When Newer is Not Better: Does Deep Learning Really Benefit Recommendation From Implicit Feedback?

Yushun Dong
yd6eb@virginia.edu
University of Virginia

Jundong Li
jundong@virginia.edu
University of Virginia

Tobias Schnabel
toschnab@microsoft.com
Microsoft

ABSTRACT

In recent years, neural models have been repeatedly touted to exhibit state-of-the-art performance in recommendation. Nevertheless, multiple recent studies have revealed that the reported state-of-the-art results of many neural recommendation models cannot be reliably replicated. A primary reason is that existing evaluations are performed under various inconsistent protocols. Correspondingly, these replicability issues make it difficult to understand how much benefit we can actually gain from these neural models. It then becomes clear that a fair and comprehensive performance comparison between traditional and neural models is needed.

Motivated by these issues, we perform a large-scale, systematic study to compare recent neural recommendation models against traditional ones in top-*n* recommendation from implicit data. We propose a set of evaluation strategies for measuring memorization performance, generalization performance, and subgroup-specific performance of recommendation models. We conduct extensive experiments with 13 popular recommendation models (including two neural models and 11 traditional ones as baselines) on nine commonly used datasets. Our experiments demonstrate that even with extensive hyper-parameter searches, neural models do not dominate traditional models in all aspects, e.g., they fare worse in terms of average HitRate. We further find that there are areas where neural models seem to outperform non-neural models, for example, in recommendation diversity and robustness between different subgroups of users and items. Our work illuminates the relative advantages and disadvantages of neural models in recommendation and is therefore an important step towards building better recommender systems.

CCS CONCEPTS

• **General and reference** → **Evaluation**; • **Information systems** → **Recommender systems**.

KEYWORDS

Recommender Systems, Deep Learning, Evaluation

ACM Reference Format:
Yushun Dong, Jundong Li, and Tobias Schnabel. 2023. When Newer is Not Better: Does Deep Learning Really Benefit Recommendation From Implicit Feedback?. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3539618.3591785

SIGIR '23, July 23–27, 2023, Taipei, Taiwan

ACM ISBN 978-1-4503-9408-6/23/07.
https://doi.org/10.1145/3539618.3591785

1 INTRODUCTION

Recent years have witnessed a surge of research interest in deep learning-based approaches for recommendation, including point-of-interest (POI) recommendation [39, 81, 91], e-commerce recommendation [15, 77, 86], and news recommendation [23, 85, 96], to name a few. Deep learning-based approaches parameterize user preferences through a series of neural network layers and then optimize ranking performance over an inventory of items [41, 42, 60, 94]. With their large modeling capacity, neural models can, in theory, capture more complex and non-linear relationships from implicit user feedback. This, in turn, should increase the chances of finding a highly accurate recommendation model.

Despite the undisputed success of neural models in many other areas, several studies have pointed out that the reported performance improvements of many neural recommendation models are difficult to reproduce [32, 33, 62]. For example, several recent neural recommendation models are found to be outperformed by very simple recommendation models, such as linear models [33, 61] and those based on nearest neighbors [61]. A primary reason for such replicability issues is that the reported performance advantages are concluded under many different experimental settings [19, 74, 94, 95], such as dataset splitting ratios, evaluation protocols, metrics, recommendation tasks, etc. This leaves us the important question of how much and what kind of benefits neural models can add to recommendation, which is a critical question for practitioners and system designers [22]. To answer this question, we need to have a more accurate understanding of when (i.e., under what scenarios) and how neural recommendation models and non-neural ones perform differently.

In this paper, we focus on exploring the performance differences between neural recommendation models and non-neural models in the context of top-*n* item recommendation as one of the most common recommendation tasks [25]. While there has been prior work on benchmarking top-*n* recommender systems [2, 20, 76], many of these efforts either did not include neural models or still left important questions unanswered. For example, recent work analyzing both neural and non-neural models [82, 83] still lacks a thorough analysis of the performance difference between these two model classes and thus does not answer what benefits neural models may have over traditional ones. Another line of work only investigates the performance differences between neural and non-neural recommendation models in limited aspects or settings [33, 62] and ignores other aspects where neural and non-neural models may differ. For instance, [33] only focuses on the ranking accuracy over

the test set. Ludwig et al. [62] compare neural and non-neural models under session-based recommendation tasks, ignoring the widely studied top-n recommendation scenario. More importantly, these benchmarks only focus on the *weak generalization* setting [63, 64], i.e., all users and items are known during training. This is clearly in contrast to real-world applications in which the recommender system needs to accommodate newly joined users after deployment [50]. To move towards a more realistic scenario, we analyze performance in a strong generalization setting in this paper where new users are part of the test set.

To gain a deeper understanding of what benefits neural recommendation models can provide, this paper conducts a large-scale and systematic study of neural and non-neural models and explores a wide range of model properties beyond accuracy. Specifically, we propose to empirically evaluate both the memorization and generalization abilities of recommendation models. By memorization, we mean the ability of a model to memorize the user data it has seen during training, while generalization refers to the ability to make accurate predictions for new users during the test phase. As mentioned above, we adopt the setting of *strong generalization* [63, 64] for generalization evaluation, where users are partitioned into non-overlapping sets for training, validation, and test, respectively. Note that there is usually a trade-off between memorization and generalization, which is also referred to as the bias-variance trade-off in machine learning [3, 87]. Robust recommendation performance usually requires a careful balance between them [16]. We therefore join these two perspectives, and formally state our first research question as (**RQ1**) *how do neural models differ from non-neural models in terms of both memorization and generalization?* Moreover, most datasets are inhomogeneous and contain diverse subgroups of items and users with differing characteristics (e.g., item popularity and user preferences). Under such a context, a recommendation model that performs well on one subgroup may not perform well on others [24, 26, 28]. Hence an important dimension for evaluating the differences between neural and non-neural models is their recommendation performance over specific subgroups of users or items. This understanding helps analyze the potential limitations of recommendation models, which is an important step in coming up with possible improvements in the future. We state our second research question as (**RQ2**) *how do neural models differ from non-neural models in terms of subgroup-specific performance?*

To answer the research questions above, we propose an array of practical and comprehensive evaluation strategies. We survey 13 popular recommendation models (including two neural models and 11 traditional models as baselines) and conduct extensive experiments over nine commonly used real-world datasets. Experimental results indicate that even if hyper-parameters are sufficiently optimized, neural models do not outperform traditional models in terms of HitRate in most cases. However, we find that neural models can have certain advantages over traditional non-neural ones. For example, we find that neural models achieve better performance in terms of MeanRanks, recommendation diversity, map out semantic relationships between items more accurately, and show improved robustness between subgroups of instances (i.e., users and items).

To summarize, our contributions are three-fold: (1) **Experimental Design.** We propose a set of comprehensive evaluation strategies. To the best of our knowledge, our work serves as the first step towards evaluating top-n recommendation performance differences between neural and non-neural models on both memorization and generalization. (2) **Performance Comparison.** We chose 13 popular recommendation models and nine commonly used datasets for our empirical investigation. We conduct a large-scale study assessing all dimensions defined above between neural recommendation models and non-neural ones. (3) **Comprehensive Analysis.** We present a comprehensive analysis of neural and non-neural recommendation models based on the experimental results. In addition, we also point out concrete directions for future research.

2 EVALUATION STRATEGIES

Through our experiments, we aim to answer the following two research questions in this paper:

RQ1: *How do neural models differ from non-neural models in terms of performance on memorization and generalization tasks?*

RQ2: *How do neural models differ from non-neural models in terms of subgroup-specific performance?*

Below we present notations and our strategies to evaluate memorization, generalization, and subgroup-specific performance.

2.1 Notation

We use calligraphic letters (e.g., $\mathcal{A}$), bold lower-case letters (e.g., $\mathbf{a}$), and normal lower-case letters (e.g., a) to denote sets, vectors, and scalars, respectively. In recommendation from implicit user feedback, the historical interactions between a specific user and item are binary, i.e., a user either likes an item or the relationship between them is unknown and can thus be expressed as a set. We denote the set of users and items in a recommender system as $\mathcal{U} = \{U_1, ..., U_{|\mathcal{U}|}\}$ and $\mathcal{I} = \{I_1, ..., I_{|\mathcal{I}|}\}$, respectively.

