# SimpleX: A Simple and Strong Baseline for Collaborative Filtering

Kelong Mao[1*], Jieming Zhu[2*], Jinpeng Wang[3], Quanyu Dai[2], Zhenhua Dong[2]
Xi Xiao[3⋆], Xiuqiang He[2]

[1]Gaoling School of AI, Renmin University of China
[2]Huawei Noah's Ark Lab, Shenzhen, China
[3]Tsinghua Shenzhen International Graduate School, Tsinghua University, China
<mkl@ruc.edu.cn><jiemingzhu@ieee.org><wjp20@mails.tsinghua.edu.cn><daiquanyu@huawei.com>
<dongzhenhua@huawei.com><xiaox@sz.tsinghua.edu.cn><hexiuqiang1@huawei.com>

## ABSTRACT

Collaborative filtering (CF) is a widely studied research topic in recommender systems. The learning of a CF model generally depends on three major components, namely interaction encoder, loss function, and negative sampling. While many existing studies focus on the design of more powerful interaction encoders, the impacts of loss functions and negative sampling ratios have not yet been well explored. In this work, we show that the choice of loss function as well as negative sampling ratio is equivalently important. More specifically, we propose the cosine contrastive loss (CCL) and further incorporate it to a simple unified CF model, dubbed SimpleX. Extensive experiments have been conducted on 11 benchmark datasets and compared with 29 existing CF models in total. Surprisingly, the results show that, under our CCL loss and a large negative sampling ratio, SimpleX can surpass most sophisticated state-of-the-art models by a large margin (e.g., max 48.5% improvement in NDCG@20 over LightGCN). We believe that SimpleX could not only serve as a simple strong baseline to foster future research on CF, but also shed light on the potential research direction towards improving loss function and negative sampling.

## CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Collaborative filtering**.

## KEYWORDS

Recommender systems; collaborative filtering; graph neural networks; contrastive loss

## 1 INTRODUCTION

Nowadays, personalized recommendation is ubiquitous in various applications, such as video recommendation in YouTube [5], product recommendation in Amazon [24], and news recommendation in Bing [34]. The goal of recommendation is to predict whether a user will interact (e.g., click or purchase) with an item and thus help users discover potential items of interests. Collaborative filtering (CF) [27] is a fundamental task in recommendation that leverages the collaborative information among users and items to predict users' preferences on candidate items. The simplicity and effectiveness make it one of the most popular techniques in recommender systems.

Generally, the learning process of a CF model can be separated to three major components, including interaction encoder, loss function, and the negative sampling strategy used when only positive (i.e.,implicit) feedbacks are available. Most existing studies focus on the design of more powerful interaction encoders to capture collaborative signals among users and items. Especially, the prevalence of deep learning motivates a rich line of work that applies various neural networks to CF, including multi-layer perceptrons (MLPs) [5, 11], auto-encoders [16], attention networks [3], transformers [28], graph neural networks (GNNs) [10], and so on. Nevertheless, these models tend to become more and more complex to show performance improvements. This somehow limits their practical applicability in industrial recommender systems that demand high efficiency.

On the contrary, few research efforts have been devoted to investigating the impacts of the latter two components. Specifically, while multiple loss functions have been used in CF, such as Bayesian personalized ranking (BPR) loss [22], binary cross-entropy loss [11], softmax cross-entropy loss [5], pairwise hinge loss [12], and mean square error loss [2], there is still a lack of systematic evaluation and comparisons among different loss functions. Furthermore, many recent GNN-based studies [10, 29, 30, 32, 33] experiment with the BPR loss [22] and simply set the negative sampling ratio to a small value (i.e., sampling 1 or 10 negative samples per positive user-item pair). In this way, they can justify the superiority of their proposed interaction encoders, but they neglect the importance of loss functions and negative sampling in the learning of CF models.

In fact, we empirically observed that training with the BPR loss and a small negative sampling ratio results in inferior results for many CF models. In this paper, we show that choosing a suitable loss function and a proper number of negative samples plays an equal or more important role than an interaction encoder. Towards this goal, we systematically compare multiple commonly-used loss functions and also investigate the impact of negative sampling ratio on each loss function. Moreover, inspired by the widely used

[*] Both authors contributed equally to the work.
[⋆] Corresponding Author.

*CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia*

The definitive Version of Record was published in *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia*, https://doi.org/10.1145/3459637.3482297.

contrastive loss [8, 38] in computer vision, we propose a cosine contrastive loss (CCL) tailored for CF. Our CCL loss optimizes the embedding by maximizing the cosine similarity of a positive user-item pair, while minimizing the similarity of a negative pair to a certain margin. Surprisingly, we found that even a simple model (e.g., MF), if paired with our proposed CCL loss, is sufficient to surpass many sophisticated state-of-the-art models.

These findings raise questions about whether the current baselines are strong enough to verify the performance improvements of the state-of-the-art CF models, and how much these sophisticated models have really improved. Our work aims to answer these questions. We argue that the current baselines might not be strong enough, which could mislead us to overestimate the real improvements of many new CF models. Instead of criticizing the contributions of any existing work, the main goal of our work is to build a simple and strong baseline model to foster future research on CF.

In the design of SimpleX, we keep simplicity in mind and borrow ideas from several existing studies (e.g., average pooling in YouTubeNet [5], attention in ACF [3]). We build Simplex as a unified model that integrates matrix factorization and user behaviour modeling. Specifically, it comprises a behavior aggregation layer (e.g., average pooling) to obtain a user's preference vector from the historically interacted items, and then fuses with the user embedding vector via a weighted sum. More importantly, SimpleX is optimized with our CCL loss and a large negative sampling ratio. Although the interaction encoder of SimpleX seems quite simple and might not be novel at all, we show that it could serve as a super-strong baseline model and have great potential for industrial applications because of its high efficiency.

For evaluation, we conduct comprehensive experiments on 11 benchmark datasets in total and compare with a total of 29 popular CF models of different types. The results show that SimpleX outperforms most sophisticated state-of-the-art methods by a large margin (up to 48.5% improvement in NDCG@20 over LightGCN [10] on Amazon-Books). We also empirically compare the performance of six representative loss functions and investigate the impact of different negative sampling ratios on each loss function, which demonstrates the superiority of our proposed CCL loss for CF tasks. Furthermore, we evaluate the efficiency of SimpleX, which shows more than 10x speedup over the simplified GNN-based CF model, LightGCN [10]. We hope that our work could not only serve as a simple and strong baseline to foster future research on CF, but also attract more research efforts towards the co-design of interaction encoders, loss functions, and negative sampling strategies.

The main contributions of our work are summarized as follows:

- We highlight the importances of loss functions and negative sampling in CF, and propose the cosine contrastive loss accordingly.
- We present a simple and strong baseline model, SimpleX, which could even attain much better performance than most sophisticated state-of-the-art models.
- We perform experiments on 11 benchmark datasets and compare SimpleX with 29 existing CF models to show its superiority in terms of both effectiveness and efficiency.

# 2 BACKGROUND AND RELATED WORK

In this section, we first give a formulation of collaborative filtering and point out three important aspects in CF modeling. We then summarize different categories of CF models.

## 2.1 Formulation of CF

The research of collaborative filtering includes implicit CF and explicit CF. Implicit CF models learn from implicit feedback data, e.g., click, visit, and purchase, while explicit CF models learn from explicit feedbacks such as ratings. In this work, we focus on implicit CF since it is more common in real recommendation scenarios. Besides, it is also easy to transform explicit feedback to implicit feedback via binarization. In implicit CF, a matrix $\mathbf{Y}$ is used to denote the user-item interactions, where $y_{ui} = 1$ if user $u$ has observed interaction with item $i$ and $y_{ui} = 0$ otherwise.

As mentioned in Section 1, we highlight three vital aspects that have a large impact to the learning process of CF models:

**(1) Interaction Encoder.** The function of the interaction encoder is to learn embeddings for each user and each item, which capture collaborative signals in the interaction matrix that reflect the behavioral similarity between users (or items). It is undoubtedly the core of CF models and has been well studied. We give a brief summary of interaction encoders in section 2.2.

**(2) Loss Function.** In general, there are two common types of loss functions in CF. Pointwise loss functions such as binary cross-entropy (BCE) and mean square error (MSE) treat the learning process as a binary classification or a regression task. Pairwise loss such as Bayesian personalized ranking loss (BPR) is optimized to make the similarities of positive user-item pairs larger than the negative ones.

**(3) Negative Sampling.** Since there are a lot of unobserved entries, in most cases we need to perform negative sampling to improve training efficiency. A few studies have been made to improve the uniform random sampling for recommendation, including mining informative negative samples (e.g., RNS [6], and NBPO [41]), tackling the selection bias of implicit user feedback (e.g., MSN [36]) and so on. In this work, we mainly investigate the influence of the negative sampling ratio. The existing studies are complementary to our work and potential to be applied to our SimpleX model for further improvement.

