N^2)$ complexity)

In the second round, each user and item node is connected to additional intent entities from the fixed pool of intents generated in the first round. We call these new connections "related intents". During this second round of related intent selection, the existing intents for a given user/item node are excluded from the retrieval step.

Denote the intent entities obtained after the first round extraction as $\bar{\Omega}$. Given a user profile or an item description $text$, the second round intent extraction prompt can be formulated as: LLM(###Knowledge Context: ... ### Options: $R(x,\mathcal{N}\backslash\Omega_x,K)$ What are the intents mentioned in x that are the most relevant?) , where $R(x,\bar{\Omega}\backslash\Omega'_x,K)$ retrieves a group of K intents that are semantically similar to the input text x, yet have not been extracted during the first round output (Ω'_x). The full prompt is provided in Appendix A.

New users and items undergo this two-round extraction to connect them to the existing, well-connected knowledge graph. Intent construction is always applied to items (representing an item satisfying a user intent). When available, user profile data is also used for intent construction, and the resulting user and item intent nodes are merged using case-insensitive exact matches. Both rounds prompt the LLM for structured output.

4.4 Generating Recommendation Candidates with Graph Module

After the KG has been built, a graph module is used to generate the recommendation candidate list. While multiple combinations of GNN architectures and loss functions could fit, we find a simple translation layer based architecture [30] that injects the learned intents as structural priors outperforms vanilla GNN options. We leverage this intent prior GNN as the default option in the experiment sections, and discuss the details in Appendix B. We also benchmark across three other options and show results in Section 5.3.

5 Experiments

In this section, we verify the performance of the proposed recommender using both proprietary and open-source data [2]. For the proprietary data, we introduce IKGR to an enterprise knowledge search platform to test IKGR's effectiveness in re-ranking search results. We also benchmark against existing baselines on four open-source recommendation datasets to verify the results in diverse scenarios. The details of these datasets are presented in Table 1.

We use Llama-3.1-8B model for LLM inference, and all-mpnet-base-v2 from sentence transformer for encoding textual features. We retrieve the top 100 prebuilt intent candidates in the kNN retrieval step of RAG. We apply an 8:1:1 ratio when sampling the positive/negative edges for train/dev/test sets in the graph.

5.1 Baselines and Datasets

Enterprise Search. In the enterprise search setting, a set of developers query the search engine to search for datasets published by other developers within the enterprise. The search engine first uses

[2] code release: https://github.com/CapitalOne-Research/IKGR

Table 1: Statistics of datasets: Density is the ratio of interactions over #Users·#Items, #IntEdges is the total number of connectivities between intent node and other graph nodes. AvgIntDeg is the average intent node degree.

Datasets	Search	Beauty	Books	Steam	Yelp2022
#Users	36 033	40 226	251 394	281 428	1 987 898
#Items	872 678	54 542	25 606	13 044	150 347
#Inter	3.5M	0.35M	3.2M	3.5M	6.9M
Density	0.011%	0.02%	0.05%	0.095%	0.002%
#Intents	495 285	39 305	45 932	53 940	65 040
#IntEdges	6.9M	209K	390K	231K	409K
AvgIntDeg	13.9	5.3	8.5	4.3	6.3

Table 2: Performance comparison of different methods. Bold scores are the best in each row, while underlined scores are the second best.

Datasets	Metric	KGIN	CSRec	HAKG	LLMRec	RippleNet	KGCN	KTUP	IKGR
Search	HR@1	0.0074	0.0079	0.0080	0.0075	0.0024	0.0051	0.0078	**0.0086**
	HR@5	0.0162	0.0156	0.0187	0.0158	0.0118	0.0161	0.0166	**0.0202**
	HR@10	0.0263	0.0258	0.0255	0.0253	0.0218	0.0232	0.0262	**0.0267**
	NDCG@5	0.0142	0.0144	0.0139	0.0148	0.0068	0.0101	0.0137	**0.0151**
	NDCG@10	0.0154	0.0153	0.0161	0.0164	0.0087	0.0123	0.0142	**0.0172**
	MRR	0.0135	0.0136	0.0128	0.0143	0.0091	0.0100	0.0130	**0.0153**
Beauty	HR@1	0.1103	0.1401	**0.1623**	0.1563	0.0532	0.0984	0.1783	0.1369
	HR@5	0.3092	0.2044	0.2984	0.2803	0.1972	0.3120	**0.3388**	0.3316
	HR@10	0.4194	0.4293	0.4817	0.4204	0.3695	0.4204	0.4610	**0.4846**
	NDCG@5	0.2398	0.2390	0.2184	0.2293	0.1307	0.1729	0.2583	**0.2806**
	NDCG@10	0.2643	0.2433	0.2580	0.2930	0.1713	0.2345	0.2814	**0.2939**
	MRR	0.2294	0.2300	0.2402	0.2203	0.1382	0.2254	0.2581	**0.2641**
Books	HR@1	0.1218	0.1194	0.1177	0.1020	0.0480	0.0853	0.1125	**0.1251**
	HR@5	0.2983	0.3093	0.2764	0.2674	0.1691	0.2465	0.2652	**0.3197**
	HR@10	0.3204	0.3449	0.3781	0.4094	0.3553	0.3582	0.3614	**0.4248**
	NDCG@5	0.1910	0.1980	0.1874	0.1877	0.1031	0.1579	0.1921	**0.2097**
	NDCG@10	0.2573	0.2673	**0.2963**	0.2673	0.1607	0.2016	0.2415	0.2814
	MRR	0.2130	0.2203	0.2599	0.2563	0.1407	0.1832	0.2193	**0.2672**
Steam	HR@1	0.0783	0.0847	0.0960	0.0744	0.0304	0.0641	0.1060	**0.1095**
	HR@5	0.2653	0.2483	0.2665	0.2174	0.1429	0.1966	0.2445	**0.2759**
	HR@10	0.2901	0.2899	0.3170	0.3237	0.2735	0.3218	0.3126	**0.3574**
	NDCG@5	0.1691	0.1335	0.1221	0.1694	0.0915	0.1496	0.1614	**0.1735**
	NDCG@10	0.1739	0.1562	0.1771	0.1884	0.1161	0.1905	0.1809	**0.2212**
	MRR	0.2007	0.1965	0.1882	0.1872	0.1294	0.1528	0.1957	**0.2168**
Yelp2022	HR@1	0.0771	0.0936	**0.1077**	0.0724	0.0495	0.0917	0.1005	0.0989
	HR@5	0.2766	0.2355	0.2687	0.2211	0.1513	0.2405	0.2395	**0.2869**
	HR@10	0.3108	0.3362	0.3760	0.3228	0.2996	0.3395	0.3317	**0.3966**
	NDCG@5	0.1733	0.1823	0.1995	0.1931	0.1201	0.1397	0.2067	**0.2078**
	NDCG@10	0.2034	0.2164	0.2234	0.2127	0.1225	0.1861	0.2145	**0.2277**
	MRR	0.2093	0.2029	0.1842	0.1980	0.1356	0.1578	0.1906	**0.2100**

