ProtoCF: Prototypical Collaborative Filtering for Few-shot Recommendation

Aravind Sankar*
asankar3@illinois.edu
University of Illinois at Urbana-Champaign
Illinois, USA

Junting Wang*
junting3@illinois.edu
University of Illinois at Urbana-Champaign
Illinois, USA

Adit Krishnan
aditk2@illinois.edu
University of Illinois at Urbana-Champaign
Illinois, USA

Hari Sundaram
hs1@illinois.edu
University of Illinois at Urbana-Champaign
Illinois, USA

ABSTRACT

In recent times, deep learning methods have supplanted conventional collaborative filtering approaches as the backbone of modern recommender systems. However, their gains are skewed towards popular items with a drastic performance drop for the vast collection of *long-tail* items with sparse interactions. Moreover, we empirically show that prior neural recommenders lack the resolution power to accurately rank relevant items within the long-tail.

In this paper, we formulate long-tail item recommendations as a *few-shot learning* problem of *learning-to-recommend* few-shot items with very few interactions. We propose a novel *meta-learning* framework ProtoCF that learns-to-compose robust prototype representations for few-shot items. ProtoCF utilizes episodic few-shot learning to extract meta-knowledge across a collection of diverse meta-training tasks designed to mimic item ranking within the tail. To further enhance discriminative power, we propose a novel architecture-agnostic technique based on knowledge distillation to *extract, relate, and transfer* knowledge from neural base recommenders. Our experimental results demonstrate that ProtoCF consistently outperforms state-of-art approaches on overall recommendation (by 5% Recall@50) while achieving significant gains (of 60-80% Recall@50) for tail items with less than 20 interactions.

KEYWORDS

Recommendation System, Collaborative Filtering, Meta Learning, Few-shot learning.

ACM Reference Format:
Aravind Sankar, Junting Wang, Adit Krishnan, and Hari Sundaram. 2021. ProtoCF: Prototypical Collaborative Filtering for Few-shot Recommendation. In *Fifteenth ACM Conference on Recommender Systems (RecSys '21), September 27-October 1, 2021, Amsterdam, Netherlands.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3460231.3474268

1 INTRODUCTION

Neural Collaborative Filtering (NCF) methods have recently enabled substantial advances in modern recommender systems that are critical to diverse e-commerce applications. However, a close examination of neural recommenders' performance reveals a *paradox*: while the overall recommendation accuracy is high, accuracy levels are poor for most items in the inventory. A majority of recommendations are biased towards popular items [3], while ignoring *long-tail* items in under-represented categories. *Popularity bias* restricts personalization and impedes suppliers of long-tail items, who struggle to attract consumers given the low exposure. Targeting long-tail items can enhance diversity and bring relatively larger marginal profits. The increasing impact of recommendations (*e.g.*, 80% of Netflix activity is a result of recommendations [30]), also raises *ethical* questions: can items that are never recommended by a system be considered an instance of *discrimination* [14]? Thus, we focus on learning robust models for long-tail item recommendations.

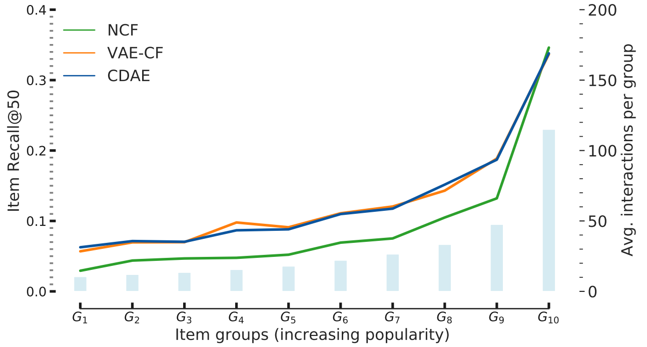

Figure 1: Item Recall@50 of three neural recommenders for item-groups (increasing popularity) in Epinions. Model performance is considerably lower for *long-tail* items.

Empirical evidence of long-tail challenges: We highlight two key observations on prior neural models [12, 24, 51]:

- Performance gains are *skewed* towards popular items with abundant historical interactions. Figure 1 depicts Item Recall@K (fraction of correctly ranked items within top-K) of three neural recommenders for different item-groups ordered by increasing

Item Subset	Top 50% Items		Bottom 50% Items	
Metric	N@50	R@50	N@50	R@50
NCF [12]	0.0906	0.1874	0.0352	0.0973
VAE-CF [24]	0.1055	0.2106	0.0457	0.1125
CDAE [51]	0.1050	0.2102	0.0471	0.1149

Table 1: Recommendation performance within top-50% *head* and bottom-50% *tail* items by item popularity on Epinions. R@50 and N@50 denote Recall@50 and NDCG@50 metrics. We observe poor ranking *resolution* within the long-tail.

*Equal contribution

RecSys '21, September 27-October 1, 2021, Amsterdam, Netherlands

ACM ISBN 978-1-4503-8458-2/21/09…$15.00
https://doi.org/10.1145/3460231.3474268

item interaction count. Performance is considerably lower for long-tail items (low popularity), indicating a clear *popularity bias*.
- Prior recommenders lack the *resolution power* to accurately rank relevant items within the long-tail. To show this, we split the item inventory into two equal-sized sets by interaction count (popular and long-tail), and evaluate personalized ranking independently within each set. Table 1 shows a big performance gap between ranking within the popular and long-tail item sets.

Prior efforts towards the long-tail have been two-fold. First, regularizing an existing recommender via inter-item associations (derived from item co-occurrences) [4, 19, 23, 54] or distributional priors over the latent space [15, 24]. While regularization strategies can partially alleviate overfitting, they typically involve static hypotheses that may be counter-productive for massive inventories with a diverse collection of long-tail items. Second, incorporating external side information [22], *e.g.*, item attributes [7] and knowledge graphs [47] to overcome sparsity. Note that exploiting side information is typically application-dependent, thus lacking generalizability to diverse scenarios. In contrast, we exploit the capabilities of any neural base recommender for long-tail item recommendation without access to side information.

Learning a latent space tailored to long-tail items poses two key challenges: first, *sparsity and heterogeneity*: although tail items have sparse interactions, they belong to diverse item categories; we need sufficient resolution power to learn discriminative representations for tail items, while being careful to avoid overfitting. Second, *distribution mismatch* between head items (substantial interactions) and tail items (sparse interactions). Specifically, neural recommenders typically sample user-item pairs from the overall interaction distribution (biased to head items) for model training; this results in overfitting and degenerate representations for tail items.

Present Work: To overcome item interaction sparsity, we *extract, relate, and transfer* the knowledge learned by a neural base recommender over *head* items, to learn robust and discriminative tail item representations. We eliminate the distribution inconsistency between head and tail items with an *episodic few-shot learning* setup [33] to *simulate* the distribution of tail items during model training by *sub-sampling* interactions from data-rich head items.

We present a novel *metric-based* few-shot learning framework ProtoCF for long-tail item recommendation. First, we pretrain a *base recommender* to extract preferences over head items and item-item relationships. Then, we design a *few-shot recommender* that learns a *shared metric space* of users and items by extracting *meta-knowledge* across a collection of meta-training tasks designed to *mimic* long-tail item recommendation. ProtoCF *learns-to-compose* a representative *prototype* for each item from a small set of user interactions and recommends relevant items by finding the nearest prototypes to each user. To transfer knowledge from the base recommender, we introduce a *knowledge distillation* strategy to distill the discovered *item-to-item* relationships into a *compact* set of *group embeddings*; we compose *discriminative* item prototypes via learnable mixtures of group embeddings. We summarize our key contributions below:

- **Few-shot Item Recommendation**: To our knowledge, ours is the first to formulate long-tail item recommendation as *learning-to-recommend* items with few interactions. We eliminate the distribution mismatch between head and tail items via episodic few-shot learning. Our problem formulation applies to diverse scenarios without requiring side information.
- **Discriminative Prototype Learning**: We learn to compose *discriminative prototypes* for tail items from their *sparse* interactions. Unlike prior gradient based meta-learning for cold-start scenarios [20, 27, 46], our item prototypes directly cluster the interacted users in a *metric space*, which avoids expensive online adaptation.
- **Architecture-agnostic Knowledge Transfer**: We enhance item prototypes by knowledge transfer from neural base recommenders. In contrast to layer transfer or adaptation methods [18], our *knowledge distillation* strategy extracts a compact representation of the item relationships discovered by arbitrary base recommenders. Significantly, ProtoCF is complementary to advances in base recommenders and enables flexible adaptation to the tail.

We instantiate ProtoCF by transferring meta-knowledge from three base recommenders. Our experiments show that ProtoCF outperforms state-of-the-art baselines (by 5% Recall@50) in *overall* recommendation, with notably significant *few-shot* gains (of 60-80% Recall@50) on tail items with less than 20 training interactions.

2 RELATED WORK

We briefly review a few related lines of work relevant to long-tail item recommendations with interaction sparsity.