In the memorization setting, we treat the whole set of users $\mathcal{U}$ as users visible during training (i.e., the training users), which is denoted as $\mathcal{U}_{\text{trn}}$. In the strong generalization setting, we split the users into three non-overlapping sets, i.e., the user set for training $\mathcal{U}_{\text{trn}}$ ($\mathcal{U}_{\text{trn}} \subset \mathcal{U}$), the user set for validation $\mathcal{U}_{\text{val}}$ ($\mathcal{U}_{\text{val}} \subset \mathcal{U}$), and the user set for test $\mathcal{U}_{\text{tst}}$ ($\mathcal{U}_{\text{tst}} \subset \mathcal{U}$, and $\mathcal{U}_{\text{trn}} \cup \mathcal{U}_{\text{val}} \cup \mathcal{U}_{\text{tst}} = \mathcal{U}$).

2.2 Memorization Evaluation

Memorization loosely describes a model's ability to recall items seen during training [16, 31, 38]. Following this notion, we measure memorization in recommendation as a model's performance in recovering the implicit user feedback that was seen during training. We now introduce two different tasks that let us probe a specific recommendation model for its memorization ability. Both tasks measure how well a model can recover an item that was seen during training but approach this goal in different ways.

Leave-One-Out Memorization Task. In this task, we adapt the widely adopted leave-one-out protocol [46, 47, 58, 65] to evaluate memorization. Different from the original protocol, we do not hold out a test item, but pick an item that was seen during training. Specifically, we randomly select one implicit feedback entry for each user in $\mathcal{U}_{\text{trn}}$. We then measure how close to the top a model ranks this entry when recommending items to users in $\mathcal{U}_{\text{trn}}$.

Reranking Memorization Task. The setting of this task is similar to the one above, but here we focus on the performance of recovering *all* training items for a user. More specifically, for

each user in $\mathcal{U}_{\text{trn}}$, we measure whether their training items will be ranked at the top when recommending items to this user.

To summarize, both tasks focus on performing evaluations over the set of training users $\mathcal{U}_{\text{trn}}$. The first strategy measures the ability of a recommendation model to predict single missing (hold-out) items, while the second strategy focuses on how well all training items can be recalled after fitting the model.

2.3 Generalization Evaluation

Generalization is loosely defined as applying the learned underlying data patterns to predict on unseen data [9, 14, 16, 53, 56, 92]. Correspondingly, when we refer to the generalization performance of a recommendation model, we mean the performance of making accurate predictions on user-item combinations that were not seen during training. It is worth noting that the generalization ability of a recommendation model has important real-world consequences as such models often have to make predictions for new users [50]. We then discuss strategies to evaluate generalization below.

Strong Generalization Task. We argue that strong generalization reflects the performance of recommendation models better in practice (compared with weak generalization). Specifically, evaluating the capability of strong generalization requires to test models on new users (i.e., those invisible ones during training) and existing items [63]. Following the partitioning introduced in Section 2.1, we use users in $\mathcal{U}_{\text{trn}}$ with one hold-out item per user for training. After that, we use users in $\mathcal{U}_{\text{tst}}$ (with one hold-out item per user) for recommendation performance evaluation.

Semantic Coherence under Strong Generalization. To gain a deeper understanding of the generalization performance, it is also critical to zoom in and examine how well those item-item semantics are captured [90]. Here, the intuition of our proposed strategy is that: given a user with only one interaction with a certain item, a model is considered to capture item-item semantics well if other items that are semantically similar (to this interacted item) can be identified and recommended. Specifically, we propose to generate a set of dummy users $\tilde{\mathcal{U}}$. Each of these dummy users only interacts with one unique item out of $\mathcal{I}$ (thus $|\tilde{\mathcal{U}}| = |\mathcal{I}|$). For each dummy user $\tilde{U}_i \in \tilde{\mathcal{U}}$, we regard the generated user profile (i.e., the generated one user-item interaction) as the known historical implicit feedback, and employ the recommendation model to generate preference scores over all other items. In addition, we collect the semantics of all items (such as manual semantic tags) as side information. For the item interacted with $\tilde{U}_i$, we also compute its similarity scores (e.g., based on cosine similarity) between the semantics of itself and that of all other items. Finally, we compute the Pearson correlation between the predicted preferences and the semantic similarity for $\tilde{U}_i$. We propose to consider the average Pearson correlation value over $\tilde{\mathcal{U}}$ as a general indicator of how well a recommendation model exploits the item-item semantics.

2.4 Subgroup-Specific Performance Evaluation

A recommendation model may bear different performance over different subgroups of instances (e.g., users and items) for a specific dataset, where such subgroups are partitioned out of the original dataset w.r.t. certain instance-level characteristics. For example, when users are divided into active and inactive groups out of an e-commerce dataset, most recommendation models may deliver recommendations with significantly higher quality to the group of active users [59, 72]. Correspondingly, we define subgroup-specific performance as "*the performance over subgroups of instances when the original dataset is partitioned following certain instance-level characteristics*". In fact, one of the most widely studied partitionings is the warm-start subgroup versus cold-start subgroup [1, 11, 43, 73, 89]. Motivated by this, we propose to partition warm-start and cold-start subgroups from the perspectives of both users and items. In this section, we discuss our strategies to evaluate the subgroup-specific performance below.

Partitioning Users: Active vs. Inactive. We propose to partition users into active user and inactive user subgroups according to their total number of interactions. We aim to explore the recommendation quality difference (across different models) over the two user subgroups. Specifically, we select the top $\frac{1}{3}$ active and top $\frac{1}{r}$ inactive users in $\mathcal{U}_{\text{tst}}$ to construct an active user set $\mathcal{U}_{\text{act}}$ and an inactive user set $\mathcal{U}_{\text{ina}}$, respectively. Here we propose to determine the value of r by ensuring the total number of positive interactions in the two sets of users are generally the same, which ensures a fair comparison. Given a specific recommendation model, we evaluate its recommendation performance on $\mathcal{U}_{\text{act}}$ and $\mathcal{U}_{\text{ina}}$ separately.

Partitioning Users: Similar vs. Dissimilar. To evaluate the generalization capability at a finer granularity, we propose to divide $\mathcal{U}_{\text{tst}}$ into two groups based on the similarity of their implicit feedback with the users in $\mathcal{U}_{\text{trn}}$. Specifically, we first compute the similarity score between every test user (i.e., users in $\mathcal{U}_{\text{tst}}$) and training user (i.e., users in $\mathcal{U}_{\text{trn}}$). Then for each test user, we truncate its top-l similarity scores with those training users and compute the sum of these truncated similarity scores. Here l is a parameter that controls the computation of similarity truncation length. We consider top $\frac{1}{3}$ test users with the largest values of the top-l similarity score sum as similar users. In a similar vein, we consider top $\frac{1}{3}$ users with the smallest top-l similarity scores sums as dissimilar users. Given a specific recommendation model, we evaluate its performance over the two sets of test users separately.

Partitioning Items: Head vs. Tail. We finally explore how neural and non-neural recommendation models differ in recommendation quality for both head items (popular items with a large total number of interactions with test users) and tail items (items with fewer interactions). Specifically, we first split the item set $\mathcal{I}$ into a head item set $\mathcal{I}_{\text{head}}$ and a tail item set $\mathcal{I}_{\text{tail}}$ according to the total number of users in $\mathcal{U}_{\text{tst}}$ who have interacted with them. The total interactions occupied by items from the two sets should generally be the same in order to ensure a fair comparison. To perform evaluation, we propose to divide the test users into two sets determined by which set the items associated with their hold-out interactions belong to. Given a specific recommendation model, we evaluate its performance over the two sets of test users separately.

3 BENCHMARKED MODELS

As previously mentioned, the generalization setting that this paper focuses on is strong generalization (as introduced in Section 2.3). As a consequence, we only include the recommendation models that naturally support the strong generalization setting in our performance comparison. Overall, we compare two neural models with

Table 1: The statistics for the adopted datasets. Note that the presented statistics are the corresponding ones after pre-processing with binarizing and 5-core filtering. "Orig. Sparsity" indicates the level of sparsity before pre-processing.

Dataset	#Users	#Items	Orig. Sparsity	5-Core Sparsity	#Ratings	#Ratings Per Item	#Ratings Per User
ml100k	938	1,008	6.30×10^{-2}	5.75×10^{-2}	54.4 K	54.0	58.0
lastfm	1,859	2,823	2.78×10^{-3}	1.36×10^{-2}	71.4 K	25.3	38.4
kuai	1,411	3,065	9.96×10^{-1}	5.01×10^{-2}	216.7 K	70.7	153.6
bookx	13,854	34,609	3.21×10^{-5}	1.09×10^{-3}	521.1 K	15.1	37.6
ml1m	6,034	3,125	4.47×10^{-2}	3.05×10^{-2}	574.4 K	183.8	95.2
jester	50,109	100	5.63×10^{-1}	2.03×10^{-1}	1.0 M	10,172.0	20.3
amazon-e	124,895	44,843	3.91×10^{-6}	1.92×10^{-4}	1.1 M	23.9	8.6
ml20m	136,674	13,680	5.40×10^{-3}	5.34×10^{-3}	10.0 M	729.3	73.0
netflix	463,435	17,721	1.18×10^{-2}	6.93×10^{-3}	56.9 M	3,209.7	122.7

five types of non-neural recommendation baselines: unpersonalized models, factorization-based models, nearest neighbor-based models, linear models, and graph-based models.