BM25 [31] to retrieve a list of item candidates, then the IKGR is applied to rerank the search results. Evaluations are done over user's feedback on how high the item of interest could be ranked among the final reranked list. In this setting, the items are enterprise datasets, which consist of text features such as dataset name, description, column names, ID labels, etc. The users are all registered within the enterprise and their developer profiles could be collected through separate channels to offer a hint about their dataset consumption preference. The IKGR is trained over historical user-dataset consumption interactions collected from separate channels. This dataset is labeled as *Search* in dataset description table and result tables.

Books, Beauty. These datasets are obtained from Amazon review[3] in [32], which contains a variety of categories. We utilize the Books and Beauty categories. We leverage the features of title, sales type, sales rank, categories, price, and brand.

Steam[4]. This is a dataset collected from Steam [33], a large online video game distribution platform. We leverage the item features of app name, genres, publisher, sentiment, specs, tags.

Yelp2022[5]. This is a popular dataset for business recommendation. Given the large size, we use the transaction records after *January 31st, 2022*. We treat the categories of businesses as attributes for items, and user compliment types as attributes for users.

[3] http://jmcauley.ucsd.edu/data/amazon/

[4] https://cseweb.ucsd.edu/~jmcauley/datasets.html#steam_data

[5] https://www.yelp.com/dataset

We employ Hit Ratio (HR), Normalized Discounted Cumulative Gain (NDCG), and Mean Reciprocal Rank (MRR) as evaluation metrics. We report HR and NDCG with $k = 1, 5, 10$. For all these metrics, the higher the value, the better the performance.

To verify the effectiveness of our method, we compare it with the following representative baselines. **KGIN** [13] extracts non-readable intent in the latent embedding space. **CSRec** [10] adds new nodes into KG with LLM, then post align the graph. **HAKG** [14] is a hierarchy KG built upon the hyperbolic space. **ConvNCF** [34] employs conv layers to learn correlations in neural collaborative filtering. **FPMC** [35] captures users' general taste as well as their sequential behaviors by combining MF with first-order Markov chains. **LLMRec** [20], an LLM as interaction synthesizer approach. **RippleNet** [7] uses a single layer of embedding translation by applying transH to user-item interaction. **KGCN** [28], a KG based GNN that captures inter-item relatedness. **KTUP** [30] uses a single layer of embedding translation to user-item interaction.

5.2 Result Analysis

Observing the results in Table 2, different datasets have different levels of knowledge gaps, with the Search dataset possibly having the largest knowledge gap for the LLM. This could be indicated by the number of intents and average intent node degrees in Table 1, where the Search dataset's intent count is about one magnitude larger than other datasets. From the approaches perspective, IKGR shows new state-of-the-art performances especially on the Search dataset.

Each reported score in Table 2 corresponds to the average of five independent runs with different random seeds. To assess statistically significancy, we employed a two-tailed paired t-test with $\alpha = 0.05$ for the per-run results of IKGR against the strongest baseline shown in Table. 3. The best baseline is selected per dataset according to the average MRR score. A result is considered statistically significant if the null hypothesis (*no difference between means*) is rejected at the $p < 0.05$ level. In addition, we report 95% confidence intervals computed via bootstrapping (with 10,000 resamples) to quantify the uncertainty of performance estimates. As shown in Table 3, the performance gains of IKGR are statistically significant ($p < 0.05$) on four out of five datasets. Even on the *Yelp2022* dataset, where margins are smaller, IKGR maintains a consistent advantage of performance gap from baselines.

Table 3: Statistical significance test with baselines.

