Mitigating Extreme Cold Start in Graph-based RecSys through Re-ranking

Alessandro Sbandi
TIM S.p.A.
Rome, Italy
Sapienza University of Rome
Rome, Italy
alessandro.sbandi@telecomitalia.it

Federico Siciliano
Sapienza University of Rome
Rome, Italy
siciliano@diag.uniroma1.it

Fabrizio Silvestri
Sapienza University of Rome
Rome, Italy
fsilvestri@diag.uniroma1.it

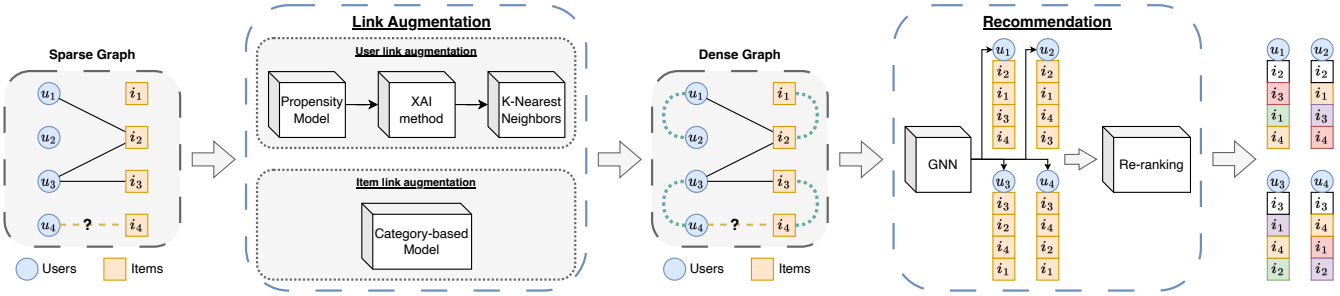

Figure 1: Overview of the proposed framework. The input sparse graph is augmented with edges derived from user similarities and item categories, resulting in an enriched dense graph. Subsequently, a Graph Neural Network model is trained on this augmented graph. Finally, re-ranking algorithm is applied to the model predictions to increase recommendation diversity.

Abstract

Recommender systems based on Graph Neural Networks (GNN) have become the state-of-the-art approach in recommendation, but they struggle with in extreme cold-start settings, where most users or items lack interaction data. This paper proposes a novel framework to address this challenge in four steps: (i) a propensity model to predict item purchase behaviour, with associated explainability to identify the most relevant features, (ii) a link augmentation module to connect users based on previously obtained similarities, (iii) a GNN-based link prediction step on the obtained dense graph and (iv) a final re-ranking stage to increase diversity in predictions leveraging users embeddings. By exploiting the enriched graph structure, the framework generates embeddings for cold-start users and items, enabling diverse recommendations, containing long tail and unsold items, for both established and new users. We validate the framework's effectiveness on real-world industrial data from TIM S.p.A.

CCS Concepts

• **Information systems** → **Recommender systems**; • **Mathematics of computing** → *Graph algorithms*; • **Computing methodologies** → *Learning to rank*.

CIKM '24, October 21–25, 2024, Boise, ID, USA

ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3680069

Keywords

Recommender systems, Cold Start, Graph Neural Networks

ACM Reference Format:
Alessandro Sbandi, Federico Siciliano, and Fabrizio Silvestri. 2024. Mitigating Extreme Cold Start in Graph-based RecSys through Re-ranking. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3627673.3680069

1 Introduction

Recommender systems aim to suggest items to users that are most likely to appeal to them. With the rise of the Internet and the availability of huge amounts of data, these algorithms have become essential. Traditional methods like collaborative filtering rely on user-item interaction similarity [36]. However, these approaches remain significantly limited as their prediction and training stages overlook high-order structural information present in observed data [3, 22].

The emergence of deep learning has led to Graph Neural Networks (GNNs) outperforming traditional methods. GNNs overcome the aforementioned problems by exploiting embedding propagation, iteratively aggregating information from neighboring nodes in a graph structure [21]. Through the stacking of propagation layers, each node gains access to information beyond immediate neighbours, surpassing the limitations of traditional methods that only consider first-order neighbors. The strength of GNNs lies in their ability to learn good user and item representations (embeddings) and generate high-quality recommendations, especially for established users and items (warm scenarios) [30, 40].

Nevertheless, GNNs face challenges in extreme cold-start scenarios, where most users and items lack interaction data. These

users and items appear as isolated nodes in the user-item bipartite graph, hindering information propagation and embedding updates.

Many existing approaches [20, 38] have used side information to alleviate this problem. [17] propose pre-trained GNNs and neighbor sampling for cold-start items. More recently, [7] presents a two-step framework to treat warm and cold items differently.

Our paper proposes a practical solution to the extreme cold-start problem in a real large-scale industrial setting, focusing on mitigating the problem for both new users (no purchase history) and new items (minimal or missing sales data). Our contributions include:

- **Identifying User Similarities with XAI:** We leverage Explainable AI (XAI) to identify characteristics of warm users. This allows us to connect users with similar behavior patterns, even if they haven't interacted with the same items.
- **Building a Dense Graph for Information Propagation:** We enrich the initial bipartite heterogeneous user-item graph by linking cold nodes to their nearest warm counterparts, according to the XAI patterns, enabling information flow across the entire graph.
- **GNN-based Recommendation with Re-ranking:** Finally we implement a re-ranking algorithm to promote recommendation diversity while simultaneously maintaining the quality of the previous GNN predictions.

