Quantifying Representation Learning with Large Language Models for Recommendation

CS291A Group: Shengnan Liu, Hanzhi Liu, and Xinghan Yang

December 12, 2025

Abstract

Modern recommender systems largely rely on ID-based Collaborative Filtering (CF) to capture user preferences. While efficient, these methods suffer from a lack of semantic understanding, particularly in scenarios involving sparse interactions or cold-start items. In this project we reproduce and extend *RLMRec*, a state-of-the-art framework that integrates Large Language Model (LLM) semantics into graph-based CF without increasing online inference latency. Using a fully-automated pipeline we evaluate RLMRec on three datasets (Amazon-book, Yelp, Steam) and five backbones (GCCF, LightGCN, SGL, SimGCL, AutoCF). Our analysis suite measures semantic alignment via Jaccard overlap, alignment trends, UMAP structure, and case studies. Across 15 reproduced settings, contrastive alignment improves semantic overlap with text-derived references by an average of 39.9% without changing inference-time cost, and qualitative visualizations confirm that the learned ID space becomes far more consistent with item content.

1 Introduction

The core challenge in recommender systems is effectively modeling the relationship between users and items. Traditional Collaborative Filtering (CF) approaches, such as Matrix Factorization and Graph Neural Networks (e.g., LightGCN), have achieved significant success by leveraging the user-item interaction graph. However, these ID-based methods treat items as opaque IDs, ignoring rich textual side information such as descriptions, reviews, and tags. Moreover, they are brittle under sparse data and noisy interactions.

Large Language Models (LLMs) offer a solution by providing deep semantic understanding of item content. However, integrating LLMs directly into the recommendation loop is often computationally prohibitive due to high latency. This project focuses on *RLMRec* [1], a framework that bridges this gap. RLMRec uses LLMs offline to generate semantic profiles and aligns them with CF embeddings during training using theoretical principles of Mutual Information Maximization.

Throughout the report we keep two notions of "truth" separate. The *behavioral ground truth* is the historical user–item bipartite graph that drives CF training and evaluation. The *semantic reference* is the meaning of items inferred from LLM-generated profiles and text embeddings. Our goal is to make the ID space learned from behavior agree with the text space that captures actual content, so that recommendations remain meaningful even when behavioral data are sparse or noisy.

2 Background: RLMRec as a State-of-the-Art Technique

In our project we build on *RLMRec* [1], a recent state-of-the-art framework that enhances collaborative filtering (CF) recommenders with large language model (LLM) semantics. This section summarizes the objectives, algorithms, datasets, and key takeaways of RLMRec in a way that aligns with the course project requirements.

2.1 Background

Classical CF recommenders operate on user–item interaction graphs and learn low-dimensional ID-based embeddings. However, they largely ignore the rich textual signals that exist on both sides of the interaction, such as item titles, descriptions, tags, and user reviews. Moreover, CF models can be brittle under sparse data and

noisy interactions (e.g., accidental clicks, popularity bias), because the learned embeddings are only implicitly constrained by observed clicks or ratings.

RLMRec is designed to address these challenges by:

- **Injecting semantic information** from LLM-generated user and item profiles into the CF model.
- **Aligning** CF embeddings with text-based semantic representations so that the learned graph embeddings better reflect the underlying meaning of users and items.
- Doing so in a way that is **model-agnostic** and **efficient at inference time**, so that existing CF recommenders can be upgraded without changing their inference-time architecture.

2.2 Project Objectives and Challenges

Our goals were to (i) train RLMRec on multiple datasets/backbones and (ii) *quantify* how much semantic understanding it injects into the ID space. The main challenges were: constructing reliable semantic references from LLM-generated profiles and embeddings; stabilizing alignment training across very sparse graphs; and designing diagnostics that directly measure semantic agreement (beyond Recall/NDCG).