Dataset	**Best Baseline**	p**-value**	**95% CI (IKGR)**
Search	LLMRec	0.021	[0.0148, 0.0158]
Beauty	KTUP	0.038	[0.2605, 0.2677]
Books	HAKG	0.034	[0.2614, 0.2730]
Steam	KTUP	0.018	[0.2089, 0.2347]
Yelp2022	KGIN	0.061	[0.2055, 0.2144]

5.3 Ablation Studies

To investigate the influence of each component over the enterprise search data, we conduct a list of experiments that drop each component in the proposed pipeline and compare with the original pipeline on the Search and Beauty dataset. Results are presented in Table 4.

Table 4: Ablation Study

Dataset	Versions	HR@1	HR@5	HR@10	NDCG@5	NDCG@10	MRR
Search	Opt1. Int. Prior GNN	**0.0086**	**0.0202**	**0.0267**	**0.0151**	**0.0172**	**0.0153**
	Opt2. Vanilla GNN	0.0078	0.0184	0.048	0.0132	0.0152	0.0130
	Opt3. Vanilla Trans.	0.0082	0.0193	0.0256	0.0140	0.0164	0.0139
	Opt4. Vanilla Scoring	0.0084	0.0199	0.0258	0.0142	0.0161	0.0137
	No Related Intent	0.0077	0.0185	0.0254	0.0147	0.0163	0.0134
	No Intent	0.0073	0.0175	0.0240	0.0131	0.0150	0.0125
Beauty	Opt1. Int. Prior GNN	**0.1369**	**0.3316**	**0.4846**	**0.2806**	**0.2939**	**0.2641**
	Opt2. Vanilla GNN	0.1294	0.3201	0.4299	0.2537	0.2674	0.2502
	Opt3. Vanilla Trans.	0.1311	0.3193	0.4272	0.2519	0.2566	0.2439
	Opt4. Vanilla Scoring	0.1332	0.3214	0.4249	0.2555	0.2579	0.2522
	No Related Intent	0.1266	0.3284	0.4462	0.2643	0.2741	0.2536
	No Intent	0.1183	0.2984	0.4093	0.2463	0.2453	0.2399

In Table 4, *Opt1. Int. Prior GNN* means the full IKGR version without component dropping. *Opt2. Vanilla GNN* means to use a plain GNN to make predictions over the generated user-intent-item graph. *Opt3. Vanilla Trans.* means removing both GNN and intent-aware scoring function, and only use a plain graph translation layer. *Opt4. Vanilla Scoring* means removing the intent prior scoring. These four options correspond to four candidates in modeling the intent graph, all detailed in Appendix B. *No Related Intent* means dropping the second round of *related intent* retrieval using RAG, and only exact intent edges are presented in the graph, without the related intent edges. *No Intent* drops all intent nodes and simply use the GNN of IKGR to predict over user-item graph.

As observed in the results shown in Table 4, the relative contributions of each components of IKGR can be estimated. The most significant observation is that the intent edges augmentation steps as a whole helps to densify the graph almost twice, as reflected by the #IntEdges and #Inter in Table 1, and the MRR performance correspondingly improved 22% (0.0125 to 0.0153 in Search dataset). This confirms that adding intent relations to the graph improves the IKGR performance while densifying the graph with meaningful knowledge connectivities.

The proposed intent node embedding improves the performance by offering a straightforward embedding structure between user and item vectors, hence guiding the learning procedure, as shown in Table 4 that learning the relation vectors from scratch and removing the intent based scoring both harms the performance.

5.4 Cold Start Metrics

We use the Books, Steam, Yelp2022 datasets to evaluate the cold start setting. The evaluation set is chosen to be a subset of edges whose end nodes both have less than or equal to 3 interactions (node degrees). The results are presented in Table 5. As can be seen from the table, IKGR achieves better performance than other methods on the tail edge set, validating its effectiveness of graph augmentation in dealing with cold start.

Table 5: Performance comparison over tail edges

Dataset	**Books**			**Steam**			**Yelp2022**		
Setting	**HR@10**	**NDCG@10**	**MRR**	**HR@10**	**NDCG@10**	**MRR**	**HR@10**	**NDCG@10**	**MRR**
IKGR	**0.4085**	**0.2791**	**0.2630**	**0.3482**	**0.2218**	**0.2005**	**0.3684**	**0.2191**	**0.1904**
ConvNCF	0.2699	0.1373	0.1382	0.2546	0.1548	0.1225	0.2436	0.1256	0.1194
FPMC	0.2810	0.1427	0.1326	0.2897	0.2010	0.1453	0.2573	0.1634	0.1429
KTUP	0.3114	0.2044	0.1679	0.3340	0.2186	0.1898	0.3104	0.1944	0.1774

5.5 Hyperparameter Sensitiveness

To test how model behaves under different hyperparameters, we computed top-k in the kNN retrieval step of RAG, and number of GNN layers in the model architecture. These experiments are conducted on the Books dataset.

The hyperparameter sensitivity results are shown in Table 6. As seen from the table, the model kNN saturates at k=100, which validates our architecture choice of k that both ensures performance and avoid over lengthy token sequences.

Table 6: Results on hyperparameter configurations

Configuration	**HR@10**	**NDCG@10**	**MRR**
$k = 120$, #conv= 1	0.4262	0.2835	0.2711
$k = 100$, #conv= 1	0.4248	0.2814	0.2672
$k = 80$, #conv= 1	0.3610	0.2619	0.2520
$k = 50$, #conv= 1	0.3023	0.2205	0.1945
$k = 100$, #conv= 2	0.3735	0.2517	0.2482

5.6 Impact of Knowledge Base

To quantify the influence of the knowledge base, we use the enterprise search data to compare the extracted intent sets for two scenarios: the knowledge base appended and dropped. We computed the average number of entity extracted, and the Jaccard Similarity Coefficient ($J(A, B) = \frac{|A \cap B|}{|A \cup B|}$) for the entities under these two settings in Table 7.

