

CLASP: Collaborative Latent Alignment from Semantic Preference Descriptions for Natural Language Based Recommendation

Dilan Dinushka, Antoine Ledent and Hady W. Lauw

Singapore Management University

Singapore

{diland.sa.2024,aledent,hadywlaw}@smu.edu.sg

Abstract

We propose CLASP, a lightweight model which performs recommendations for cold-start users for which only textual user description is provided and no previous interactions are observed. Our architecture learns lightweight latent alignment functions that project textual preference representations into a pretrained collaborative filtering latent space. The CF component can be pretrained on a large set of dense interactions from warm-start users, whilst the alignment may be trained on a distinct set of users for which textual descriptions are present, allowing an efficient use of heterogeneous information. Extensive experiments on Steam, MovieLens-1M, and Netflix demonstrate that CLASP consistently outperforms retrieval-oriented and encoder-only semantic recommendation baselines while remaining substantially more computationally efficient than LLM-based recommendation methods. Remarkably, this happens despite the fact that some of the baselines have access to item text descriptions, whilst CLASP doesn't. Lastly, we also provide a dataset which allows a granular study of the variability in user profile generation from small subsets of interactions.

1 Introduction

Deep learning has become a dominant paradigm in recommender systems, enabling models to capture complex user-item interaction patterns beyond traditional matrix factorization techniques (Koren et al., 2009; Xue et al., 2017). More recently, large language models (LLMs) have motivated extensive research on integrating semantic textual understanding into recommendation pipelines. Existing LLM-based recommender systems generally follow two directions: (1) using LLMs as auxiliary encoders to enhance collaborative filtering (CF) with textual information such as reviews and metadata (Ren et al., 2024; Liu et al., 2024; Li et al., 2025), and (2) reformulating recommendation as

an autoregressive language generation task conditioned on textual interaction histories (Kim et al., 2024; Zhang et al., 2025).

Despite their effectiveness, existing LLM-based recommendation methods exhibit several limitations. Autoregressive approaches require long textual prompts, leading to increased computational cost and context-length constraints. Moreover, generative recommendation models are susceptible to hallucination, where generated outputs may not correspond to valid catalogue items (Wu et al., 2024). In addition, many hybrid architectures continue to rely heavily on interaction histories and identifier-based representations, limiting their applicability in pure cold-start scenarios.

At the same time, lightweight collaborative filtering architectures such as matrix factorization (Koren et al., 2009), autoencoder-based recommenders (Liang et al., 2018; Truong et al., 2021), and simplified graph collaborative filtering models (He et al., 2020; Cai et al., 2023) remain highly competitive despite their relatively small parameter counts (Steck, 2019; Vančura et al., 2022). Motivated by these observations, we investigate whether lightweight architectures can directly transform textual user preferences into recommendation-oriented latent representations (Sheng et al., 2024).

We introduce CLASP, a lightweight text-based recommendation framework for cold-start recommendation. Unlike prior approaches that utilize text as auxiliary side information, CLASP relies solely on semantic user preference descriptions during inference. Furthermore, instead of generating recommendations auto-regressively, the model directly predicts recommendation scores over a fixed candidate item set through alignment with a pretrained collaborative filtering latent space.

This formulation is particularly relevant for pure cold-start settings where users possess no prior interactions. While text-based representations may not fully capture long-term interaction behaviour in

warm-start scenarios, they provide a practical mechanism for users to express preferences before sufficient interaction data becomes available (Sheng et al., 2024). Our experiments further confirm this distinction: collaborative filtering models remain stronger in dense warm-start conditions, whereas semantic preference modeling becomes increasingly beneficial under sparse/cold-start regimes.

To systematically evaluate this setting, we construct custom textual user-profile datasets derived from standard recommendation benchmarks, enabling controlled analysis under varying interaction sparsity levels. Experiments on Steam, MovieLens-1M, and Netflix demonstrate that CLASP consistently outperforms retrieval-oriented and encoder-only semantic baselines while remaining substantially more computationally efficient than autoregressive LLM-based recommendation frameworks. Our main contributions are as follows.

- We propose CLASP, a lightweight latent alignment model that maps user preference texts into a pretrained collaborative filtering latent space without autoregressive generation.
- We demonstrate experimentally that CLASP significantly outperforms semantic retrieval and encoder-only recommendation baselines in cold-start recommendation settings.
- We provide extensive ablation studies analyzing the impact of latent alignment, token-level scoring, pooling strategies, and training data availability on recommendation performance.
- We construct and release two custom textual user-profile datasets derived from standard recommendation benchmarks to support fine-grained evaluation of text-driven recommendation under varying sparsity conditions.

2 Related Work

Recent research on integrating large language models (LLMs) into recommender systems broadly follows two paradigms: (i) using LLMs as auxiliary encoders to enhance collaborative filtering (CF) representations, and (ii) employing LLMs as the primary recommendation mechanism.

In the first paradigm, LLMs encode textual information such as reviews, descriptions, and metadata to enrich user and item representations within traditional CF pipelines (Ren et al., 2024; Liu et al., 2024; Zhou et al., 2023; Alves et al., 2026). These

approaches combine semantic textual embeddings with interaction-based signals during training and inference. Although effective, they remain fundamentally dependent on historical user-item interactions, limiting their applicability in sparse or cold-start settings.

The second paradigm formulates recommendation as a language generation task, where recommendations are produced autoregressively from textual prompts describing user preferences or interaction histories (Kim et al., 2024; Zhang et al., 2025; Li et al., 2025; Sun et al., 2024). These methods leverage prompting, instruction tuning, retrieval-augmented generation, and fine-tuning strategies to adapt pretrained LLMs for recommendation objectives (Santhanam et al., 2022; Xu et al., 2026). While such approaches provide flexible semantic modeling, they introduce challenges related to computational cost, hallucination, controllability, and alignment with fixed recommendation catalogues (Fan et al., 2023; Guan et al., 2025).

Empirical evidence further suggests that purely LLM-based recommendation models often underperform interaction-based collaborative filtering approaches in warm-start settings, where abundant interaction data is available. However, in cold-start scenarios, textual preference descriptions provide a practical alternative for modeling users or items with limited interaction history (Kim et al., 2024; Sanner et al., 2023). This is particularly relevant for user cold-start recommendation, where conventional systems typically rely on popularity-based recommendations or simple onboarding strategies (Wu et al., 2023; Ganhör et al., 2024). In contrast, LLM-based approaches allow users to express preferences directly through natural language, enabling more personalized recommendations during the early stages of interaction (Lee et al., 2019; Dong et al., 2020).