Neural Recommendation Models. Only a few neural recommendation models are able to yield recommendations under the strong generalization setting. We selected MultiVAE and MultiDAE as they are both common as well as provide (near) state-of-the-art performance on some datasets [60]. Both MultiVAE and MultiDAE are autoencoder-based recommendation models where the basic idea is to reconstruct the entire user profiles from partial versions. Extending MultiDAE, MultiVAE adopts a fully Bayesian approach to fit per-user variance, which could make it more prone to overfitting.

Unpersonalized Models (Non-Neural). Two unpersonalized recommendation models are adopted as baselines, namely Random and Popularity. Random generates a random permutation of items for each user, while Popularity generates recommendation results for users based on the popularity (total number of positive interactions) of items.

Factorization-based Models (Non-Neural). Two factorization-based models are selected, including PureSVD and ALS. Their main difference is that PureSVD performs a vanilla singular value decomposition on the user-item matrix, while ALS performs a weighted matrix factorization.

Nearest Neighbor-based Models (Non-Neural). ItemKNN and UserKNN are adopted as two traditional nearest neighbor-based recommendation models. Recommendations are derived based on the item-item and user-user similarity in the two models, respectively.

Linear Models (Non-Neural). SLIM [68] and Ease [80] are two popular linear baselines. Both of them aim to learn a linear function to capture the similarity for item-based collaborative filtering. However, SLIM aims to learn a sparse linear function, while Ease does not have the sparsity constraint. Hence Ease adopts the Frobenius norm of the learnable weight matrix as the regularization term instead of leveraging an l_1-norm as in SLIM.

Graph-based Models (Non-Neural). P3alpha [17], RP3beta [69], and Graph Filter based Collaborative Filtering (GFCF) [79] are adopted as the graph-based recommendation models. In general, these models consider the users and items as nodes, and the corresponding input matrix describes the existing edges between users and items with implicit user feedback. Specifically, in P3alpha, items are ranked based on the reaching probability of a three-step walk to every user for recommendation; RP3beta is a modified version of P3alpha, where the outcomes are normalized by the corresponding item popularity; GFCF performs prediction based on the propagated input implicit user feedback matrix through existing edges.

4 EXPERIMENTAL SETUP

4.1 Datasets

Dataset Selection. A lot of previous work has published results only on a relatively small number of datasets [83]. Our guiding principle for dataset selection is to choose the most commonly used public datasets to ensure replicability. Nine popular publicly available datasets are selected in this paper, namely ml100k [40], ml1m [40], ml20m [40], lastfm [45], kuai [35], bookx [12], jester [36], amazon-e [66], and netflix [10].

Pre-Processing Strategy. We conduct experiments based on implicit user feedback (such as binary entries representing clicks and purchases) and convert all ordinal rating data to binary. We follow the general consensus and treat all ratings greater or equal to four as positive and negative for the rest [83]. In addition, it is also often necessary in practice to filter out users and items with insufficient ratings, i.e., to perform h-core filtering. We choose a common setting [94] to assign h as five, and we present the statistics after filtering in Table 1.

4.2 Metrics

We present our evaluation metric choices in this subsection. Specifically, we include three types of metrics, namely metrics for utility, diversity, and semantic coherence. We present the details below.

Utility Metrics. There is a wide range of utility metrics that have been used to evaluate the ranking performance in top-n recommendation. However, multiple popular metrics can be unstable during evaluation. For example, nDCG is affected by the total number of items and may flip the model comparison conclusion by only changing the gain factor used in computation [13]. Here, we adopt three commonly used metrics that are more stable to measure the recommendation utility, namely Recall@k, HitRate@k,

and MeanRanks [34]. We define Recall@k for a user U_i as

$$\text{Recall@k} = \frac{|R(U_i) \cap T(U_i)|}{|T(U_i)|}, \quad U_i \in \mathcal{U}, \tag{1}$$

where $R(U_i)$ denotes the set of retrieved items among the top-n recommendations for user U_i; $T(U_i)$ represents the set of relevant items that are used during the test phase for user U_i. To be consistent with prior works, when $|T(U_i)| = 1$, we refer to Recall@k instead of HitRate@k. We present its formulation as

$$\text{HitRate@k} = |R(U_i) \cap T(U_i)|, \quad U_i \in \mathcal{U}^*. \tag{2}$$

Here $\mathcal{U}^*$ is $\mathcal{U}$ and $\mathcal{U}_{\text{tst}}$ in the memorization and generalization setting, respectively. Finally, we formulate MeanRanks as

$$\text{MeanRanks} = \frac{\sum_{I_i \in T(U_i)} \text{Rank}(I_i)}{|T(U_i)|}, \quad U_i \in \mathcal{U}^*, \tag{3}$$

where the function Rank$(\cdot)$ takes an item as input and outputs the corresponding rank position in the output recommendations. We leverage both Recall@k and MeanRanks to measure the model utility under the reranking memorization task. In other settings, we utilize HitRate@k and MeanRanks as the corresponding metrics.

Diversity Metrics. Diversity is among the most popular beyond-accuracy ranking metrics [6, 29, 44, 71, 84, 93]. In our experiments, we employ the Gini Index [88] and Shannon Entropy [37] to measure diversity. The Gini Index comes from economics, where it was originally used to measure disparity. To compute it, one counts the number of times an item appears in the top-n recommendations for users in the test set, and then normalizes this frequency so that the item frequencies form a probability distribution (e.g., $p(I_i)$ for item I_i). After sorting the items in increasing order $i_1, i_2, \ldots, i_n$, the Gini Index is defined as

$$\text{Gini\_Index} = \frac{1}{n-1} \sum_{j=1}^{n} (2j - n - 1) p\left(I_{i_j}\right). \tag{4}$$

As a second measure of diversity, we adopt Shannon Entropy. It is computed as follows:

$$\text{Shannon\_Entropy} = -\sum_{i=1}^{n} p(I_i) \log p(I_i). \tag{5}$$

We use both metrics above to measure diversity in order to get a more accurate picture of performance differences between models.

Semantic Coherence Metrics. Another important aspect of differences between neural and non-neural models is the degree to which a model captures semantic relationships between items in the embedding space (see also Section 2.3). Inspired by user-centric studies such as [90], we propose a metric named Semantic Coherence Index (SCI). For each item $I_i \in \mathcal{I}$, we build a vector representation based on information that expresses human perception of similarity and relatedness (e.g., manual semantic tags). Let $\mathbf{r}_i$ be the item similarity vector that a model produces. For models that use an item-item matrix, we simply use the corresponding entry. For models that do not have such a matrix, we input a one-hot user vector with entry i set to one. We also compute the cosine similarity between the semantic vector of I_i and that of every other item and denote the resulting vector as $\mathbf{s}_i$. The SCI is then computed as

$$\text{SCI} = \frac{1}{|\mathcal{I}|} \sum_{I_i \in \mathcal{I}} \text{Pearson}\left(\mathbf{s}_i, \mathbf{r}_i\right), \tag{6}$$

where the function f is the recommendation model that takes two items as input, and outputs a vector depicting the predicted preference scores over all items in $\mathcal{I}$; function Pearson$(\cdot)$ takes two vectors as input and outputs the value of their Pearson correlation. We leverage SCI to measure how well recommendation models capture the item-item semantics.

4.3 Experimental Settings

Evaluation Protocol. For each model, we perform training and testing under nested cross-validation with five user-based folds. For each round of evaluation, we use three folds as the training data, one fold for validation, and one fold for test. We set the value for the truncation length l for test-train user similarity ranking truncation (under subgroup-specific performance evaluation) to ten.

Hyper-parameter Tuning. We conduct an extensive hyper-parameter search for all models. Specifically, the hyper-parameter search was done with Bayesian optimization for 50 iterations in every round of cross-validation. We used HitRate@50 as the target metric. The search space and the values of optimal hyper-parameters for each dataset in every round of cross-validation will be released upon acceptance.