Table 7: Statistics of extracted intents

Metric	**Value**
Average num entity with KB	15.5
Average num entity without KB	14.5
Avg Jaccard Similarity Coefficient	0.892

The result shows that the average number of intents does not differ much, and the Jaccard Similarity Coefficient is close to 1, meaning the knowledge base has limited impact on the extracted entity set. Our intention with the knowledge base is to serve as a lightweight information source in the event of heavy domain gap, rather than a bottleneck key component. Indeed, we have made the LLM generation task as a simple entity extraction task, hence the impact of missing structured knowledge is minimized.

6 Conclusions

In this work, we propose IKGR, a knowledge graph based recommender built with a Large Language Model (LLM). The proposed method features a data augmentation step to explicitly extract entities that the users have intents to interact with, and learns node embeddings over the knowledge graph using an embedding translation layer to combine the intent structure knowledge. This work takes the enterprise search personalization as a case study, and verifies that (1) when knowledge gap exists, using a simplified node-level augmentation task helps learn embeddings, while synthesizing interactions harms the model performance and introduces noise; (2) injecting intent structure prior into the modeling helps better capturing the semantic structure and boosts embedding learning.

References

[1] Chetan Verma, Michael Hart, Sandeep Bhatkar, Aleatha Parker-Wood, and Sujit Dey. Improving scalability of personalized recommendation systems for enterprise knowledge workers. *IEEE Access*, 4:204–215, 2015. 1

[2] Deepjyoti Roy and Mala Dutta. A systematic review and research perspective on recommender systems. *Journal of Big Data*, 9(1):59, 2022. 1

[3] Yehuda Koren, Steffen Rendle, and Robert Bell. Advances in collaborative filtering. *Recommender systems handbook*, pages 91–142, 2021. 1

[4] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, et al. Personalized re-ranking for recommendation. In *Proceedings of the 13th ACM conference on recommender systems*, pages 3–11, 2019. 17

[5] Weiwen Liu, Qing Liu, Ruiming Tang, Junyang Chen, Xiuqiang He, and Pheng Ann Heng. Personalized re-ranking with item relationships for e-commerce. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 925–934, 2020. 17

[6] Chandler Zuo, Jonathan Castaldo, Hanqing Zhu, Haoyu Zhang, Ji Liu, Yangpeng Ou, and Xiao Kong. Inductive modeling for realtime cold start recommendations. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6400–6409, 2024. 1, 16

[7] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 417–426, 2018. 1, 8, 16

[8] Hongwei Wang et al. Knowledge-aware recommendation with graph neural networks: A survey. *arXiv preprint arXiv:2112.14936*, 2021. URL `https://arxiv.org/abs/2112.14936`. 1, 2, 3

[9] Yankai Chen, Menglin Yang, Yingxue Zhang, Mengchen Zhao, Ziqiao Meng, Jian Hao, and Irwin King. Modeling scale-free graphs for knowledge-aware recommendation. *arXiv preprint arXiv:2108.06468*, 2021. URL `https://arxiv.org/abs/2108.06468`. 1, 2, 3

[10] Shenghao Yang, Weizhi Ma, Peijie Sun, Min Zhang, Qingyao Ai, Yiqun Liu, and Mingchen Cai. Common sense enhanced knowledge-based recommendation with large language model. *arXiv preprint arXiv:2403.18325*, 2024. URL `https://arxiv.org/abs/2403.18325`. 1, 3, 8

[11] <first name> Author et al. Large language models for knowledge graph construction: A survey. In *International Conference on Learning Representations (ICLR), OpenReview*, 2023. URL `https://openreview.net/pdf?id=MipDf3C38E`. 2, 3

[12] <first name> Zhang et al. Llm-enhanced knowledge graphs for recommendation: Opportunities and challenges. *arXiv preprint arXiv:2402.13840*, 2024. URL `https://arxiv.org/abs/2402.13840`. 2, 3

[13] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. Learning intents behind interactions with knowledge graph for recommendation. In *Proceedings of The Web Conference (WWW)*, pages 878–887. ACM / IW3C2, 2021. URL `https://arxiv.org/pdf/2102.07057`. 2, 3, 8

[14] Yuntao Du, Xinjun Zhu, Lu Chen, Baihua Zheng, and Yunjun Gao. Hakg: Hierarchy-aware knowledge gated network for recommendation. In *Proceedings of the 45th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 1390–1400, 2022. 8

[15] Yankai Chen, Menglin Yang, Yingxue Zhang, Mengchen Zhao, Ziqiao Meng, Jianye Hao, and Irwin King. Modeling scale-free graphs with hyperbolic geometry for knowledge-aware recommendation. In *Proceedings of the fifteenth ACM international conference on web search and data mining*, pages 94–102, 2022. 2

[16] <first name> Zhao et al. Learning from rich external knowledge for recommendation. *arXiv preprint arXiv:2204.04959*, 2022. URL `https://arxiv.org/abs/2204.04959`. 2, 3

[17] <first name> Li et al. Recformer: Large language model for recommendation. *arXiv preprint arXiv:2305.07001*, 2023. URL `https://arxiv.org/abs/2305.07001`. 2, 3

[18] <first name> Lin et al. Alignrec: Alignment-enhanced recommendation with large language models. *arXiv preprint arXiv:2306.10933*, 2023. URL `https://arxiv.org/abs/2306.10933`. 3

[19] Priyanka Dey, Daniele Rosa, Wenqing Zheng, Daniel Barcklow, Jieyu Zhao, and Emilio Ferrara. Gravity: A framework for personalized text generation via profile-grounded synthetic preferences. *arXiv preprint arXiv:2510.11952*, 2025.