3 Methodology

In this section, we describe our main contributions: subsection 3.1 describes our dataset construction and qualitative experimental observations, whilst subsection 3.2 describes our CLASP model.

3.1 Exploratory Analysis/ Dataset Generation

Existing recommendation datasets often construct a single user profile from the *complete interaction history* of a user (Ren and Huang, 2025). While such profiles provide rich semantic information,

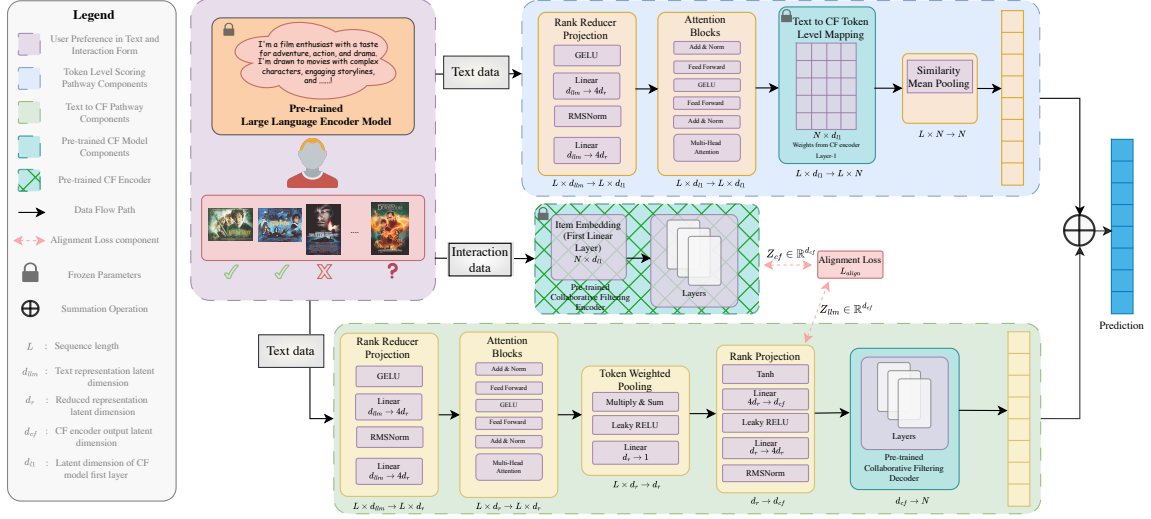


Figure 1: Overall architecture of our simplified CLASP model. The explicit interaction used to pretrain the CF model is assumed to be available for a large corpus of previous users, whilst text data may be available either for all users or a subset of them. For test users, only the text data is available.

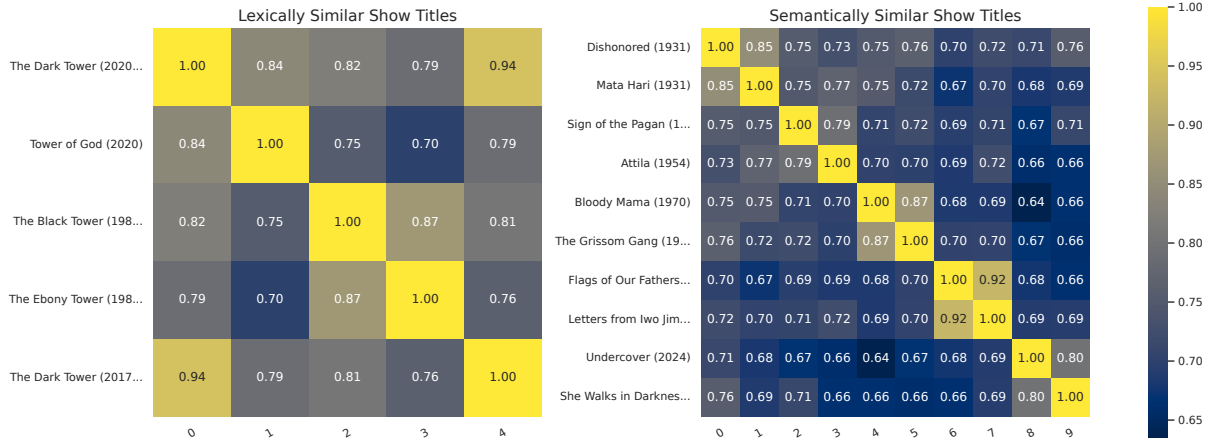


Figure 2: Similarity comparison between *item* titles across different conditions. The figure on left considers a case with shows with lexically similar titles yet different content while figure on right consider a case of semantically similar content based on [Twin Movies](#): each *consecutive* pair of movies is semantically similar whilst other movies are unrelated.

they make it difficult to isolate whether recommendation models learn meaningful preference representations or implicitly exploit full interaction data. Moreover, they do not support controlled analysis of the variability introduced by LLM-based profile generation.

To address this limitation, we construct controlled textual user-profile datasets by generating multiple preference descriptions from randomly sampled subsets of user interactions. For each user, we sample multiple interaction subsets with varying sparsity levels and generate corresponding textual preference profiles using an *Llama - 3.2B*

Instruct model, resulting in $r \times k$ profiles per user where r defines the number of subsets of interactions sampled from a given user interaction set and k defines the number of preference profile texts generated per selected subset. Importantly, all interaction subsets are specifically annotated along with the preference profiles such that they can be suppressed during evaluation processes to avoid data leakage issues. We set $r = 3$ and $k = 3$, resulting in nine textual profiles per user. We analyse the consistency of our user embeddings with respect to the variation in the profile generation procedure. As shown in Figure 3, profiles from the

same user exhibit higher similarity, indicating that textual preferences capture meaningful semantic signals.

As a point of comparison to illustrate that our cosine similarity variation is within normal ranges for similar methods, we also analyse *item* embeddings generated from movie titles, which form a key component of the success of various LLM based recommender systems such as (Ren and Huang, 2025; Hou et al., 2024; Ren et al., 2024). The item title representations are obtained from Mistral-7B (Wang et al., 2023, 2022). Figure 2 shows that LLM-based embeddings can successfully distinguish both semantic and lexical variations between item titles, explaining their strong retrieval capabilities. However, the predictive signal of lexical similarity remains stronger than semantic similarity and both are within a similar range to our own user embeddings. This is in line with the observation that CF methods still perform best in the warm-start setting, and shows that our generation strategy is sufficient to generate meaningful embeddings even from a relatively small number of interactions.