5 EMPIRICAL INVESTIGATION

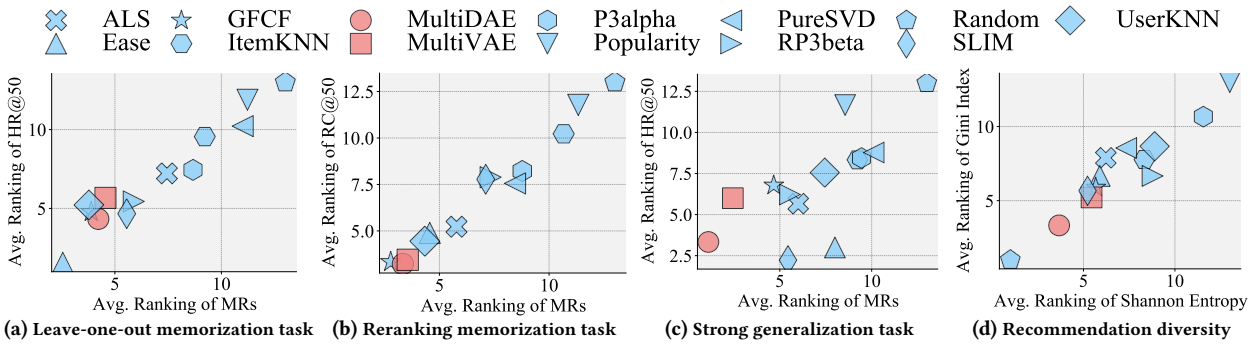
5.1 Finding 1: Neural models excel on datasets with larger sizes in memorization

We first perform experiments on the two memorization settings introduced in Section 2.2. In the leave-one-out memorization task, we collect the values of HitRate@50 and Meanranks. In the reranking memorization task, we collect the values of Recall@50 and MeanRanks. We compute the ranking of all models for each dataset, and then rank them by their average rank over all datasets. We present the results in Fig. 1(a) and Fig. 1(b).

Leave-One-Out Memorization Task. We first discuss the differences between neural and non-neural models in the leave-one-out memorization tasks. From the experimental results, we can observe that neural models do not exhibit superiority over other non-neural models, and simpler non-neural models are still able to outperform neural ones. For example, Ease, as a linear recommendation model, outperforms neural models on both utility metrics.

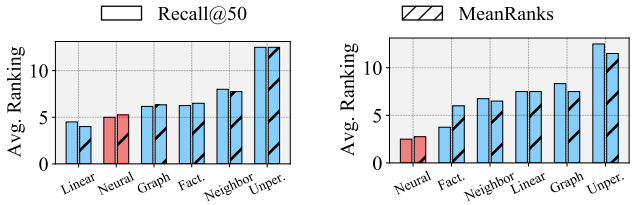
Reranking Memorization Task. We examine the differences between neural and non-neural models in the so-called reranking memorization task. We observe that neural models are among the ones with the best reranking performance. A potential reason could be that the high model capacity enables neural models to perform well in memorizing all implicit user feedback that has been seen during training. Nevertheless, it is also worth noting that several non-neural models also perform well, e.g., linear models, which is consistent with the findings from existing work (e.g., [33]) in weak generalization settings.

Memorization under Different Dataset Sizes. Based on the previous results, we further zoom in on the differences regarding the fitting capabilities of neural models and non-neural models. Specifically, we compute the average performance ranking in the reranking memorization task for each baseline model over datasets with the top-k smallest and largest sizes (in terms of the total number of implicit user feedback). We present the results for $k = 2$



(a) Leave-one-out memorization task **(b) Reranking memorization task** **(c) Strong generalization task** **(d) Recommendation diversity**

Figure 1: In (a) and (b), we present the results of the two different memorization tasks, respectively. In (c), we show the generalization results in terms of recommendation utility. In (d), we present the results for recommendation diversity in terms of Gini Index and Shannon Entropy. We use "RC", "HR", and "MRs" to represent Recall, HitRate, and MeanRanks, respectively. Average rankings are computed over all nine datasets for every recommendation model. The two neural recommendation models are in light coral while others use a light blue.



(a) Average ranking on the datasets with top-two smallest sizes **(b) Average ranking on the datasets with top-two largest sizes**

Figure 2: Results of the reranking memorization task between neural models and other five groups of baselines. The performance of each baseline group is averaged over the top two smallest/largest datasets. The two neural models are in light coral and others are in light blue.

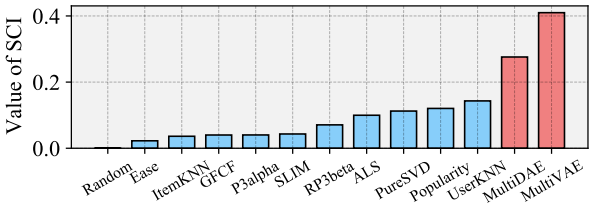


Figure 3: Results for semantic coherence on the ml20m dataset. The two neural recommendation models are marked in light coral and others in light blue.

in Fig. 2(a) and Fig. 2(b). We observe that linear recommendation models achieve the best performance over all other models on the top-k smallest datasets. Nevertheless, neural recommendation models achieve the best performance on the top-k largest datasets. Although not shown, these trends also hold for other values of k. One hypothesis for these results is that smaller datasets with less implicit user feedback (i.e., user-item interactions) tend to possess simpler user preference patterns. As a consequence, simpler models, such as linear ones, are able to memorize the user profiles well. Another explanation might be that as neural models are higher capacity, they need more data in order to not overfit.

To summarize the observations, we found that neural models did not show superior performance in our leave-one-out memorization task but were among the best-performing ones in the reranking memorization task. We hypothesize that neural models perform better in the reranking task because it emphasizes the performance on recovering all training items correctly at the same time. With their smooth parameterization, neural models may be able to do this soft clustering better than sparse linear models.

5.2 Finding 2: Neural and non-neural models generalize differently

We now turn to investigate how neural and non-neural models differ in their generalization abilities. Going beyond pure accuracy, we discuss their differences from three perspectives, including recommendation utility, diversity, and semantic coherence.

Recommendation Utility. This assesses recommendation utility by measuring HitRate and MeanRanks in the strong generalization setting. We first present a general summary of the rankings of each recommendation model in Fig. 1(c). We observe that in terms of HitRate@50 (the vertical axis), although neural models are among the ones with the best performance, both linear models (Ease and SLIM) achieve better performance compared with the neural ones. Put another way: neural models did not perform any better than traditional linear models. However, when considering how far relevant items appear in the ranking by measuring MeanRanks (the horizontal axis), neural models are superior to all other recommendation models. These experimental results point again to the ability of neural models to have soft and coherent clusters – similar to the reranking memorization task.

In addition, we also present the detailed performance statistics in Table 2. These detailed results are with our previous summary: neural models tend to outperform other non-neural models in terms of MeanRanks, but they do not exhibit any superiority compared with linear models in terms of HitRate.

Recommendation Diversity. To study the performance differences in recommendation diversity, we computed the Gini Index and Shannon Entropy of the resulting recommendations. We look at the top-10 ($k = 10$) performance scores for more discriminative power and present the diversity rankings of each model in Fig. 1(d). Our observations are as follows. First and unsurprisingly,

the unpersonalized Random method achieves the best performance in terms of both metrics for recommendation diversity. However, Random is not an effective recommendation approach in real-world applications due to its poor utility, e.g., low HitRate and MeanRanks indicated in Fig. 1(c). Among the remaining models, neural models achieve the best performance on both diversity metrics. These results indicate that neural models deliver recommendations with a higher level of diversity, which implies that neural models are less prone to putting high scores on popular items.

Recommendation Semantic Coherence. We finally analyze the semantic coherence of recommendation, where we use side information about the items collected from user-specified tags. The results we present here are based on the MovieLens ml20m dataset, where human-annotated semantic information is available from the semantic tags submitted by users. We follow the methodology outlined in Section 2.3 and present the experimental results in Fig. 3. We observe that neural models achieve the highest values in terms of Semantic Coherence Index. Such an observation suggests that neural models are better at capturing item-item semantics as they are learning from user feedback.

To summarize, we found that in the strong generalization setting, non-neural models (on average) achieve better HitRate, while neural models achieve better performance in terms of MeanRanks. This at least gives some guidance for practitioners – non-neural models should be preferred when a better HitRate is desired. In addition, neural models achieve a higher level of diversity and can better capture the item-item semantic relationship compared with non-neural ones. Thus neural models should be preferred when diversity and having adequate item-item relationships are of importance.

5.3 Finding 3: Neural models exhibit stronger robustness among different subgroups

We now explore subgroup-specific performance differences between neural and non-neural models as discussed in Section 4. We examine performance differences between user subgroups and item subgroups in strong generalization settings as before.