Neural Collaborative Filtering: The core paradigm of latent-factor CF models is to parameterize users and items with latent representations learned from historical user-item interactions. Recent neural recommenders enhance the representational capacity of CF models via non-linear latent representations [24, 51], neural interaction modeling [12, 42], and graph-based representation learning [48]. While these neural models learn expressive models to significantly outperform conventional CF approaches, sparsity concerns owing to long-tail items remain a critical challenge.

Sparsity-aware Recommendation: Clustering is one popular way to address interaction sparsity by modeling group-level behaviors; early methods generate recommendations for tail items at the granularity of item clusters [32], *e.g.*, cluster-based smoothing [53], user-item co-clustering [52] and joint clustering and CF [29]. However, clustering in the presence of skew can lead to uninformative results [2] and degrades the extent of personalization.

Recent techniques leverage *item-item co-occurrence* statistics and *distributional priors* to regularize latent-factor CF methods. One common approach is *regularization* of recommenders, *e.g.*, factorization of an item co-occurrence matrix that shares latent factors with a CF model [4, 23]. Variational auto-encoders (VAEs) [15, 24, 37] employ Gaussian priors as regularizers on the representational space to handle sparsity. Another strategy to alleviate sparsity is *data augmentation* for items (or users) in the tail via *adversarial* regularization [19] or rating generation [5, 6] techniques. However, regularization techniques typically impose static hypotheses with restrictive assumptions, while adversarial learning is computationally expensive and does not scale to massive inventories.

Cold-start recommendation is a related problem that targets *new* users or *new* items with *no* historical interactions. In such a scenario, models rely on auxiliary information, *e.g.*, user profiles [20], item content [7], social connections [17, 35, 36], and knowledge graphs [47]. An effective strategy is randomized feature dropouts to enable generalization to missing inputs [45, 57]. In contrast, we focus on *few-shot* recommendations over long-tail items with very few interactions, *without* any auxiliary information.

Few-shot Learning: designing models capable of learning new tasks rapidly given a limited number of training examples. Here, the *meta-learning* (learning to learn) paradigm has achieved considerable success in several domains including computer vision [38, 44], natural language [10], and data mining [20, 43].

A few recent methods adapt meta-learning to cold-start recommendation [28]. Prior work have examined gradient-based parameter [8, 20, 27, 46, 56] and hyper-parameter initialization [9], and layer sharing with user-specific (or item) parameter adaptation [43, 50, 55]. However, these methods mostly address the cold-start (zero-shot) scenario with side information, thus inapplicable to our general setting of recommendation given limited interactions.

Our work is conceptually related to *metric-learning* approaches [38, 39, 44] for few-shot learning, which *learn-to-learn* a metric space that generalizes to new tasks without any need for adaptation. A few recent explorations address task heterogeneity [40] in few-shot classification settings, by designing architectures for explicit knowledge transfer from data-rich classes [21, 26, 49] to construct few-shot classifiers. However, such techniques are suited to learning classifiers from a *limited* number of classes, hence cannot scale to handle *massive* item inventories in recommendation applications. To our knowledge, ours is the *first* investigation of metric-based few-shot learning for long-tail item recommendations.

3 PROBLEM DEFINITION

We consider the implicit feedback setting (only clicks, no explicit ratings) with a user set $\mathcal{U}$, an item set $\mathcal{I}$, and a binary $U \times I$ user-item interaction matrix $\boldsymbol{X}$. Let N_i denote the set of all users who have interacted with item $i \in \mathcal{I}$. Prior neural recommenders learn a scoring function $f(i \mid u, \boldsymbol{X}), i \in \mathcal{I}$ personalized to each user $u \in \mathcal{U}$ for item ranking over $\mathcal{I}$. From Table 1, we find their performance for tail items (bottom 50%) to be significantly inferior to head items (upper 50%). Thus, our focus is to develop personalized recommenders tailored to the *long-tail* items (with sparse interactions), while ensuring reasonable performance over the entire item set $\mathcal{I}$.

Problem (Long-Tail Item Recommendation). *Given user-item interaction matrix* $\boldsymbol{X}$*, learn a scoring function* $f_T(i \mid u, \boldsymbol{X}), i \in \mathcal{I}$ *to generate a ranked list of items personalized to each user* $u \in \mathcal{U}$ *that improves recommendation quality on the long-tail items without compromising overall model performance on the entire item set* $\mathcal{I}$*.*

4 PROTOCF FRAMEWORK

Learning informative representations for tail items poses the key challenge of *interaction sparsity.* Without access to side information (*e.g.*, item content or contextual attributes), we critically remark that prior neural recommenders [24, 51] learn informative representations for head items with abundant training interactions. Thus, to learn discriminative tail item representations, we propose a novel few-shot learning framework ProtoCF with two key steps:

First, we train a neural *base recommender* $\boldsymbol{R}_B$ to learn high-quality user representations that mainly capture preferences over head items and infer item-item relationships (Section 4.1). Then, we present a *few-shot recommender* $\boldsymbol{R}_F$ that *extracts and transfers* knowledge from the base recommender $\boldsymbol{R}_B$ to *learn-to-recommend* few-shot items (Sections 4.2, 4.3).

Symbol	Description
$p_B(i,j)$	proximity induced by $\boldsymbol{R}_B$ for items $i, j \in \mathcal{I}$
$p(\mathbb{T})$	meta-training task distribution over items $\mathcal{I}$
$\mathcal{T}$	meta-training task sampled from $p(\mathbb{T})$
$\mathcal{I}_{\mathcal{T},N}$	set of N items sampled from $\mathcal{I}$ for task $\mathcal{T}$
$\mathcal{S}_i$	support set for item $i \in \mathcal{I}_{\mathcal{T},N}$ in task $\mathcal{T}$
$\mathcal{Q}_i$	query set for item $i \in \mathcal{I}_{\mathcal{T},N}$ in task $\mathcal{T}$
$G_U(\cdot \mid \theta)$	user encoder in few-shot recommender $\boldsymbol{R}_F$
$\mathcal{Z}_M$	set of M group embeddings $\{\mathbf{z}_m \in \mathbb{R}^D\}_{m=1}^{M}$
sim_m	similarity metric for meta-recommender $\boldsymbol{R}_F$
$\mathbf{p}_i$	initial item prototype for item $i \in \mathcal{I}$
$\mathbf{g}_i$	group-enhanced item prototype for item $i \in \mathcal{I}$
$\mathbf{e}_i$	final gated item prototype item for item $i \in \mathcal{I}$

Table 2: Notations

4.1 Neural Base Recommender

In this section, we outline the architecture of a differentiable base recommender $\boldsymbol{R}_B$ to learn a ranking $f_\phi(i \mid u), i \in \mathcal{I}$ (with parameters ϕ) over the interactions $\boldsymbol{X}$. Neural CF models [12, 24, 51] typically learn latent representations for users and items, followed by an interaction function and learning objective for model training.

4.1.1 ***User and Item Representations****.* Latent-factor CF methods adopt a variety of representation learning strategies, including matrix factorization [16, 34], autoencoders [24, 51] and graph neural networks [48]. We define preference encoders $F_U(\cdot \mid \phi)$ and $F_I(\cdot \mid \phi)$ to learn user $\mathbf{h}_u \in \mathbb{R}^D$ and item $\mathbf{h}_i \in \mathbb{R}^D$ embeddings for user $u \in \mathcal{U}$ and item $i \in \mathcal{I}$, which can be described by:

$$\mathbf{h}_u = F_U(u, \boldsymbol{X} \mid \phi) \qquad \mathbf{h}_i = F_I(i, \boldsymbol{X} \mid \phi) \tag{1}$$

4.1.2 ***Training Objective****.* We define a user-item interaction function $F_{\text{INT}}(\cdot \mid \phi)$ to compute a score $\hat{y}_b(u,i)$ that indicates the relevance of item i to user u. The training objective L_B of the base recommender $\boldsymbol{R}_B$ is given by:

$$\hat{y}_b(u,i) = F_{\text{INT}}(\mathbf{h}_u, \mathbf{h}_i \mid \phi) \qquad L_B = l(\hat{y}_b(u,i), y_{ui})) \tag{2}$$

where $y_{ui} = 1$ for observed (u,i) pairs and $l(\cdot)$ is a pairwise [34] or pointwise [12] loss function. The interaction function F_{INT} measures user-item relevance and is typically modeled using an inner product [24, 34, 48, 51].