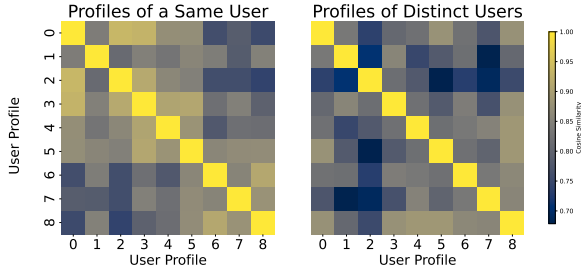


Figure 3: Similarity comparison between user profiles generated across 2 cases. The figure on the left shows the similarity between preference texts for a single user and figure on right depicts the similarity between the representations obtained for 9 distinct user profiles.

3.2 The CLASP Model

We design CLASP under two primary considerations. First, the model relies exclusively on textual user preferences, enabling direct applicability in pure cold-start settings where interaction histories are unavailable. Second, instead of generating recommendations autoregressively, the model predicts recommendation scores over a fixed item set, similar to autoencoder-based recommender systems. This formulation avoids hallucination issues and guarantees that all predictions belong to the known item catalogue. To isolate the effect of textual user preferences, the proposed framework intentionally

minimizes dependence on item-side features and identifier-based representations.

3.2.1 Model Formulation

The proposed architecture learns a mapping between semantic textual representations and a pre-trained CF latent space through two complementary pathways: (i) a *Text-to-CF* pathway that projects user preference text into the CF embedding space, and (ii) a *Token-level scoring* pathway that aligns token representations with CF item representations for ranking.

Let L denote the sequence length, N the number of items, and d_t the dimensionality of the textual embedding space. Given token-level representations extracted from a pretrained language model, the input representation is defined as $\mathbf{X} \in \mathbb{R}^{L \times d_t}$, where $\mathbf{X}$ represents the semantic preference embeddings of a user. Furthermore, d_{cf} , d_{llm} , and d_r denote the latent dimensions of the collaborative filtering model, text encoder, and proposed architecture, respectively.

Text-to-CF Pathway. Our proposed CLASP model leverages a collaborative filtering autoencoder model, specifically MultDAE. This pre-trained CF encoder has been trained on a full dataset of explicit interactions, is frozen during optimization, used as a target for our alignment loss guiding our text-to-CF pathway (cf. eq. (13)). Thus, our model is able to map preference texts to CF representations to leverage rich domain-specific knowledge of user space which cannot be inferred from limited text data alone.

First, to reduce dimensionality and adapt the textual embeddings to the recommendation space, the input representations are projected through a lightweight feed-forward transformation:

$$\bar{\mathbf{X}} = \text{RMSNorm}(\mathbf{X}\mathbf{W}_1 + \mathbf{b}_1^\top), \quad (1)$$

$$\mathbf{H}_0 = \text{GELU}(\bar{\mathbf{X}}\mathbf{W}_2 + \mathbf{b}_2^\top), \quad (2)$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_t \times 4d_r}$, $\mathbf{W}_2 \in \mathbb{R}^{4d_r \times d_r}$, $\mathbf{b}_1 \in \mathbb{R}^{4d_r}$, $\mathbf{b}_2 \in \mathbb{R}^{d_r}$, and d_r denotes the reduced latent dimensionality and the biases $\mathbf{b}_1$, $\mathbf{b}_2$ are broadcast throughout each row during the sum operation. The RMS normalization operation denotes a stabilized row-wise normalization with component-wise trainable weights as described in (Zhang and Sennrich, 2019) and in Appendix A.

The motivation behind this projection component is to reduce the parameter complexity of the model

while preserving semantically informative structures from the pretrained language representations. The projected token representations are subsequently processed through a stack of self-attention blocks (Vaswani et al., 2017):

$$\mathbf{H}_{k+1} = \text{AttentionBlock}_k(\mathbf{H}_k), \quad (3)$$

for $k = 1, \dots, N_a$, where N_a denotes the number of attention layers.

These attention modules enable the model to capture higher-order semantic dependencies across the preference text sequence, thereby improving the expressiveness of the learned user representation.

The transformed token representations $X_{N_a} \in \mathbb{R}^{L \times d_r}$ are then aggregated into a fixed-dimensional user embedding using a learnable weight-based pooling mechanism (Ennadir et al., 2026). For each token representation $\mathbf{h}_i$, an importance score is computed as

$$\mathbf{s} = \text{LeakyReLU}(\mathbf{X}_{N_a} \mathbf{w}_p), \quad (4)$$

where $\mathbf{w}_p \in \mathbb{R}^{d_r}$ is a learnable parameter vector. The normalized token weights are obtained through

$$\alpha_i = \frac{\exp(s_i)}{\sum_{j=1}^L \exp(s_j)}. \quad (5)$$

The pooled representation is then defined as token-wise weighted summation.

$$\hat{\mathbf{z}} = \sum_{i=1}^L \alpha_i \mathbf{X}_{N_a i}. \quad (6)$$

This pooling strategy enables the model to emphasize semantically informative tokens while suppressing less relevant contextual information. Alternative aggregation strategies, such as CLS pooling, mean pooling, or max pooling, may also be employed though empirically we noticed that token-weighted poolings provide better results in general.

The pooled user preference representation $\hat{\mathbf{z}} \in \mathbb{R}^{d_r}$ is subsequently projected into the latent space expected by the collaborative filtering decoder:

$$\mathbf{z} = \text{Tanh}(\mathbf{W}_4 \text{Tanh}(\mathbf{W}_3 \text{RMSNorm}(\hat{\mathbf{z}}))) \quad (7)$$

, where $\mathbf{W}_3 \in \mathbb{R}^{d_r \times 4d_r}$ and $\mathbf{W}_4 \in \mathbb{R}^{4d_r \times d_{cf}}$. The resulting vector $\mathbf{z} \in \mathbb{R}^{d_{cf}}$ represents the inferred user embedding aligned with the collaborative filtering latent space. Recommendation scores are then obtained through the pretrained CF decoder:

$$\mathbf{P}_{cf} = \text{Decoder}(\mathbf{z}). \quad (8)$$

We use the pretrained MultDAE (Liang et al., 2018) as a decoder, parametrized as a shallow MLP.

Token-Level Scoring Pathway. In addition to the latent alignment pathway, the architecture includes a secondary token-level scoring mechanism designed to align textual token representations directly with item representations derived from the collaborative filtering model inspired by ColBERT (Santhanam et al., 2022).