## 2.2 Summary of representative CF methods

We summarize representative CF methods into four categories:

**(1) MF-based methods.** Matrix factorization (MF) based algorithms decompose the user-item interaction matrix into two low-dimensional latent matrices for user and item representation. Due to its effectiveness, MF has been wildly studied in CF. Manotumruksa et al. proposed GRMF [19] that smoothed MF through adding the graph Laplacian regularizer to introduce graph information. Yang et al. devised a unified and efficient method called HOP-Rec [37] that incorporated both MF and graph-based models for implicit CF. Chen et al. designed ENMF [2], which is an efficient MF-based CF model with modified MSE loss function. It can be optimized efficiently without negative sampling for implicit feedback.

**(2) Autoencoder-based methods.** Autoencoder-based CF methods leverage the autoencoder network architectures to learn item embeddings. Such models are suitable to perform inductive recommendation, i.e., learning from one group of users while performing recommendation for another group of users with the same candidate items. For example, Liang et al. proposed Mult-VAE [16], which applied variational autoencoder (VAE) for CF. Ma et al. proposed MacridVAE [18] by disentangling user intents behind user-item and leveraging $\beta$-VAE to simulate the generative process of a user's personal history interactions. Steck et al. designed a linear model called EASE$^{R}$ [26] that is geared toward sparse data, in particular implicit feedback data, for the recommendation.

**(3) GNN-based methods.** Since the interaction data can be naturally modelled as a user-item bipartite graph, recent studies propose graph neural network (GNN) based CF models and report state-of-the-art performance. GNN-based methods model the recommendation as the link prediction task between user nodes and item nodes, where the higher-order collaborative signals can be effectively captured through multi-layers message passing. Ying et al. proposed PinSage [39] that improved GraphSage [9] to model the item-item relationships for Pinterest. Wang et al. devised NGCF [32] that explicitly encoded the collaborative signals as high-order connectivities by performing embedding propagation. He et al. proposed LightGCN [10], which removed the feature transformation and non-linear activation in NGCF and improved both performance and efficiency. These successful applications of GNN in recommendation further inspire many good studies, including BGCF [29] which models the uncertainty in the user-item graph with bayesian graph neural networks, DGCF [33] which models a distribution over intents for each user-item interaction, NIA-GCN [30] and NGAT4Rec [25] that learn neighborhood relationships, and SGL-ED [35], DHCF [14], LCFN [40], and so on.

**(4) Others.** We put methods that do not fall into the first three categories into this "Others" category. Here we list some representative models such as SLIM [21] which is a simple linear model that combines the advantages of neighborhood- and model-based CF approaches, MLPs-based NeuMF [11] and YouTubet [5], memory network-based CMN [7], metric learning-based CML [12], and NBPO [41] that leverages noisy-label robust learning techniques.

# 3 SIMPLEX

In this section, we first present our cosine contrastive loss and the SimpleX model architecture for CF. We then analyze its connections to other existing models.

## 3.1 Cosine Contrastive Loss

In the CF literature, many different loss functions have been employed, including BPR loss [22], binary cross-entropy [11], softmax cross-entropy [5], pairwise hinge loss [12], etc. However, there is still a lack of a systematic comparison among them, leaving their effects on model performance not well understood. In this work, we not only make such a comparison, but also propose a new loss function for CF, namely cosine contrastive loss (CCL). Given a positive user-item pair $(u, i)$ and a set of randomly sampled negative samples (i.e., $\mathcal{N}$), the CCL loss is expressed as follows:

$$\mathcal{L}_{CCL}(u,i) = (1-\hat{y}_{ui}) + \frac{w}{|\mathcal{N}|}\sum_{j\in\mathcal{N}} \max(0,\ \hat{y}_{uj} - m) \qquad (1)$$

where $\hat{y}_{ui}$ calculates the cosine similarity between the representation vectors of user $u$ and item $i$. $|\mathcal{N}|$ denotes the number of negative samples. $m$ is the margin to filter negative samples, which is usually set to 0~1. Intuitively, CCL is optimized to maximize the similarity between positive pairs and minimize the similarity of negative pairs below the margin $m$. $w$ is a hyper-parameter to control the relative weights of positive-sample loss and negative-sample loss.