[20] Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 806–815, 2024. 2, 3, 4, 5, 8

[21] Hyeyoung Ko, Suyeon Lee, Yoonseo Park, and Anna Choi. A survey of recommendation systems: recommendation models, techniques, and application fields. *Electronics*, 11(1):141, 2022. 3

[22] Ke Liang, Lingyuan Meng, Meng Liu, Yue Liu, Wenxuan Tu, Siwei Wang, Sihang Zhou, and Xinwang Liu. Learn from relational correlations and periodic events for temporal knowledge graph reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval Conference on Research and Development in Information Retrieval*, pages 1559–1568, 2023. 3

[23] Yuting Zhang, Yiqing Wu, Ruidong Han, Ying Sun, Yongchun Zhu, Xiang Li, Wei Lin, Fuzhen Zhuang, Zhulin An, and Yongjun Xu. Unified dual-intent translation for joint modeling of search and recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6291–6300, 2024. 3

[24] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. Learning intents behind interactions with knowledge graph for recommendation. In *Proceedings of the web conference 2021*, pages 878–887, 2021. 3

[25] Yashar Deldjoo, Zhankui He, Julian McAuley, Anton Korikov, Scott Sanner, Arnau Ramisa, René Vidal, Maheswaran Sathiamoorthy, Atoosa Kasirzadeh, and Silvia Milano. A review of modern recommender systems using generative models (gen-recsys). In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6448–6458, 2024. 4

[26] Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952*, 2024. 4

[27] 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*, pages 1441–1450, 2019. 4

[28] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. Knowledge graph convolutional networks for recommender systems. In *The world wide web conference*, pages 3307–3313, 2019. 5, 8

[29] Yunjia Xi, Weiwen Liu, Jianghao Lin, Xiaoling Cai, Hong Zhu, Jieming Zhu, Bo Chen, Ruiming Tang, Weinan Zhang, and Yong Yu. Towards open-world recommendation with knowledge augmentation from large language models. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 12–22, 2024. 5

[30] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The world wide web conference*, pages 151–161, 2019. 6, 8, 14

[31] John S Whissell and Charles LA Clarke. Improving document clustering using okapi bm25 feature weighting. *Information retrieval*, 14:466–487, 2011. 7

[32] J. J. McAuley, C. Targett, Q. Shi, and A. van den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR 2015*, pages 43–52, 2015. 7

[33] W.-C. Kang and J. J. McAuley. Self-attentive sequential recommendation. In *ICDM 2018*, pages 197–206, 2018. 7

[34] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. Outer product-based neural collaborative filtering. *arXiv preprint arXiv:1808.03912*, 2018. 8

[35] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 811–820, 2010. 8

[36] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013. 14

[37] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28, 2014. 14

[38] Wenqing Zheng, Edward W Huang, Nikhil Rao, Sumeet Katariya, Zhangyang Wang, and Karthik Subbian. Cold brew: Distilling graph node representations with incomplete or missing neighborhoods. *arXiv preprint arXiv:2111.04840*, 2021. 16

[39] Sein Kim, Hongseok Kang, Seungyoon Choi, Donghyun Kim, Minchul Yang, and Chanyoung Park. Large language models meet collaborative filtering: An efficient all-round llm-based recommender system. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1395–1406, 2024. 17

[40] Junda Wu, Cheng-Chun Chang, Tong Yu, Zhankui He, Jianing Wang, Yupeng Hou, and Julian McAuley. Coral: collaborative retrieval-augmented large language models improve long-tail recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3391–3401, 2024. 17

[41] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1007–1014, 2023. 17

[42] Huizhi Liang, Yue Xu, Yuefeng Li, Richi Nayak, and Xiaohui Tao. Connecting users and items with weighted tags for personalized item recommendations. In *Proceedings of the 21st ACM conference on Hypertext and hypermedia*, pages 51–60, 2010. 17

[43] Ariel Evnine, Stratis Ioannidis, Dimitris Kalimeris, Shankar Kalyanaraman, Weiwei Li, Israel Nir, Wei Sun, and Udi Weinsberg. Achieving a better tradeoff in multi-stage recommender systems through personalization. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4939–4950, 2024. 17

[44] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 1893–1902, 2020. 17

A Full Prompts And Visualizations

For the enterprise search scenario, the full prompt in the first round of intent generation is provided as follows.

```
### Task
You are given a short paragraph of description of a dataset.
   Extract a few entities mentioned in the description that
   the users may be interested in when they consume this
   dataset. Acronyms can also be included in your answer, if
   any. Acronyms connected via underscores can be broken
   apart and partially used, so as to only include the core
   part into your answer. Please discard entities that have
   too general and broad meaning, and only pick those
   specific to this dataset. Please also discard entities
   that are over-specific, such as date or seemingly highly
   specified serial numbers/markers.
Please return answer in the format of python list of string,
   such as: ["answer 1", "answer 2"]. Only return the python
   list of string. Do NOT return any other explaination words.
### Dataset Description:
{desc}
### Answer:
```