Similar Users vs. Dissimilar Users. We first study the performance differences between similar and dissimilar users. Here similar users refer to test users that have the largest top-ranked cosine similarities with the users that have been seen during training (in terms of implicit user feedback), while dissimilar users are those who are most dissimilar to those training users. We present the experimental results over the two subgroups of users in Fig. 4(a) and Fig. 4(b), respectively. We make the following observations. First, neural and non-neural models mimic the performance ranking we found in the generalization experiments (introduced in Section 5.2): neural models do better in terms of MeanRanks but cannot outperform traditional linear recommendation models on HitRate. Second, relative to its performance over similar users, MultiDAE achieves better performance over dissimilar ones. Other models don't experience much change in terms of HitRate. However, in terms of MeanRanks, both neural models fare substantially better than non-neural models on dissimilar users. These findings indicate that neural models are more robust with respect to their performance across the two different user subgroups (i.e., similar and dissimilar), both for HitRate as well as MeanRanks.

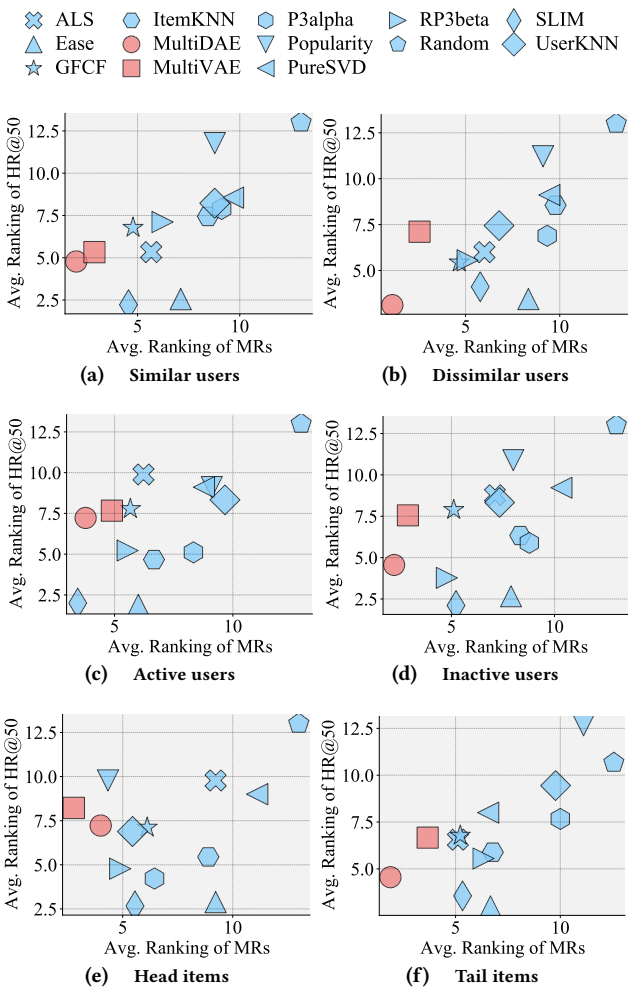


Figure 4: Evaluation of subgroup-specific performance. We use "HR" and "MRs" to represent HitRate and MeanRanks, respectively. The two neural models are marked out with color light coral against others as light blue.

Active Users vs. Inactive Users. We also analyze the performance differences between active and inactive users. Here active refers to users with the largest numbers of ratings, while inactive denotes the opposite (thresholds were picked based on percentiles). Our findings are as follows. First, in terms of active user subgroups, both neural models exhibit better performance for MeanRanks than for HitRate. However, they do not consistently outperform non-neural models. For example, SLIM is a much simpler non-neural model but achieves better performance compared to neural models in terms of both utility metrics. Second, when considering inactive users, MultiDAE achieves better performance, and the performance of most models remains stable for HitRate. However, the MeanRanks performance of most non-neural models declines for inactive users, while both neural models fare best. These observations imply that neural models demonstrate better robustness compared with non-neural models (especially on MeanRanks) between users with different activity levels.

(a) ml100k.

Model	HR@50	95% CI	MeanRanks	95% CI
Random	0.047	0.030	476.3	39.5
Popularity	0.324	0.067	193.5	30.1
UserKNN	0.553	0.071	107.3	23.8
ItemKNN	0.569	0.071	102.9	22.0
P3alpha	0.570	0.071	111.3	24.8
RP3beta	0.571	0.071	95.4	19.8
PureSVD	0.577	0.071	108.7	24.9
MultiVAE	0.579	0.071	92.1	19.6
GFCF	0.582	0.071	97.7	21.6
MultiDAE	0.597	0.070	85.8	18.6
ALS	0.598	0.070	96.7	22.0
Ease	0.600	0.070	99.1	23.9
SLIM	0.604	0.070	101.5	23.2

(b) ml1m.

Model	HR@50	95% CI	MeanRanks	95% CI
Random	0.018	0.007	1518.1	50.0
Popularity	0.251	0.024	410.6	29.1
ItemKNN	0.441	0.028	276.3	30.2
UserKNN	0.449	0.028	258.4	29.0
P3alpha	0.456	0.028	296.3	31.6
RP3beta	0.462	0.028	242.1	26.3
PureSVD	0.471	0.028	271.1	32.3
GFCF	0.487	0.028	219.9	25.6
MultiVAE	0.492	0.028	180.3	18.5
SLIM	0.496	0.028	208.4	23.1
MultiDAE	0.503	0.028	173.8	18.1
ALS	0.504	0.028	216.8	25.6
Ease	0.512	0.028	224.4	27.1

(c) ml20m.

Model	HR@50	95% CI	MeanRanks	95% CI
Random	0.004	0.001	6814.6	46.6
Popularity	0.241	0.005	695.9	14.7
P3alpha	0.432	0.006	957.8	24.9
ItemKNN	0.462	0.006	559.9	18.4
PureSVD	0.469	0.006	808.5	30.9
RP3beta	0.476	0.006	374.7	12.6
UserKNN	0.496	0.006	517.4	18.6
GFCF	0.516	0.006	385.9	17.5
ALS	0.519	0.006	494.7	22.2
MultiVAE	0.522	0.006	253.8	7.8
SLIM	0.523	0.006	405.1	14.8
Ease	0.548	0.006	744.3	30.4
MultiDAE	0.564	0.006	197.7	6.6

(d) lastfm.

Model	HR@50	95% CI	MeanRanks	95% CI
Random	0.024	0.015	1374.2	82.2
Popularity	0.206	0.041	660.6	78.9
PureSVD	0.464	0.051	337.8	64.7
UserKNN	0.467	0.051	311.0	60.4
P3alpha	0.491	0.051	220.0	44.0
ALS	0.492	0.051	295.4	58.9
ItemKNN	0.505	0.051	193.2	37.8
RP3beta	0.508	0.051	191.5	38.1
Ease	0.522	0.051	287.5	59.0
GFCF	0.527	0.051	205.0	43.3
SLIM	0.530	0.051	201.6	41.4
MultiVAE	0.540	0.051	172.3	34.6
MultiDAE	0.544	0.051	167.7	33.9

(e) kuai.

Model	HR@50	95% CI	MeanRanks	95% CI
Random	0.016	0.015	1419.9	98.9
P3alpha	0.318	0.054	601.7	90.2
MultiVAE	0.319	0.054	567.4	84.7
Popularity	0.319	0.054	588.9	86.0
GFCF	0.320	0.054	590.4	87.6
ItemKNN	0.322	0.055	621.6	90.8
RP3beta	0.322	0.055	578.9	87.2
MultiDAE	0.323	0.055	577.3	84.9
UserKNN	0.323	0.055	598.6	87.7
PureSVD	0.327	0.055	628.8	93.6
ALS	0.330	0.055	599.8	90.7
Ease	0.330	0.055	617.3	92.7
SLIM	0.331	0.055	607.4	89.0

(f) bookx.

Model	HR@50	95% CI	MeanRanks	95% CI
Random	0.001	0.001	17270.8	372.2
Popularity	0.055	0.008	9118.9	395.9
PureSVD	0.109	0.012	10701.4	450.7
ALS	0.117	0.012	9704.5	430.6
MultiVAE	0.129	0.012	4976.9	275.0
GFCF	0.132	0.013	4852.8	282.9
MultiDAE	0.137	0.013	4704.4	262.9
UserKNN	0.137	0.013	7749.4	373.8
ItemKNN	0.145	0.013	10098.6	400.6
P3alpha	0.153	0.013	8949.8	392.8
Ease	0.156	0.013	10142.6	463.3
RP3beta	0.157	0.014	7526.9	374.7
SLIM	0.161	0.014	8243.6	376.2

(g) jester.

