Promoting Tail Item Recommendations in E-Commerce

Tamar Didi
Ben-Gurion University of the Negev
Beer Sheva, Israel
tamarin@post.bgu.ac.il

Ido Guy *
Ben-Gurion University of the Negev
Beer Sheva, Israel
idoguy@acm.org

Amit Livne
Ben-Gurion University of the Negev;
Beer Sheva, Israel
livneam@post.bgu.ac.il

Arnon Dagan*
Apiiro
Tel Aviv, Israel
arnon@apiiro.com

Lior Rokach
Ben-Gurion University of the Negev
Beer Sheva, Israel
liorrk@post.bgu.ac.il

Bracha Shapira
Ben-Gurion University of the Negev
Beer Sheva, Israel
bshapira@bgu.ac.il

ABSTRACT

The research area of recommender systems (RS) in e-commerce has become extremely popular in recent years. However, traditional RSs tend to recommend popular items, while niche (long-tail) items are often neglected, which is known as the long-tail problem. However, recent studies found that tail items are one of the key success factors in the e-commerce world. The availability of such items encompasses relatively high marginal profits and boosts the sales of popular short-head items. We suggest promoting long-tail items by leveraging the short-head items' advantages and exposing the user to a tail item that may have not been considered otherwise. We use a classification model and statistical tools to generate personalized recommendations of a long-tail item considering a short-head item that has already been clicked. The uniqueness of our method lies in the combination of tail and head items to uplift the exposure of the latter and in using an applicable solution to deal with the extreme volume of tail items. We demonstrate the effectiveness of our method on real-world data from eBay and provide an analysis of the long-tail phenomenon and consumption behavior.

CCS CONCEPTS

• **Information systems** → **Electronic commerce**; **Recommender systems**.

KEYWORDS

machine learning, classification, tree-based methods, long tail items, e-commerce, Bayes' theorem

ACM Reference Format:
Tamar Didi, Ido Guy, Amit Livne, Arnon Dagan, Lior Rokach, and Bracha Shapira. 2023. Promoting Tail Item Recommendations in E-Commerce. In *UMAP '23: Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization (UMAP '23), June 26–29, 2023, Limassol, Cyprus.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3565472.3592968

*Research was conducted while working at eBay.

UMAP '23, June 26–29, 2023, Limassol, Cyprus

ACM ISBN 978-1-4503-9932-6/23/06.
https://doi.org/10.1145/3565472.3592968

1 INTRODUCTION

Recommender systems (RS) are a class of information filtering applications whose main goal is to provide predictions to users based on their preferences, [4, 11, 27]. However, traditional RSs tend to focus on recommended items, rather than uncovering niche (long-tail) items and showcasing them to potential users, which is well known as the long tail problem [3]. Long-tail is the phenomenon in which niche items gain a significant share of demand among all items, while the "short-head" refers to a small number of popular items that account for the highest relative share of the demand [15]. Traditionally, most retailers manage sales by using the Pareto principle, which states that roughly 80% of the outcome results from 20% of the causes. However, this principle is weak in e-commerce [6]. The success of "infinite-inventory" retailers, such as Amazon, eBay, and Netflix has been largely attributed to the long-tail phenomenon. Although the majority of the tail item inventory is in low demand, these hard-to-find items, unavailable at limited-inventory competitors, embrace relatively large marginal profit compared to the short-head items [6]. Furthermore, niche items can boost head sales by providing users with a "one-stop shopping" convenience, which can entice customers to purchase both short-head items and long-tail items in one step, increasing the total sales [40]. Most RSs are evaluated based on accuracy-oriented metrics, which is not necessarily reflective of user needs, as RSs may provide accurate but likely obvious suggestions [10, 21]. In addition, accuracy-driven RSs are prone to popularity bias, which is the tendency of popular items to be recommended more often [1]. Long-tail recommendations can enhance the user's experience with the recommender system, as recommending popular (short-head) items only may bore the users [2, 24].

In this work, We introduce a novel method to handle the long-tail challenge by combining tail items with head items. Our approach personally recommends a tail item most likely to be clicked together with the head item that has already been clicked, while considering context information, such as historical user behavior and time. Our method is expected to significantly improve the visibility of the tail items in the recommendation systems [40]. The head and tail pair could also be offered for sale as a package to enhance the exposure of the latter and expand customers' buying scope.

Recommending a personalized long-tail item considering a clicked short-head item is a challenging task, due to the high volume of the long-tail items and their niche characteristics. The latter makes the multi-class task sparse as well. Moreover, a straightforward

solution model that utilizes a feature set to predict a long-tail item often requires adaptations and powerful computing resources, so it may not be a scalable solution for real-world scenarios. To tackle these challenges, we suggest a two-stage approach: First, a tree-based model that predicts the short-head category to combine with a given tail item. Following, Bayes' theorem is used on the predicted categories from the first phase to estimate the personal probability for a user to click a tail item together with the short-head category already clicked. We use these probabilities to measure the improvement in the exposure of the tail items, in order to reduce popularity bias. Similar to [22], we utilize head categories instead of head items due to the dynamics of the large-scale e-commerce websites as well as the sparsity and fine-grained granularity of the head items' data. Through our two-step method, we address the aforementioned challenges by leveraging the denser and much smaller number of short-head categories, which serves to decrease the learned parameters and the training resources required. We demonstrate the effectiveness of our method on a real-world dataset provided by eBay, which includes co-clicked transactions of tail and head items. Figure 1 presents an example of a user's clicks on tail and head items in one day.

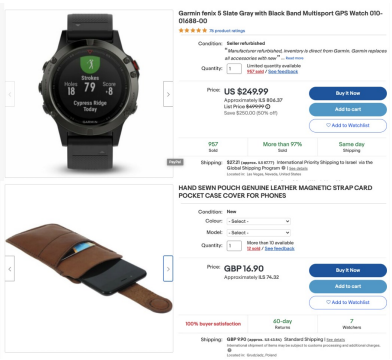


Figure 1: Example of co-clicked short-head (top) and long-tail (bottom) items from eBay.com

The main contribution of this work is twofold:

(1) We introduce a novel approach leveraging short-head items for promoting long-tail items, which is a unique strategy for addressing the well-known long-tail challenge in RSs. The strategy is shown to be effective on a real-world dataset, demonstrating success in exposure lift of long-tail items in a recommendation list.
(2) We analyze the long-tail phenomenon on a real-world dataset from eBay. Our analysis provides insights about long-tail consumption behavior and users' preferences over the categories.