Our approach demonstrates success in generating diverse and accurate recommendations in cold-start scenarios: for new items - with minimal, or no sales history or where sales history is missing - and new users - those who have never made a purchase.

2 Background

2.1 XAI for Feature Selection

Understanding the inner workings of a predictive model is crucial for trusting and effectively utilizing its outputs. In recent years, a large number of XAI methodologies have emerged to tackle the complexities inherent in addressing this specific problem [26, 34]. Often, a simpler *explanation model* is used [31]. This simpler model is essentially a more straightforward version designed to help you understand the complexities of the original, more "complicated" model. Shapley values are also a popular XAI technique to explain the contribution of each feature to the output of a Machine Learning (ML) model. The goal is to decompose the model prediction and assign importance scores (Shapley values) to individual features [35]. The explanation method SHAP [25] is a common tool to compute Shapley values. In the context of feature selection, Shapley values can be used to identify the most relevant features of an ML model. In this situation, the features themselves become the *players* in a cooperative selection game, and model performance acts as the *payoff* [11, 15]. Formally, the feature selection game is defined as follows [33]:

Definition 2.1. *Feature selection game.*

Let the players' set be $\mathcal{N} = \{1, ..., n\}$, where n is the total number of features. A feature subset $\mathcal{S} \subseteq \mathcal{N}$ represents a specific combination (coalition) of features. For each coalition, the train and test feature vector sets are defined as $X_{\mathcal{S}}^{Train} = \{\boldsymbol{x}_i^{Train} | i \in S\}$ and $X_{\mathcal{S}}^{Test} = \{\boldsymbol{x}_i^{Test} | i \in \mathcal{S}\}$. Let $f_{\mathcal{S}}(\cdot)$ be a machine learning model trained using $X_{\mathcal{S}}^{Train}$ as input, then the payoff is $v(\mathcal{S}) = g(\boldsymbol{y}, \hat{\boldsymbol{y}}_{\mathcal{S}})$, where $g(\cdot)$ is a goodness of fit function, $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}_{\mathcal{S}} = f_{\mathcal{S}}(X_{\mathcal{S}}^{Test})$ are the ground truth and predicted targets.

Shapley values have been studied as a measure of feature importance in various contexts [9] and applied in several tasks, including human action recognition [13] and natural language processing [29]. Model features can be ranked, selected, or removed by exploiting the Shapley importance.

2.2 Graph Construction and Link Augmentation

We consider an unweighted heterogeneous bipartite graph $G = (U, I, E, X)$, where U is the set of user nodes, I is the set of item nodes, E is the set of edges connecting users and items based on past interactions, and $X \subseteq \mathbb{R}^{d \times |U|}$ is a matrix of d user features. The task is then formalized as a link prediction task [23]. In a cold-start scenario, many users and items lack interaction data, resulting in isolated nodes that hinder information propagation to the local neighbors, making it impossible to capture graph topology and obtain recommendations. To overcome this problem, Gupta and Shrinath [14] leverage the K-Nearest Neighbors (KNN) algorithm to identify similar users and items based on their features and connect the closest ones. This approach has been shown to outperform traditional recommender systems when applied to huge sparse networks. We employ a similar technique to perform link augmentation over the graph, connecting user-user and item-item nodes based on their similarity.

2.3 GNNs for Recommendation

GNNs are powerful tools that can capture the structural information of a graph and learn node representations. This operation is performed by message passing, where nodes aggregate and update information from their neighbours in iterative steps. The feature matrix X can either represent existing node features or be randomly initialized, which significantly increases the expressiveness at the cost of slower model convergence [1].

The message-passing paradigm has been widely used in user-item GNNs [18, 39]. The messages are iteratively aggregated and updated to obtain informative node embeddings. A single message passing iteration is written as:

$$\mathbf{h}_u^{(k+1)} = UPD^k(\mathbf{h}_u^{(k)}, AGG^k(\{\mathbf{h}_v^{(k)}, \forall v \in N(u)\})) \tag{1}$$

where $\mathbf{h}_u^{(k)}$ is node u embeddings at the k-th step, with $\mathbf{h}_u^{(0)} = X_u$, i.e. the u-th row of the feature matrix X. $N(u)$ denotes the set of neighbors of node u and UPD and AGG are the update and aggregation permutation-invariant operation, respectively. Many differentiable approaches exist for the aggregation operation [21, 37]. After multiple iterations, each node j obtains a final embedding h_j that captures its local and global context within the graph.

Finally, the predicted likelihood of u buying item i is obtained using the inner product of their embeddings:

$$\hat{y}_{ui} = \mathbf{h}_u^T \mathbf{h}_i \tag{2}$$

Those obtained likelihoods allow us to rank the items for each user u, recommending the ones most likely relevant for him.

Figure 2: A schematic representation of the proposed framework for graph construction. (A) The initial sparse graph. (B) A propensity model is trained, then (C) Shapley values are computed to select the most representative features. (D) A bipartite graph is built using interaction data, with link augmentation for users and items. Finally, a dense graph (E) is obtained.

2.4 Re-ranking for Recommendation Diversity

As the field of Recommender systems continues to evolve, various critical issues have been identified in the literature [6, 19], like the *Matthew effect* [24, 41] and *filter bubble* [28].