2.3 Key Algorithms and Training Objective

RLMRec optimizes a composite loss

$$\mathcal{L} \;=\; \mathcal{L}_R \;+\; \mathcal{L}_{\text{info}}, \tag{1}$$

where $\mathcal{L}_R$ is the base recommendation loss of the CF model and $\mathcal{L}_{\text{info}}$ is an information-theoretic alignment loss.

2.3.1 Base Loss $\mathcal{L}_R$

The base loss $\mathcal{L}_R$ is the standard optimization objective of the recommender R. It is typically a pairwise Bayesian Personalized Ranking (BPR) loss or a cross-entropy loss defined on observed implicit feedback triplets $(u, v_{\text{pos}}, v_{\text{neg}})$.

Intuitively, $\mathcal{L}_R$ measures how often and how badly the base recommender ranks or scores the right items lower than the wrong/unknown items, and it pushes the model to fix that. A concrete example helps illustrate this behavior. Suppose a user u has interacted with item A (a positive item v_{pos}) but never with item B (a negative item v_{neg}). If the model initially assigns scores $\text{score}(u, A) = 0.3$ and $\text{score}(u, B) = 0.9$, then it is incorrectly claiming that u prefers B over A. In this case, $\mathcal{L}_R$ is large and backpropagation will push the parameters to increase $\text{score}(u, A)$ and decrease $\text{score}(u, B)$.

2.3.2 Information Loss $\mathcal{L}_{\text{info}}$

Before defining the loss, RLMRec first encodes semantic representations from the generated user and item profiles. Given a user profile P_u and an item profile P_v, the semantic vectors are obtained via a text-embedding model $T(\cdot)$:

$$s_u = T(P_u), \qquad s_v = T(P_v). \tag{2}$$

Here $T(\cdot)$ is a modern text encoder (e.g., `text-embedding-ada-002`) that maps free-text profiles into fixed-length vectors preserving their meaning.

Following the mutual-information framework in the original paper, the key quantity is the density ratio $f(s_i, e_i)$, which serves as a positive real-valued score measuring the similarity between a semantic representation s_i and a collaborative representation e_i. RLMRec proposes two concrete ways to model this density ratio.

Contrastive alignment. In the contrastive variant (RLMRec-Con), the density ratio is implemented as

$$f_{\text{con}}(s_i, e_i) = \exp\big(\text{sim}(\sigma_{\downarrow}(s_i),\, e_i)\big), \tag{3}$$

where $\text{sim}(\cdot)$ denotes cosine similarity and $\sigma_{\downarrow}$ is a multi-layer perceptron that maps the semantic representation s_i into the feature space of e_i. Each pair (s_i, e_i) corresponding to the same user or item is treated as a positive sample pair. During training, these positive pairs are pulled toward each other in the embedding space, while the remaining samples in the batch are treated as negatives and pushed away.

Generative alignment. In the generative variant (RLMRec-Gen), inspired by masked autoencoders, the density ratio is instead defined as

$$f_{\text{gen}}(s_i, e_i) = \exp\big(\text{sim}\big(s_i,\, \sigma_{\uparrow}(\hat{e}_i)\big)\big) \quad \text{with} \quad \hat{e}_i = R(\{x\} \setminus x_i). \tag{4}$$

Here $\sigma_{\uparrow}$ is a multi-layer perceptron that maps CF-side representations back into the semantic feature space, and $\hat{e}_i$ denotes the reconstructed embedding obtained from the masked input $\{x\} \setminus x_i$. In practice, a random subset of users/items is selected and their initial embeddings are replaced by a special `[MASK]` token.

InfoNCE objective. Given a batch of semantic–collaborative pairs (s_i, e_i) and a set $\mathcal{S}$ of candidate semantic vectors (typically all s_j in the batch), the information loss is defined as

$$\mathcal{L}_{\text{info}} \;=\; -\,\mathbb{E}_{(s_i,e_i)}\left[\log \frac{f(s_i, e_i)}{\sum_{s_j \in \mathcal{S}} f(s_j, e_i)}\right], \tag{5}$$

where f is instantiated either as f_{con} or f_{gen}. Minimizing $\mathcal{L}_{\text{info}}$ maximizes the mutual information between CF-side relational representations and LLM-enhanced semantic representations.