Model	HR@50	95% CI	MeanRanks	95% CI
Random	0.633	0.009	40.8	0.5
PureSVD	0.837	0.007	23.6	0.5
Popularity	0.891	0.006	20.0	0.4
ItemKNN	0.912	0.006	18.3	0.4
P3alpha	0.927	0.005	17.4	0.4
GFCF	0.928	0.005	17.1	0.4
ALS	0.936	0.005	15.4	0.3
Ease	0.939	0.005	14.9	0.3
MultiVAE	0.939	0.005	15.7	0.3
RP3beta	0.939	0.005	16.0	0.3
SLIM	0.939	0.005	15.1	0.3
UserKNN	0.940	0.005	15.4	0.3
MultiDAE	0.947	0.004	14.7	0.3

(h) amazon-e.

Model	HR@50	95% CI	MeanRanks	95% CI
Random	0.001	0.000	22423.5	160.7
Popularity	0.063	0.003	9779.2	153.7
PureSVD	0.070	0.003	12358.1	179.7
ALS	0.089	0.004	12863.8	187.2
GFCF	0.114	0.004	10004.1	165.5
MultiVAE	0.114	0.004	7169.4	128.3
UserKNN	0.119	0.004	14265.9	178.9
MultiDAE	0.126	0.004	6424.7	120.2
Ease	0.128	0.004	17552.1	221.5
ItemKNN	0.128	0.004	15762.3	178.8
P3alpha	0.129	0.004	14208.5	178.4
RP3beta	0.129	0.004	14341.7	178.4
SLIM	0.132	0.004	12844.7	174.0

(i) netflix.

Model	HR@50	95% CI	MeanRanks	95% CI
Random	0.003	0.000	8794.4	32.7
Popularity	0.172	0.002	1005.6	12.2
P3alpha	0.361	0.003	1759.7	25.7
ItemKNN	0.374	0.003	1426.6	23.0
UserKNN	0.387	0.003	1144.4	20.4
RP3beta	0.389	0.003	1415.7	23.0
PureSVD	0.398	0.003	1281.5	24.2
GFCF	0.404	0.003	556.6	12.3
MultiVAE	0.417	0.003	420.9	6.8
ALS	0.424	0.003	742.4	16.5
SLIM	0.450	0.003	830.3	17.2
MultiDAE	0.462	0.003	327.8	5.8
Ease	0.467	0.003	1099.6	23.0

Table 2: Performance of all recommendation models over nine real-world datasets. The results on each dataset are ranked in an ascending order from top to bottom in terms of HitRate@50 (denoted as HR@50). "95% CI" represents the 95% confidence interval. Neural models are marked in grey.

Head Items vs. Tail Items. We finally compare the performance of the recommendation models on head and tail items. Again, head items refer to items with the largest number of implicit user feedback, while tail items are the opposite. The performance of each model is reported over two user subgroups partitioned by which type of item is held out randomly. We observe that neural models do not outperform non-neural ones in terms of HitRate, especially for users whose hold-out items are head items. However, neural models do have an advantage when considering MeanRanks in both settings. Second, both neural models maintain their performance advantage over other non-neural models on tail items, while the performance of most other non-neural models drops on tail items. Several non-neural models (e.g., ALS and PureSVD) show better performance on tail items, however, they are behind the neural models. These observations suggest that neural models not only achieve better MeanRanks scores in general but also exhibit better robustness across head and tail items.

Based on the discussion above, our results indicate that neural models tend to have stronger robustness over warm/cold-start instances compared to non-neural ones. As a consequence, we suggest that neural models should be preferred if better MeanRanks or robustness is desired in practice.

6 RELATED WORK

Researchers have long been noticing that the results of multiple recommender models are hard to compare due to differences in baseline implementations [76], dataset pre-processing strategies [52], and evaluation methodologies [27]. These findings have cast doubt on whether the reported performance increases of each new model

can actually add up. This leads researchers to re-investigate existing recommendation models [4, 33]. Most of the replication studies found it hard to obtain similar results as published previously [19]. This further boasts the claim that reproducibility and replicability in recommender systems are central issues[4, 8, 33, 62, 74]. For example, Sedhain et al. [78] pointed out that simple linear classification models compare quite favorably with state-of-the-art; Pérez et al. [70] found several neural recommendation models are to be non-reproducible; Kolesnikov et al. [51] pointed out potential flaws with popular evaluation strategies.

Neural vs. Non-Neural Recommendation Models. Compared with non-neural models, neural models have been claimed to outperform traditional recommender systems in recent years. To the best of our knowledge, there are only three other relevant studies on the difference between neural and non-neural models. Anelli et al. [4] found that neural models cannot outperform most traditional models, e.g., linear models and matrix factorization. Ludewig et al. [62] found similar results for session-based recommendation. Finally, Zhao et al. [94] argued that the training efficiency of neural models is significantly lower than that of other traditional models, such as linear models. This casts further doubt on whether or not it is worthwhile to adopt neural models in practice.

Recommender System Evaluation. It is worth noting that recommender systems do not exist in isolation but are part of a user-facing system [18]. In real-world live recommender systems, any potential flaw can have a potentially huge negative impact on users. Hence it is critical to systematically understand the properties of recommendation models offline before users are exposed to them. Multiple studies have tried to systematically evaluate existing recommendation models [4, 8, 33, 62, 74]. For example, Sun et al. [83] argued that certain essential factors (e.g., data splitting methods and hyper-parameter tuning strategies) can influence the performance recommendation model dramatically. Therefore, they proposed standardized procedures for a more rigorous evaluation. Other recent studies have also looked at other parts of the whole evaluation pipeline [5, 6, 33, 49, 55, 62]. Furthermore, there are also existing studies focusing on specific factors in evaluation. (1) **Dataset Construction.** Datasets have been playing a critical role in delivering accurate recommendations in existing recommendation models. The strategies to pre-process and construct the dataset have attracted much research attention [55, 57, 67, 75, 94]. For example, Sachdeva et al. [75] evaluated the efficacy of sixteen different sampling strategies for benchmarking recommendation models. (2) **Baselines.** Whether appropriate baselines are included or not is also a critical factor in the evaluation of recommender systems. As an example, Ji et al. [48] proposed to re-visit the commonly chosen baselines in recommender systems. (3) **Model Optimization.** Model optimization can also exert a significant influence on the performance of recommender systems, e.g., hyper-parameter tuning strategies. For example, Zhao et al. [94] argued that the search range of hyper-parameters often affects the performance substantially, while Anelli et al. [7] pointed out that only tuning a few parameters will usually help achieve adequate performance. (4) **Evaluation Metrics.** It is also important to study whether the adopted evaluation metrics are appropriate or not [21, 30, 54, 55, 94]. For instance, Zhao et al. [94] found that adopting sampling-based metrics could introduce bias in the evaluation process.

Significance of Our Work. Despite the progress in evaluating recommendation models, there is not much work comparing different types of recommendation models [49, 62]. However, we note that it is a critical issue since such an understanding facilitates the researchers and engineers to choose appropriate models under different application scenarios. Meanwhile, existing insights on the difference between neural and non-neural models are also limited. For example, Anelli et al. [4] conducted an evaluation between neural and non-neural models. However, they adopted weak generalization as the evaluation strategy, which we argue is not representative of real-world recommendation systems that are needed to make recommendations to new users. Ludewig et al. [62] compared the performance between neural and non-neural models. However, their work only focuses on session-based recommendation, ignoring the most common recommendation problem with implicit user feedback. Zhao et al. [94] evaluated popular recommendation models considering different factors. However, they did not provide a comprehensive discussion of the performance between these two types of recommendation models. Different from most existing work on recommendation model evaluation, this paper provides comprehensive evaluation results under a strong generalization setting, which gives a better understanding of the characteristics of the two types of recommendation models.

7 CONCLUSION AND DISCUSSION

In this work, we present a thorough investigation into the performance differences between neural recommendation models and non-neural ones. We empirically explored what type of benefit one may gain when using neural models instead of traditional non-neural ones for recommendation in practice. We introduced a number of practical and diverse evaluation strategies for benchmarking and then conducted extensive experiments on nine publicly available real-world datasets over 13 popular recommendation models. We show that in most cases, neural models do not show superior performance over other non-neural models when considering HitRate. They do, however, fare better in terms of MeanRanks. In addition, neural models, on average, showed a higher level of diversity, were better at capturing item-item semantics, and exhibited stronger robustness in warm-start and cold-start scenarios.