The *Matthew effect*, which can be summarized as "*the rich get richer and the poor get poorer*", occurs in recommendation when popular items or categories have increased chances of being recommended, while the less popular ones are overlooked. This is a critical concern to users seeking diverse recommendations, negatively impacting user satisfaction, even if they initially favoured such content. This lack of diversity can lead to the *filter bubble* phenomenon, where individuals are repeatedly exposed to content that reinforces their existing preference, isolating them from diverse perspectives and information. As a consequence, it fortifies previous preferences and forces the user to delve deeper into them. This situation is also called "group polarization" [12].

Extremely personalized recommendations can exacerbate these issues. To avoid this situation, it is imperative to introduce diversity into them. One way to do so is by solving a multi-objective optimization problem, balancing personalization and diversity.

3 Methodology

3.1 Graph construction

Interaction data. The first step involves building a bipartite heterogeneous graph, where nodes represent users and items, and edges connect users to items they have purchased.

In extreme cold-start scenarios, most nodes are isolated, due to the lack of interaction data for most users and items. We need to address this sparsity in order to make a viable prediction for isolated nodes. To do so, we employ link augmentation to create additional edges. As discussed in Sec. 2.2, the main idea behind our approach is to connect similar users and items. While for items we can just use their categorization, for users we need a way to describe them based on the most relevant factors influencing purchasing behavior.

Propensity model. Each user is represented by 600 features. Due to computational complexity, which is a major concern in industrial applications, we decide to reduce the number of features to 150 by performing PCA. This results in capturing over 80% of the explained variance, but with a 75% reduction of space.

A *propensity model* is trained to simulate a user's likelihood to purchase items: users who have bought at least one item are labeled as positive, while all the others are considered negative. For the actual algorithm, we decided to use an XGBoost binary classifier [8], a scalable end-to-end tree boosting model. Regarding the data, "no purchase" users significantly outnumber purchasing users, leading to a strongly unbalanced dataset. To avoid data splitting and overfit the model for retrieving the most relevant features, we address the imbalance by undersampling the majority class ("no purchase"). We then perform a grid search to optimize the XGBoost hyperparameters: maximum tree depth, learning rate and regularization. We consider optimal the model that achieves the highest F1 score.

Once the propensity model is trained, we compute its Shapley values. Typically used in XAI, they quantify the contribution of each feature to the model's prediction. This allows us to identify the top-10 most influential features for explaining user purchase behavior. Ultimately, this method reduces the computational complexity of similarity by 60 times, making the solution highly valuable for industrial applications where efficiency and interpretability are crucial.

3.1.1 Link augmentation. The key features of the *propensity model* identified via the Shapley values are utilized to connect users with similar characteristics in the graph construction stage. Users with similar behaviour are linked via Approximate Nearest-Neighbor (ANN). Connecting users with similar profiles is beneficial as they tend to exhibit similar purchasing patterns. This strategy helps predict new interactions based on established user behaviour. Especially in cold-start scenario, a user might influence other users connected to him, even though these haven't made a purchase yet.

We employ a similar approach connecting item nodes based on their categorization. This is based on the assumption that a user who already expressed interest in one item is more likely to buy another item in the same category. Linking isolated nodes within the same category facilitates information propagation between them, enabling the GNN model to capture latent relationships and similarities among items, even if initially missing direct connections. In this way, every item, even the unsold ones, is linked to the final graph.

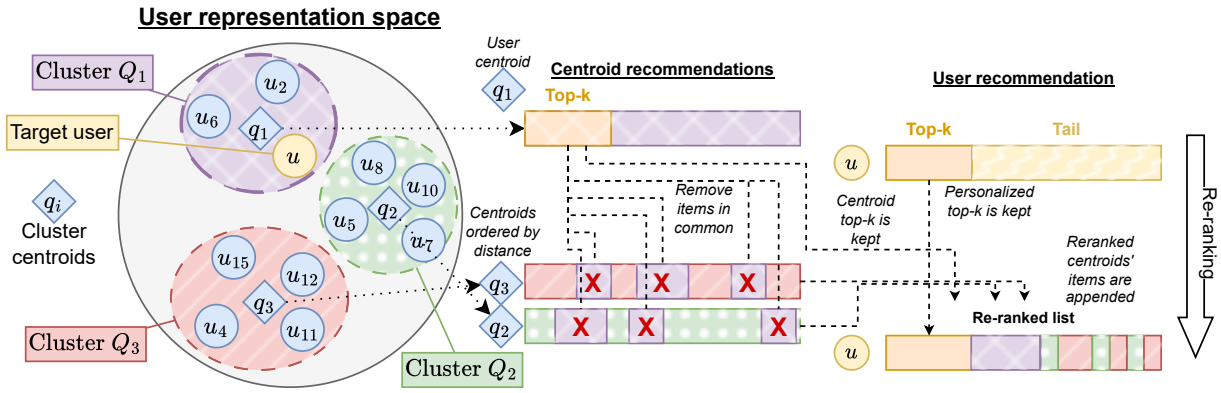

Figure 3: Proposed re-ranking algorithm. In each of the three clusters, the closest user to the centroid is considered as the centroid recommendation list. For each user, the top recommendations are frozen, and the others are re-ranked. Its cluster is not taken into consideration; items already present are excluded, while those not present in the intersection remain relevant; finally, in addition to the initially frozen ones, the remaining candidates are concatenated, and sorted by descending probability.