We train the base recommender $\boldsymbol{R}_B$ using cosine similarity since normalized representations generalize better to few-shot settings [11] (compared to unnormalized inner products), and facilitate unified recommendation of head and tail items during model inference. We denote the *item-item proximity* $\text{sim}_b(\cdot)$ in the latent space of $\boldsymbol{R}_B$ by:

$$p_B(i,j) \propto \text{sim}_b(\mathbf{h}_i, \mathbf{h}_j) = \cos(\mathbf{h}_i, \mathbf{h}_j) \qquad i,j \in \mathcal{I} \tag{3}$$

We pre-train $\boldsymbol{R}_B$ to extract knowledge of user preferences over head items via encoder $F_U(\cdot)$ and item-item proximities among

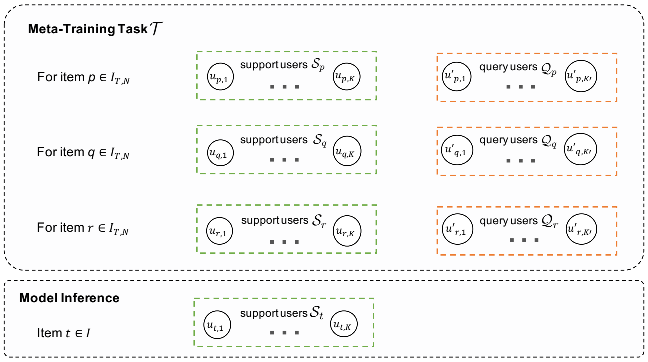

Figure 2: Episodic few-shot learning with meta-training task $\mathcal{T}$ and item embedding inference at meta-testing.

different items via $p_B(i, j)$. Below, we present our proposed framework ProtoCF that transfers knowledge from the *data-rich* head to the *data-poor* tail to enable robust few-shot recommendations.

4.2 Few-shot Item Recommendation

In this section, we formulate long-tail item recommendation as *few-shot item representation learning* given a small *support set* of upto K interacted users for each tail item (typically $K \approx 5$ to 20).

4.2.1 *Few-shot Task Formulation.*

Our framework is grounded on *episodic* learning [44] with a collection of *meta-training* tasks or episodes. One approach [11, 38] is to construct meta-training tasks from the interactions of data-rich head items; however, excluding the diverse collection of tail items may impede generalization. Thus, our meta-training tasks operate on the entire item set $\mathcal{I}_T$.

Each meta-training task $\mathcal{T}$ is a personalized ranking problem over a subset of N items $\mathcal{I}_{\mathcal{T},N}$ randomly sampled from $\mathcal{I}$. We *simulate the interaction distribution of few-shot items during meta-training* by sampling K training interactions (from N_i) for each item $i \in \mathcal{I}_{\mathcal{T},N}$ in task $\mathcal{T}$. During inference, we generate *few-shot* item recommendations from samples of upto K training interactions per item. By ensuring consistency between meta-training and inference, we bridge the *distribution mismatch* between head and tail items. The meta-knowledge extracted across diverse meta-training tasks benefits tail items with sparse interactions.

The episodic few-shot training process operates over a collection of meta-training tasks $\{\mathcal{T}_1, \mathcal{T}_2, \dots\}$ sampled from a task distribution $p(\mathbb{T})$ over the item set $\mathcal{I}$. Specifically, a *K-shot, N-item* meta-training task $\mathcal{T}$ sampled from $p(\mathbb{T})$ consists of support $\mathcal{S}$ and query $\mathcal{Q}$ user sets over N items $\mathcal{I}_{\mathcal{T},N} \subset \mathcal{I}$ (analogous to the usual sense of training and testing sets respectively), defined by:

$$\begin{aligned}
&\mathcal{T} = \{\mathcal{I}_{\mathcal{T},N}, \mathcal{S}, \mathcal{Q}\} && \mathcal{I}_{\mathcal{T},N} \subset \mathcal{I} && && (4)\\
&\mathcal{S} = \{\mathcal{S}_i : i \in \mathcal{I}_{\mathcal{T},N}\} && \mathcal{S}_i = \{u_{i,1}, \dots, u_{i,K}\} && u_{i,k} \in N_i &&\\
&\mathcal{Q} = \{\mathcal{Q}_i : i \in \mathcal{I}_{\mathcal{T},N}\} && \mathcal{Q}_i = \{u'_{i,1}, \dots, u'_{i,K'}\} && u'_{i,k'} \in N_i &&
\end{aligned}$$

where the support user set $\mathcal{S}_i$ contains K interacted users sampled for each item $i \in \mathcal{I}_{\mathcal{T},N}$ and the corresponding query set $\mathcal{Q}_i$ includes K' interacted users sampled from each of the N items.

We learn a *few-shot recommender* $\boldsymbol{R}_F$ that takes as input the support users $\mathcal{S}$ to *learn-to-compose* representations for items $i \in \mathcal{I}_{\mathcal{T},N}$ in task $\mathcal{T}$. The few-shot recommender $\boldsymbol{R}_F$ is trained by optimizing a learning objective designed to match the item recommendations generated by $\boldsymbol{R}_F$ for its query users $\mathcal{Q}$ with their corresponding ground-truth interactions over the item $\mathcal{I}_{\mathcal{T},N}$ in each task $\mathcal{T}$.

4.2.2 *Initial Item Prototype.*

We learn a shared *metric space* of users and items to synthesize a representation for each item $i \in \mathcal{I}_{\mathcal{T},N}$ based on its support set $\mathcal{S}_i$ (with K interactions). The metric space (with similarity $\text{sim}_m(\cdot)$) compactly clusters the interacted users $\mathcal{S}_i$ of each item $i \in \mathcal{I}_{\mathcal{T},N}$ around a *prototype* $\mathbf{p}_i \in \mathbb{R}^D$ [38].

We first define a user preference encoder $G_U(\cdot \mid \theta)$ that maps each user $u \in \mathcal{U}$ into a latent embedding space. To transfer knowledge from the base recommender $\boldsymbol{R}_B$, the few-shot user encoder $G_U(\cdot \mid \theta)$ has parameters initialized from its pre-trained encoder $F_U(\cdot \mid \phi)$ (Equation 1), but is parameterized with learnable parameters θ. The prototype $\mathbf{p}_i$ for item $i \in \mathcal{I}_{\mathcal{T},N}$ is computed as the mean vector of the embedded support user set $\mathcal{S}_i$, defined by:

$$\mathbf{p}_i = \frac{1}{\mathcal{S}_i} \sum_{u_{i,k} \in \mathcal{S}_i} G_U(u_{i,k}, X_H \mid \theta) \qquad i \in \mathcal{I}_{\mathcal{T},N} \tag{5}$$

Equation 5 encourages the prototype $\mathbf{p}_i$ to learn a representative vector summarizing its cluster of interacted users. However, we face two critical challenges in handling tail items: first, due to sparse support sets, the prototypes are often *noisy* and sensitive to *outliers*; second, due to substantial *heterogeneity* in the tail, simplistic averaging may lack the *resolution* to discriminate across diverse tail items. Thus, the few-shot recommender $\boldsymbol{R}_F$ needs a strong *inductive bias* during prototype learning to avoid overfitting, and yet have sufficient expressivity to learn *discriminative* prototypes.

4.2.3 *Head-Tail Meta Knowledge Transfer.*

We now exploit the *item-to-item* relationship knowledge acquired by the neural base recommender $\boldsymbol{R}_B$ (Section 4.1) as an inductive bias to enhance the item prototype $\mathbf{p}_i$ (Equation 5). Given an item $i \in \mathcal{I}_{\mathcal{T},N}$ with limited support users, extracting knowledge from the most related items to item i (based on the item-to-item proximities) can provide valuable guidance to synthesize robust item prototypes.

However, a direct knowledge transfer from $\boldsymbol{R}_B$ to $\boldsymbol{R}_F$ is computationally challenging: dynamically identifying related items for each few-shot item during prototype construction is not scalable due to the arbitrary complexity of $\boldsymbol{R}_B$ and the high cardinality of item sets in massive inventories. Thus, we extract a *compact* representation of the *item-item proximity* knowledge (Equation 3) discovered by $\boldsymbol{R}_B$ and transfer this knowledge to enhance item prototypes.

Group-Enhanced Item Prototype Learning. We learn a set of M group embeddings $\mathcal{Z}_M$ as *basis vectors* modeling *item-item proximity* in the latent space of the base recommender $\boldsymbol{R}_B$. Formally, the group embeddings $\mathcal{Z}_M$ are given by:

$$\mathcal{Z}_M = \{z_m \in \mathbb{R}^D : 1 \leq m \leq M\} \qquad M \ll |\mathcal{I}| \tag{6}$$

As depicted in Figure 3 (bottom left), we intuitively visualize the group embeddings as *discriminative centroids* of *overlapping* clusters of items identified by $\boldsymbol{R}_B$, and conversely view items as *mixtures* over the group embeddings, *e.g.*, each centroid may represent a contextual factor such as a restaurant type (Cafe) or location (Chicago), while restaurants are mixture of multiple centroids (Cafe located in Chicago) and belong to overlapping item clusters.