**Design Choices**. The formulation of CCL is simple and largely inspired by the widely used contrastive loss [8, 38] in the computer vision tasks, such as face recognition and image retrieval. But we make several design choices that differ from most widely-used loss functions in CF and greatly facilitate model training. First, instead of applying dot product (e.g., in LightGCN [10]) or Euclidean distance (e.g., in CML [12]) to measure the similarity (or distance) between a user-item pair, we choose to compute the cosine similarity between them. By applying L2 normalization on both representation vectors, cosine similarity only calculates the angle difference and thus avoid the effect of representation magnitude. This is favorable since the magnitude of a user/item representation could be strongly biased by its popularity in CF tasks. This is also similar to the calculation of word similarity in Word2Vec [20], where cosine similarity is usually used.

Second, when the number of negative samples becomes large, there usually exist many redundant yet uninformative samples. But existing loss functions (e.g., BPR [22]) treat every negative sample equivalently. As such, model training could be overwhelmed by these uninformative samples, which significantly degrade the model performance and also slows the convergence. In contrast, CCL alleviates this problem by using a proper margin $m$ to filter uninformative negative samples. Intuitively, uninformative negative samples will get zero loss in CCL when they have a small cosine similarity below the margin $m$. As a result, it helps automatically identify those hard negative samples with cosine similarity larger than $m$, and thus facilitates better training of the model.

Third, we found that directly summing or averaging the loss terms of all negative samples could degrade the model performance, especially when the number of negative samples is large. This is partially due to the high imbalance between positive and negative samples (e.g., 1:1000 when $|\mathcal{N}| = 1000$). We thus introduce a data-dependent weight $w$ to control the balance between positive loss and negative loss. We emphasize that it also achieves a similar effect to the confidence weight imposed on negative samples in weighted matrix factorization [13].

## 3.2 Model Architecture

To leverage the advantages of CCL, we further propose a simple CF model, dubbed SimpleX. In the design of SimpleX, we keep simplicity in mind and borrow ideas from several successful models such as YouTubeNet [5], ACF [3], and PinSage [39].

Figure 1 illustrates the overall architecture of SimpleX. It largely follows the mechanism of MF, which factorizes users and items into a common latent space. Yet, SimpleX also takes the interacted item sequence of each user as additional input to better model user behaviors. This also has been shown effective in many existing studies, such as YouTubeNet [5] and ACF [3]. The key part of SimpleX lies in its aggregation layer for behavior sequence aggregation.

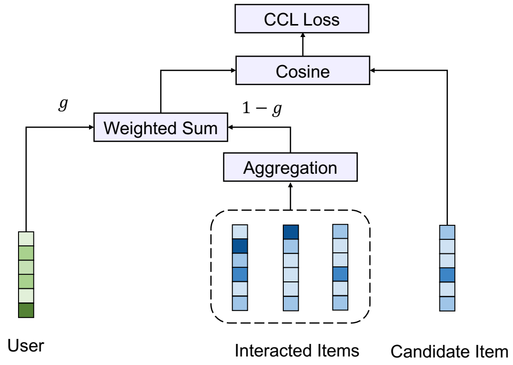

**Figure 1: The Simple Model Architecture of SimpleX.**

Here we introduce three common aggregation choices, including average pooling, self-attention, and user-attention, but Simplex is a unified architecture that any other aggregation method should also be applicable.

Suppose the historically interacted item set of user $u$ as $\mathcal{H}_u$, and we set its maximal size to $K$. For users with a different size of interacted items, either padding or chunking can be applied accordingly. As such, the aggregated vector can be obtained as follows:

$$p_u = \sum\nolimits_{k \in \mathcal{H}_u} I_k \cdot \alpha_k \boldsymbol{e}_k \ , \tag{2}$$

where $e_k \in \mathbb{R}^{d \times 1}$ is the $d$-dimensional embedding vector of item $k$. $I_k$ denotes the mask indices to $\mathcal{H}_u$ during padding, where $I_k = 0$ indicates a padding token; otherwise $I_k = 1$. $\alpha_k$ denotes the aggregation weight, which can be computed according to different aggregation types as follows.

$$\alpha_k = \begin{cases} \frac{I_k}{\sum_{k \in \mathcal{H}_u} I_k} \ , & \text{if average pooling,} \\ \frac{I_k \cdot exp(\beta_k)}{\sum_{j \in \mathcal{H}_u} I_k \cdot exp(\beta_j)} \ , & \text{if self-attention or user-attention.} \end{cases} \tag{3}$$