The full prompt in the second round of intent linkage densification is provided as follows.

```
You are given a short paragraph of description of a dataset,
   and a list of entity candidates. Based on the available
   entity candidates, which of them are likely to attract the
   user's attention, if the user has proven interest in the
   dataset? Please include related entities, which may not be
   diretly mentioned in the dataset description. Please make
   selections only from the available options given below.
   Please return answer in the format of python list of
   string.
### Dataset Description:
{desc}
### Available Options:
{RAG_options}
### Important Notes:
1. You can ONLY choose from the Available Options above. If
   certain entity is not mentioned in the Available Options,
   you CANNOT return that entity.
2. Please return as many related entities as possible.
3. Please only return the python list of string. Do NOT
   return any other explaination words.
### Answer:
```

For the open source dataset (e.g. Steam), the prompt for description summarization using the open world knowledge agent is as follows.

```
{item_name} is a game on Steam. Collect necessary information
   for this game, and write one paragraph summary of why a
   user is potentially interested in it. Use objective tone.
```

The above prompt converts the item name into a paragraph of LLM summary. Then LLM is used to extract the intent using a similar prompt as above. An example of extracted intents for steam dataset is as follows.

```
Origin item name: Half-Life 2
LLM summary: Half-Life 2 is a landmark first-person shooter
    that remains highly relevant today for its exceptional
    storytelling, physics-based gameplay, and rich
    world-building. Players could be interested in it because
    it offers more than just combat; the game’s silent
    protagonist, Gordon Freeman, navigates a deeply immersive
    narrative that unfolds entirely through gameplay, pulling
    you into a desperate human resistance against an alien
    empire. The revolutionary Gravity Gun allows for creative
    problem-solving and combat by manipulating objects, while
    the game’s detailed environments and memorable
    characters-like the resourceful Alyx Vance-create a
    compelling and lasting experience that set a new standard
    for the genre. Ultimately, it’s a masterclass in game
    design that is both historically significant and still
    incredibly fun to play.
Extracted Intent entities: ["storytelling", "physics-based
    gameplay", "world-building", "combat", "narrative", "human
    resistance", "alien empire", "gravity gun",
    "problem-solving", "characters", "alyx vance", "game
    design"]
```

B Intent Prior GNN

Below we specify the details of intent prior GNN and how it improves scoring with intent awareness at test time.

B.1 Knowledge Graph Embedding Translation Formulations

Due to the incomplete nature of KGs, KG completion is often leveraged as a self-supervised learning task, which predicts the missing entity e_h or e_t for a triplet (e_h, e_t, r). To this end, TransE [36], a popular knowledge graph embedding model commonly used for KG completion, enforces a translation in the embedding space: $\mathbf{e}_h + \mathbf{e}_r \approx \mathbf{e}_t$. This embedding translation is achieved via the following training objective: $\min \sum\limits_{\mathbf{e}_h, \mathbf{r}, \mathbf{e}_t} \parallel \mathbf{e}_h + \mathbf{r} - \mathbf{e}_t \parallel$.

One issue with TransE embedding is that a single relation type may correspond to multiple head entities or tail entities, leading to significant 1-to-N, N-to-1, and N-to-N issues [37]. As an improvement, TransH [37] learns different representations for an entity conditioned on different relations. It assumes that each relation owns a hyperplane, and the translation between head entity and tail entity is valid only if they are projected to the same hyperplane. It defines an energy score function for a triplet as follows:

$$f(\mathbf{e}_h, \mathbf{e}_t, \mathbf{r}) = \parallel \mathbf{e}_h^{\perp} + \mathbf{r} - \mathbf{e}_t^{\perp} \parallel \tag{4}$$

where a lower score of $f(e_h, e_t, r)$ indicates that the triplet is likely valid. $\mathbf{e}_h^{\perp}$ and $\mathbf{e}_t^{\perp}$ are projected entity vectors:

$$\mathbf{e}_h^{\perp} = \mathbf{e}_h - \mathbf{w}_r^{\mathrm{T}} \mathbf{e}_h \mathbf{w}_r \tag{5}$$

$$\mathbf{e}_t^{\perp} = \mathbf{e}_t - \mathbf{w}_r^{\mathrm{T}} \mathbf{e}_t \mathbf{w}_r \tag{6}$$

where $\mathbf{w}_r$ and $\mathbf{r}$ are two learned vectors for a specified relation r. $\mathbf{w}_r$ denotes the projection vector of its corresponding hyperplane where $\mathbf{r}$ is the relationship embedding vector.