To enhance the prototype $\mathbf{p}_i$ of item $i \in \mathcal{I}_{\mathcal{T},N}$, we synthesize a *group-enhanced prototype* $\mathbf{g}_i \in \mathbb{R}^D$ as a mixture over the M group

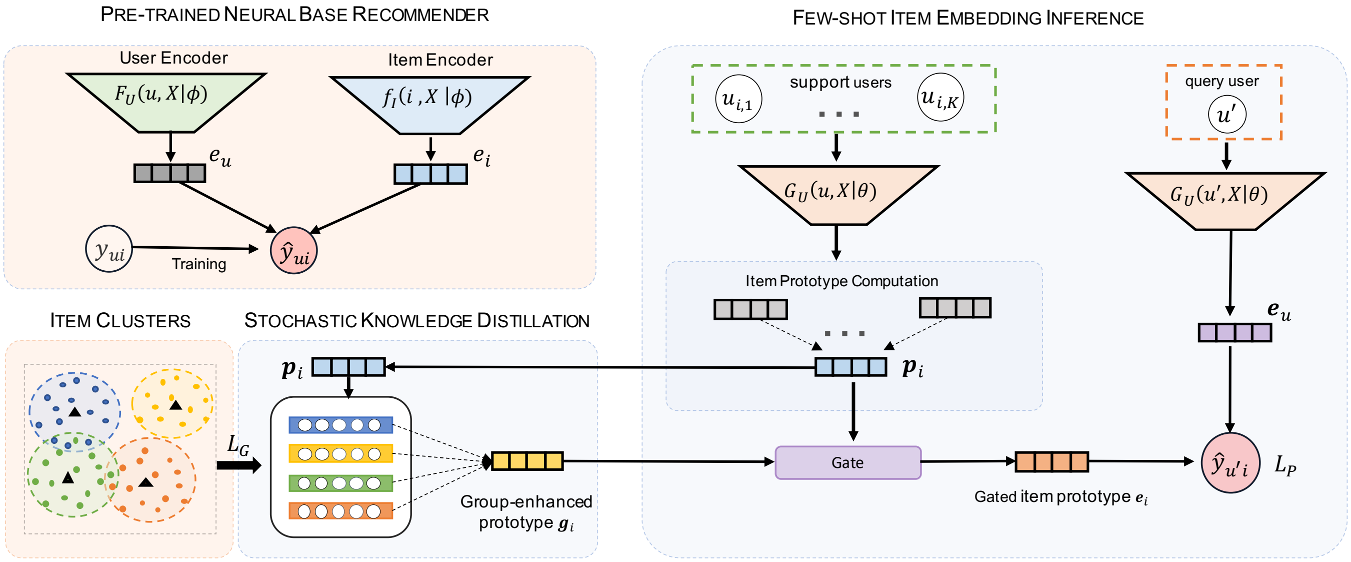

Figure 3: Architecture diagram of ProtoCF depicting the different model components: pre-trained neural base recommender R_B (top left), group embedding learning via stochastic knowledge distillation L_G (bottom left), initial item prototype construction via support set averaging followed by group-enrichment and adaptive gating to construct gated item prototype $\mathbf{e}_i$ (right).

embeddings. The mixture coefficients are estimated by a learnable attention mechanism [1] to measure *compatibility* α_{im} between the prototype $\mathbf{p}_i$ and each centroid $z_m \in \mathcal{Z}_M$. We parameterize the attention function with a lightweight network comprised of an auxiliary set $\mathcal{K}_M = \{\mathbf{k}_m \in \mathbb{R}^D : 1 \leq m \leq M\}$ of trainable keys to index the group embeddings. We implement the attention function with an inner product followed by softmax normalization, as:

$$\mathbf{g}_i = \sum_{m=1}^{M} \alpha_{im}\mathbf{z}_m \qquad \alpha_{im} = \frac{\exp\left(\boldsymbol{W}_q\mathbf{p}_i \cdot \mathbf{k}_m\right)}{\sum_{m'=1}^{M}\exp\left(\boldsymbol{W}_q\mathbf{p}_i \cdot \mathbf{k}_{m'}\right)} \tag{7}$$

where $\boldsymbol{W}_q \in \mathbb{R}^{D\times D}$ is projects the prototype $\mathbf{p}_i$ into a query to index the centroids. The group-enhanced item prototype $\mathbf{g}_i$ relates the different centroids via attention, transferring knowledge to few-shot items with sparse support sets.

Task-level Stochastic Knowledge Distillation. We present a *knowledge distillation* strategy [13] to learn compact *group embeddings* $\mathcal{Z}_M$ that capture *item-item relationships* in $\boldsymbol{R}_B$. We transfer knowledge from a high-capacity *teacher* model (base recommender $\boldsymbol{R}_B$) to a compact *student* model (group embeddings $\mathcal{Z}_M$) by encouraging the student to emulate predictions of the teacher; $\mathcal{Z}_M$ is trained to emulate the item proximity distribution in $\boldsymbol{R}_B$.

Aligning pairwise item proximities over all items in $\mathcal{I}$ is not scalable; thus, we operate at the granularity of each meta-training task $\mathcal{T}$. For each item $i \in \mathcal{I}_{\mathcal{T},N}$, we compute a soft probability distribution $p_B(j \mid i, \boldsymbol{R}_B)$ for the teacher model $\boldsymbol{R}_B$ (based on equation 3) over other items $j \in \mathcal{I}_{\mathcal{T},N}$ in the task $\mathcal{T}$, decribed by:

$$p_B(j \mid i, \boldsymbol{R}_B) = \frac{\exp\left(p_B(i,j)/T\right)}{\sum_{k\in\mathcal{I}_{\mathcal{T},N}}\exp\left(p_B(i,k)/T\right)} \qquad i,j \in \mathcal{I}_{\mathcal{T},N} \tag{8}$$

where $T > 0$ is a temperature scaling hyper-parameter to regulate the rate of knowledge transfer. We analogously define the item similarity distribution $p_F(j \mid i, \mathcal{Z}_M)$ for the student model $\mathcal{Z}_M$ based on the metric space proximity $\text{sim}_m(\cdot)$ of group-enhanced prototypes $\mathbf{g}_i$ and $\mathbf{g}_j$ for items $i, j \in \mathcal{I}_{\mathcal{T},N}$, defined by:

$$p_F(j \mid i, \mathcal{Z}_M) = \frac{\exp\left(\text{sim}_m(\mathbf{g}_i, \mathbf{g}_j)\right)}{\sum_{k\in\mathcal{I}_{\mathcal{T},N}}\exp\left(\text{sim}_m(\mathbf{g}_i, \mathbf{g}_k)\right)} \quad i,j \in \mathcal{I}_{\mathcal{T},N} \tag{9}$$

We align the two distributions by minimizing cross-entropy between their task-level similarities [13]. Since each item is typically related to very few items within the task $\mathcal{T}$, our *stochastic knowledge distillation* loss L_G minimizes distribution divergence over the top-n ($n \approx 10$) related items (out of N) identified by the teacher $\boldsymbol{R}_B$, by:

$$L_G = -\frac{1}{nN}\sum_{i\in\mathcal{I}_{\mathcal{T},N}}\ \sum_{j\in\mathbb{T}_{B,n}(i)} p_B(j \mid i, \boldsymbol{R}_B)\log p_F(j \mid i, \mathcal{Z}_M) \tag{10}$$

where $\mathbb{T}_{B,n}(i) = \text{Top}_n\big(p_B(\cdot \mid i, \boldsymbol{R}_B)\big)$ denotes the top-n most related items to item i within $\mathcal{I}_{\mathcal{T},N}$ based on the teacher $\boldsymbol{R}_B$. Distinct from prior distillation approaches [13, 31] that match student and teacher predictions over a fixed set of classes, L_G is stochastic and thereby more efficient since it matches the top-n item proximity distributions over different sets of sampled items in each task. The distillation loss L_G transfers knowledge from $\boldsymbol{R}_B$ to the group embeddings $\mathcal{Z}_M$ and is trained jointly with the rest of the framework.

4.2.4 *Item Prototype Fusion via Neural Gating.*

The initial prototype $\mathbf{p}_i$ for item $i \in \mathcal{I}_{\mathcal{T},N}$ direct encodes its support users $\mathcal{S}_i$, while the group-enhanced prototype $\mathbf{g}_i$ captures the knowledge transferred from related items (via base recommender $\boldsymbol{R}_B$). We design a *gating mechanism* [41] that adaptively selects salient feature dimensions from $\mathbf{p}_i$ and $\mathbf{g}_i$ to infer the final *gated item prototype* $\mathbf{e}_i \in \mathbb{R}^D$. Specifically, we introduce a neural gating layer to merge $\mathbf{p}_i$ and $\mathbf{g}_i$ by learning a non-linear gate to flexibly modulate information flow via dimension re-weighting, defined by:

$$\begin{aligned} \textbf{gate} &= \sigma\big(\boldsymbol{W}_{g_1}\mathbf{p}_i + \boldsymbol{W}_{g_2}\mathbf{g}_i + \mathbf{b}_g\big) \quad i \in \mathcal{I}_{\mathcal{T},N} \\ \mathbf{e}_i &= \textbf{gate} \odot \mathbf{p_i} + (\mathbf{1} - \textbf{gate}) \odot \mathbf{g_i} \end{aligned} \tag{11}$$

where $\boldsymbol{W}_{g_1} \in \mathbb{R}^{D \times D}$, $\boldsymbol{W}_{g_2} \in \mathbb{R}^{D \times D}$, and $\mathbf{b}_g \in \mathbb{R}^D$ are learnable parameters in the neural gating layer, $\odot$ denotes the element-wise product operation, and σ is the sigmoid non-linearity.