To the best of our knowledge, this is the first work to investigate the performance differences between neural and non-neural recommendation models in top-n recommendation with implicit user feedback. Investigations in terms of the widely studied memorization, under-explored generalization, and rarely discussed subgroup-specific performance are included. Future works based on this paper may explore whether neural models exhibit superiority over other popular metrics and whether the benefits of neural models are worth the training costs. However, we note that they are beyond the scope of this paper. We hope this work can help practitioners with their choice of recommendation models, inspire more research around understanding recommender models, and facilitate better model design in the future.

8 ACKNOWLEDGEMENTS

Yushun Dong and Jundong Li are supported by the National Science Foundation under grants IIS-2006844, IIS-2144209, and IIS-2223769.

REFERENCES

[1] Nor Aniza Abdullah, Rasheed Abubakar Rasheed, Mohd Hairul Nizam Md Nasir, and Md Mujibur Rahman. 2021. Eliciting auxiliary information for cold start user recommendation: A survey. *Applied Sciences* 11, 20 (2021), 9608.

[2] Gediminas Adomavicius and Jingjing Zhang. 2016. Classification, ranking, and top-K stability of recommendation algorithms. *INFORMS Journal on Computing* 28, 1 (2016), 129–147.

[3] Charu C Aggarwal et al. 2016. *Recommender systems*. Vol. 1. Springer.

[4] Vito Walter Anelli, Alejandro Bellogín, Tommaso Di Noia, Dietmar Jannach, and Claudio Pomo. 2022. Top-n recommendation algorithms: A quest for the state-of-the-art. *arXiv preprint arXiv:2203.01155* (2022).

[5] Vito Walter Anelli, Alejandro Bellogín, Tommaso Di Noia, and Claudio Pomo. 2021. Reenvisioning the comparison between neural collaborative filtering and matrix factorization. In *ACM Conference on Recommender Systems*. 521–529.

[6] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and Tommaso Di Noia. 2021. Elliot: a comprehensive and rigorous framework for reproducible recommender systems evaluation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2405–2414.

[7] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Claudio Pomo, and Azzurra Ragone. 2019. On the discriminative power of hyper-parameters in cross-validation and how to choose them. In *Proceedings of the 13th ACM conference on recommender systems*. 447–451.

[8] Timothy G Armstrong, Alistair Moffat, William Webber, and Justin Zobel. 2009. Improvements that don't add up: ad-hoc retrieval results since 1998. In *Proceedings of the 18th ACM conference on Information and knowledge management*. 601–610.

[9] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. 2017. A closer look at memorization in deep networks. In *International conference on machine learning*. PMLR, 233–242.

[10] James Bennett, Stan Lanning, et al. 2007. The netflix prize. In *Proceedings of KDD cup and workshop*, Vol. 2007. New York, 35.

[11] Jiajun Bu, Xin Shen, Bin Xu, Chun Chen, Xiaofei He, and Deng Cai. 2016. Improving collaborative recommendation via user-item subgroups. *IEEE Transactions on Knowledge and Data Engineering* 28, 9 (2016), 2363–2375.

[12] Joseph A. Konstan Georg Lausen Cai-Nicolas Ziegler, Sean M. McNee. 2005. Improving Recommendation Lists Through Topic Diversification. In *Proceedings of the 14th International World Wide Web Conference (WWW '05)*.

[13] Rocío Cañamares, Pablo Castells, and Alistair Moffat. 2020. Offline evaluation options for recommender systems. *Information Retrieval Journal* 23, 4 (2020), 387–410.

[14] Satrajit Chatterjee. 2018. Learning and memorization. In *International Conference on Machine Learning*. PMLR, 755–763.

[15] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 2019. Behavior sequence transformer for e-commerce recommendation in alibaba. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*. 1–4.

[16] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[17] Colin Cooper, Sang Hyuk Lee, Tomasz Radzik, and Yiannis Siantos. 2014. Random walks in recommender systems: exact computation and simulations. In *Proceedings of the 23rd international conference on world wide web*. 811–816.

[18] Paolo Cremonesi, Franca Garzotto, and Roberto Turrin. 2013. User-centric vs. system-centric evaluation of recommender systems. In *Ifip conference on human-computer interaction*. Springer, 334–351.

[19] Paolo Cremonesi and Dietmar Jannach. 2021. Progress in recommender systems research: Crisis? What crisis? *AI Magazine* 42, 3 (2021), 43–54.

[20] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*. 39–46.

[21] Maurizio Ferrari Dacrema, Nicolò Felicioni, and Paolo Cremonesi. 2022. Offline Evaluation of Recommender Systems in a User Interface With Multiple Carousels. *Frontiers in Big Data* 5 (2022).

[22] Aminu Da'u and Naomie Salim. 2020. Recommendation system based on deep learning methods: a systematic review and new directions. *Artificial Intelligence Review* 53, 4 (2020), 2709–2748.

[23] Gabriel de Souza Pereira Moreira, Felipe Ferreira, and Adilson Marques da Cunha. 2018. News session-based recommendations using deep neural networks. In *Proceedings of the 3rd workshop on deep learning for recommender systems*. 15–23.

[24] Yashar Deldjoo, Alejandro Bellogin, and Tommaso Di Noia. 2021. Explaining recommender systems fairness and accuracy through the lens of data characteristics. *Information Processing & Management* 58, 5 (2021), 102662.

[25] Mukund Deshpande and George Karypis. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 143–177.

[26] Michael D Ekstrand, Anubrata Das, Robin Burke, and Fernando Diaz. 2012. Fairness in recommender systems. In *Recommender systems handbook*. Springer, 679–707.

[27] Michael D Ekstrand, Michael Ludwig, Joseph A Konstan, and John T Riedl. 2011. Rethinking the recommender research ecosystem: reproducibility, openness, and lenskit. In *Proceedings of the fifth ACM conference on Recommender systems*. 133–140.

[28] Michael D Ekstrand, Mucun Tian, Mohammed R Imran Kazi, Hoda Mehrpouyan, and Daniel Kluver. 2018. Exploring author gender in book rating and recommendation. In *Proceedings of the 12th ACM conference on recommender systems*. 242–250.

[29] Farzad Eskandanian and Bamshad Mobasher. 2020. Using stable matching to optimize the balance between accuracy and diversity in recommendation. In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization*. 71–79.

[30] Nicolò Felicioni, M Ferrari Dacrema, FB Perez Maurera, and Paolo Cremonesi. 2021. Measuring the ranking quality of recommendations in a two-dimensional carousel setting. In *11th Italian Information Retrieval Workshop, IIR 2021*, Vol. 2947. CEUR-WS, 1–14.

[31] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep session interest network for click-through rate prediction. *arXiv preprint arXiv:1905.06482* (2019).

[32] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. 2021. A troubling analysis of reproducibility and progress in recommender systems research. *ACM Transactions on Information Systems (TOIS)* 39, 2 (2021), 1–49.

[33] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM conference on recommender systems*. 101–109.

[34] Norbert Fuhr. 2018. Some common mistakes in IR evaluation, and how they can be avoided. In *Acm sigir forum*, Vol. 51. ACM New York, NY, USA, 32–41.

[35] Chongming Gao, Shijun Li, Wenqiang Lei, Biao Li, Peng Jiang, Jiawei Chen, Xiangnan He, Jiaxin Mao, and Tat-Seng Chua. 2022. KuaiRec: A Fully-observed Dataset for Recommender Systems. *arXiv preprint arXiv:2202.10842* (2022).

[36] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. 2001. Eigentaste: A constant time collaborative filtering algorithm. *information retrieval* 4, 2 (2001), 133–151.

[37] Robert M Gray. 2011. *Entropy and information theory*. Springer Science & Business Media.

[38] 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* (2017).

[39] Junpeng Guo, Wenxiang Zhang, Weiguo Fan, and Wenhua Li. 2018. Combining geographical and social influences with deep learning for personalized point-of-interest recommendation. *Journal of Management Information Systems* 35, 4 (2018), 1121–1153.

[40] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[41] 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*. 639–648.

[42] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[43] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S Yu. 2018. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1531–1540.

[44] Neil Hurley and Mi Zhang. 2011. Novelty and diversity in top-n recommendation–analysis and evaluation. *ACM Transactions on Internet Technology (TOIT)* 10, 4 (2011), 1–30.

[45] Peter Brusilovsky Iván Cantador and Tsvi Kuflik. 2011. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys'11)*.

[46] Mohsen Jamali and Martin Ester. 2009. Trustwalker: a random walk model for combining trust-based and item-based recommendation. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 397–406.