Finally, the training of TransH encourages the discrimination between valid triplets and incorrect ones using margin-based ranking loss. Similar to KTUP [30], we employ:

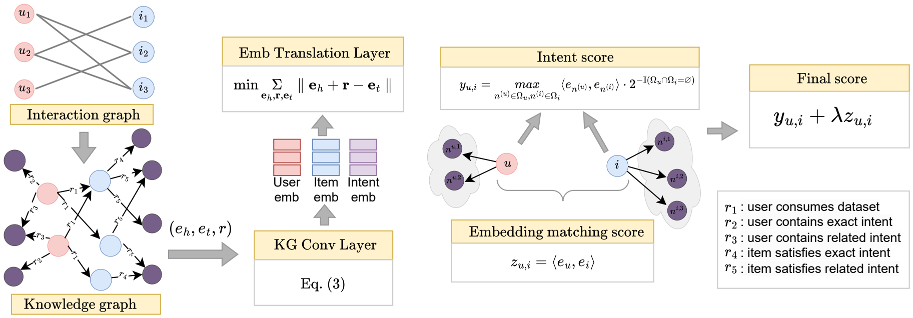
u_1
u_2
u_3
i_1
i_2
i_3
Interaction graph
r_1
r_2
r_3
r_4
r_5
Knowledge graph
(e_h, e_t, r)
Emb Translation Layer
$\min \sum_{\mathbf{e}_h,\mathbf{r},\mathbf{e}_t} \| \mathbf{e}_h + \mathbf{r} - \mathbf{e}_t \|$
User emb
Item emb
Intent emb
KG Conv Layer
Eq. (3)
Intent score
$y_{u,i} = \max_{n^{(u)} \in \Omega_u, n^{(i)} \in \Omega_i} \langle e_{n^{(u)}}, e_{n^{(i)}} \rangle \cdot 2^{-\mathbb{I}(\Omega_u \cap \Omega_i = \varnothing)}$
u
i
$n^{u,1}$
$n^{u,2}$
$n^{i,1}$
$n^{i,2}$
$n^{i,3}$
Embedding matching score
$z_{u,i} = \langle e_u, e_i \rangle$
Final score
$y_{u,i} + \lambda z_{u,i}$
r_1 : user consumes dataset
r_2 : user contains exact intent
r_3 : user contains related intent
r_4 : item satisfies exact intent
r_5 : item satisfies related intent

Figure 3: The intent prior GNN in IKGR. This module is the knowledge graph convolution layer followed by an embedding translation layer, where the intent translation is encoded as structure prior in the embedding space.

$$\mathcal{L}_{KG} = - \sum_{\substack{(\mathbf{e}_h, \mathbf{e}_t^+, \mathbf{r}^+) \in \mathcal{KG} \\ (\mathbf{e}_h, \mathbf{e}_t^-, \mathbf{r}^-) \in \mathcal{KG}^-}} \log \sigma [f(\mathbf{e}_h, \mathbf{e}_t^-, \mathbf{r}^-) - f(\mathbf{e}_h, \mathbf{e}_t^+, \mathbf{r}^+)] \tag{7}$$

where $\mathcal{KG}^-$ contains corrupted triplets constructed by randomly sampling a tail entity and relation. In practice, weight decay and normalization-enforcing losses are also applied to prevent overfitting.

B.2 Intent Prior Injected Translation layers

Having obtained the intent nodes from items and users, we train a knowledge graph-based GNN to predict user-item interaction probabilities. The architecture is shown in Figure 3, which is a knowledge graph convolution layer followed by a transH layer. We note that this component of the system can be swapped for any KG-compatible graph learning method. We posit that as long as the node-level features are efficiently decoupled and connections are made, semantic meaning can be effectively captured even with a relatively simple graph model. Therefore, we leverage a simple embedding translation-based approach to verify that building intent-item and intent-user connections helps enable more accurate user-item interaction predictions even with simpler architectures. This GNN module is implemented for the ease of the benchmarking and ablation study, so that the gain from the intent graph augmentation can be readily observed. Next, we discuss the embedding translation mechanisms and how we connect the user-item preference translation with the newly added intent embeddings at inference time.

First, we leverage a pre-trained natural language encoder to process textual node features into embeddings as input to IKGR. Given the raw embedding input, we apply a KG convolution layer described in Equation 3 to pass signals among intent, item, and user nodes to prepare for interaction prediction. Then, the KG embedding translation layer is applied to encourage semantic meaning alignment given the identified intent nodes in the graph.

In the intent-augmented knowledge graph, there are three main types of relations: user possesses intent, item satisfies intent, and user consumes item. For the first two relation types, we leverage Equation 7 to train relation representation $\mathbf{r}$ and projection vector $\mathbf{w}_r$ without modification. For the third type, user consumes item, the projection vector $\mathbf{w}_r$ is still independently trained, but the translation vector $\mathbf{r}$ is based on intent node embeddings.

The motivation behind leveraging the intent nodes to compute relation representation is that both the user and the item are decoupled into lists of intents, and when there are shared or similar intent nodes, their difference is expected to be small, i.e., $\mathbf{e}_h^\perp \approx \mathbf{e}_t^\perp$ or $\| \mathbf{r} \| \approx 0$. In this way, the intent embedding introduces direct insight for the relation embedding translations. We hence build the relation embeddings $\mathbf{r}^{u,i} \in R^d$ as follows.

Denote the intent embeddings for user node u and item node i as $\mathbf{Z}^u \in R^{|\mathcal{S}(u)| \times d}$ and $\mathbf{Z}^i \in R^{|\mathcal{S}(i)| \times d}$, where $\mathcal{S}(u)$ and $\mathcal{S}(i)$ are the intent node neighbor sets.