4.2.5 *Few-shot Recommender Training.*

We induce few-shot item rankings for the query users Q using the gated prototypes $\{\mathbf{e}_i : i \in \mathcal{I}_{\mathcal{T},N}\}$ in task $\mathcal{T}$. We generate item recommendations for each query user $u' \in Q$ (over items $\mathcal{I}_{\mathcal{T},N}$) by measuring similarity $\text{sim}_m(\cdot)$ with each prototype $\mathbf{e}_i$. Each task $\mathcal{T}$ minimizes a negative log-likelihood L_P between the few-shot recommendations for query users Q and their ground-truth interactions in $\mathcal{T}$, defined by:

$$L_P = -\frac{1}{KN} \sum_{i \in \mathcal{I}_{\mathcal{T},N}} \sum_{u'_{i,k'} \in Q_i} \log p_F(i \mid u'_{i,k'}, \theta) \qquad (12)$$

where $p_F(i \mid u'_{i,k'}, \theta)$ is computed based on cosine similarity and the choice of likelihood function for few-shot training.

The overall loss L is composed of two terms, the few-shot recommendation loss L_P (Equation 12) and the knowledge distillation loss L_G for group embedding learning (Equation 10), given by:

$$L = L_P + \lambda L_G \qquad (13)$$

where λ is a tunable hyper-parameter. Algorithm 1 summarizes the training procedure of our entire framework ProtoCF.

4.2.6 *Model Inference.*

We infer the *gated prototype* $\mathbf{e}_i$ for each item $i \in \mathcal{I}$ by sub-sampling K interactions from its historical interactions N_i as the the support set (Equation 11). We generate item recommendations for each user $u \in \mathcal{U}$ by:

$$\hat{y}_f(u, i) = \text{sim}_m(\mathbf{e}_u, \mathbf{e}_i) \quad i \in \mathcal{I} \qquad \mathbf{e}_u = G_U(u, \boldsymbol{X} \mid \theta) \qquad (14)$$

The few-shot recommender $\boldsymbol{R}_F$ is designed for recommendations over tail items (sparse interactions), while the base recommender $\boldsymbol{R}_B$ is effective for head items (abundant interactions). In ProtoCF, we compute rankings over the item set $\mathcal{I}$ by ensembling predictions from $\boldsymbol{R}_B$ and $\boldsymbol{R}_F$. One simple interpolation approach is given by:

$$\hat{y}(u, i) = (1 - \eta) \cdot \hat{y}_b(u, i) + \eta \cdot \hat{y}_f(u, i) \qquad (15)$$

where $0 < \eta < 1$ balances $\boldsymbol{R}_F$ and $\boldsymbol{R}_B$. We empirically show effective overall item recommendations with $\eta = 0.5$.

4.3 Model Details

We now discuss different choices of likelihood functions for few-shot training and details of the base recommender $\boldsymbol{R}_B$.

4.3.1 *Few-shot Likelihood.*

We examine two likelihood functions in neural CF models: *multinomial* and *logistic*.

- **Multinomial log-likelihood:** The scores $\text{sim}_m(\mathbf{e}_{u'}, \mathbf{e}_i)$ for each query user $u' \in Q_i$, over the N gated item prototypes in meta-training task $\mathcal{T}$, are normalized to produce probabilities $p_F(i \mid u', \theta)$ over the N items $\mathcal{I}_{\mathcal{T},N}$, defined by:

$$p_F(i \mid u', \theta) = \frac{\exp\left(\text{sim}_m(\mathbf{e}_{u'}, \mathbf{e}_i)\right)}{\sum_{j \in \mathcal{I}_{\mathcal{T},N}} \exp\left(\text{sim}_m(\mathbf{e}_{u'}, \mathbf{e}_j)\right)} \qquad u' \in Q_i \qquad (16)$$

The resulting loss L_G (with Equation 12) is also known as the *cross-entropy* loss, defined over the N items of task $\mathcal{T}$.

Algorithm 1 ProtoCF: Prototypical Collaborative Filtering

```
Input: User-item interactions $X$, meta-training task distribution $p(\mathbb{T})$.
Output: Function $f_T(i \mid u)$, $i \in \mathcal{I}_T$ for few-shot item recommendations.
Pre-train the neural base recommender $R_B$ on the training interactions $X$ to learn $f_{H,\phi}(i \mid u)$ (eqn 2) and freeze parameters $\phi$.
Initialize user preference encoder $G_U(\cdot \mid \theta)$ of few-shot recommender $R_F$ with parameters $F_U(\cdot \mid \phi)$ from the base recommender $R_B$.
▹ Meta-training: learn-to-recommend few-shot items.
while not converged do
    Sample a meta-training task $\mathcal{T} = \{\mathcal{I}_{\mathcal{T},N}, \mathcal{S}, Q\} \sim p(\mathbb{T})$.
    Calculate the initial $\mathbf{p}_i$ and group-enhanced $\mathbf{g}_i$ prototypes for items $i \in \mathcal{I}_{\mathcal{T},N}$ (eqn 5 and eqn 7).
    Compute the knowledge distillation loss $L_G$ (eqn 10).
    Compute gated prototype $\mathbf{e}_i$ for items $i \in \mathcal{I}_{\mathcal{T},N}$ (eqn 11).
    Estimate few-shot recommendation loss $L_P$ (eqn 12) with multinomial (eqn 16) or logistic (eqn 17) log-likelihoods.
    Minimize overall loss $L$ (eqn 13) using mini-batch gradient descent.
▹ Model Inference: few-shot and overall item recommendations.
Generate item recommendations over $\mathcal{I}$ for user $u \in \mathcal{U}$ (eqn 15)
```

- **Logistic log-likelihood:** The relevance $\hat{y}_{u'i} = \text{sim}_m(\mathbf{e}_{u'}, \mathbf{e}_i)$ for item $i \in \mathcal{I}_{\mathcal{T},N}$ to query user $u' \in Q_i$, is transformed into a probability using the sigmoid function σ. We use a *confidence* weight $\beta > 0$ to re-weight the likelihood of observed 1's which are far fewer than the unobserved 0's in implicit feedback.

$$\log p_F(i \mid u', \theta) = \beta \log \sigma(\hat{y}_{u'i}) + \sum_{j \in \mathcal{I}_{\mathcal{T},N}, u' \notin N_j} \log(1 - \sigma(\hat{y}_{u'j})) \qquad (17)$$

The cosine similarity scores are appropriately scaled to match the non-saturating regimes of the softmax and sigmoid functions in the multinomial and logistic log-likelihoods respectively.

4.3.2 *Neural Base Recommender Architecture.*

We consider three neural CF methods, matrix factorization (MF) [34], denoising (CDAE) [51] and variational (VAE-CF) [24] autoencoders, as base recommenders $\boldsymbol{R}_B$ (Equation 1) for ProtoCF.

- **Matrix Factorization (MF) [34]**: The user F_U and item F_I encoders are learnable latent embeddings for each user and item respectively, trained using a logistic log-likelihood function.
- **Variational AutoEncoder (VAE-CF) [24]**: The user encoder F_U is a two-layer Multi-Layer Perceptron transforming the binary user preference vector $\mathbf{x}_u \in \mathbb{R}^I$ into a D-dimensional user embedding $\boldsymbol{h}_u \in \mathbb{R}^D$, defined by:

$$\boldsymbol{h}_u = F_U(\boldsymbol{x}_u \mid \phi) = \sigma(\boldsymbol{W}_2^T(\sigma(\boldsymbol{W}_1^T \boldsymbol{x}_u + b_1) + b_2) \qquad (18)$$

The item encoder F_I is a latent embedding. VAE-CF uses a multinomial likelihood with KL-divergence regularization.
- **Denoising AutoEncoders (CDAE) [51]**: The user encoder F_U operates on partially corrupted inputs $\mathbf{x}_u$ and adds an auxiliary per-user embedding to the encoder output from Equation 18.

5 EXPERIMENTS

We present experiments on four real-world datasets to evaluate our framework ProtoCF. We introduce datasets, baselines and experimental setup in Sections 5.1, 5.2 and 5.3. We propose four research questions to guide our experiments:

(RQ$_1$) Does ProtoCF beat state-of-the-art NCF and sparsity-aware methods on *overall* recommendation performance?