Several open-source Python libraries like FAISS [10] and Hnswlib [27] offer GPU-accelerated ANN implementations. However, for industrial tasks, scalability becomes challenging. We leverage the Annoy library [5] which, thanks to its optimized memory scaling, allows us to augment hundreds of billions of edges.

In Fig. 2 are summarized these steps to achieve a denser graph.

3.2 Recommendation model

3.2.1 GNN model. We adopt a 2-layer GraphSAGE [16] model for our recommender systems. In GraphSAGE, the update function (*UPD*) combines concatenation and a linear layer, while the aggregation function (*AGG*) is a mean-pooling. According to Eq. (2), we compute the dot product between the user and item embeddings obtained from the GNN, providing a score for each item-user pair. Finally, for each user, the items sorted based on their predicted score give a ranked recommendation list. We randomly split the edges of the graph into training, validation and test sets with a 70%-20%-10% ratio. During training, we mask a portion of the edges for link prediction. For mini-batch training on the large-scale graph, we choose batches containing equal numbers of positive (real) edges and negative (non-existing), which are randomly sampled.

We utilize a weighted binary cross-entropy loss function for link prediction, as shown in the equation below:

$$\mathcal{L}(x, y) = -\frac{1}{N}\sum_{n=1}^{N}[y_n \log(x_n) + w_{neg}(1 - y_n)\log(1 - x_n)] \quad (3)$$

where N is the batch size, x_n is the model prediction for the $n-th$ element in the batch, y_n the corresponding target label (0 or 1), and w_{neg} is the parameter to weight the negative loss term.

To identify the best configuration for our GNN model, we perform hyperparameter tuning of input feature vector node dimension, number of neighbours considered in message-passing, model hidden space dimensions, optimizer used in training, and negative weight w_{neg} in the loss function. The objective of hyperparameter tuning is to minimize the loss function on the validation set.

3.2.2 Re-ranking. The GNN produces user and item embeddings, which are vector representations of user preferences and item characteristics, as well as their connections within the graph. To promote recommendation diversity and user serendipity (discovering unexpected but interesting items), we use a re-ranking approach.

Our re-ranking strategy leverages the user embeddings learned by the GNN. We proceed as follows:

(1) **Community Detection:** We aim to identify behavioural communities in the user's embeddings latent space. We achieve this by clustering the users using *k-means||* [4]. The optimal number of clusters is determined by maximizing the Silhouette score [32], a metric for evaluating clustering quality. Let $Q = \{Q_1, \ldots, Q_K\}$ be the set of detected clusters and $q = \{q_1, ..., q_K\}$ their respective centroids.
(2) **Distance to Other Centroids:** We compute the Euclidean distance between each user embedding h_u and cluster centroid q_k: $d = \sqrt{(h_u - q_k)^2}$.
(3) **Cluster Centroid Recommendations:** For each cluster centroid q_k, we determine the closest user u. This user's recommendation item list is considered as the base recommendation for all users in the same cluster Q_k.
(4) **Distance between centroids:** For each cluster centroid q_k, we compute the Euclidean distance among all of them q_k: $d = \sqrt{(q_u - q_k)^2}$. Then we choose the furthest as good candidates for the centroid re-ranking, $\hat{q_k}$, where $\hat{q_k} = \{q_1, ..., q_F\}$ is the set of candidates for q_k.
(5) **Re-ranking Centroid Tail Recommendations:** We freeze the top K recommendations for each centroid and the tails are re-ranked. First, we remove from each centroid candidate q_f, the top recommendations of the centroid q_k. We consider for re-ranking those items that remain both in the recommended tail list of centroid q_k, Y_k, and the other centroids lists candidates, Y_f. This set is formally indicated as $Y_{k,f} = Y_k \cap Y_f$, $f \in \{1, \ldots, F\}$, with f the set of furthest

centroids. The selected items $Y_{k,f}$ are sorted based on importance (position) in other lists, ordered from the furthest centroid list to the closest, and appended to the top K frozen centroids recommendations to form the final centroid recommendation list $\hat{Y}_k$.

(6) **Re-ranking User Tail Recommendations:** Once the centroid lists have been re-ranked, we re-rank user recommendations based on his cluster centroid. In order to maintain the user personalization provided by the GNN, we decided to freeze the top-K user recommendations and re-rank the user tails based on their re-ranked cluster. To achieve this, once again, we initially removed the top-K when comparing user recommendations with those of their centroid, and then appended the remaining ones following the cluster recommendation's order.

This re-ranking approach preserves the quality of GNN results by retaining the first K recommendations. Simultaneously, it increases diversity by taking into account other communities' interests, potentially leading to serendipitous discoveries for users. Additionally, it helps reducing the group polarization problem. A potential drawback of this approach is that re-ranking all the tail items in the same order within a cluster might lead to some homogeneity. To bypass the problem, we create a large number of micro-clusters. By grouping users into smaller communities, the overlap in tail recommendations among users is reduced. Essentially. the tail recommendations are identical only for users belonging to the same micro-cluster. Re-ranking based only on centroids offers a significant computational advantage compared to using individual user recommendations. It reduces the cost from $|U| * K$ to K^2. Additionally, pre-computing tail recommendations for each cluster allows for faster response times for new users. This translates to substantial efficiency gains, especially for large user bases, making the solution highly scalable for industrial deployments. The described re-ranking algorithm is visualized in Fig. 3.