2 RELATED WORK

2.1 Long Tail Recommendation

Traditional RSs are accuracy-driven and tend to recommend the most popular items that have many ratings or clicks, since it is difficult to predict the relevance of long-tail items with a small number of transactions. Consequently, these items are less likely to appear in the recommendation lists, do not gain more predictions, and may be discarded [27]. This problem is known as the long-tail problem. Recent studies on long-tail recommendation methods mainly focus on graph-based methods, multi-objective-function-optimization-based methods, and clustering-based methods.

In the realm of graph-based methods, Yin et al. [40] represented user-item information with an undirected edge-weighted graph and applied the hitting time algorithm for long-tail item recommendation. Shi [35] aimed to find a trade-off among multiple criteria of measurements, such as long-tail, accuracy, similarity, and diversity. Yet, while we utilize time and item properties, those works do not consider any context information when recommending.

Other studies adopted multi-objective optimization to address the long-tail challenge. Wang et al. [39] introduced a multi-objective system with a new evolutionary algorithm for long-tail recommendations. Two contradictory objective functions were designed to recommend unpopular items while minimizing accuracy loss. Then, a multi-objective evolutionary algorithm was proposed to find a set of trade-off solutions by optimizing the two objectives simultaneously. Nevertheless, evolutionary algorithms usually have high runtime, so they are not optimized for the rapid rhythm of e-commerce. Hamedani and Kaedi [27] used multi-objective simulated annealing with three objectives: increasing accuracy, personalizing diversity, and reducing the popularity of recommendation lists. Using simulated annealing, however, is not guaranteed to find an optimal solution. In the long-tail item recommendation method proposed by Pang et al. [30], a non-dominated sorting genetic algorithm (NSGA-II) was used. To achieve the recommendation of long-tail items, both accuracy and coverage were taken as the objective functions simultaneously. All of these methods, however, do not distinguish between the user's long-term preferences and short-term preferences, while we model the user behavior and preferences over time.

Finally, clustering methods are used to alleviate the issue of data sparsity and improve the accuracy of recommendations. Park [31] proposed an adaptive clustering method that groups items according to their popularity, so that the recommendations for tail items are based on the ratings in more intensively clustered groups, and for the head items are based on the ratings of individual items or groups, clustered to a lesser extent. Unfortunately, the challenge of this method is finding an appropriate clustering standard. Huang and Fu [17] suggested a long-tail item recommendation method that combines topic model-based methods with clustering techniques. The proposed model incorporates the hidden experience information discovered through online customer reviews with numeric information from product/service providers. However, those methods are based on ratings and do not take into account the item's attributes or user preferences.

2.2 Head and Tail Recommendations

Similar to our work, some researchers rely on the head and tail distribution to improve recommendations. Park et al. [32] split the whole item set into head and tail and clustered only the tail items. For the tail items, recommendations were made based on ratings within these clusters, and for the head items, they were based on individual ratings. Li et al. [23] simultaneously handled both cold-start and long-tail recommendations in multiple objectives. For the cold-start problem, they used side information; and for the long-tail

recommendation, they decomposed the overall set of interesting items into two parts: short-head items and long-tail items. The two parts were independently revealed in the training stage and transferred into the final recommendation for new users. While these works split the item set into head and tail items as we do, these two sets are independent during the training process. Therefore, relations between them are neglected.

In [41], the authors presented a model that transferred knowledge from head items to tail items, leveraging the rich information of head items and the semantic connections between head and tail items. The framework proposed by Liu and Zheng [24] was designed based on neural networks and attention mechanism to identify long-tail items from session data. The proposed method aimed to determine user preference between short-head and long-tail items, based on click frequency, while we propose a method to personally recommend a long-tail item considering a short-head item that has already been clicked.

In summary, our work differs from the above-mentioned long-tail recommender systems in various aspects. First, several works [27, 31, 32, 35, 39–41] retrieve the long-tail items through analysis of rating, whereas we focus on e-commerce recommendations and use the click distributions. Second, some methods [17, 27, 30–32, 35, 39, 40] do not take into account the user's preferences or any context information. Finally, most of the presented methods [23, 27, 31, 32, 35, 39, 41] aim to optimize the recommendation list and suggest a long-tail item alongside other items. In our novel approach to boosting long-tail recommendations, we combine long-tail items most likely to be clicked with short-head items already clicked. This method suggests a unique solution to the high volume of long-tail items by leveraging the popularity of head items.

3 DATASET

Our dataset was provided by eBay, an e-commerce platform that offers people and businesses the option to buy and sell various goods and services worldwide. The dataset consists of "click" transactions that occurred during February 2020 within the United States version of the eBay site. Each record in our dataset represents a co-click of two items clicked by the same user on the same day. We define two sets of items: H and T, corresponding to the short-head and long-tail items, and label each item accordingly. In this work, we aim to boost the long-tail items recommendations by leveraging short-head items, thus we focus on the transactions that include clicks on pairs of H and T items. To determine the partitioning between H and T, we utilized a "views" dataset provided to us by eBay. A *view* refers to the appearance of an item in search results. The views dataset includes transactions that occurred during November 2019. Similarly to [14, 32], we experienced various view values along the x-axis (depicted in Figure 2) to determine the optimal thresholds for H and T. The impact of selecting different thresholds on the results is depicted in subsection 6.2. Our final selected thresholds for short-head and long-tail are 1000 and 10 monthly views, respectively. Table 1 and Table 2 provide a summary of the clicks dataset used in this study and statistics of the transactions respectively.