The textual embeddings are first transformed as

$$\mathbf{C}_0 = \text{GELU}(\mathbf{W}_6 \text{RMSNorm}(\mathbf{X}\mathbf{W}_5)), \quad (9)$$

where $\mathbf{W}_5 \in \mathbb{R}^{d_t \times 4d_r}$, $\mathbf{W}_6 \in \mathbb{R}^{4d_r \times d_{cf}^{l1}}$ and d_{cf}^{l1} denotes the dimensionality of the first layer of the CF autoencoder.

The transformed representations are processed through additional attention blocks:

$$\mathbf{C}_i = \text{AttentionBlock}(\mathbf{C}_{i-1}). \quad (10)$$

Recommendation scores are computed by matching the transformed token representations against item representations from the frozen CF encoder:

$$\mathbf{P}_c = \frac{1}{L} \sum_{i=1}^L \mathbf{C}_i \mathbf{W}_{cf}^\top, \quad (11)$$

where $\mathbf{W}_{cf} \in \mathbb{R}^{N \times d_{cf}^{l1}}$ denotes the weight matrix of the first layer of the frozen CF encoder.

This component is particularly useful in highly sparse datasets where collaborative filtering signals become weak. In such settings, the token-level pathway acts similarly to a content-based recommendation mechanism by directly associating semantic textual features with item representations (Ostendorff, 2020; Grootendorst, 2022).

The final predictions for a user are obtained by

$$P = P_{cf} + \mu P_c \quad (12)$$

where μ is a hyperparameter affecting the weightage from the token scoring path to the prediction.

Training Objective: Our training objective is comprised of 2 components: (1) An *alignment objective*, which guides the text-to-CF branch's from the user text to usable CF representations and (2) A *ranking objective*, trains the model to perform the ranking task directly from the available text.

We use the mean squared error between the pooled text user preference and corresponding CF-model based user representation. Formally,

$$L_{alignment} = \|z - u_{cf}\|^2 = \sum_{i=1}^{d_{cf}} [z - u_{cf}]_i^2 \quad (13)$$

where $u_{cf} \in \mathbb{R}^{d_{cf}}$ denotes the user representation obtained via the CF encoder model.

The *ranking loss* is defined as multinomial loss between the final prediction of the user and their corresponding true interaction. Formally, writing $y_i = 1$ to indicate that the user has interacted with item i and $z \in \mathbb{R}^N$ for the set of prediction scores output by the model,

$$\tilde{y}_i = \frac{y_i}{\sum_{k=1}^N y_k}, \quad p_i = \frac{e^{z_i}}{\sum_{k=1}^N e^{z_k}} \quad (14)$$

$$L_{\text{ranking}}(\text{user}) = - \sum_{j=1}^N \tilde{y}_j \log p_j. \quad (15)$$

Thus, final loss is defined as,

$$L = \lambda L_{\text{alignment}} + L_{\text{ranking}}. \quad (16)$$

where $\lambda \in \mathbb{R}$ is a hyper-parameter adjusting the weight from each of the alignment related components (CF pathway).

4 Experimental Evaluation

To evaluate the effectiveness of the proposed architecture, we conduct experiments on three widely used recommendation benchmarks: Steam (Ren et al., 2024), MovieLens-1M (ML-1M) (Harper and Konstan, 2015), and Netflix (Community, 2025). These datasets were selected due to their varying sparsity characteristics, interaction distributions, and domain diversity, enabling evaluation under both relatively dense and highly sparse recommendation settings. Our dataset preparation follows the standard Autoencoder literature (Liang et al., 2018) which consider user-level splits. This decision naturally models the validation and test set users as cold start users during evaluation phases as these users will not be involved during training phase thus avoiding data leakage.

Table 1 summarizes the dataset statistics computed from the training splits. Among the evaluated datasets, Steam represents the sparsest setting, with substantially fewer interactions per user and item compared to ML-1M and Netflix. In contrast, Netflix exhibits significantly denser interaction patterns, with a large number of users and high interaction counts.

Training splits and hyperparameter configurations: Following standard recommendation evaluation protocols within Autoencoder paradigm where the data splits were done based on user level which helps to mimic the cold start user cases

	Steam	ML-1M	Netflix
Num. Users	13986	32598	277891
Num. Items	5237	3452	17766
Min interactions / item	1	9	1
Max interactions / item	2379	15498	91919
Mean inter. / item	60	907.73	1921.62
Min interactions / user	9	9	5
Max interactions / user	202	536	2142
Mean inter. / user	22.67	96.13	122.85

Table 1: Dataset statistics (training split).

with validation/test sets, we report Recall@ K and NDCG@ K metrics. Recall measures the proportion of relevant items successfully retrieved within the top- K predictions, while NDCG evaluates ranking quality by accounting for the positional ordering of relevant recommendations. In all cases, we used single attention block in each pathway. For hyperparameters we use $(\mu = 1, \lambda = 1)$, $(\mu = 0.01, \lambda = 1)$, and $(\mu = 1, \lambda = 0.001)$ for Netflix, MovieLens-1M and Steam datasets respectively. Further details regarding the experimental setup is included in the appendix.

Base model and item information: For all experiments involving CLASP, we used the BERT-based *all-MiniLM-L6-v2* (Reimers and Gurevych, 2020) model provided by Hugging Face to generate the token embeddings. For baselines that rank items via similarity scores between user and item representations, we relied on item titles to construct item representations. These representations constitute auxiliary information available to those baselines during prediction, which is not incorporated into our proposed model. Nevertheless, as demonstrated in the experiments below, our model consistently achieves superior performance despite both the *absence* of such *additional information* and a significantly *lower parameter count* compared to the best baseline available, namely EASYRec (~120M vs. ~10M trainable parameters).

Computational Efficiency: In addition, we note that training the second best method, EASYRec, requires individually processing all the (user text, item text) combinations through a trainable LLM, which dramatically increases the compute time in the case of large datasets such as MovieLens and Netflix. In contrast, our CLASP model only processes text through a frozen LLM once for each user, resulting in much faster training. In our experiments on MovieLens, our model converges in 10 epochs with a training time of 30 minutes whilst EASYRec takes around 3 hours to reach convergence in 0.15 epochs.

4.1 Baselines

We compare our model against a diverse set of recommendation baselines spanning collaborative filtering, retrieval-based, and encoder-only text representation methods. The performance is reported in Table 2. The considered baselines include:

- **Popularity**: a non-personalized recommendation baseline that ranks items according to training set interaction frequency of items.
- **EasyRec**: an LLM-based collaborative filtering model showing strong performance with text inputs (Ren and Huang, 2025).
- **Blair, ColBERT**, and **SiBraR**: retrieval-oriented methods utilizing textual similarity and token-level matching strategies (Hou et al., 2024; Santhanam et al., 2022).
- **Encoder-only baselines**: models constructed directly from pretrained text encoders without collaborative alignment, including Mistral-7B, BGE-Multilingual-Gemma2, RoBERTa variants, and Sentence-T5 models (Wang et al., 2023, 2022; Xiao et al., 2023; Chen et al., 2024; Ni et al., 2022).