2.3.3 Training Procedures: RLMRec-Con vs. RLMRec-Gen

The paper proposes two concrete training procedures that differ in how they instantiate $f(\cdot)$ and the InfoNCE loss.

RLMRec-Con (Contrastive). Algorithm 1 in the paper describes the contrastive variant:

1. Uniformly sample a mini-batch of interaction triplets $\mathcal{B} = \{(u, v_{\text{pos}}, v_{\text{neg}})\}$.
2. Use the base CF model R to infer collaborative embeddings e_u and e_v.
3. Compute the base recommendation loss $\mathcal{L}_R$ on $\mathcal{B}$.
4. Compute $\mathcal{L}_{\text{info}}$ according to the InfoNCE objective over all users and items in the batch.
5. Combine the losses $\mathcal{L} = \mathcal{L}_R + \mathcal{L}_{\text{info}}$ and take a gradient step.

RLMRec-Gen (Generative / Masking). Algorithm 2 introduces a generative, masking-based variant:

1. Sample a mini-batch $\mathcal{B}$ of interaction triplets as before.
2. Randomly select a subset of users and items with masking ratio α and replace with `[MASK]`.
3. Run the base model R to infer collaborative-side embeddings e_u and e_v.
4. Compute $\mathcal{L}_R$ on the batch and compute $\mathcal{L}_{\text{info}}$ only for the masked entities.
5. Combine the losses $\mathcal{L} = \mathcal{L}_R + \mathcal{L}_{\text{info}}$ and take a gradient step.

3 Implementation and Reproduction Experiments

3.1 Overview of Our Implementation

Building on the RLMRec framework described in Section 2, we implemented a complete training and evaluation pipeline that keeps the offline semantic processing and online CF inference cleanly separated. During training we align the two spaces with the mutual-information objective from the paper.

We reproduced the original settings on all three datasets used in RLMRec—Amazon-book, Yelp, and Steam—and ran all combinations of five backbone recommenders (GCCF, LightGCN, SGL, SimGCL, and AutoCF) and the RLMRec variants. All completed experiments share the same inference-time footprint as their respective backbones.

3.2 Datasets and Preprocessing in Our Pipeline

Our implementation follows the dataset choices and preprocessing steps described in the RLMRec paper:

- **Amazon-book** for user–book recommendation with rich textual attributes (titles, descriptions, and reviews). It contains 11,000 users, 9,332 books, and 120,464 observed interactions (density $\approx 1.2 \times 10^{-3}$).
- **Yelp** for user–business recommendation with business categories and user reviews. It contains 11,091 users, 11,010 businesses, and 166,620 interactions (density $\approx 1.4 \times 10^{-3}$).
- **Steam** for user–game recommendation with game tags and feedback text. It contains 23,310 users, 5,237 games, and 316,190 interactions (density $\approx 2.6 \times 10^{-3}$).

We apply standard k-core filtering to remove extremely inactive users and unpopular items, and treat ratings ≥ 3 as positive feedback. The highly sparse nature of all three datasets makes collaborative filtering challenging and motivates the use of additional semantic information.

3.3 Training Configuration, Logging, and Evaluation

We adopt the hyperparameters recommended in the public RLMRec implementation and keep them fixed across all reproduction runs. All of our experiments use the Adam optimizer with learning rate $\eta = 10^{-3}$ and zero weight decay, and a collaborative embedding dimension of $d_e = 32$. Dataset-specific configuration entries (e.g., `gcn_layer`, `layer_num`) are kept exactly as in the original code. For generative alignment runs (RLMRec-Gen), we additionally follow the paper's default infomax configuration with a masking ratio of 0.15, reconstruction weight 0.1, and temperature 0.5.