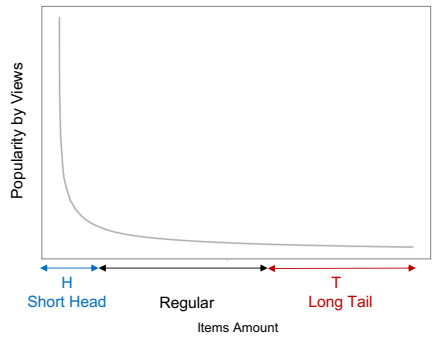


Figure 2: The long-tail of item popularity by the number of views on November 2019 at eBay dataset.

	eBay Dataset
# Records	3,532,938
# Long-tail items	1,044,051
# Users	665,608
Long-tail MCs	Books (5%), Business & Industrial (8%), Cell Phones & Accessories (10%), Clothing & Shoes (27%), Collectibles (4%), Consumer Electronic (3%), Health & Beauty (10%), Home & Garden (19%), Jewelry & Watches (12%)
Head labels	Fashion, Beauty, Home & Garden, Electronics, Others

Table 1: Dataset summary, including the distribution of MCs over long-tail items.

	Long-tail item	User
Mean	3.38	5.38
Median	2	2
Std	9.29	13.77
Min	1	1
Max	3,933	3,884

Table 2: Transaction statistics per long-tail item and user.

3.1 Item Meta-Categories

eBay spans a variety of shopping domains. Each item on eBay is associated with one out of 43 *meta-categories* (MCs) [9] (e.g., Jewelry & Watches, Cameras & Photo, DVDs & Movies). For the long-tail items, we focused on nine major MCs as depicted in Table 1. We selected these MCs based on the richness of the long-tail items, as well as on balance in the number of pairs over the MCs given the threshold values mentioned previously. We used short-head items from all 43 MCs without any filtering. Nevertheless, for the head label, we grouped the MCs into the following five different labels: Fashion, Beauty, Home & Garden, Electronics, and Others. The head labels within the dataset are balanced as the interactions are distributed approximately evenly over the head labels. Each of the five labels accounted for approximately a fifth of the total transactions, specifically ranging from 17% (Electronics) to 24% (Fashion).

In an effort to better understand the MCs on eBay, we explored the monthly click-through rate (CTR) and purchase rate over the

MCs	CTR Ratio	Purchase Rate Ratio
Books	2.71	1.11
Business & Industrial	1.90	0.98
Cell Phones & Accessories	1.89	1.07
Clothing & Shoes	1.81	0.38
Collectibles	2.06	0.48
Consumer Electronics	1.87	0.44
Health & Beauty	1.59	0.60
Home & Garden	2.05	0.86
Jewelry & Watches	2.35	0.92

Table 3: CTR and purchase rate ratios by MC.

categories by short-head and long-tail items. The values are presented as a ratio between the short-head to long-tail items (Table 3). We used the following signals:

- $CTR = \frac{total\ clicks}{total\ views}$
- $purchase\ rate = \frac{total\ purchases}{total\ views}$

We gained two important insights regarding user click behavior patterns. First, user behavior often varies across MCs. Second, there is a difference between short-head and long-tail consumption patterns. Specifically, short-head items receive higher CTR than long-tail items over the different MCs ($ratio > 1$). Nevertheless, the purchase rate for the long-tail items is higher than the short-head for most MCs ($ratio < 1$). This latter finding emphasizes the important role tail items play in driving e-commerce platforms' revenue.

3.2 Features

The dataset includes four major families of features: item's characteristics (e.g., price, category), time-related features (e.g., hour/day the short-head item was clicked), cumulative transactions (e.g., the user's total clicks until the current transaction, days since the last record for the user, user behavior: most clicked category, popular search keys, etc.), and cumulative class transactions (e.g., the total number of times that the user has clicked the head label Fashion).

4 METHOD

In this section, we describe our method to handle product-to-product recommendations, especially for short-head and long-tail items. We aimed to increase the visibility of long-tail items by leveraging head item's popularity. Thus, we recommend a long-tail item that is most likely to be clicked together with the short-head item that has already been clicked.

The proposed method consists of two stages. The first stage is responsible for predicting the head label that should be paired with a long-tail item for a specific user at a specific time. The second stage uses the estimated probabilities obtained from the first stage to recommend the long-tail item that is most likely to be clicked with the given clicked head label by user and time. Figure 3 demonstrates the difference between predicting the long-tail item end-to-end (Figure 3a) vs. our two-stages method (Figure 3b), which utilizes the probability of clicking on a small number of head labels to recommend a personalized long-tail item.

Using a two-stage approach introduces multiple benefits:

- On large e-commerce platforms, inventory long-tail items are huge in numbers and dynamic in nature. Therefore, personally recommending a long-tail item considering a clicked short-head item is a sparse task. A straightforward solution model that utilizes a feature set to predict a long-tail item often requires adaptations and might not be a scalable solution for real-world scenarios. Our two-step method handles the sparsity problem by leveraging the denser and dramatically smaller short-head categories amount, which decreases both the learned parameters and the required resources for training.
- To reduce latency and resource budget, the first stage can be performed offline or pre-computed.
- Predicting the short-head categories at the first step, which are almost static, allows flexibility in adding new items to the marketplace. Newly seen items do not require a new model to be trained since the labels are not changed. Instead, incremental training can be conducted.
- The proposed approach is modular, where each of the two stages can be separately optimized.