For all the baselines, we utilised the code provided by the authors and followed their recommended hyper-parameters. Similarly, for LLM based models we used their provided checkpoints to initialise the training wherever required.

4.2 Results

The results shown in table 2 demonstrate that the proposed model consistently outperforms all baseline methods across all datasets and evaluation metrics. The largest improvements are observed on the MovieLens-1M and Netflix datasets, which exhibit comparatively larger interaction variance.

Compared to encoder-only retrieval baselines, the proposed architecture achieves substantially higher Recall and NDCG scores. These results indicate that directly utilizing pretrained semantic embeddings is insufficient for recommendation tasks without alignment to recommendation-oriented latent structures, especially when the item catalogue is large. However, we observe that the number of parameters of the model used to generate the text representation seems to have a positive correlation towards ranking performance as well. Although EASYRec and Blair remain the strongest baselines

as Large Language based model trained for recommendation task, CLASP consistently surpasses them across all datasets. This suggests that the current LLM architectures are not entirely suitable for ranking tasks directly and still require architectural changes to embrace the unique properties that arise in recommendation datasets such as sparsity, overlapping preferences and many to one mapping of text preference representation to collaborative filtering representations. Similarly, retrieval-based approaches such as ColBERT and SiBraR perform considerably worse than the proposed framework (and other baselines), indicating a lack of explicit optimization toward recommendation objectives.

Overall, the results demonstrate that lightweight latent alignment architectures can effectively bridge textual preference representations with collaborative recommendation spaces, achieving strong recommendation performance without relying on autoregressive generation mechanisms or large-scale end-to-end LLM fine-tuning.

4.3 Impact of Textual Training Data Size

To analyse the data efficiency of the proposed architecture, we evaluate model performance under varying fractions of the available training data, i.e. the number of users where a profile text is available. Table 3 presents the results averaged across three independent runs. Here, the frozen CF component still has access to the full training data: thus, these experiments showcase our model’s ability to efficiently use heterogeneous information sources.

The results indicate that the proposed CLASP remains strong even under moderate reductions in training data. Reducing the training set to 50% results in only minor performance degradation across all metrics, suggesting that the model can effectively utilise semantic information encoded in textual preferences even under reduced supervision. In addition, CLASP consistently outperforms the strongest baseline EASYRec in all sparsity regimes. For more extreme sparsity regimes, the degradation remains gradual rather than catastrophic, indicating that textual representation can be meaningfully exploited by our model even in extreme cold-start scenarios where (1) no interactions are provided for text users and (2) very few textual profiles are provided during training.

4.4 Ablation Study

To understand the contribution of each architectural component, we conduct an ablation study by selec-

Model	Steam			ML-1M			Netflix		
	N@100	R@20	R@50	N@100	R@20	R@50	N@100	R@20	R@50
Blair	0.1886	0.1091	0.1841	0.1605	0.0963	0.1916	0.0303	0.0154	0.0328
ColBERT	0.0939	0.0540	0.0788	0.0118	0.0042	0.0127	0.0009	0.0003	0.0009
EASYRec	0.2046	0.1205	0.1968	0.1513	0.0862	0.1824	0.1010	0.0659	0.1315
SiBraR	0.0531	0.0274	0.0527	0.1580	0.1037	0.1989	0.1404	0.0851	0.1683
Popularity	0.1643	0.0883	0.1627	0.1939	0.1306	0.2273	0.1462	0.0752	0.1585
CLASP	0.2963	0.1488	0.2387	0.2747	0.1594	0.2808	0.2255	0.1311	0.2426
Encoder-based Retrieval Baselines									
Mistral-7B	0.0886	0.0504	0.0824	0.0800	0.0474	0.0971	0.0124	0.0046	0.0139
BGE-Multilingual-Gemma2	0.0367	0.0201	0.0335	0.1121	0.0681	0.1358	0.0110	0.0023	0.0109
RoBERTa-Large-v1	0.0393	0.0183	0.0358	0.0276	0.0137	0.0320	0.0087	0.0023	0.0081
RoBERTa-Base-v2	0.0417	0.0223	0.0412	0.0267	0.0134	0.0300	0.0034	0.0017	0.0035
Sentence-T5-Base	0.0476	0.0252	0.0443	0.0378	0.0214	0.0439	0.0122	0.0050	0.0135
Sentence-T5-XXL	0.0668	0.0376	0.0646	0.0532	0.0295	0.0621	0.0106	0.0041	0.0108

Table 2: Comparison of the proposed method against collaborative filtering, retrieval-based encoder-only baselines across Steam, ML-1M, and Netflix datasets. Best results for each dataset are highlighted in bold.

Data ratio	EASYRec			CLASP		
	Recall@20	Recall@50	NDCG@100	Recall@20	Recall@50	NDCG@100
1.0	0.1205	0.1968	0.2046	0.1488	0.2387	0.2963
0.8	0.1183	0.1947	0.2040	0.1439	0.2328	0.2903
0.5	0.1145	0.1876	0.1990	0.1366	0.2252	0.2783
0.3	0.1147	0.1871	0.1986	0.1291	0.2164	0.2668
0.1	0.1081	0.1780	0.1886	0.1121	0.1974	0.2440

Table 3: Impact of textual training data size on recommendation performance based on Steam dataset compared to EASYRec. Results are averaged across two runs.

Removed Component	MovieLens-1M			Steam		
	Recall@20	Recall@50	NDCG@100	Recall@20	Recall@50	NDCG@100
Content pathway	0.1585	0.2781	0.2737	0.1448	0.2357	0.2907
CF pathway (no align. loss)	0.1250	0.2262	0.2307	0.1056	0.1842	0.2291
Alignment objective	0.1565	0.2773	0.2711	0.1483	0.2391	0.2968
Recommendation loss	0.1441	0.2588	0.2620	0.1328	0.2225	0.2718
Token-weighted pooling	0.1519	0.2703	0.2667	0.1470	0.2370	0.2944
Proposed Model	0.1594	0.2808	0.2747	0.1488	<u>0.2387</u>	<u>0.2963</u>

Table 4: Ablation study on the architecture. Results are averaged across three runs. The second best method is underlined in cases where the gap is smaller than 0.0005.

tively removing or modifying individual modules. The results are reported in Table 4. The ablation results demonstrate that all of our model components contribute non-trivially to the model performance. In addition, the collaborative filtering alignment pathway is the most critical component of the architecture. Removing the CF pathway component results in the largest performance degradation across both datasets, confirming that compatibility with the pretrained CF latent manifold is essential for effective recommendation generation. Similarly, removing the recommendation loss significantly reduces performance, suggesting that the two objectives play complementary roles. On the Steam dataset, we observed the alignment loss doesn’t significantly improve performance, likely due to the higher sparsity undermining the robustness of the CF representations (due to underperforming CF en-

coder) which leads to lower reliance on alignment objective for ranking performance.