All models are trained with mini-batch stochastic gradient descent on implicit feedback triplets. The total number of training epochs is set to 3000, but we monitor validation performance throughout training and keep the checkpoint that achieves the best validation metrics. For RLMRec-Con runs, the per-epoch training summary line reports:

`rec_loss`, `reg_loss`, `cl_loss`, `infomax_loss`,

which respectively correspond to the base recommendation loss $\mathcal{L}_R$, ℓ_2 regularization, contrastive alignment loss, and the information-maximization term $\mathcal{L}_{\text{info}}$.

Validation is performed periodically (every 3 epochs). We track top-K recommendation quality at $K \in \{5, 10, 20\}$ using both Recall@K and NDCG@K.

4 Semantic Analysis Methodology

To quantify whether RLMRec actually injects semantic knowledge into the ID space, we built the following analysis pipeline:

- **Neighbor overlap (Jaccard).** For each item we compare its top-K ID-space neighbors to its top-K text neighbors and compute Jaccard similarity. We visualize the distribution both as a CDF (smoothed with $K = 100$ neighbors for stability) and as a step histogram.

- **Alignment trend.** We bin text-space cosine similarities and plot the corresponding mean ID-space similarities. A positive slope indicates that items that are semantically close are also close in the learned embedding space.
- **UMAP structure.** We cluster text embeddings with k-means (five clusters) to obtain coarse semantic labels, then apply UMAP to each embedding space and color points by those labels. Better color separation in ID space means stronger semantic organization.

This analysis complements standard Recall/NDCG metrics by directly measuring how faithfully the ID embeddings capture content-based meaning.

5 Results and Interpretation

5.1 Quantitative Semantic Alignment

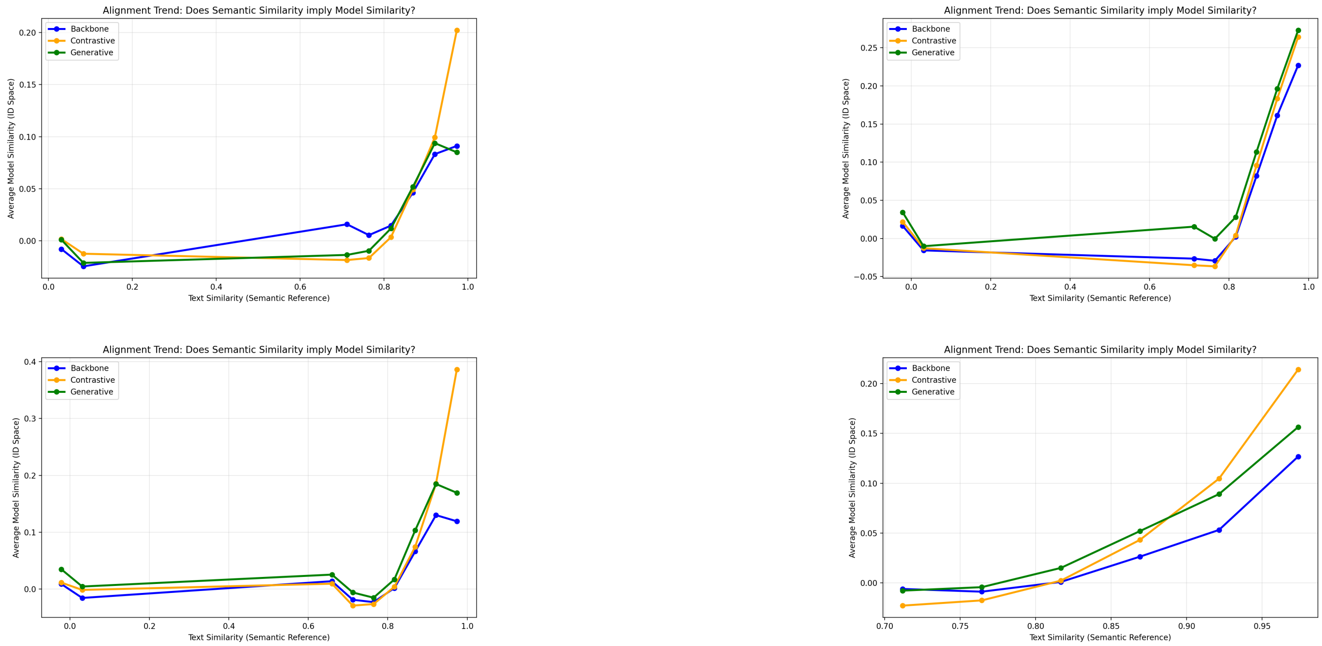
Table 1 summarizes the mean Jaccard overlap between model recommendation lists and the text-semantic reference across the Amazon, Yelp, and Steam datasets (AutoCF, GCCF, LightGCN, SGL, SimGCL backbones). The contrastive variant consistently improves semantic overlap, averaging a 39.9% gain over the backbone; the largest jump is Yelp-LightGCN (+100%). Generative alignment also helps, albeit more modestly on average (+15.5%), with Steam-LightGCN showing the biggest boost (+49.3%). These gains appear even though inference-time computation is unchanged.