The input for the first stage is a set D composed of N records. Each record $(u, t, f, h) \in D$ denotes an interaction event where user u clicked on a long-tail item t together with short-head item associated with head label h, while considering a feature set f regarding this interaction. f may include categorical features (e.g., the item's MC) represented as a one-hot encoding vector and continuous features (e.g., age). The task is building a multi-class classification model to estimate the personalized probability $p(x)$ for user u clicking a specific head label h while considering already clicked long-tail item t and given features f.

$$p(x) = p(h|(u, t, f)) \tag{1}$$

During the second stage, we use the probabilities $p(x)$ from the first stage (Equation 1) to estimate the personalized probability that user u will click on a specific long-tail item t while considering the set of features f given that the user has already clicked on head label (h). Using Bayes' theorem:

$$p(t|(u, f, h)) = \frac{p(t, u, f, h)}{p(u, f, h)} = \frac{p(t, u, f|h) \times p(h)}{p(u, f|h) \times p(h)}$$

$$= \frac{\frac{p(h|t,u,f) \times p(t,u,f)}{p(h)}}{\frac{p(h|u,f) \times p(u,f)}{p(h)}} = \frac{p(h|t, u, f) \times p(t, u, f)}{p(h|u, f) \times p(u, f)}$$

$$= p(x) \times \left(\frac{p(t, u, f)}{p(h|u, f) \times p(u, f)} \right) =$$

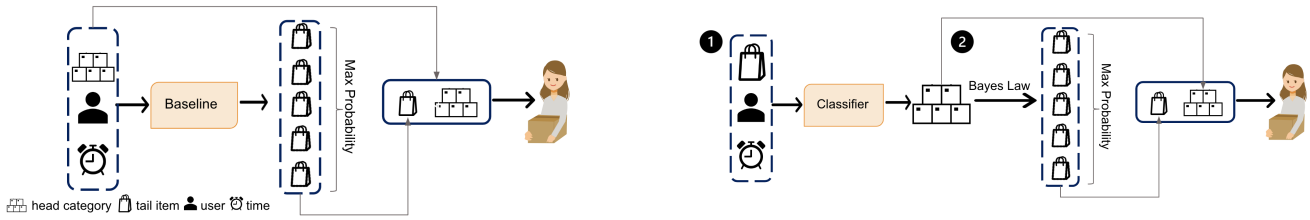
$$p(x) \times \left(\frac{p(t|u, f) \times p(u, f)}{p(h|u, f) \times p(u, f)} \right) = p(x) \times \left(\frac{p(t|u, f)}{p(h|u, f)} \right)$$

It is worth noting that tail item t in the dataset is associated with only a few records (Table 2), in particular when considering features f and user u. Thus, to address the sparsity of the data, we used the Laplace smoothing as follows:

$$p(t|u, f) = \frac{p(t, u, f)}{p(u, f)} \approx \frac{p(t, u, f) + 1}{p(u, f) + |T|}$$

where $|T|$ stands for the number of tail items in our training set. Accordingly, the probability that a tail item t will be recommended with a head category h given a user u and features f is:

$$p(t|(u, f, h)) \approx p(x) \times \left(\frac{\frac{p(t,u,f)+1}{p(u,f)+|T|})}{p(h|u, f)} \right) \tag{2}$$



(a) Baselines- end to end. **(b) Our two steps method**

Figure 3: Our two-stage method generates a personalized long-tail item recommendation. The method utilizes Bayes Law on the probability of recommending already clicked head label to recommend the long-tail item. The baselines are recommending the long-tail item on one step, end-to-end

Finally, the proposed method can be used to generate recommendations that are valuable both to the user and the web store. Whenever a user clicks a head item, we recommend a long-tail item with the maximum probability of $p(t|(u, f, h))$.

5 EXPERIMENTS

In this section, we describe our extensive experiments for evaluating our method's effectiveness for uplifting tail items. Specifically, the following elements (research goals) were investigated:

(1) For Stage I
- (G1): Different classifiers for the task of predicting the head label for long-tail item.
- (G2): Identify the most important features to consider when predicting the head label to pair with a long-tail item.
- (G3): Explore the long-tail consumption patterns by eBay over MCs. Analyze the effect of the item's MC on short-head and long-tail pairs generation.

(2) Finally, for Stage II
- (G4) Demonstrate that our method is effective for uplifting the exposure to the proper long-tail items.

5.1 First Stage

5.1.1 Experimental Settings. First, we aimed at predicting the head label to pair with a tail item given a user and time (G1). To this end, we experimented with popular tree-based classification models and deep learning models as follows:

(1) *Tree-based algorithms:*
- ***XGBoost*** [8], an open source library[1] that implements gradient boosted trees, designed for efficiency.
- ***LightGBM (LGBM)*** [18], an open source library [2] that implements gradient-boosted trees, designed for speed and scalability. It splits the tree leaf-wise with the best fit to reduce more loss than the level-wise algorithms.
- ***CatBoost*** [34], an open source library[3] that provides a gradient boosting model and supports categorical features.
- ***RandomForest*** [5], an ensemble learning method that operates by combining a multitude of decision trees.

(2) *Deep learning algorithms* [4]*:*
- ***DeepFM***[12], integrates the architectures of FM and DNNs by modeling low-order feature interactions as FMs and modeling high-order feature interactions as DNNs.
- ***FiBiNET*** [16], a DNN model that calculates feature interactions using a bilinear function.

5.1.2 Evaluation Metrics.
- ***Accuracy***: the proportion of samples that were correctly classified.
- ***Recall*** [36]: the sum of true positives across all classes divided by the sum of TPs and FNs across all classes.
- ***Precision*** [36]: an average per-class agreement of the data class labels with those of a classifier.

Since the labels in the dataset are uniformly distributed (20%), we used a macro-average to calculate the overall precision and recall, which computes the metric independently for each class and then averages it.

5.1.3 Train-Test Split. Our dataset is chronologically ordered. To avoid data leakage, we split the data based on timestamps. This split simulates the real-world scenario, as real systems train on available data up to a certain timestamp and predict for the following days. We split the dataset as follows: the 30% most recent records are considered as the test set and the rest are used for training (the 10% most recent records used for validation and hyper parameters tuning).

5.1.4 Hyper Parameter tuning. For the tree-based model, we used grid search to tune the following hyper-parameters: "n_estimators" (100,150,200), "max_depth" (from 5 to 15, in two steps jumps), and "learning rate" (0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1). The best results were achieved by 100 estimators with a maximum depth of 10 and a learning rate of 0.1. The objective task is determined to be 'multiclass'. In XGBoost, we also tuned the "min_child_weight" parameter to 5. For the deep models, we used the Adam optimizer [20]. We tried varying values of learning rate from 0.0001 to 0.1 and mini-batch sizes (32, 64, 256, 512, 1,024, 2,048) resulting in an optimal value of 0.001 and 1,024 respectively. For multi-class classification, we changed the last layer to a fully connected layer with Softmax activation over five neurons, equal to the number of head labels in our classification task.