4 Experimental Results

4.1 Experimental Setup

Research Questions. We conduct a series of experiments to evaluate the effectiveness of our proposed approach, particularly focusing on its ability to handle cold-start and long-tail item recommendations. Our evaluation aims to answer the following research questions.

- **RQ1:** does the re-ranking algorithm improve recommendation diversity?
- **RQ2:** does the re-ranking algorithm promote long-tail item recommendations?
- **RQ3:** is the re-ranking algorithm robust to hyperparameter variations?

Table 1: The description of the dataset.

Dataset	Industrial			
	# tot	# warm	# cold	# extremely cold
items	693	29	412	252
users	1080681	13622	169505	897554

Dataset. TIM S.p.A. supplies various commodities to the customers, generally called IT items. However, the dataset provided by the company exhibits a significant cold-start challenge, as only 17% of the customer base has purchased any item. Among those, 93% have only bought one item. Approximately 40% of the items have never been sold, and the top three best-selling items collectively account for half of all sales. This extreme cold-start scenario presents a significant challenge for recommender systems. A detailed summary of the data is shown in Tab. 1.

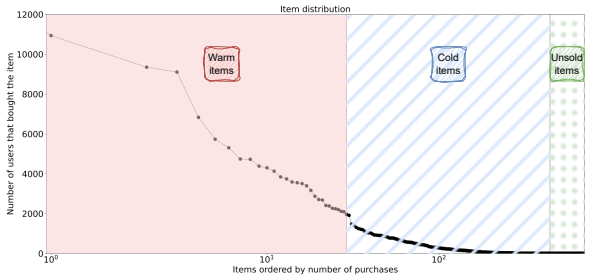

Figure 4: Internal dataset distribution: warm, cold, and unsold items are highlighted.

Given the dataset's characteristics, we deviate from the traditional approach of splitting items into warm and cold categories according to the Pareto Principle [2]. Instead, as presented in Tab. 1, we define user and item categories based on purchase behaviour: **warm users** have purchased more than one item; **cold users** have only one purchase; **extremely cold users** have no purchases. **warm items** represent more than 1% of the total stock.; **cold items** represent less than 1% of the total stock; **unsold items** have never been purchased. The item distribution is illustrated in Fig. 4.

Evaluation Metrics. We employ several metrics to assess the performance of our method.

Intra-List Distance (ILD) measures the average pairwise distance between items in a user's recommendation list.

$$ILD(R_u) = \frac{1}{|R_u|(|R_u|-1)} \sum_{i \in R_u} \sum_{j \in R_u \setminus i} d(i,j) \tag{4}$$

where $d(i,j)$ is the Euclidean distance between item embeddings derived from the graph, and $|R_u|$ is the cardinality of the set of items.

Coverage@K: This metric calculates the proportion of unique items appearing in the top-K recommendations across all users:

$$Coverage@K = \frac{|\cup_{u \in U} Y_K(u)|}{|R_u|} \tag{5}$$

where $Y_K(u) = [i_1, ..., i_k]$ represent the top-K recommendation list for user u and R_u the set of all items.

Type-Coverage@K: Similarly, this metric focuses on the coverage given to long-tail items:

$$Type-Coverage@K = \frac{|\cup_{u \in U} Y_K^{Type}(u)|}{|R_u^{Type}|} \tag{6}$$

where $Y_K^{Type}(u)$ and $|R_u^{Type}|$ refer to a specific item type.

Table 2: ILD results of GNN, Random Shifted GNN and re-ranking algorithm. Experiments are repeated five times with a different weight initialization; mean and variance are reported. Bold indicates the best result, while our metrics surpassing the baselines are underlined. * denotes statistically significant (p<0.01) improvements over the top-performing baseline.

Method	ILD@K				
	K=5	K=10	K=20	K=50	K=100
GNN	6.5249 ± 1.9822	6.4746 ± 1.7203	6.6528 ± 1.7000	6.6797 ± 1.7088	7.1063 ± 1.8448
Random Shifted GNN	7.0976 ± 4.2660	1.5773 ± 0.9480	0.3735 ± 0.2270	0.0580 ± 0.0348	0.0143 ± 0.0086
Re-ranked F@5	-	**16.9174 ± 3.5631***	**22.2285 ± 1.4671***	**22.1086 ± 0.5082***	**21.9093 ± 0.2157***
Re-ranked F@10	-	-	10.8625 ± 1.8004*	10.3264 ± 0.9937*	10.1021 ± 0.4554*
Re-ranked F@20	-	-	-	11.0949 ± 0.6530*	12.6525 ± 0.2326*
Re-ranked F@50	-	-	-	-	15.2856 ± 1.5690*

Table 3: Coverage results of GNN results and after re-ranking. Bold indicates the best result, while second-best is underlined.