5 Conclusion

We propose CLASP, a text-to-CF architecture which performs recommendations for cold-start users where only a self-descriptive user preference text is available. The model relies on an alignment loss to leverage information from a large CF model trained on warm-start users. Despite its relatively simple architecture and the lack of reliance on item textual descriptions, we show that CLASP can significantly outperform pure LLM based non-sequential ranking models. Finally we release two new datasets based on the MovieLens-1M and Netflix datasets, where user descriptions were generated from explicitly logged subsets of items in the training set. This allows a fair evaluation of user

text-based methods and the study of the natural variability in the LLM profile generation.

6 Limitations

Our model is primarily targeted at cold-start users where only a textual description is available, and we observe that CF methods still outperform both our method and all the baselines on warm-start users. We generate user profile texts from small sets of interactions *present only in the training set*, in contrast to existing methods such as EASYRec (Ren and Huang, 2025) which generate the profile texts from the full interaction history, introducing a risk of data leakage. Whilst this enables a well-principled separation of the errors originating from model training and profile generation, it also unmasks a key vulnerability in all text-based recommendation methods: the variability inherent in the text generation step is large and some information must be lost. Next, we note that unlike (Ren and Huang, 2025), our method doesn't incorporate item-side text. Incorporating item text into our model is a natural extension for future works and may help further improve performance on sparser datasets. However, we find it noteworthy that on the three datasets considered, we are able to outperform even the methods which *do* exploit item text. Lastly, we acknowledge that we only perform a comprehensive evaluation for a single prompting strategy.

References

- Rodrigo Alves, Vojtěch Vančura, Pavel Kordík, and Antoine Ledent. 2026. Language embeddings meet shallow autoencoders. In *Proceedings of the 34th ACM Conference on User Modeling, Adaptation and Personalization*, pages 398–402.
- Xuheng Cai, Chao Huang, Lianghao Xia, and Xubin Ren. 2023. [LightGCL: Simple yet effective graph contrastive learning for recommendation](#). In *The Eleventh International Conference on Learning Representations*.
- Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. [M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335, Bangkok, Thailand. Association for Computational Linguistics.
- HuggingFace Community. 2025. [hugginglearners/netflix-shows](https://huggingface.co/datasets/hugginglearners/netflix-shows) dataset.
- <https://huggingface.co/datasets/hugginglearners/netflix-shows>. Accessed on Hugging Face.
- Manqing Dong, Feng Yuan, Lina Yao, Xiwei Xu, and Liming Zhu. 2020. [Mamo: Memory-augmented meta-optimization for cold-start recommendation](#). In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '20*, page 688–697, New York, NY, USA. Association for Computing Machinery.
- Sofiane Ennadir, Levente Zólyomi, Oleg Smirnov, Tianze Wang, John Pertoft, Filip Cornell, and Lele Cao. 2026. [Pool me wisely: On the effect of pooling in transformer-based models](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Wenqi Fan, Zihuai Zhao, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Jiliang Tang, and Qing Li. 2023. [Recommender systems in the era of large language models \(llms\)](#). *IEEE Transactions on Knowledge and Data Engineering*, 36:6889–6907.
- Christian Ganhör, Marta Moscati, Anna Hausberger, Shah Nawaz, and Markus Schedl. 2024. [A multi-modal single-branch embedding network for recommendation in cold-start and missing modality scenarios](#). In *Proceedings of the 18th ACM Conference on Recommender Systems, RecSys '24*, page 380–390, New York, NY, USA. Association for Computing Machinery.
- Maarten R. Grootendorst. 2022. [Bertopic: Neural topic modeling with a class-based tf-idf procedure](#). *ArXiv*, abs/2203.05794.
- Ziyuan Guan, Weiyi Zhong, Weiming Liu, Yuwen Liu, Huaizhen Kou, Xiaoxiao Chi, and Xiaoran Zhao. 2025. [Large language models for recommender systems: A problem-driven survey](#). *Tsinghua Science and Technology*.
- F. Maxwell Harper and Joseph A. Konstan. 2015. [The movielens datasets: History and context](#). *ACM Trans. Interact. Intell. Syst.*, 5(4).
- Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. [Lightgcn: Simplifying and powering graph convolution network for recommendation](#). In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '20*, page 639–648, New York, NY, USA. Association for Computing Machinery.
- Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. 2024. [Bridging language and items for retrieval and recommendation](#). *ArXiv*, abs/2403.03952.
- Sein Kim, Hongseok Kang, Seungyeon Choi, Donghyun Kim, Minchul Yang, and Chanyoung Park. 2024. [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*, KDD '24, page 1395–1406, New York, NY, USA. Association for Computing Machinery.
- Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.
- Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. 2019. [Melu: Meta-learned user preference estimator for cold-start recommendation](#). In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 1073–1082, New York, NY, USA. Association for Computing Machinery.
- Xiangyang Li, Bo Chen, Lu Hou, and Ruiming Tang. 2025. [Ctrl: Connect collaborative and language model for ctr prediction](#). *ACM Trans. Recomm. Syst.*, 4(2).
- Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. [Variational autoencoders for collaborative filtering](#). In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 689–698, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.
- Yifan Liu, Kangning Zhang, Xiangyuan Ren, Yanhua Huang, Jiarui Jin, Yingjie Qin, Ruilong Su, Ruiwen Xu, Yong Yu, and Weinan Zhang. 2024. [Alignrec: Aligning and training in multimodal recommendations](#). In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, page 1503–1512, New York, NY, USA. Association for Computing Machinery.
- Ilya Loshchilov and Frank Hutter. 2019. [Decoupled weight decay regularization](#). In *International Conference on Learning Representations*.
- Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022. [Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models](#). In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, Dublin, Ireland. Association for Computational Linguistics.
- Malte Ostendorff. 2020. [Contextual document similarity for content-based literature recommender systems](#). *ArXiv*, abs/2008.00202.
- Nils Reimers and Iryna Gurevych. 2020. [Making monolingual sentence embeddings multilingual using knowledge distillation](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4512–4525, Online. Association for Computational Linguistics.
- Xubin Ren and Chao Huang. 2025. [EasyRec: Simple yet effective language models for recommendation](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 17717–17732, Suzhou, China. Association for Computational Linguistics.
- 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, page 3464–3475, New York, NY, USA. Association for Computing Machinery.
- Scott Sanner, Krisztian Balog, Filip Radlinski, Ben Wedin, and Lucas Dixon. 2023. [Large language models are competitive near cold-start recommenders for language- and item-based preferences](#). In *Proceedings of the 17th ACM Conference on Recommender Systems*, RecSys '23, page 890–896, New York, NY, USA. Association for Computing Machinery.
- Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. [ColBERTv2: Effective and efficient retrieval via lightweight late interaction](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734, Seattle, United States. Association for Computational Linguistics.
- Leheng Sheng, An Zhang, Yi Zhang, Yuxin Chen, Xiang Wang, and Tat-Seng Chua. 2024. [Language representations can be what recommenders need: Findings and potentials](#). In *International Conference on Learning Representations*.
- Harald Steck. 2019. [Embarrassingly shallow autoencoders for sparse data](#). In *The World Wide Web Conference*, WWW '19, page 3251–3257, New York, NY, USA. Association for Computing Machinery.
- Zhongxiang Sun, Zihua Si, Xiaoxue Zang, Kai Zheng, Yang Song, Xiao Zhang, and Jun Xu. 2024. [Large language models enhanced collaborative filtering](#). In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, page 2178–2188, New York, NY, USA. Association for Computing Machinery.
- Quoc-Tuan Truong, Aghiles Salah, and Hady W. Lauw. 2021. [Bilateral variational autoencoder for collaborative filtering](#). In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 292–300, New York, NY, USA. Association for Computing Machinery.
- Vojtěch Vančura, Rodrigo Alves, Petr Kasalický, and Pavel Kordík. 2022. [Scalable linear shallow autoencoder for collaborative filtering](#). In *Proceedings of the 16th ACM Conference on Recommender Systems*, RecSys '22, page 604–609, New York, NY, USA. Association for Computing Machinery.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, L ukasz

- Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.
- Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*.
- Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, Hui Xiong, and Enhong Chen. 2024. [A survey on large language models for recommendation](#). *World Wide Web*, 27(5).
- Shiguang Wu, Yaqing Wang, Qinghe Jing, Daxiang Dong, Dejing Dou, and Quanming Yao. 2023. [Coldnas: Search to modulate for user cold-start recommendation](#). In *Proceedings of the ACM Web Conference 2023*, WWW '23, page 1021–1031, New York, NY, USA. Association for Computing Machinery.
- Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. [C-pack: Packaged resources to advance general chinese embedding](#). *Preprint*, arXiv:2309.07597.
- Changliang Xu, Yukun Kang, Quan Feng, Jinghua Hua, Piji Li, Feiran Wu, Hu Wei, Xiang Chen, Sheng-Jun Huang, and Songcan Chen. 2026. [Itemrag: Retrieval-augmented generation with item-based knowledge computing for e-commerce product question answering](#). *Big Data Mining and Analytics*, 9(2):407–424.
- Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep matrix factorization models for recommender systems. In *Ijcai*, volume 17, pages 3203–3209. Melbourne, Australia.
- Biao Zhang and Rico Sennrich. 2019. *Root mean square layer normalization*. Curran Associates Inc., Red Hook, NY, USA.
- Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. 2025. [Collm: Integrating collaborative embeddings into large language models for recommendation](#). *IEEE Transactions on Knowledge and Data Engineering*, 37(5):2329–2340.
- Zhihui Zhou, Lilin Zhang, and Ning Yang. 2023. [Contrastive collaborative filtering for cold-start item recommendation](#). In *Proceedings of the ACM Web Conference 2023*, WWW '23, page 928–937, New York, NY, USA. Association for Computing Machinery.

A Additional Mathematical details

For the sake of completeness, we include definitions for some of the functions used in the main text which represent well-known operations.

Each **attention block** takes in an initial sequence $\mathbf{X}_{k-1} \in \mathbb{R}^{L \times d}$ and outputs $\mathbf{X}_k = \text{AttentionBlock}(\mathbf{X}_{k-1})$ computed through the sequence of operations below.

$$\mathbf{A}_k = \text{MultiHeadAttention}(\mathbf{X}_{k-1}), \quad (17)$$

$$\mathbf{Y}_k = \text{RMSNorm}(\mathbf{A}_k + \mathbf{A}_{k-1}), \quad (18)$$

$$\mathbf{H} = \mathbf{Y}_k \mathbf{W}_{k1}, \quad (19)$$

$$\mathbf{F} = \text{GELU}(\mathbf{H}) \mathbf{W}_{k2}, \quad (20)$$

$$\mathbf{X}_k = \text{RMSNorm}(\mathbf{F} + \mathbf{Y}_k), \quad (21)$$

where

$$\mathbf{W}_{k1} \in \mathbb{R}^{d_r \times 4d_r}, \quad \mathbf{W}_{k2} \in \mathbb{R}^{4d_r \times d_r}. \quad (22)$$

The MultiHeadAttention is calculated as

$$\mathbf{Q}_k^h = \mathbf{X}_{k-1} \mathbf{W}_Q^h, \quad (23)$$

$$\mathbf{K}_k^h = \mathbf{X}_{k-1} \mathbf{W}_K^h, \quad (24)$$

$$\mathbf{V}_k^h = \mathbf{X}_{k-1} \mathbf{W}_V^h, \quad (25)$$

$$\mathbf{A}_k^{(h)} = \text{Softmax} \left(\frac{\mathbf{Q}_k^{(h)} \mathbf{K}_k^{(h)\top}}{\sqrt{d_h}} \right) \mathbf{V}_k^{(h)} \quad (26)$$

$$\mathbf{X}_k = \text{Concat} \left(\mathbf{A}_k^{(1)}, \dots, \mathbf{A}_k^{(H)} \right), \quad (27)$$

where

$$\mathbf{W}_Q^h, \mathbf{W}_K^h, \mathbf{W}_V^h \in \mathbb{R}^{d_r \times \frac{d_r}{h}}, \quad (28)$$

$$h \in [H] \quad (29)$$

and H defines the number of attention heads.