5.1.5 Baseline. ***Statistic-Based Baseline*** As part of our dataset analysis, we recognized that there is a connection between the

[1]XGBoost: https://xgboost.readthedocs.io/en/latest/
[2]https://lightgbm.readthedocs.io/en/latest/
[3]CatBoost: https://catboost.ai/
[4]deepctr: https://github.com/shenweichen/DeepCTR

Long-tail Item MC	Head labels
Books	Other
Business & Industrial	Other
Cell Phones & Accessories	Electronics
Clothing, Shoes & Accessories	Fashion
Collectibles	Other
Consumer Electronics	Electronics
Health & Beauty	Beauty
Home & Garden	Home & Garden
Jewelry & Watches	Beauty

Table 4: Head label mapping by the statistic-based baseline. We select the head label that has been most co-clicked with each tail MC.

long-tail item MC and the head label (61% of the pairs included items from one MC). As a baseline we choose the most frequent head label by long-tail MC for the head label prediction task (G1), to demonstrate that our method captures MCs-heterogeneous pairs as well. Table 4 presents the label per each long-tail item MC.

5.2 Second Stage

5.2.1 Experimental Settings. Based on the short-head item clicked by the user, we set out to recommend the long-tail item with the highest probability to be clicked out of 736, 530 unique long-tail items.

5.2.2 Baselines. The goal of this paper is to recommend a specific tail item in combination with a head item. Therefore, our method is evaluated by scalable baselines from the product-to-product domain and classification in recommendation systems. Studies in the field of long-tail recommendations have multiple objectives: to improve tail item recommendations without negatively affecting overall performance. Specifically, they provide a recommendation list that includes both head and tail items. However, we recommend a personalized long-tail item given a short-head item that has already been clicked. Therefore, we use the following algorithms as baselines:

- **Ranking based on item popularity (RBIP)**, this method, broadly used in previous long-tail studies [7, 28, 29, 33, 37], ranks all the available items based on their popularity from highest to lowest. The popularity measure is based on the item's number of views, calculated in the same way as described in Section 3, for identifying the short-head and long-tail thresholds. Equally-popular items (same number of views) were randomly ordered.
- **Item-NN**, a neighborhood-based method [13, 19, 24]. It recommends a set of items with the highest similarity. The similarity is calculated by dividing the number of pairs in which two items are clicked together by the square root of the product of the number of pairs in which the individual items occur.
- **Deep learning algorithms** For the multi-class classification, we changed the last layer to a fully connected layer with Softmax activation over 736, 530 neurons, which equals the number of tail items: ***DeepFM***[12], ***FiBiNET*** [16].

5.2.3 Evaluation Metrics.

- ***AUC- Lift Chart*** A lift chart presents the added gain of applying the proposed recommendation model over a null model that assumes all items are equal. We create a separate lift chart for each user. The X-axis represents the cumulative proportion of the ranked items out of 736, 530 long-tail items and the Y-axis represents the cumulative proportion of the actual clicks by that user. The rank of the items was determined according to the probability estimated by our model. Similar to AUC, the area under the lift chart is a summary measure that is often used to estimate the usefulness of a model (the null model has an area of 0.5) [38].

To emphasize our contribution to increasing the click likelihood and visibility of the long-tail item (G4), we also follow a test methodology applied in previous works, using "negative sampling" [40, 41]. For each record in the test set, we randomly select 99 additional "negative" items out of 736, 530 unique long-tail items. Negative items are those that are not clicked together with the head label on the same day. Then, we compute predicted scores for the test item i as well as the additional 99 items and form a ranked list of these 100 items. We use the following metrics to evaluate the quality of the list:

- ***Mean Reciprocal Rank (MRR)***: An average of the reciprocal ranks of results for a sample of queries Q.
- ***Mean Rank***: The average of the true tail item ranks.
- ***Median Rank***: The median of the true tail items ranks.
- ***Recall@K***: $Recall@K = \frac{\sum hit@K}{|N|}$. Where hit@K stands for a single test case, as either the value 1 if the test item i appears in the top-K results, otherwise 0. |N| is the number of test records.

6 RESULTS

6.1 First Stage Results

Table 5 presents the performance results of the different classifiers described earlier (G1). The best results are highlighted in bold. It can be seen that the Statistic-Based baseline reaches an accuracy of about 75%. The deep learning classifiers achieve substantial improvement compared to the baseline's performance. The tree-based classifiers yield the highest performance, with XGBoost achieving the highest result with a high 91.51% accuracy, and both precision and recall above 90%. The differences between the various tree-based classifiers' performances are not significant (Table 5).

We conjecture that the tree-based classifiers substantially outperform the deep classifiers due to task compliance and the training data size. Deep learning algorithms usually rely on vast training data and may not perform as well when using a relatively small amount of data [25]. Overall, it is shown that the clicked head label can be predicted at high accuracy, given long-tail item, time, and user (G1).

Table 6 presents the performance of XGBoost over the different labels. As explained earlier, the labels are the head labels that are recommended alongside long-tail item. It can be observed that the Electronics MC achieved the highest precision with 95.90%, while Fashion reached the highest recall with 94.82%. The "Other" MC had the lowest performance. We can assume that this finding lies in the fact that this is an eclectic group of meta-categories.

Classifier	Accuracy	Recall	Precision
Statistic-Based Baseline	75.04%	73.93%	75.05%
XGBoost	**91.51**%	**91.14**%	**91.43**%
LGBM	91.47%	91.11%	91.38%
CatBoost	91.42%	91.03%	91.33%
RandomForest	91.40%	91.01%	91.32%
DeepFM	84.75%	84.36%	86.38%
FiBiNET	84.39%	83.94%	86.67%

Table 5: Performance results of different algorithms for the head label classification task.