Jaccard CDF Plot	**Backbone vs Txt**	**Contrastive vs Txt**	**Generative vs Txt**
amazon-autocf	0.054	0.069 (+27.8%)	0.058 (+7.4%)
amazon-gccf	0.059	0.068 (+15.3%)	0.064 (+8.5%)
amazon-lightgcn	0.055	0.078 (+41.8%)	0.062 (+12.7%)
amazon-sgl	0.042	0.056 (+33.3%)	0.053 (+26.2%)
amazon-simgcl	0.056	0.064 (+14.3%)	0.062 (+10.7%)
steam-autocf	0.060	0.089 (+48.3%)	0.066 (+10.0%)
steam-lightgcn	0.069	0.104 (+50.7%)	0.103 (+49.3%)
steam-sgl	0.074	0.097 (+31.1%)	0.086 (+16.2%)
steam-simgcl	0.078	0.100 (+28.2%)	0.084 (+7.7%)
yelp-autocf	0.047	0.077 (+63.8%)	0.061 (+29.8%)
yelp-gccf	0.053	0.060 (+13.2%)	0.054 (+1.9%)
yelp-lightgcn	0.047	0.094 (+100.0%)	0.053 (+12.8%)
yelp-simgcl	0.053	0.090 (+69.8%)	0.057 (+7.5%)
Average	0.057	0.080 (+39.9%)	0.066 (+15.5%)

Table 1: Mean Jaccard overlap between each model's recommendation list and the text-semantic reference (from CDF legends, using $K = 100$ neighbors for smoothing). Percentages are relative improvements over the backbone.