Method	Coverage@K									
	K=5	K=10	K=20	K=50	K=100	K=200	K=300	K=400	K=500	K=600
GNN	**0.3255**	**0.3967**	**0.4877**	**0.6582**	**0.8172**	**0.9389**	**0.9895**	**1.0000**	**1.0000**	**1.0000**
Re-ranked F@5	-	0.3578	0.3839	0.4176	0.4688	0.5679	0.6510	0.7471	0.8323	0.9248
Re-ranked F@10	-	-	0.4294	0.4825	0.4983	0.5788	0.6488	0.7312	0.8323	0.9219
Re-ranked F@20	-	-	-	0.5004	0.5365	0.6137	0.6941	0.7890	0.8699	0.9364
Re-ranked F@50	-	-	-	-	0.7036	0.7736	0.8255	0.8800	0.9205	0.9624

Table 4: Type-Coverage improvement results for different K. Experiments are repeated freezing at different ranges in re-ranking algorithm. The coverage improvements are presented for warm (W), cold (C) and unsold (U) items. Bold indicates the type of item that has lower decreases for a specific K, underlining represents the general best results for the type of item.

Method	Type-Coverage@K					
	W@100	C@100	U@100	W@200	C@200	U@200
GNN	0.8534	0.7384	0.9385	0.9493	0.9017	0.9921
Improvements F@5	-0.5557	-0.6589	**-0.1204**	-0.4828	-0.5730	**-0.1191**
Improvements F@10	-0.4500	-0.5876	**-0.1623**	-0.3461	-0.5309	**-0.1841**
Improvements F@20	**-0.2501**	-0.3799	-0.3018	**-0.2414**	-0.3873	-0.2939
Improvements F@50	-0.0833	-0.1931	**-0.0736**	-0.0812	-0.2434	**-0.0755**

We evaluate the Type-Coverage@K differentiating between warm, cold and unsold items.

Baselines. Currently, TIM S.p.A. is not using any machine learning algorithm to recommend IT items. In this extreme cold-start setting, good performances would correspond to overfitting on the few items sold. This is opposite to the idea of pushing a limited amount of items previously decided in every campaign. All these reasons add up to why we consider the GNN as one of the baselines against which to compare our re-ranking approach.

Additionally, we decide to compare our results with a random baseline. For each user, we randomly shuffle the recommended item list. In Tab. 2, we refer to it as Random Shifted GNN, and the ILD results are calculated based on the obtained GNN item embeddings.

4.2 Results

This section presents the evaluation results for our proposed approach, addressing the research questions outlined earlier.

Diversity. To answer **RQ1**, we evaluate the impact of re-ranking on recommendation diversity by comparing the ILD obtained by the GNN with that resulting from the re-ranking algorithm. The results presented in Tab. 2 show that re-ranking methods increase the ILD of the recommendations by **+126.7%** in average, across list lengths ranging from 10 to 100. This improvement is attributed to the re-ranking strategy. By freezing the top-K recommendations and incorporating items from the furthest cluster centroids, the model injects items with a lower user-community overlap, leading to more diverse recommendations.

Long tails. However, altering the user tails using those of the centroids, introduces a trade-off. Since users within the same cluster

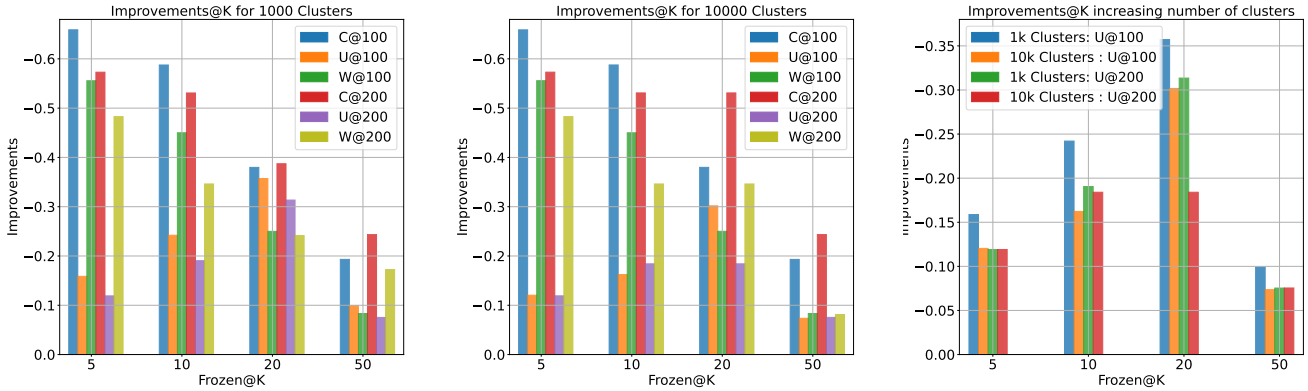

Figure 5: From left to right: Type-Coverage@K improvements for a low number of clusters, Type-Coverage@K improvements for a high number of clusters, and Unsold-Coverage@K improvements between low and high number of clusters.

probably receive similar tail recommendations, the overall item coverage is expected to decrease. Tab. 3 summarize the coverage results to answer **RQ2**. As expected, overall coverage decreases due to re-ranking. However, since our focus is on long-tail items, we should turn our attention to Type-Coverage@K, shown in Tab. 4. The coverage for unsold items shows the smallest decrease, while cold items show less coverage than warm items. This suggests that the re-ranking algorithm is effective at promoting long-tail recommendations, particularly for items that have never been purchased before.

Hyperparameter sensitivity. To assess the robustness of our approach to hyperparameter variations **RQ3**, we present the effect of four hyperparameters in re-ranking algorithm: the number of clusters, the number of candidates for each centroid, the number of items to freeze in the centroid-centroid and in the user-centroid re-ranking step. In order to reduce the number of experiments, we set the same number of frozen items in centroid re-ranking and user re-ranking, reducing to three hyperparameters.