The **RMSNormalization function** (Zhang and Sennrich, 2019) is defined for each vector $x \in \mathbb{R}^d$ by

$$\text{RMSNorm}(x)_i = \gamma_i \frac{x_i}{\sqrt{\frac{1}{d} \sum_{j=1}^d x_j^2 + \varepsilon}}, \quad (30)$$

where $\gamma \in \mathbb{R}^\gamma$ is a learnable scaling parameter and ε is a small constant introduced for numerical stability. When taking a matrix/sequence argument $\mathbf{Z} \in \mathbb{R}^{L \times d}$, the function is applied row by row to return $\text{RMSNorm}(\mathbf{Z})$ with a single set of $\gamma \in \mathbb{R}^d$ being used for each row. Each separate application of the RMSNorm function in the algorithm introduces a new set of parameters γ .

B User Semantic Preference Profiles

Here, we give some example user profiles provided both by our method and by the existing methods to highlight some of the differences.

The following is an example user profile from Ren and Huang (2025):

“This user is likely to enjoy puzzle games with challenging gameplay and cooperative elements, as well as games that offer unique storytelling, intellectual experiences, and explore historical events. They may also enjoy platformer games with creative puzzle-solving and fantasy settings. Additionally, this user appreciates immersive first-person shooter games with engaging stories and atmospheric environments.”

We note this profile mostly describes the items interacted by the user through relatively typical item labels such as “platformer game”, “fantasy”, etc.

In contrast, our prompting strategy yields more granular profiles such as the one below.

“I’m a bit of a cinematic connoisseur, with a preference for films that explore the human condition through complex characters and thought-provoking themes. I enjoy movies that delve into the darker aspects of life, often with a touch of drama and romance. I’m drawn to films with intricate plots, well-developed characters, and cinematography that captures the mood and atmosphere of a particular setting. I appreciate movies that push the boundaries of storytelling, whether it’s through unconventional narrative structures or unflinching depictions of the human experience. When it comes to actors and actresses, I look for performances that are nuanced and emotionally resonant. I’m a fan of actors who can convey depth and complexity, and actresses who can bring a sense of vulnerability and relatability to their characters. In terms of genre, I’m open to a wide range of films, but tend to lean towards dramas, thrillers, and romantic movies that explore mature themes. I’m not a fan of overly simplistic or predictable films, preferring those that challenge me and keep

me on the edge of my seat. Overall, I'm looking for movies that are intelligent, thought-provoking, and emotionally resonant – films that will stay with me long after the credits roll."

Such a text is able to draw connections between the items enjoyed by the user at a deeper level than traditional categories such as genre, extending to more abstract descriptors such as "well-developed characters" or "actors which convey depth". This is in line with our observation that our model performs especially well compared to the baselines on dense datasets such as MovieLens.

B.1 Profile Generation Details

In order to generate the user semantic preference profiles, we utilised *Llama - 3.2B Instruct* model provided by [Huggingface](#). The related metadata generation procedure was conducted as explained in Section 3.1.

During the generation we set the LLM parameters to below values to get diverse set of profiles.

Temperature :	2.5
Top-K :	100
Top-P :	0.8
Number of Beams :	6

We primarily utilised the huggingface library to obtain and run the model for profile generation task.

The used prompts are given below.

B.1.1 MovieLens-1M

System Prompt

Assume you are a person who is interested in movies. You have rated the following movies (tagged with their respective genres) on a scale of 1 to 5. Do not explicitly mention the rating values unless absolutely necessary. You may describe your preferences using relative comparisons based on these ratings.

item1 | 5 | Action, Adventure
item2 | 3 | Adventure
....

You may consider the following aspects when explaining your preferences: [Movie title, Genre, Movie length (runtime), Director, Actors/Actresses,

Cinematography, Plot, Popularity, and whether the movie is based on a book] Note, you are in P percentile based on your interactions with the system. Use this information only to explain your behaviour and preferences. DO NOT MENTION EXPLICITLY.

Explain your preferences informally, using as few sentences as possible. Keep your response under 250 words.

User Prompt

"Write a short description of your movie preferences."

During the generation process, we randomly select the subset of items a particular user interaction along with percentile details of that particular user is provided to fill the relevant parts of the prompt.

B.1.2 Netflix

System Prompt

Assume you are a person who is interested in movies. You have rated the following movies on a scale of 1 to 5. Do not explicitly mention the rating values unless absolutely necessary. You may describe your preferences using relative comparisons based on these ratings.

item1 | 5 | Action, Adventure
item2 | 3 | Adventure
.....

You may consider the following aspects when explaining your preferences: [Movie title, Genre, Movie length (runtime), Director, Actors/Actresses, Cinematography, Plot, Popularity, and whether the movie is based on a book]

Explain your preferences informally, using as few sentences as possible. Keep your response under 250 words.

User Prompt

"Write a short description of your movie preferences."

Similar to previous case, the prompts are completed with randomly selected item interaction subsets selected from the considered user's past interaction as explained in Section 3.1.

C Additional details of experimental setup

For the all the experiments we used the same splits across the datasets for fair comparison and primarily utilised the preprocessing schemes used in (Liang et al., 2018). the additional details of the data splits as follows.

Since our primary focus is cold start user problem, we split the datasets based on the users. For all the datasets, we chose 6:2:2 ratio splits based on the users across training, validation and test sets. And for each user within each split, we held-out 20% of their true interactions for prediction as usual under Autoencoder paradigm. One important distinction is that, we explicitly avoid considering the items which was used to generate the text preference descriptions of that particular user within the validation and test splits, thus leading to exact evaluation of ranking performance. In our datasets, we explicitly mark the items which we used to generate the profile texts of users and can be used to evaluate retrieval models and ranking models in more precise manner under different conditions without having data leakage issues. Also, in experiments we considered full item set as candidates and suppress the items which was used as inputs during forward pass as typically done in Autoencoder literature (Liang et al., 2018).

During optimization, we used AdamW optimizer (Loshchilov and Hutter, 2019) with learning rate of $1e^{-4}$ in general. We used batch size of 128 for our own model training whereas for LLM based baseline models we used the batch size accordingly based on our available GPU capabilities at the time of the experiments. All the baseline models are initialised using the provided Huggingface checkpoints and used the official code repositories provided by the corresponding authors to obtain the necessary parameter whenever required.

All experiment results related to *our model* are conducted for 3 random seeds for 10 epochs and selected the best model based on the results on the validation set. The results are provided as mean values across such 3 runs.

For the baseline models which required training, we ran the experiment until we observe a consistent degradation of ranking metrics or 3 epochs. Note, some of our baselines which utilise pairwise training objectives took extremely long time (5 days+) under our prototype implementations for dense datasets such as Netflix therefore, we have

only considered single run values in such cases.