	Recall	Precision
Beauty	94.04%	94.24%
Electronics	90.51%	**95.90**%
Fashion	**94.82**%	91.84%
Home & Garden	89.29%	90.18%
Other	87.08%	85.02%

Table 6: XGBoost performance analysis for each head label.

In order to address our second and third research questions (G2, G3), we used two different analyses:

(1) we studied the impact of each feature family by training XGBoost on subsets of features; and (2) we used a common explainable AI tool, Shapley Additive Explanations (SHAP). SHAP is a method to explain individual predictions, based on the game's theoretically optimal Shapley values [26].

First, we conducted a series of experiments to examine the impact of feature families on XGBoost performance. Specifically, we trained XGBoost with different subsets of feature families. In each experiment, we excluded one feature family when training XGBoost.

We can observe (Figure 4) that excluding the "cumulative class transactions" features decreases the model performance substantially, followed by the "Item characteristics" features family (which includes the "tail MC" feature). We also compare the impact of user history against items within the "cumulative class transactions" features. The historical transactions of the user (i.e., "user-Cumulative class transactions") have a greater impact on the results than the history of the tail item. It emphasizes the importance of user behavior on long-tail recommendation.

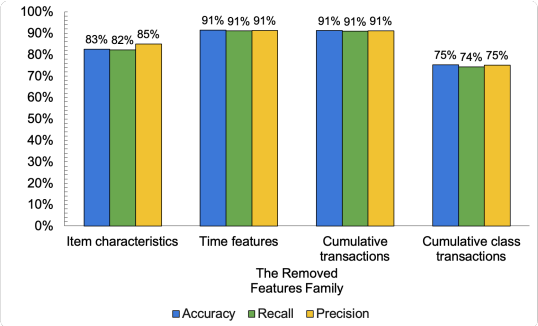


Figure 4: The impact of excluding different feature families on the performance of XGBoost.

Second, we also used SHAP to determine the impact of a feature on each class (Figure 5).

Train Size	Accuracy	Recall	Precision
10%	91.03%	90.80%	90.71%
30%	91.30%	91.10%	90.81%
50%	91.42%	91.11%	91.35%
70%	91.46%	91.13%	91.42%
90%	91.51%	91.14%	91.43%

Table 7: XGBoost results over varied training set sizes.

It can be seen that the user's clicking behavior has a high contribution to predictions (e.g., cumulative class transactions features: user_Beauty_cum, user_Fashion_cum, etc.). The importance lies in whether the user has already clicked a head label along with a specific long-tail item or not. It could be the result of the user's consumption habits in e-commerce or it may be another search attempt for products from a specific MC. Furthermore, our data analysis revealed a connection between the tail MC and the head label. It seems that users often click on items within the same MC.

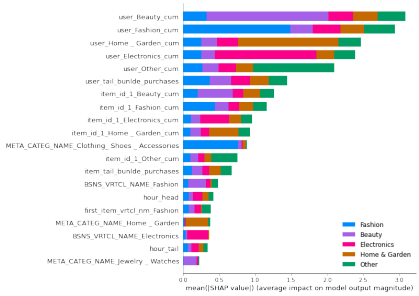
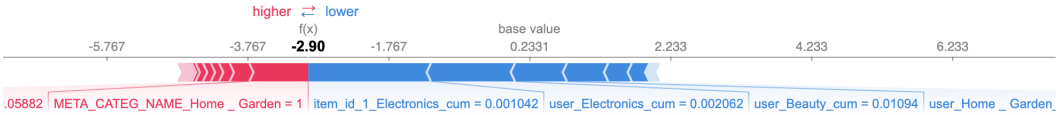


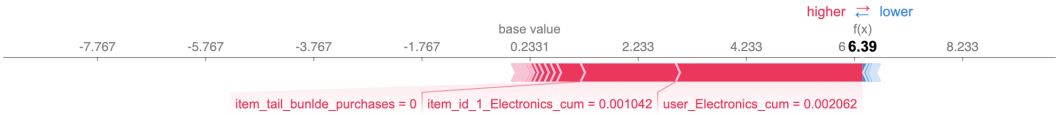
Figure 5: SHAP summary plot - The impact of a feature on the classes is stacked to create the feature importance plot. Features are sorted by the sum of the SHAP value magnitudes across all samples.

Finally, to better understand the contribution of the features when the user clicked on items from different MCs, we use the SHAP Force plot (Figure 6). The plot presents the contribution of each feature to pushing the model output from the base value to the model output. We looked at an observation where a user clicked on a tail item belonging to the Home & Garden MC together with a head item from the Electronics BV. When observing the plot for the Home & Garden class (Figure 6a), it appears that the tail MC pushed the model score higher (red). However, the cumulative features (e.g. user_Electronic_cum, item_id_1_Electronic_cum) reduced (blue) the property value. In Figure 6b, we can see that the Electronics cumulative features increase (depicted in red) the likelihood of Electronics as the true label.

We also set out to examine the impact of the training set size on the model performance. Similarly to [25], we trained XGBoost using only a portion of the available training set, while reserving a chronologically constant test set (30% as described in 5.1.3). We experimented with varying sizes of the training set, ranging from 10% to 90% of the original training set, always considering the most recent portion. 100% represents the original training set. Table 7 presents the performance results. It can be seen that even a relatively small training set leads to high performance, which gives an indication of the robustness of the model in this task.



(a) SHAP force plot for class Home & Garden



(b) SHAP force plot for class Electronic

Figure 6: SHAP force plot for individual prediction from the test set

Tail \Head	1000	3000	5000
2	90.28	90.76	90.03
5	90.52	91.08	90.44
10	91.51	91.35	91.11

Table 8: XGBoost Performance based on varied Head and Tail views thresholds.

In summary, the results indicate the effectiveness of our method. We found that tree-based models yield the best performance on predicting the head label task. Moreover, the cumulative class features have a high contribution to the model prediction, which emphasizes the importance of user preferences on long-tail consumption. We also recognize that head and tail items from the same MC are frequently clicked together. Finally, we found that our model can be trained on a relatively small amount of data and still yield high performances.