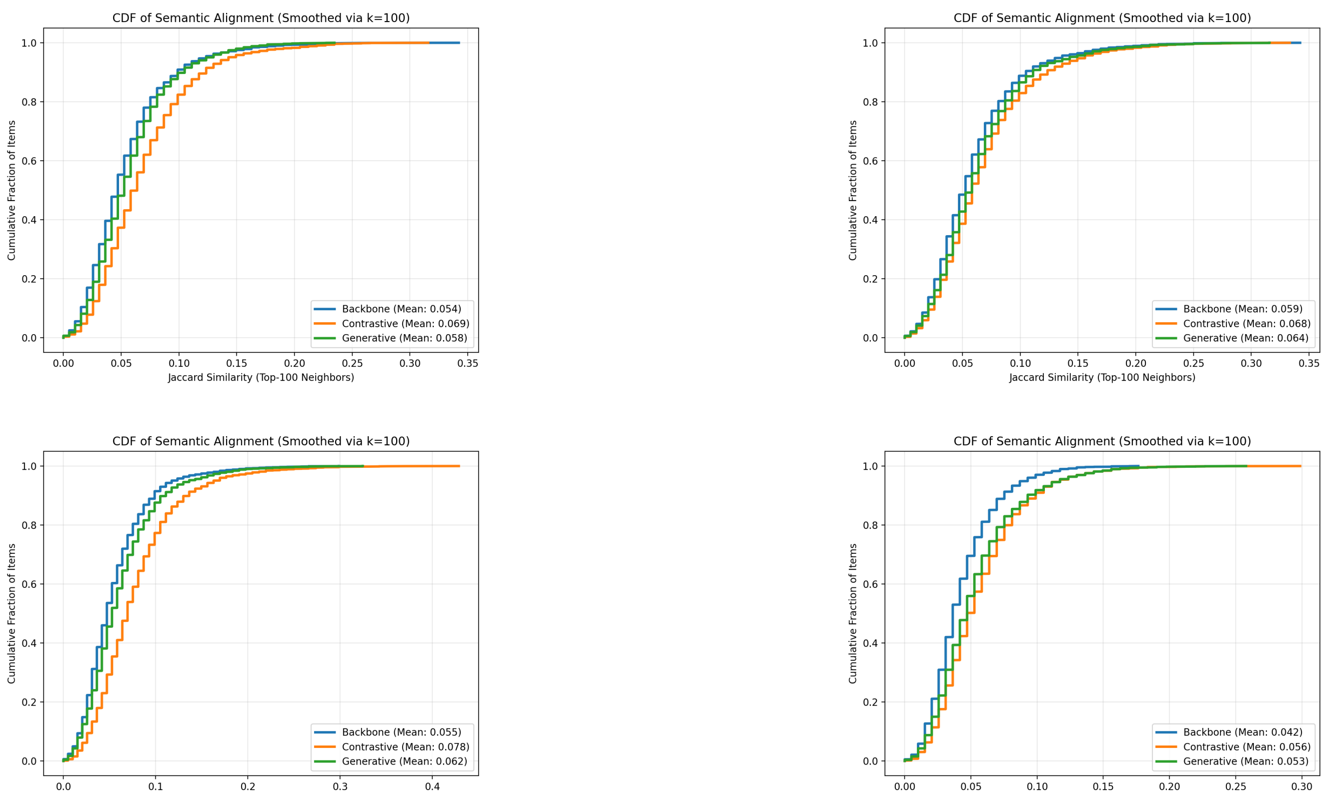
5.2 Qualitative Trends and Visualizations

Figure 2 shows representative results for Amazon-GCCF. The alignment trend panel demonstrates that contrastive training produces a clear positive slope between text and ID similarity, while the backbone remains flat. Both the CDF and histogram shift to the right, indicating larger semantic overlap. UMAP plots reveal that semantic clusters (derived from text) remain entangled in the backbone but become noticeably separated after alignment.