Average pooling provides a straightforward way to aggregate the interacted items, which has been successfully applied in YouTubeNet [5]. But it treats each item equally and fails to account for the relative importances of different items as well as a user's preference on each item. The attention mechanism, such as self-attention and user-attention, can be applied in such cases as calculated in the lower part of Equation 3. The difference between them lies in the computation of $\beta_k$, which is:

$$\beta_k = \begin{cases} q^T tanh(\boldsymbol{W}_1 e_k + b_1) \ , & \text{if self-attention,} \\ e_u^T tanh(\boldsymbol{W}_2 \times e_k + b_2) \ , & \text{if user-attention.} \end{cases} \tag{4}$$

where $q \in \mathbb{R}^{d \times 1}$ is a learnable global query vector for self-attention and $e_u$ is the user-specific query vector for user $u$ in user-attention. $\boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{d \times d}$ and $b_1, b_2 \in \mathbb{R}^{d \times 1}$ are learnable parameters. Note that similar attention mechanisms can be found in some existing work [3, 34].

However, after behavior aggregation via Equation 2, the pooling vector $p_u$ may lie in a different latent space with user vector $e_u$. We further fuse both parts to get the final user representation $h_u$:

$$h_u = g \cdot e_u + (1 - g) \cdot V p_u \tag{5}$$

where $\boldsymbol{V} \in \mathbb{R}^{d \times d}$ is a learnable parameter and $g$ is a hyperparameter weight. Finally, we measure the cosine similarity $\hat{y}_u i$ between user $u$ and item $i$ as the input to our CCL loss.

$$\hat{y}_{ui} = cos(h_u, e_i) \tag{6}$$

The above three aggregation layers provide different views for aggregation, including global-average view, global-weighed view and user-specific weighted view. The choice among them is quite data-dependent. In our experiment, we show that average-pooling is a robust aggregation method that always demands a first attempt when applying SimpleX. The other two usually needs more efforts to tune and in some cases brings marginal improvements.

## 3.3 Relationships to Existing Models

SimpleX is also related to multiple popular CF models.

- **MF**. MF is the most common model for CF. SimpleX follows the similar mechanism of MF. When setting g = 1 in SimpleX, it reduces to a MF model trained with CCL (i.e., MF-CCL).
- **YouTubeNet**. YouTubeNet is a successful model that has been widely used in industry. SimpleX can be also seen as a simplified YouTubeNet model (without using side features) when average pooling is employed. The only difference is that YouTubeNet employs concatenation instead of weighted sum to fuse $e_u$ and $p_u$. But the latter performs better in our experiments.
- **GNN-based models**. Simplex is also similar to GNN-based models. For instance, when choosing the user-attention aggregation layer, it almost equals to a graph attention (GAT) layer applied on user nodes only. If using the self-attention aggregation layer, it works like the neighbor interaction in NIA-GCN [30] as well.

We emphasize that although the design of SimpleX is simple and might not be novel to some extent, it unifies several key components in existing CF models. **Surprisingly, such a simple model is sufficient to surpass most state-of-the-art CF models by a large margin, which could serve as simple and strong baseline for future research**.

## 4 EXPERIMENTS

In this section, we conduct comprehensive experiments to evaluate SimpleX, including: 1) studying the impacts of loss functions and negative sampling ratios, 2) making performance comparisons to existing models on three main datasets, 3) incorporating CCL to other models, 4) performing parameter analysis and efficiency evaluation, 5) further validating SimpleX on some other datasets.

## 4.1 Experimental Setup

*4.1.1 Dataset.* We use 11 benchmark datasets in our study. For fairness and ease of comparison, we choose those open datasets that have been already split and preprocessed. Specifically:

(1) We employ three main datasets **Amazon-Books**, **Yelp2018**, and **Gowalla**, which are commonly used in recent GNN-based CF models [4, 10, 25, 32, 33, 35]. We perform most of our experiments on them and further make comparisons to these GNN-based models.