[47] Mohsen Jamali and Martin Ester. 2009. Using a trust network to improve top-n recommendation. In *Proceedings of the third ACM conference on Recommender systems*. 181–188.

[48] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2020. A re-visit of the popularity baseline in recommender systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1749–1752.

[49] Ruoming Jin, Dong Li, Jing Gao, Zhi Liu, Li Chen, and Yang Zhou. 2021. Towards a better understanding of linear models for recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 776–785.

[50] Michael Jugovac, Dietmar Jannach, and Mozhgan Karimi. 2018. Streamingrec: a framework for benchmarking stream-based news recommenders. In *Proceedings of the 12th ACM conference on recommender systems*. 269–273.

[51] Sergey Kolesnikov, Oleg Lashinin, Michail Pechatov, and Alexander Kosov. 2021. TTRS: Tinkoff Transactions Recommender System benchmark. *arXiv preprint arXiv:2110.05589* (2021).

[52] Joseph A Konstan and Gediminas Adomavicius. 2013. Toward identification and adoption of best practices in algorithmic recommender systems research. In *Proceedings of the international workshop on Reproducibility and replication in recommender systems evaluation*. 23–28.

[53] Georgia Koutrika. 2018. Modern recommender systems: from computing matrices to thinking with neurons. In *Proceedings of the 2018 International Conference on Management of Data*. 1651–1654.

[54] Karl Krauth, Sarah Dean, Alex Zhao, Wenshuo Guo, Mihaela Curmei, Benjamin Recht, and Michael I Jordan. 2020. Do Offline Metrics Predict Online Performance in Recommender Systems? *arXiv preprint arXiv:2011.07931* (2020).

[55] Sara Latifi, Dietmar Jannach, and Andrés Ferraro. 2022. Sequential recommendation: A study on transformers, nearest neighbors and sampled metrics. *Information Sciences* 609 (2022), 660–678.

[56] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.

[57] Dong Li, Ruoming Jin, Jing Gao, and Zhi Liu. 2020. On sampling top-k recommendation evaluation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2114–2124.

[58] Ruifeng Li, Yin Zhang, Haihan Yu, Xiaojun Wang, Jiangqin Wu, and Baogang Wei. 2011. A social network-aware top-N recommender system using GPU. In *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*. 287–296.

[59] Yunqi Li, Hanxiong Chen, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2021. User-oriented fairness in recommendation. In *Proceedings of the Web Conference 2021*. 624–632.

[60] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 world wide web conference*. 689–698.

[61] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction* 28, 4 (2018), 331–390.

[62] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. 2019. Performance comparison of neural and non-neural approaches to session-based recommendation. In *Proceedings of the 13th ACM conference on recommender systems*. 462–466.

[63] Benjamin M Marlin. 2003. Modeling user rating profiles for collaborative filtering. *Advances in neural information processing systems* 16 (2003).

[64] Benjamin M Marlin and Richard S Zemel. 2009. Collaborative prediction and ranking with non-random missing data. In *Proceedings of the third ACM conference on Recommender systems*. 5–12.

[65] Paolo Massa and Paolo Avesani. 2007. Trust-aware recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*. 17–24.

[66] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 43–52.

[67] Zaiqiao Meng, Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2020. Exploring data splitting strategies for the evaluation of recommendation models. In *Fourteenth ACM conference on recommender systems*. 681–686.

[68] Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In *2011 IEEE 11th international conference on data mining*. IEEE, 497–506.

[69] Bibek Paudel, Fabian Christoffel, Chris Newell, and Abraham Bernstein. 2016. Updatable, accurate, diverse, and scalable recommendations for interactive applications. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 7, 1 (2016), 1–34.

[70] Fernando Benjamín Pérez Maurera, Maurizio Ferrari Dacrema, and Paolo Cremonesi. 2022. An Evaluation Study of Generative Adversarial Networks for Collaborative Filtering. In *European Conference on Information Retrieval*. Springer, 671–685.

[71] Lijing Qin and Xiaoyan Zhu. 2013. Promoting diversity in recommendation by entropy regularizer. In *Twenty-Third International Joint Conference on Artificial Intelligence*. Citeseer.

[72] Hossein A Rahmani, Yashar Deldjoo, Ali Tourani, and Mohammadmehdi Naghiaei. 2022. The unfairness of active users and popularity bias in point-of-interest recommendation. *arXiv preprint arXiv:2202.13307* (2022).

[73] Chamsi Abu Quba Rana, Hassas Salima, Fayyad Usama, and Chamsi Hammam. 2014. From a" cold" to a" warm" start in recommender systems. In *2014 IEEE 23rd International WETICE Conference*. IEEE, 290–292.

[74] Steffen Rendle, Li Zhang, and Yehuda Koren. 2019. On the difficulty of evaluating baselines: A study on recommender systems. *arXiv preprint arXiv:1905.01395* (2019).

[75] Noveen Sachdeva, Carole-Jean Wu, and Julian McAuley. 2022. On sampling collaborative filtering datasets. *arXiv preprint arXiv:2201.04768* (2022).

[76] Alan Said and Alejandro Bellogín. 2014. Comparative recommender system evaluation: benchmarking recommendation frameworks. In *Proceedings of the 8th ACM Conference on Recommender systems*. 129–136.

[77] J Ben Schafer, Joseph A Konstan, and John Riedl. 2001. E-commerce recommendation applications. *Data mining and knowledge discovery* 5 (2001), 115–153.

[78] Suvash Sedhain, Aditya Menon, Scott Sanner, and Darius Braziunas. 2016. On the effectiveness of linear models for one-class collaborative filtering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.

[79] Yifei Shen, Yongji Wu, Yao Zhang, Caihua Shan, Jun Zhang, B Khaled Letaief, and Dongsheng Li. 2021. How Powerful is Graph Convolution for Recommendation?. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1619–1629.

[80] Harald Steck. 2019. Embarrassingly shallow autoencoders for sparse data. In *The World Wide Web Conference*. 3251–3257.

[81] Ke Sun, Tieyun Qian, Tong Chen, Yile Liang, Quoc Viet Hung Nguyen, and Hongzhi Yin. 2020. Where to go next: Modeling long-and short-term user preferences for point-of-interest recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 214–221.

[82] Zhu Sun, Hui Fang, Jie Yang, Xinghua Qu, Hongyang Liu, Di Yu, Yew-Soon Ong, and Jie Zhang. 2022. DaisyRec 2.0: Benchmarking Recommendation for Rigorous Evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[83] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. 2020. Are we evaluating rigorously? benchmarking recommendation for reproducible evaluation and fair comparison. In *Fourteenth ACM conference on recommender systems*. 23–32.

[84] Daniel Valcarce, Alejandro Bellogín, Javier Parapar, and Pablo Castells. 2018. On the robustness and discriminative power of information retrieval metrics for top-N recommendation. In *Proceedings of the 12th ACM conference on recommender systems*. 260–268.

[85] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 world wide web conference*. 1835–1844.

[86] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 839–848.

[87] Wenjie Wang, Fuli Feng, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2021. Denoising implicit feedback for recommendation. In *Proceedings of the 14th ACM international conference on web search and data mining*. 373–381.

[88] Colin Wilkie and Leif Azzopardi. 2014. Best and fairest: An empirical analysis of retrieval system bias. In *Advances in Information Retrieval: 36th European Conference on IR Research, ECIR 2014, Amsterdam, The Netherlands, April 13-16, 2014. Proceedings 36*. Springer, 13–25.

[89] Bin Xu, Jiajun Bu, Chun Chen, and Deng Cai. 2012. An exploration of improving collaborative recommender systems via user-item subgroups. In *Proceedings of the 21st international conference on World Wide Web*. 21–30.

[90] Yuan Yao and F Maxwell Harper. 2018. Judging similarity: a user-centric study of related item recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 288–296.

[91] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat Thalmann. 2013. Time-aware point-of-interest recommendation. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. 363–372.

[92] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2017. Understanding deep learning requires rethinking generalization (2016). *arXiv preprint arXiv:1611.03530* (2017).

[93] Mi Zhang and Neil Hurley. 2008. Avoiding monotony: improving the diversity of recommendation lists. In *Proceedings of the 2008 ACM conference on Recommender systems*. 123–130.

[94] Wayne Xin Zhao, Zihan Lin, Zhichao Feng, Pengfei Wang, and Ji-Rong Wen. 2022. A Revisiting Study of Appropriate Offline Evaluation for Top-N Recommendation Algorithms. *ACM Transactions on Information Systems (TOIS)* (2022).

[95] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, et al. 2021. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4653–4664.

[96] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 world wide web conference*. 167–176.