6.2 Head and Tail Thresholds Analysis

In this section, we examine the sensitivity of our first-stage method to the selected head and tail thresholds. To this end, we create different short-head and long-tail item sets based on varied threshold values. Then, we trained XGBoost, the best performing algorithm in Section 6.1, on those datasets. Specifically, we experienced different values of thresholds as follows: for short-head items, we considered items with at least 1000, 3000, or 5000 views, and for long-tail items, we considered items with up to 2, 5, or 10 views. We examined all possible combinations of these thresholds. The results are presented in Table 8.

6.3 Second Stage Results

Table 9 provides the results of the second stage experiments described in Section 5.2. Our method yields the highest Lift score 0.897, which indicates the ability to promote the appearance of long-tail items in the recommendation list. Among the baselines, Item-NN achieves the highest lift, while RBIP yields the lowest. As for the "Negative Sampling" experiments, the RBIP algorithm achieves an MRR of 0.057, with the mean and median rank having a similar value of 47. FiBiNET obtains higher results with 0.063 MRR and mean and median rank of ≈ 44. DeepFM performs 0.088 MRR and mean and median rank of ≈ 43. Item-NN shows a significant improvement with 0.122 MRR, ≈ 41 mean rank, and 39 median rank. Our method yields the best performances with 0.185 MRR. The median and mean ranks are 18 and 35 respectively. Recall@K performance is reported by Figure 7. The performance is

	MRR	Mean Rank	Median Rank	Lift
RBIP	0.057	48.42	47	0.763
Item-NN	0.122	40.96	39	0.808
FiBiNET	0.063	44.23	45	0.790
DeepFM	0.088	42.51	43	0.782
Our Method	**0.185**	**35.11**	**18**	**0.897**

Table 9: Stage 2 results of predicting the long-tail item based on the head label already clicked. MRR, mean and median ranks were calculated using "Negative Sampling" with 99 negative items. The Lift was calculated on a test set consisting of 736,530 long-tail items

only shown for $K = [5, 10, 20, 30, 40]$, as a larger value of K can be ignored in a typical top-N recommendation task. We can see that the performances improve as K increases as expected. RBIP, FiBiNET and DeepFM achieve similar performance, while Item-NN yields an improvement. Clearly, our method performs better than all other competitors.

In summary, our method outperforms the baselines in terms of MRR, mean rank, median rank, and recall@K. These results attest to the ability of our approach to enhance the visibility of long-tail items in RS. Our method enables further integration of the proper long-tail items into the recommendation list by exploiting the presence of short-head items in previous interactions between the users and the recommender.

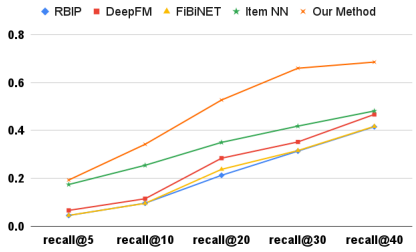


Figure 7: Recall@K over the baselines using "Negative Sampling" with 99 negative items.

7 DISCUSSION

Our results show that the tree-based models have the highest performance in the task of predicting the head label for a tail item (G1). The gaps between those classifiers are insignificant. Therefore, the

method is generic and the model can be replaced by any tree-based model depending on the task constraints.

The feature importance analysis (G2, G3) revealed several key features in the head label classification task. We observed that the cumulative features demonstrate the highest impact on the model performances. It means that the historical behavior of a user and item is a key component in generating personalized head label recommendations considering clicked long-tail item. Moreover, our method's strength lies in its ability to succeed in the classification task with a relatively small amount of data, as observed in Table 7. In the second stage, we compared our method with varied baselines on the tail item classification task (G4). Our method significantly outperformed the other algorithms with substantially increased MRR compared to the best baseline. The uniqueness of our method lies in the combination of tail and head items to enhance the visibility of long-tail items and the use of an applicable solution to deal with such a high volume of long-tail items. Using our method to include long-tail item in the personalized recommendation list alleviates the popularity bias and helps users find their favorite long-tail item.

Finally, we gain two valuable insights about long-tail consumption patterns. First, the MC factor on head label and long-tail item co-occurrence: although head and tail are defined according to the granularity level of the item, we realized that users often click on two items from the same MC. Second, The consumption patterns of head and tail items differ. It seems that short-head items are more clickable, while long-tail items are more purchased.

8 CONCLUSIONS

Recent studies found that long-tail items are one of the keys to success in the e-commerce world. In this paper, we addressed the long-tail recommendation problem alongside encouraging consumers to buy a long-tail item that they may not have considered in isolation. Our experiments demonstrated the effectiveness of our method in predicting the long-tail item that a user will click with a head label. The inclusion of long-tail items in recommendation lists has the additional benefit of reducing popularity bias. This user experience, in which a recommendation of a personalized long-tail item is provided, is currently uncommon on e-commerce platforms. We utilize the time context and historical information to produce more accurate recommendations. Moreover, We gained insights regarding the long-tail phenomenon at eBay and user consumption habits over long-tail items. This work gives rise to additional research directions in the field of long-tail recommendations. First, our experiments involve nine main tail MCs. Future research directions may include the study of long-tail items from other e-commerce domains. Second, our research focused on classification into one label. In the future, we intend to investigate the multi-label task, wherein a tail item may be purchased alongside more than one head category. In addition, conducting online experiments to evaluate the proposed approach in vivo, can also naturally extend the contributions suggested in this work.

REFERENCES

[1] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. 2019. The unfairness of popularity bias in recommendation. *arXiv preprint arXiv:1907.13286* (2019).

[2] Gediminas Adomavicius and YoungOk Kwon. 2011. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on Knowledge and Data Engineering* 24, 5 (2011), 896–911.

[3] Chris Anderson. 2007. *The long tail: How endless choice is creating unlimited demand.* Random House.