We use the top 10 and top 50 furthest centroids in re-ranking approach and we notice that the number of furthest centroids considered is not impacting the ILD and type-coverages up to one hundred frozen items. So, we go deep into the variations in the number of clusters and the amount of items to freeze.

The number of frozen recommendations determines how much GNN's initial predictions are preserved. As shown in Tabs. 2 to 4, a higher number of frozen items produces both a higher ILD and a higher coverage for all three item types. As shown in Fig. 5, the coverages decrease more as the number of frozen items increases.

Furthermore, the deterioration in coverage for unsold items increases with each K up to 20, and at 50, they have the lowest value overall. At 50, the degradation for cold items is at its highest. This is because the number of warm items is 29, and for low K values, the re-ranking algorithm places tail items (unsold) in the top positions. Once the number of warm items is exceeded, these unsold items end up disadvantaging the cold items, lowering their type coverage.

Moreover, when the number of clusters increases, the amount of unsold items is always decreasing, for each number of frozen items, @100 and @200. This happens because having a larger number of clusters allows for the identification of more behaviors within the micro-clusters, thereby favoring the unsold items more.

Finding the optimal hyperparameter configuration requires careful consideration of the desired balance between diversity and coverage within the specific application context.

The results presented in RQ1 and RQ2 are obtained with one thousand clusters and ten furthest centroid candidates.

5 Conclusions

This paper addressed the extreme cold-start problem for new users and items in a recommender system. We proposed a novel end-to-end framework designed to generate recommendations in an extreme cold-start scenario. The framework utilizes relevant customer features to connect users, aiming to prevent isolated nodes in the graph and facilitate information flow. Additionally, we introduced a re-ranking algorithm to enhance user diversity.

We conducted experiments to evaluate the effectiveness of our approach on a real-world dataset with extreme cold-start characteristics. The results show that our method significantly improves recommendation diversity compared to the baseline GNN model. Additionally, it shows promising results in promoting long-tail item recommendations, especially for unsold items.

For future work, we plan to incorporate additional information sources, such as external surveys, to improve GNN embeddings' quality. The algorithm has also been deployed in production and we will monitor user feedback to refine the framework. Finally, we aim to investigate state-of-the-art neural multi-objective re-ranking models for potentially even better performance and flexibility.

Acknowledgments

This work was partially supported by projects FAIR (PE0000013) and SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU. This work has also been supported by the NEREO (Neural Reasoning over Open Data) project funded by the Italian Ministry of Education and Research (PRIN) Grant no. 2022AEFHA.

References

[1] Ralph Abboud, Ismail Ilkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. 2020. The surprising power of graph neural networks with random node initialization.

[2] Chris Anderson. 2006. Why the future of business is selling less of more.

[3] Andrea Bacciu, Federico Siciliano, Nicola Tonellotto, and Fabrizio Silvestri. 2023. Integrating Item Relevance in Training Loss for Sequential Recommender Systems. In *Proceedings of the 17th ACM Conference on Recommender Systems* (Singapore, Singapore) *(RecSys '23)*. ACM, New York, NY, USA, 1114–1119. https://doi.org/10.1145/3604915.3610643

[4] Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. 2012. Scalable k-means++.

[5] Erik Bernhardsson. 2018. Annoy: Approximate Nearest Neighbors in C++/Python. https://pypi.org/project/annoy/ Python package version 1.13.0.

[6] Filippo Betello, Federico Siciliano, Pushkar Mishra, and Fabrizio Silvestri. 2024. Investigating the Robustness of Sequential Recommender Systems Against Training Data Perturbations. In *Advances in Information Retrieval*. Springer Nature Switzerland, Cham, 205–220.

[7] Hao Chen, Zefan Wang, Yue Xu, Xiao Huang, and Feiran Huang. 2022. GPatch: Patching Graph Neural Networks for Cold-Start Recommendations.

[8] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. ACM, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[9] Shay Cohen, Gideon Dror, and Eytan Ruppin. 2007. Feature selection via coalitional game theory. *Neural Computation* 19, 7 (2007), 1939–1961.

[10] Dimitrios Danopoulos, Christoforos Kachris, and Dimitrios Soudris. 2019. Approximate Similarity Search with FAISS Framework Using FPGAs on the Cloud. In *Embedded Computer Systems: Architectures, Modeling, and Simulation*, Dionisios N. Pnevmatikatos, Maxime Pelcat, and Matthias Jung (Eds.). Springer International Publishing, Cham, 373–386.

[11] Daniel Fryer, Inga Strümke, and Hien Nguyen. 2021. Shapley values for feature selection: The good, the bad, and the axioms. *Ieee Access* 9 (2021), 144352–144360.

[12] Alireza Gharahighehi and Celine Vens. 2023. Diversification in session-based news recommender systems. *Personal and Ubiquitous Computing* 27, 1 (2023), 5–15.

[13] Ritam Guha, Ali Hussain Khan, Pawan Kumar Singh, Ram Sarkar, and Debotosh Bhattacharjee. 2021. CGA: A new feature selection model for visual human action recognition. *Neural Computing and Applications* 33 (2021), 5267–5286.

[14] Anshul Gupta and Pravin Shrinath. 2023. Link Prediction based on bipartite graph for recommendation system using optimized SVD++. *Procedia Computer Science* 218 (2023), 1353–1365.