Dataset	Epinions	Yelp	Weeplaces	Gowalla
# Items	9,035	10,451	11,679	27,920
# Users	9,729	13,926	6,167	17,848
# Interactions	260,263	465,386	427,236	907,351
# Interactions per item	28.81	44.53	36.58	32.50

Table 3: Dataset statistics

(RQ$_2$) What is the impact of item *interaction sparsity* on the *few-shot* recommendation performance of ProtoCF?

(RQ$_3$) How do the different *architectural* choices impact the few-shot and overall performance of ProtoCF?

(RQ$_4$) How do the *hyper-parameters* (distillation loss balance factor λ and meta-training task size N) affect ProtoCF?

Finally, we discuss the limitations of our ProtoCF model and explore future directions in Section 5.8.

5.1 Datasets

We conducted experiments on four publicly available benchmark datasets, including two online product review platforms (*Epinions*, *Yelp*) and two check-in based social networks (*Weeplaces*, *Gowalla*).

- **Epinions[1]:** product ratings from an e-commerce platform; we retain interactions with ratings higher than two.
- **Yelp[2]:** user ratings on local businesses located in the state of Arizona, obtained from Yelp dataset challenge round 13.
- **Weeplaces[3]:** we extract business check-ins from Weeplaces of different categories, including Nightlife, Outdoors, Entertainment, Travel and Food, across all cities in the United States.
- **Gowalla [25]:** restaurant check-ins in Gowalla by users across different cities in the United States.

We pre-process the datasets to retain users and items with at least ten interactions (10-core) (Table 3).

5.2 Baselines

We present comparisons against prior work that broadly fall into two categories: standard *neural collaborative filtering* models, and *sparsity-aware long-tail item recommendation* models, including regularization and meta-learning techniques.

- **Neural Base Recommenders**: neural CF methods with matrix factorization (BPR) [34], and autoencoder models VAE-CF [24] and CDAE [51] (described in Section 4.3.2).
- **NCF [12]**: neural CF model with non-linear interactions (via neural layers) between the user and item embeddings.
- **NGCF [48]**: state-of-the-art graph-based NCF with embedding propagation layers on the user-item interaction graph.
- **Cofactor [23]**: regularized MF to capture inter-item associations by jointly decomposing the user-item interaction matrix and the item-item co-occurrence matrix, with shared latent item factors.
- **EFM [4]**: embedding factorization model that uses item-item co-occurrence with bayesian personalized ranking.
- **DropoutNet [45]:** randomly dropout latent CF embedding for regularization in content-based CF for cold-start (auxiliary content). We adapt it to long-tail items by replacing content embeddings with prototypes (over support set).
- **MetaRec-LWA** [43] meta-learning method to construct recommendation models for cold-start users with user-specific linear transforms; we adapt it to construct few-shot item embeddings.
- **MetaRec-NLBA** [43]: meta-learning recommendation model that learns user-specific biases parameterized by non-linear layers, to replace the user-specific weights in MetaRec-LWA.

Note that we omit baseline comparisons with gradient-based meta-learning recommenders [20, 27], since they are designed for cold-start recommendation in the presence of auxiliary attributes. We test ProtoCF by adopting BPR [34] (matrix factorization), VAE-CF [24], and CDAE [51] as base recommenders of our framework.

5.3 Experimental Setup

In each dataset, we randomly split the user interactions of all items into train (70%) and test (30%) sets, which ensures consistent interaction distributions across train and test. We also use 10% of training interactions as validation for hyper-parameter tuning. We use NDCG@K and Recall@K as evaluation metrics, computed based on the rank of ground-truth test interactions in top-K ranked lists.

We design a unified model to enhance few-shot recommendation on the tail without affecting the overall performance. We first evaluate *overall* recommendations over the entire item set $\mathcal{I}$ followed by *few-shot* performance analysis over the long-tail.

All experiments are conducted on a Tesla K-80 GPU using PyTorch. ProtoCF is trained using episodic learning for a maximum of 300 episodes with a meta-training task size of 512 items using Adam optimizer. ProtoCF uses a multinomial log-likelihood for few-shot training and uses support sets of upto $K = 10$ and query sets of $K' = 5$ interactions per item. We learn $M = 100$ group embeddings $\mathcal{Z}_M$, tune the learning rate and balance hyper-parameter λ (Equation 13) in the range $\{10^{-4}, 10^{-3}, 10^{-2}\}$, and use dropout regularization with a rate of 0.5. We tune baselines in hyper-parameter ranges centered at the author-provided values on each dataset and set the latent embedding dimension to 128 for consistency. Our implementation of ProtoCF and datasets are publicly available[4].

5.4 Overall Recommendation Results (RQ$_1$)

Our experimental results comparing the *overall* recommendation performance of ProtoCF with competing baselines is shown in Table 4. We summarize our key empirical observations below:

- Neural CF models based on autoencoders (VAE-CF, CDAE) and graph neural networks (NGCF) outperform other latent-factor models (NCF, BPR) on overall model performance (Table 4).
- Model regularization strategies that use item co-occurrence information (CoFactor, EFM) to improve long-tail item recommendations, are noticeably worse than BPR in overall performance.
- Sparsity-aware meta-learning models (MetaRec) perform poorly in overall item rankings. One potential reason is their inability to effectively exploit or transfer knowledge from head items.

[1] https://www.cse.msu.edu/ tangjili/datasetcode/truststudy.htm

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

[3] https://www.yongliu.org/datasets/

[4] https://github.com/aravindsankar28/ProtoCF

Dataset	**Epinions**		**Yelp**		**Weeplaces**		**Gowalla**	
Metric	**N@50**	**R@50**	**N@50**	**R@50**	**N@50**	**R@50**	**N@50**	**R@50**
Standard Neural Collaborative Filtering Methods								
BPR [34]	0.0860	0.1666	0.0749	0.1416	0.2537	0.3778	0.1661	0.2703
NCF [12]	0.0878	0.1694	0.0752	0.1429	0.2462	0.3694	0.1702	0.2745
NGCF [48]	0.0913	0.1725	0.0826	0.1579	0.2533	0.3764	0.1696	0.2758
VAE-CF [24]	0.0938	0.1778	0.0854	0.1602	0.2482	0.3730	0.1710	0.2769
CDAE [51]	0.0927	0.1774	0.0870	0.1611	0.2570	0.3760	0.1634	0.2644
Sparsity-aware Long-tail Item Recommendation Methods								
DropoutNet [45]	0.0881	0.1697	0.0761	0.1435	0.2516	0.3751	0.1697	0.2768
Cofactor [23]	0.0845	0.1639	0.0734	0.1402	0.2342	0.3539	0.1596	0.2642
EFM [4]	0.0742	0.1534	0.0741	0.1403	0.2306	0.3429	0.1532	0.2584
MetaRec-NLBA [43]	0.0453	0.0937	0.0381	0.0875	0.1698	0.2889	0.0753	0.1384
MetaRec-LWA [43]	0.0467	0.0943	0.0392	0.1425	0.1702	0.2997	0.0722	0.1391
Prototypical Collaborative Filtering Recommenders (ProtoCF)								
ProtoCF + BPR	0.0964	0.1812	0.0815	0.1533	0.2576	0.3879	0.1737	0.2800
ProtoCF + VAE	**0.0977**	**0.1830**	0.0857	0.1605	**0.2725**	**0.4035**	**0.1899**	**0.3004**
ProtoCF + CDAE	0.0972	0.1824	**0.0883**	**0.1623**	0.2697	0.4011	0.1786	0.2875
Percentage Gains	4.16%	2.92%	1.50%	0.75%	6.03%	6.80%	11.05%	8.49%

Table 4: Overall item recommendation results on four datasets, R@K and N@K denote Recall@K and NDCG@K metrics at $K = 50$. Sparsity-aware models are generally outperformed by standard NCF methods on overall item recommendation; ProtoCF achieves *overall* NDCG@50 gains of 6% and Recall@50 gains of 4% (over the best baseline) across all datasets.

- ProtoCF outperforms state-of-the-art baselines (NDCG@50 gains of 5% on average) on *overall* item ranking, with consistent gains for the variants of all neural base recommenders.
  Next, we examine the few-shot performance of ProtoCF.

5.5 Few-Shot Recommendation Results (RQ$_2$)

To evaluate few-shot results, we qualitatively analyze performance for long-tail items with sparse interactions. Specifically, we compare recommendation results for long-tail items with varying number of training interactions K (5 to 30) by computing Recall@50 metrics only on their associated test interactions (Figure 4). We only include the base recommenders BPR and VAE-CF here for comparison since they consistently outperform other sparsity-aware variants.

We find that performance generally increases with item interaction count; ProtoCF achieves significant performance gains for items with less than 20 interactions. Episodic training with knowledge transfer is one of the key factors responsible for our higher gains over items with sparse interactions (small values of K).

To evaluate the impact of *interaction sparsity* across the entire item set, we compare overall recommendation performance (Recall@K) across item-groups with different sparsity levels. We divide the test set into ten equal-sized item-groups, sorted in increasing order by the average number of interactions per item. Figure 5 compares ProtoCF against two base recommenders BPR and VAE-CF.