[4] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. 2013. Recommender systems survey. *Knowledge-Based Syst.* (2013). https://doi.org/10.1016/j.knosys.2013.03.012

[5] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[6] C. Anderson. 2006. The Long Tail: Why the Future of Business Is Selling Less of More by Chris Anderson. *J. Prod. Innov. Manag.* (2006). https://doi.org/10.1111/j.1540-5885.2007.00250.x

[7] Chih-Ming Chen, Chuan-Ju Wang, Ming-Feng Tsai, and Yi-Hsuan Yang. 2019. Collaborative similarity embedding for recommender systems. In *The World Wide Web Conference*. Association for Computing Machinery, New York, NY, USA, 2637–2643.

[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)*. Association for Computing Machinery, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[9] Arnon Dagan, Ido Guy, and Slava Novgorodov. 2021. An Image is Worth a Thousand Terms? Analysis of Visual E-Commerce Search. (2021).

[10] Marco De Gemmis, Pasquale Lops, Giovanni Semeraro, and Cataldo Musto. 2015. An investigation on the serendipity problem in recommender systems. *Information Processing & Management* 51, 5 (2015), 695–717.

[11] Rana Forsati. 2014. Matrix Factorization with Explicit Trust and Distrust Side Information for Improved Social Recommendation. *ACM Trans. Inf. Syst.* 32, 4 (2014), 17. https://doi.org/10.1145/2641564

[12] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* 24, 3 (2017), 262–290.

[13] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[14] Yu-Chieh Ho, Yi-Ting Chiang, and Jane Yung-Jen Hsu. 2014. Who likes it more? Mining worth-recommending items from long tails by modeling relative preference. In *Proceedings of the 7th ACM international conference on Web search and data mining*. 253–262.

[15] Liang Hu, Longbing Cao, Jian Cao, Zhiping Gu, Guandong Xu, and Jie Wang. 2017. Improving the quality of recommendations for users and items in the tail of distribution. *ACM Transactions on Information Systems (TOIS)* 35, 3 (2017), 1–37.

[16] Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. 2019. FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In *Proceedings of the 13th ACM Conference on Recommender Systems*. Association for Computing Machinery, New York, NY, USA, 169–177.

[17] Xin Huang and Feng Wu. 2019. A novel topic-based framework for recommending long tail products. *Computers & Industrial Engineering* 137 (2019), 106063.

[18] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017), 3146–3154.

[19] Yejin Kim, Kwangseob Kim, Chanyoung Park, and Hwanjo Yu. 2019. Sequential and Diverse Recommendation with Long Tail.. In *IJCAI*, Vol. 19. 2740–2746.

[20] Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations* (12 2014).

[21] Denis Kotkov, Shuaiqiang Wang, and Jari Veijalainen. 2016. A survey of serendipity in recommender systems. *Knowledge-Based Systems* 111 (2016), 180–192.

[22] Eileen Li, Eric Kim, Andrew Zhai, Josh Beal, and Kunlong Gu. 2020. Bootstrapping Complete The Look at Pinterest. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Association for Computing Machinery, New York, NY, USA, 3299–3307.

[23] Jingjing Li, Ke Lu, Zi Huang, and Heng Tao Shen. 2017. Two birds one stone: on both cold-start and long-tail recommendation. In *Proceedings of the 25th ACM international conference on Multimedia*. 898–906.

[24] Siyi Liu and Yujia Zheng. 2020. Long-tail Session-based Recommendation. *arXiv preprint arXiv:2007.12329* (2020).

[25] Amit Livne, Roy Dor, Bracha Shapira, and Lior Rokach. 2021. BNN: Boosting Neural Network Framework Utilizing Limited Amount of Data. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. Association for Computing Machinery, New York, NY, USA, 1150–1159. https://doi.org/10.1145/3459637.3482414

[26] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*. 4768–4777.

[27] Elaheh Malekzadeh Hamedani and Marjan Kaedi. 2019. Recommending the long tail items through personalized diversification. *Knowledge-Based Syst.* 164 (jan 2019), 348–357. https://doi.org/10.1016/j.knosys.2018.11.004

[28] Rasaq Otunba, Raimi A Rufai, and Jessica Lin. 2017. Mpr: Multi-objective pairwise ranking. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. 170–178.

[29] Enrico Palumbo, Giuseppe Rizzo, and Raphaël Troncy. 2017. Entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*. Association for Computing Machinery, New York, NY, USA, 32–36.

[30] Jiaona Pang, Jun Guo, and Wei Zhang. 2019. Using multi-objective optimization to solve the long tail problem in recommender system. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 302–313.

[31] Yoon-Joo Park. 2012. The adaptive clustering method for the long tail problem of recommender systems. *IEEE Transactions on Knowledge and Data Engineering* 25, 8 (2012), 1904–1915.

[32] Yoon-Joo Park and Alexander Tuzhilin. 2008. The long tail of recommender systems and how to leverage it. In *Proceedings of the 2008 ACM conference on Recommender systems*. 11–18.

[33] Rajiv Pasricha and Julian McAuley. 2018. Translation-based factorization machines for sequential recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 63–71.

[34] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. CatBoost: Unbiased Boosting with Categorical Features. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) *(NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 6639–6649.

[35] Lei Shi. 2013. Trading-off among accuracy, similarity, diversity, and long-tail: a graph-based recommendation approach. In *Proceedings of the 7th ACM conference on Recommender systems*. 57–64.

[36] Marina Sokolova and Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. *Information processing & management* 45, 4 (2009), 427–437.

[37] Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Long-Kai Huang, and Chi Xu. 2018. Recurrent knowledge graph embedding for effective recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*. Association for Computing Machinery, New York, NY, USA, 297–305.

[38] Miha Vuk and Tomaz Curk. 2006. ROC curve, lift chart and calibration plot. *Metodoloski zvezki* 3, 1 (2006), 89.

[39] Shanfeng Wang, Maoguo Gong, Haoliang Li, and Junwei Yang. 2016. Multi-objective optimization for long tail recommendation. *Knowledge-Based Systems* 104 (2016), 145–155.

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

[41] 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 *Proceedings of the Web Conference 2021*. 2220–2231.