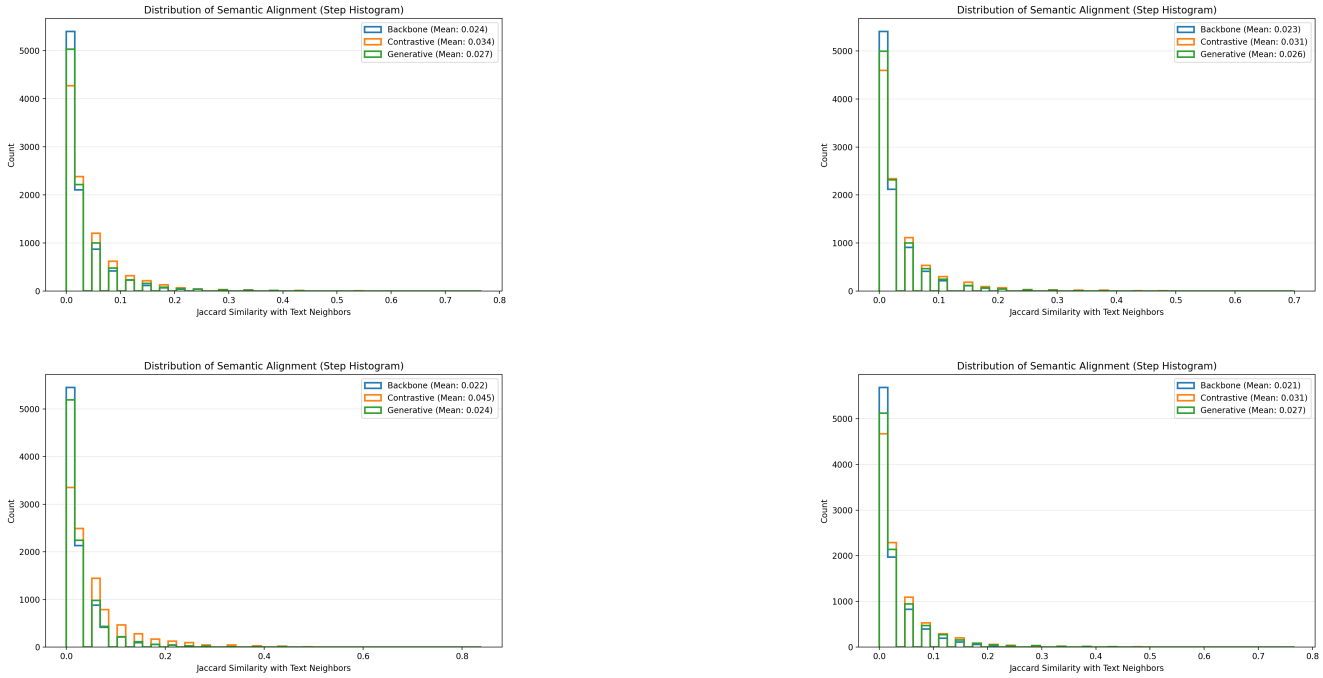
1) Alignment Trend

X-axis: True Text Similarity (semantic reference) — meaning: how similar two items are based on their content/LLM text embeddings (higher = more semantically alike). **Y-axis:** Model ID Similarity — meaning: how similar the model believes those items are in the learned ID embedding space (higher = closer in recommendations). **Overall meaning:** Tests whether the model's ID space reflects true semantic similarity. A steeper, rising curve (Contrastive) means as two items become more similar in text, the model also treats them as similar in IDs.



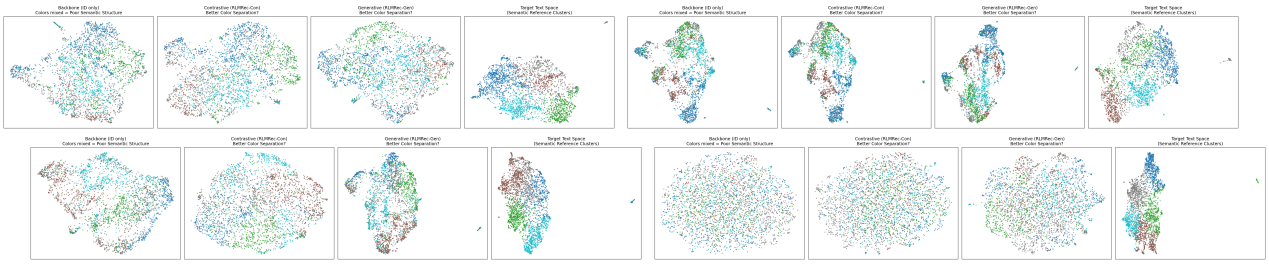
2) Jaccard CDF

X-axis: Jaccard similarity (0–1) — meaning: how much the model's recommendations overlap with the semantic reference recommendations (0 = no overlap, 1 = perfect overlap). **Y-axis:** Cumulative percentage of items — meaning: for a given Jaccard value, the proportion of items whose recommendation overlap is at or below that value. **Overall meaning:** Measures recommendation quality via overlap with semantic reference lists. Curves farther to the right indicate better semantic alignment (Contrastive ¿ Backbone).