From figure 5, model performance is lower on the *long-tail* for base recommenders BPR and VAE due to severe interaction sparsity. Notably, ProtoCF achieves much higher item recall scores on the *tail* items with significant gains over the corresponding base recommenders, while ensuring comparable performance on the *head* items. Knowledge transfer of *item-to-item relationships* via group embeddings and pre-trained user encoder enables ProtoCF to learn discriminative prototypes for tail items with sparse interactions.

5.6 Model Ablation Study (RQ$_3$)

We examine the impact of different *architectural* design choices in ProtoCF on its *overall* and *few-shot* performance (Table 5). We choose VAE-CF as the base recommender to instantiate ProtoCF due to its consistent results. Here, we report few-shot performance by only considering the test interactions of *long-tail* items with less than 20 training interactions. Note that most ablation variants do not have a significant impact on overall results since all predictions are computed by ensembling $\boldsymbol{R}_F$ and $\boldsymbol{R}_B$ (Equation 15). Our key design choices and empirical insights are shown below:

Dataset	**Epinions**		**Gowalla**	
Metric	**Overall R@50**	**Few-shot R@50**	**Overall R@50**	**Few-shot R@50**
ProtoCF	0.1830	0.1070	0.3004	0.2195
w/o Prototype Gating	0.1823	0.0948	0.2992	0.2082
w/o Knowledge Distillation	0.1805	0.0869	0.2923	0.1983
ProtoCF-Avg	0.1801	0.0712	0.2898	0.1696
ProtoCF-logistic	0.1804	0.0896	0.2853	0.1843
VAE-CF [24]	0.1778	0.0549	0.2769	0.1316
MetaRec-LWA [43]	0.0943	0.0898	0.1391	0.1804

Table 5: Model ablation study of ProtoCF; few-shot performance is reported for tail items (less than 20 training interactions). Knowledge transfer and prototype gating contribute 10-19% and 5-11% to few-shot gains respectively.

- **Remove Prototype Gating:** We replace the gating layer (Equation 11) that adaptively fuses the initial and group-enhanced item prototypes, with a simpler additive operation. Adaptive gating contributes 5-11% few-shot performance gains.

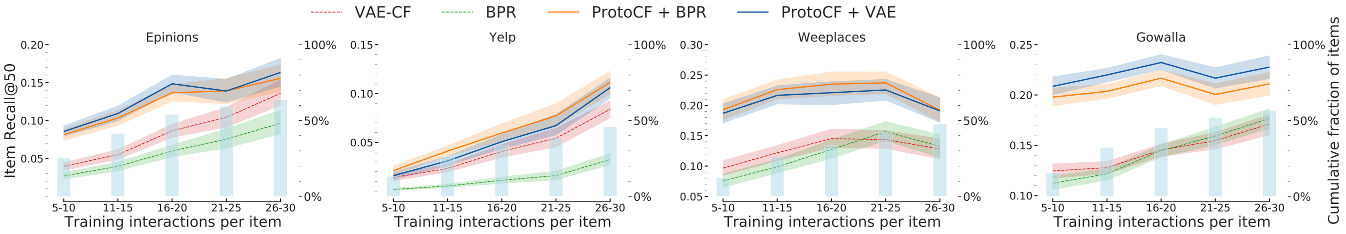

Figure 4: Few-shot item recommendation results: Performance comparison for long-tail items with varying number of training interactions K (5 to 30); lines denote model performance (Recall@50) and background histograms indicate the cumulative fraction of the item inventory covered by tail items with $\leq K$ impressions. Overall performance generally increases with K for all models; ProtoCF achieves notably stronger gains (over baselines) for items with few training interactions (small K).

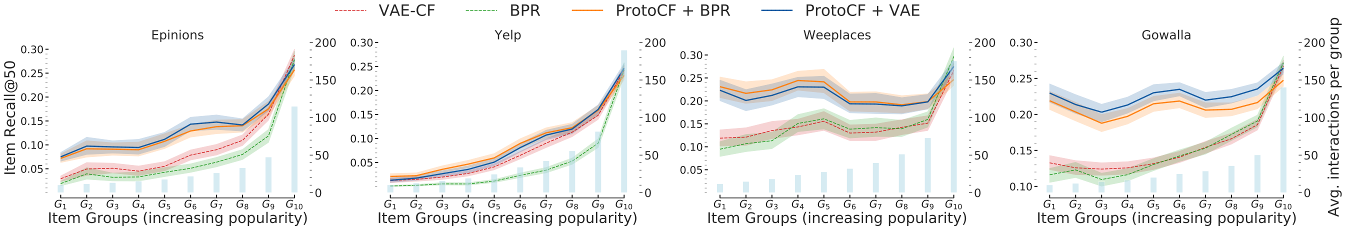

Figure 5: Impact of item interaction sparsity: Performance comparison for item-groups sorted in increasing order by their average training interaction counts; lines denote model performance (Recall@50) and background histograms indicate the average number of interactions in each item-group. ProtoCF has significant performance gains (over baselines) on the tail items (item-groups G_1 to G_8) while maintaining comparable performance on the head items (item-groups G_9 to G_{10}).

- **Remove Knowledge Distillation:** We test the importance of item-to-item relationships transferred from the base recommender via distillation loss L_G; here, we exclude knowledge transfer from group embeddings $\boldsymbol{Z}_M$ and only train on the few-shot loss L_P. Removing distillation loss L_G reduces few-shot results by 10-19%.
- **Averaging-based Prototype (ProtoCF-AVG):** We directly use the averaging-based initial item prototype (Equation 5) for inference; here, we also exclude the pre-trained parameter initialization for the user encoder G_U (from F_U in the base recommender). ProtoCF-AVG (without any knowledge transfer) is worse than ProtoCF by a margin of 20-30%, yet outperforms the base recommender VAE-CF by nearly 30% on few-shot items.
- **Logistic log-likelihood (ProtoCF-logistic):** We train ProtoCF using *logistic* log-likelihood (Equation 17). ProtoCF with multinomial log-likelihood (Equation 16) outperforms ProtoCF-logistic by a considerable margin; this validates prior findings [24] on the efficacy of multinomal for top-N recommendation.

We further note that the meta-learning baseline MetaRec-LWA improves few-shot results (compared to VAE-CF), but forgets knowledge of head items resulting in poor overall recommendations.

5.7 Parameter Sensitivity (RQ$_4$)

We analyze sensitivity to the hyper-parameter λ that weights the knowledge distillation loss L_G (Equation (13)). In Figure 6 (a), we show the effect of λ on few-shot results with base recommenders BPR and VAE-CF on Gowalla. We empirically find the optimal value of λ to be 0.01 for both models, which also transfers across datasets.

We investigate the impact of *task size*, which is the number of items N sampled in each meta-training task $\mathcal{T}$. In Figure 6 (b), performance typically increases with task size (and stabilizes ~ 400), due to a larger set of sampled items available for ranking; this observation is consistent with prior few-shot learning studies [38].

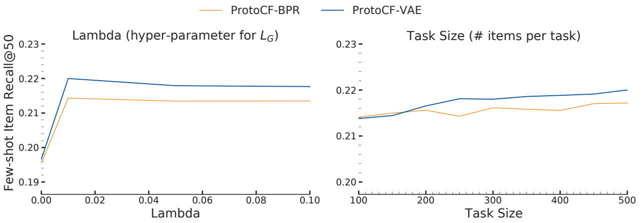

Figure 6: Few-shot performance on Gowalla (for tail items with less than 20 training interactions) is higher for larger meta-training tasks; the empirically optimal value of balance factor $\lambda = 0.01$ also transfers across all datasets.

5.8 Discussion

Our framework ProtoCF is orthogonal to advances in neural recommenders that enhance representational capacity to learn from massive interaction data. We adapt expressive neural recommenders to develop light-weight few-shot models tailored to the long-tail. Furthermore, our formulation requires no side information and can be easily adapted to address the long-tail of users.

ProtoCF learns a metric space predicated on knowledge transfer over a shared user space. Few-shot transfer learning across domains with disjoint feature spaces is a potential future direction.

6 CONCLUSION

This paper formulates long-tail item recommendation as learning-to-embed items with sparse interactions. A novel few-shot learning framework ProtoCF is introduced to extract meta-knowledge across a collection of training tasks designed to simulate tail item ranking. ProtoCF efficiently transfers knowledge from arbitrary base recommenders to construct discriminative prototypes for items with very few interactions. Our experiments indicate 5% overall performance gains (Recall@50) for ProtoCF over the state-of-the-art, with notable 60-80% few-shot performance gains (Recall@50) on long-tail items with less than 20 training interactions.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

[2] Alex Beutel, Kenton Murray, Christos Faloutsos, and Alexander J Smola. 2014. Cobafi: collaborative bayesian filtering. In *WWW*. 97–108.