We first calculate two matrices: $\mathbf{P}^{u,i} \in R^{|\mathcal{S}(u)||\mathcal{S}(i)|\times 1}$, containing cosine similarities between each pair of rows in $\mathbf{Z}^u$ and $\mathbf{Z}^i$, and $\mathbf{D}^{u,i} \in R^{|\mathcal{S}(u)||\mathcal{S}(i)|\times d}$, containing $\mathbf{Z}^i_{[q,:]} - \mathbf{Z}^u_{[p,:]}$ for each row index $p = 0, 1, 2, \cdots |\mathcal{S}(i)| - 1$ and $q = 0, 1, 2, \cdots |\mathcal{S}(u)| - 1$.

Then, to enforce the relationship of more similar intent pairs inducing a smaller translation vector, the resulting vector $\mathbf{r}^{u,i}$ is computed via:

$$\mathbf{r}^{u,i} = \text{softmax}(\mathbf{P}^{u,i})^T \mathbf{D}^{u,i} \tag{8}$$

B.3 Intent Aware Scoring

To further leverage signal from the intents extracted during KG construction, we incorporate them into the user-item interaction scoring function, in conjunction with the more traditional embedding similarity score. At inference time, given a tuple of user and item (u, i), we derive *embedding matching* and *intent matching* scores between a user and an item, and the final score is a hybrid combination of them. The *embedding matching* score between u and i is the cosine similarity between their embeddings:

$$z_{u,i} = \frac{\mathbf{e}_u \cdot \mathbf{e}_i}{\parallel \mathbf{e}_u \parallel \cdot \parallel \mathbf{e}_i \parallel} \tag{9}$$

We denote a single intent extracted from item i as $n^{(i)}$, and the collection of all intents for i as Ω_i. Similarly, a single intent extracted from u and their collections is denoted as $n^{(u)}$ and Ω_u. Additionally, $\mathbf{e}_n$ is used to represent the embedding of some entity n.

The *intent matching* score between u and i is determined by:

$$y_{u,i} = \underset{n^{(u)}\in\Omega_u, n^{(i)}\in\Omega_i}{max} \frac{\mathbf{e}_{n^{(u)}} \cdot \mathbf{e}_{n^{(i)}}}{||\mathbf{e}_{n^{(u)}} \parallel \cdot \parallel \mathbf{e}_{n^{(i)}} \parallel} \cdot 0.5^{\mathbb{I}(\Omega_u \cap \Omega_i = \varnothing)} \tag{10}$$

Equation 10 has two components, the similarity component and the non-overlap punishment component. In the event that u and i share a single intent (no need to have completely equal intent set), the non-overlap punishment component will be disabled (=1), otherwise if no intents are shared, it will punish by 0.5. The choice of 0.5 is decided based on empirical comparison along the development of the approach, which shows priority over too severe punishment values (e.g. 0.1). Finally, the score for the triplet (u, i) is the hybrid mixture of the embedding score and intent matching score:

$$score(u, i) = y_{u,i} + \lambda z_{u,i} \tag{11}$$

where λ is an empirical mixer coefficient. In the experiments, the coefficient λ is set to 0.1 based on grid search result on the validation set.

In the ablation study section Section 5.3, *Opt2. Vanilla GNN* means translation layers, simply use a two-layer GNN trained with standard ranking loss. *Opt3. Vanilla Trans.* means simply use a translation layer without GNN and the scoring function of Equation 11. *Opt4. Vanilla Scoring* means keep both GNN and translation layer, but removing intent-aware scoring function of Equation 11.

C Related Works Continued

Recommender Approaches Addressing Data Sparsity. Recent works tackle the data sparsity issue through enhanced model architectures, graph-based techniques, and knowledge-aware methods. The Item History Model (IHM) improves cold-start item recommendations by directly injecting user-interaction data into the item tower and employing an inductive structure for real-time inference [6]. Cold Brew addresses sparsity in graph neural networks by distilling node representations, mitigating the impact of missing or noisy neighbors [38]. RippleNet further alleviates sparsity by propagating user preferences through knowledge graph relations, enriching item representations beyond collaborative signals [7]. These approaches demonstrate that leveraging historical interactions, distilling structural knowledge, and integrating external information effectively mitigate data sparsity in recommendation systems.

Directly Leveraging LLM as the Recommender. A-LLMRec enhances LLM-based recommendation by leveraging embeddings from state-of-the-art collaborative filtering models, excelling in both cold and warm scenarios while maintaining efficiency and model-agnostic integration [39]. CoRAL introduces collaborative retrieval-augmented prompting, addressing LLMs' reliance on semantic information by incorporating user-item interactions through reinforcement learning-based retrieval policies, significantly improving long-tail recommendation [40]. TALLRec proposes a tuning framework to align LLMs with recommendation-specific tasks, demonstrating strong generalization and efficiency even with limited training data [41]. These approaches collectively highlight the potential of LLMs as standalone recommenders by addressing cold-start challenges, enhancing collaborative reasoning, and improving alignment with recommendation-specific objectives.

Personalized Re-ranking Recommender System. Re-ranking in recommender systems aims to refine an initially ranked list to better capture user preferences and item relationships. Traditional ranking methods optimize global performance but often overlook the mutual influence between items and user-specific intent [4]. Recent approaches address these limitations by integrating semantic tag information [42], multi-stage ranking optimization [43], and item relationships [5]. Transformer-based models effectively model global item interactions, while graph-based methods leverage item relationships for improved ranking [4, 5]. Furthermore, self-supervised learning enhances sequential recommendation by mitigating data sparsity issues [44]. These methods collectively demonstrate that incorporating user personalization, item dependencies, and efficient ranking strategies significantly enhances re-ranking effectiveness.