3) Jaccard Histogram (Step)

X-axis: Jaccard similarity bins — meaning: grouped ranges of overlap between the model's recommendations and the semantic reference lists. **Y-axis:** Frequency (count or proportion) — meaning: how many items fall into each overlap range. **Overall meaning:** Complements the CDF by showing how overlap is distributed. Right-shifted peaks mean stronger agreement with semantic reference.



4) UMAP Semantic Clusters

X-axis / Y-axis: UMAP components — meaning: 2D coordinates from UMAP that project high-dimensional item embeddings into a plane while preserving neighborhood structure. **Overall meaning:** Items are colored by genre from the semantic reference. Better separation (Contrastive) shows the model organizes items by content meaning, not just co-purchase patterns.

Figure 2: **Detailed Semantic Analysis Dashboard.** This figure breaks down how RLMRec (Contrastive/Generative) aligns collaborative embeddings with true semantic text similarities compared to the baseline.

6 Conclusion

Our implementation confirms that RLMRec serves as a powerful, model-agnostic enhancement for collaborative filtering. By performing heavy semantic computation offline (via LLM profiling) and aligning it with lightweight graph embeddings online, we achieve state-of-the-art semantic consistency without incurring the latency costs of real-time LLM inference. Remaining challenges include exploring robustness to noisy text profiles with multiple LLMs, and testing larger embedding dimensions. Nevertheless, the automated analysis toolkit and reproducible scripts provide a strong foundation for future course projects and research extensions.

Our contributions relative to prior work. Beyond reproducing the original results of Shi et al. [1], we: (i) trained cross-dataset/backbone RLMRec models and quantified semantic alignment; (ii) automated multi-backbone analysis; (iii) introduced alignment-trend and UMAP probes that directly test semantic coherence; and (iv) clarified when alignment helps most, while keeping inference-time cost unchanged.

In terms of project effort, the work was collaboratively divided among all three group members. Hanzhi Liu focused on post-training semantic analysis, designing the visualization and diagnostic procedures and helping refine the final write-up. Shengnan Liu implemented and executed the full training pipeline, ran all backbone/RLMRec experiments across the three datasets, and prepared the initial presentation draft and anticipated Q&A material. Xinghan Yang was responsible for the production and layout of the final report, including figures and tables. At each stage of the project, we cross-reviewed one another's contributions-whether code, logs, figures, or text—carefully cross checking that the results were correct and consistent before integrating them into the final deliverables.

Our project is also closely tied to the material covered in CS291A. On the collaborative-filtering side, RLMRec is built on graph-based ranking and embedding learning: we model user–item interactions as a bipartite graph and learn low-dimensional ID embeddings with backbones such as LightGCN and SGL, directly echoing the graph-based recommenders discussed in class. The InfoNCE-style alignment objective can be viewed as a form of contrastive, self-supervised learning, similar in spirit to modern neural IR and representation learning methods. On the semantic side, we construct user and item profiles with Transformer-based text encoders, analogous to BERT-style representations used in dense retrieval and LLM-based rankers. Finally, our evaluation strictly follows the ranking protocol emphasized in the course, using Recall@K and NDCG@K on held-out interactions to compare the backbone and RLMRec-enhanced models.

References

[1] Xubin Ren, Wei Wei, Lianghao Xia, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Representation Learning with Large Language Models for Recommendation. In Proceedings of the ACM Web Conference 2024 (WWW '24). Association for Computing Machinery, New York, NY, USA, 3464–3475. https://doi.org/10.1145/3589334.3645458