[3] Rodrigo Borges and Kostas Stefanidis. 2020. On Measuring Popularity Bias in Collaborative Filtering Data. In *EDBT/ICDT Workshops*.

[4] Da Cao, Liqiang Nie, Xiangnan He, Xiaochi Wei, Shunzhi Zhu, and Tat-Seng Chua. 2017. Embedding factorization models for jointly recommending items and user generated lists. In *SIGIR*. 585–594.

[5] Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jaeho Choi. 2019. Rating augmentation with generative adversarial networks towards accurate collaborative filtering. In *WWW*. 2616–2622.

[6] Dong-Kyu Chae, Jihoo Kim, Duen Horng Chau, and Sang-Wook Kim. 2020. AR-CF: Augmenting Virtual Users and Items in Collaborative Filtering for Addressing Cold-Start Problems. In *SIGIR*. 1251–1260.

[7] Zhihong Chen, Rong Xiao, Chenliang Li, Gangfeng Ye, Haochuan Sun, and Hongbo Deng. 2020. ESAM: Discriminative Domain Adaptation with Non-Displayed Items to Improve Long-Tail Performance. In *SIGIR*. 579–588.

[8] Manqing Dong, Feng Yuan, Lina Yao, Xiwei Xu, and Liming Zhu. 2020. Mamo: Memory-augmented meta-optimization for cold-start recommendation. In *KDD*. 688–697.

[9] Zhengxiao Du, Xiaowei Wang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Sequential Scenario-Specific Meta Learner for Online Recommendation. In *KDD*. 2895–2904.

[10] Miao Fan, Yeqi Bai, Mingming Sun, and Ping Li. 2019. Large margin prototypical network for few-shot relation classification with fine-grained features. In *CIKM*.

[11] Spyros Gidaris and Nikos Komodakis. 2018. Dynamic few-shot visual learning without forgetting. In *CVPR*. 4367–4375.

[12] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.

[13] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[14] Andre Holzapfel, Bob Sturm, and Mark Coeckelbergh. 2018. Ethical dimensions of music information retrieval technology. *TISMIR* 1, 1 (2018), 44–55.

[15] Daeryong Kim and Bongwon Suh. 2019. Enhancing VAEs for collaborative filtering: flexible priors & gating mechanisms. In *RecSys*. 403–407.

[16] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[17] Adit Krishnan, Hari Cheruvu, Cheng Tao, and Hari Sundaram. 2019. A modular adversarial approach to social recommendation. In *CIKM*. 1753–1762.

[18] Adit Krishnan, Mahashweta Das, Mangesh Bendre, Hao Yang, and Hari Sundaram. 2020. Transfer Learning via Contextual Invariants for One-to-Many Cross-Domain Recommendation. In *SIGIR*. 1081–1090.

[19] Adit Krishnan, Ashish Sharma, Aravind Sankar, and Hari Sundaram. 2018. An adversarial approach to improve long-tail performance in neural collaborative filtering. In *CIKM*. 1491–1494.

[20] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. 2019. MeLU: Meta-Learned User Preference Estimator for Cold-Start Recommendation. In *KDD*. 1073–1082.

[21] Aoxue Li, Tiange Luo, Zhiwu Lu, Tao Xiang, and Liwei Wang. 2019. Large-scale few-shot learning: Knowledge transfer with class hierarchy. In *CVPR*. 7212–7220.

[22] Xiaopeng Li and James She. 2017. Collaborative variational autoencoder for recommender systems. In *KDD*. 305–314.

[23] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M Blei. 2016. Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *RecSys*. 59–66.

[24] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *WWW*. 689–698.

[25] Yong Liu, Wei Wei, Aixin Sun, and Chunyan Miao. 2014. Exploiting geographical neighborhood characteristics for location recommendation. In *CIKM*. 739–748.

[26] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. 2019. Large-scale long-tailed recognition in an open world. In *CVPR*. 2537–2546.

[27] Yuanfu Lu, Yuan Fang, and Chuan Shi. 2020. Meta-learning on Heterogeneous Information Networks for Cold-start Recommendation. In *KDD*. 1563–1573.

[28] Mi Luo, Fei Chen, Pengxiang Cheng, Zhenhua Dong, Xiuqiang He, Jiashi Feng, and Zhenguo Li. 2020. MetaSelector: meta-learning for recommendation with user-level adaptive model selection. In *WWW*. 2507–2513.

[29] Jingwei Ma, Jiahui Wen, Mingyang Zhong, Liangchen Liu, Chaojie Li, Weitong Chen, Yin Yang, Hongkui Tu, and Xue Li. 2019. DBRec: Dual-Bridging Recommendation via Discovering Latent Groups. In *CIKM*. 1513–1522.

[30] Bernard Marr. 2018. (2018). https://www.forbes.com/sites/bernardmarr/2018/04/18/netflix-used-big-data-to-identify-the-movies-that-are-too-scary-to-finish/

[31] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. 2019. Relational knowledge distillation. In *CVPR*. 3967–3976.

[32] Yoon-Joo Park. 2012. The adaptive clustering method for the long tail problem of recommender systems. *IEEE TKDE* 25, 8 (2012), 1904–1915.

[33] Sachin Ravi and Hugo Larochelle. 2017. Optimization as a Model for Few-Shot Learning. In *ICLR*. OpenReview.net.

[34] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*.

[35] Aravind Sankar, Yozen Liu, Jun Yu, and Neil Shah. 2021. Graph Neural Networks for Friend Ranking in Large-scale Social Platforms. In *WWW*. 2535–2546.

[36] Aravind Sankar, Yanhong Wu, Yuhang Wu, Wei Zhang, Hao Yang, and Hari Sundaram. 2020. Groupim: A mutual information maximization framework for neural group recommendation. In *SIGIR*. 1279–1288.

[37] Aravind Sankar, Xinyang Zhang, Adit Krishnan, and Jiawei Han. 2020. Inf-vae: A variational autoencoder framework to integrate homophily and influence in diffusion prediction. In *WSDM*. 510–518.

[38] Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *NeurIPS*. 4077–4087.

[39] Rama Syamala Sreepada and Bidyut Kr Patra. 2020. Mitigating long tail effect in recommendations using few shot learning technique. *Expert Sys. App.* 140 (2020).

[40] Qiuling Suo, Jingyuan Chou, Weida Zhong, and Aidong Zhang. 2020. TAdaNet: Task-Adaptive Network for Graph-Enriched Meta-Learning. In *KDD*. 1789–1799.

[41] Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*. 1422–1432.

[42] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Latent relational metric learning via memory-based attention for collaborative ranking. In *WWW*. 729–739.

[43] Manasi Vartak, Arvind Thiagarajan, Conrado Miranda, Jeshua Bratman, and Hugo Larochelle. 2017. A meta-learning perspective on cold-start recommendations for items. In *NeurIPS*. 6904–6914.

[44] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *NeurIPS*. 3630–3638.

[45] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. Dropoutnet: Addressing cold start in recommender systems. In *NeurIPS*. 4957–4966.

[46] Huiwei Wang and Yong Zhao. 2020. ML2E: Meta-Learning Embedding Ensemble for Cold-Start Recommendation. *IEEE Access* 8 (2020), 165757–165768.

[47] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *KDD*. 950–958.

[48] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*. 165–174.

[49] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. 2017. Learning to model the tail. In *NeurIPS*. 7029–7039.

[50] Tianxin Wei, Ziwei Wu, Ruirui Li, Ziniu Hu, Fuli Feng, Xiangnan He, Yizhou Sun, and Wei Wang. 2020. Fast Adaptation for Cold-start Collaborative Filtering with Meta-learning. ICDM.

[51] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *WSDM*. 153–162.

[52] Yao Wu, Xudong Liu, Min Xie, Martin Ester, and Qing Yang. 2016. CCCF: Improving collaborative filtering via scalable user-item co-clustering. In *WSDM*.

[53] Gui-Rong Xue, Chenxi Lin, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu, and Zheng Chen. 2005. Scalable collaborative filtering using cluster-based smoothing. In *SIGIR*.

[54] Hongzhi Yin, Bin Cui, Jing Li, Junjie Yao, and Chen Chen. 2012. Challenging the long tail recommendation. *arXiv preprint arXiv:1205.6700* (2012).

[55] Runsheng Yu, Yu Gong, Xu He, Bo An, Yu Zhu, Qingwen Liu, and Wenwu Ou. 2020. Personalized Adaptive Meta Learning for Cold-start User Preference Prediction. *AAAI* (2020).

[56] Yin Zhang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Lichan Hong, and Ed H Chi. 2021. A Model of Two Tales: Dual Transfer Learning Framework for Improved Long-tail Item Recommendation. In *WWW*. 2220–2231.

[57] Ziwei Zhu, Shahin Sefati, Parsa Saadatpanah, and James Caverlee. 2020. Recommendation for New Users and New Items via Randomized Training and Mixture-of-Experts Transformation. In *SIGIR*. 1121–1130.