[15] Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *Journal of machine learning research* 3, Mar (2003), 1157–1182.

[16] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc., New York, NY, USA. https://proceedings.neurips.cc/paper_files/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf

[17] Bowen Hao, Jing Zhang, Hongzhi Yin, Cuiping Li, and Hong Chen. 2021. Pre-Training Graph Neural Networks for Cold-Start Users and Items Representation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining* (Virtual Event, Israel) *(WSDM '21)*. ACM, New York, NY, USA, 265–273. https://doi.org/10.1145/3437963.3441738

[18] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong 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* (Virtual Event, China) *(SIGIR '20)*. ACM, New York, NY, USA, 639–648. https://doi.org/10.1145/3397271.3401063

[19] Yassine Himeur, Abdullah Alsalemi, Ayman Al-Kababji, Faycal Bensaali, Abbes Amira, Christos Sardianos, George Dimitrakopoulos, and Iraklis Varlamis. 2021. A survey of recommender systems for energy efficiency in buildings: Principles, challenges and prospects. *Information Fusion* 72 (2021), 1–21.

[20] James Huang, Stephanie Rogers, and Eunkwang Joo. 2014. Improving restaurants by extracting subtopics from yelp reviews. *iConference 2014 (Social Media Expo)* 1 (2014).

[21] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks.

[22] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. 2020. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems* 33 (2020), 4465–4478.

[23] David Liben-Nowell and Jon Kleinberg. 2003. The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management* (New Orleans, LA, USA) *(CIKM '03)*. ACM, New York, NY, USA, 556–559. https://doi.org/10.1145/956863.956972

[24] Ying Chieh Liu and Min Qi Huang. 2021. Examining the Matthew Effect on YouTube Recommendation System. In *2021 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*. IEEE, New York, NY, USA, 146–148. https://doi.org/10.1109/TAAI54685.2021.00035

[25] Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc., New York, NY, USA. https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

[26] Lucie Charlotte Magister, Pietro Barbiero, Dmitry Kazhdan, Federico Siciliano, Gabriele Ciravegna, Fabrizio Silvestri, Mateja Jamnik, and Pietro Liò. 2023. Concept Distillation in Graph Neural Networks. In *Explainable Artificial Intelligence*, Luca Longo (Ed.). Springer Nature Switzerland, Cham, 233–255.

[27] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* 42, 4 (2018), 824–836.

[28] Tien T. Nguyen, Pik-Mai Hui, F. Maxwell Harper, Loren Terveen, and Joseph A. Konstan. 2014. Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd International Conference on World Wide Web* (Seoul, Korea) *(WWW '14)*. ACM, New York, NY, USA, 677–686. https://doi.org/10.1145/2566486.2568012

[29] Roma Patel, Marta Garnelo, Ian Gemp, Chris Dyer, and Yoram Bachrach. 2021. Game-theoretic Vocabulary Selection via the Shapley Value and Banzhaf Index. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (Eds.). Association for Computational Linguistics, Online, 2789–2798. https://doi.org/10.18653/v1/2021.naacl-main.223

[30] Antonio Purificato, Giulia Cassarà, Federico Siciliano, Pietro Liò, and Fabrizio Silvestri. 2023. Sheaf4Rec: Sheaf Neural Networks for Graph-based Recommender Systems.

[31] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. ACM, New York, NY, USA, 1135–1144. https://doi.org/10.1145/2939672.2939778

[32] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20 (1987), 53–65.

[33] Benedek Rozemberczki, Lauren Watson, Péter Bayer, Hao-Tsung Yang, Olivér Kiss, Sebastian Nilsson, and Rik Sarkar. 2022. The shapley value in machine learning.

[34] Federico Siciliano, Maria Sofia Bucarelli, Gabriele Tolomei, and Fabrizio Silvestri. 2022. NEWRON: A New Generalization of the Artificial Neuron to Enhance the Interpretability of Neural Networks. In *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, New York, NY, USA, 01–17. https://doi.org/10.1109/IJCNN55064.2022.9892367

[35] Erik Štrumbelj and Igor Kononenko. 2014. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems* 41 (2014), 647–665.

[36] Xiaoyuan Su and Taghi M. Khoshgoftaar. 2009. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence* 2009, 1 (2009), 421425. https://doi.org/10.1155/2009/421425 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1155/2009/421425

[37] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks.

[38] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community Preserving Network Embedding. *Proceedings of the AAAI Conference on Artificial Intelligence* 31, 1 (Feb. 2017). https://doi.org/10.1609/aaai.v31i1.10488

[39] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19)*. ACM, New York, NY, USA, 165–174. https://doi.org/10.1145/3331184.3331267

[40] Muhan Zhang and Yixin Chen. 2018. Link Prediction Based on Graph Neural Networks. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc., New York, NY, USA. https://proceedings.neurips.cc/paper_files/paper/2018/file/53f0d7c537d99b3824f0f99d62ea2428-Paper.pdf

[41] Chang Zhou, Jianxin Ma, Jianwei Zhang, Jingren Zhou, and Hongxia Yang. 2021. Contrastive Learning for Debiased Candidate Generation in Large-Scale Recommender Systems. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (Virtual Event, Singapore) *(KDD '21)*. ACM, New York, NY, USA, 3985–3995. https://doi.org/10.1145/3447548.3467102