Table 1: Performance of MF under different loss functions. The best result in each column is marked in bold.

| Loss | AmazonBooks | | Yelp18 | | Gowalla | |
|---|---|---|---|---|---|---|
| | Recall@20 | NDCG@20 | Recall@20 | NDCG@20 | Recall@20 | NDCG@20 |
| BPR Loss | 0.0338 | 0.0261 | 0.0549 | 0.0445 | 0.1616 | 0.1366 |
| Pairwise Hinge Loss | 0.0352 | 0.0267 | 0.0562 | 0.0453 | 0.1318 | 0.0996 |
| Binary Cross-Entropy | 0.0479 | 0.0371 | 0.0617 | 0.0503 | 0.1321 | 0.1159 |
| Softmax Cross-Entropy | 0.0478 | 0.0367 | 0.0639 | 0.0522 | 0.1545 | 0.1276 |
| Mean Square Error | 0.0337 | 0.0267 | 0.0624 | 0.0513 | 0.1528 | 0.1315 |
| Cosine Contrastive Loss | **0.0559** | **0.0447** | **0.0698** | **0.0572** | **0.1837** | **0.1493** |

(2) To demonstrate the universality of SimpleX, we further test SimpleX on some other datasets adopted by studies published in top-tier conferences. Three of them, **Amazon-CDs**, **Amazon-Movies**, **Amazon-Beauty**, are adopted by the work NIA-GCN [30] and BGCF [29]. The other three, **Amazon-Electronics**, **CiteUlike-A**, and **Movielens-1M**, are provided by NBPO [41], DHCF [14], and LCFN [40], respectively. Specifically, we compare SimpleX with the corresponding models on the corresponding datasets that adopted in their original papers. For example, we will compare with DHCF [14] on CiteUlike-A dataset because DHCF adopts this dataset in their original paper.

(3) The last two are **Movielens-20M** and **MillionSongData**, which are commonly used by autoencoder-based CF models, such as Mult-VAE [16] and RecVAE [23]. We follow the strong generalization setting, which split train/validation/test sets with different sets of users, and specially make comparison with those autoencoder-based CF models to further demonstrate the effectiveness of SimpleX.

*4.1.2 Compared Methods.* We compare SimpleX with 29 existing CF models of different types:

- Five methods based on MF and its variants, including MF-BPR [15], GRMF [19], HOP-Rec [37], NeuMF [11], and ENMF [2];
- Four autoencoder-based methods, including Mult-VAE [16], MacridVAE [18], EASE$^R$ [26], and RecVAE [23];
- Fourteen GNN-based methods, including GC-MC [1], Pinsage [39], GAT [31], NGCF [32], DisenGCN [17], LR-GCCF [4], NIA-GCN [30], LightGCN [10], DGCF [33], NGAT4Rec [25], SGL-ED [35], BGCF [29], DHCF [14], and LCFN [40];
- Six methods of other types, including ItemPop, SLIM [21], CML [12], YouTubeNet [5], CMN [7], and NBPO [41].

*4.1.3 Implementation Details.* We implement SimpleX in PyTorch. Specifically, we set the batch size to 1024 by default. We use the Adam optimizer and tune the learning rate among [1e-3, 5e-4, 1e-4]. We also employ $L_2$ regularization on the embedding parameters and search the regularization weight between 1e-9∼1e-2 with an increase ratio of 5. For cosine contrastive loss, we search the number of negative samples from 1 to 2000. In many cases, we pick 100, 500, or 1000. The margin $m$ is tuned among 0∼1 at an interval of 0.1, for example, we set 0.4, 0.9, and 0.9 on Amazon-Books, Yelp2018, and Gowalla, respectively. Meanwhile, we use the same embedding size with the compared model, for example, 64 in LightGCN and 128 in LCFN. For fairness of comparison with existing models, we report the results using the same evaluation metrics (e.g., Recall@20 and NDCG@20) and duplicate the reported results on their papers for consistency. To facilitate reproducible research in the community, we have contributed our source code and detailed benchmark settings to the public Open-CF-Benchmark[1].

## 4.2 Impact of Different Loss Functions

While most studies focus on the interaction encoder design, they neglect the importance of loss functions in the learning of a CF model. We make a systematic comparison on the impacts of different loss functions. For this purpose, we choose one of the simplest baseline CF models, i.e., MF, as the backbone to perform the experiments, since simple models tend to be more illustrative. In addition to our CCL loss, we evaluate MF on the following representative loss functions:

- **Bayesian personalized ranking (BPR)** loss encourages the similarity of a positive user-item pair to be higher than that of each negative user-item pair. It is one of the most commonly used loss function for CF research [10, 22, 32, 33].
- **Pairwise hinge loss (PHL)**, is also known as max-margin objective, which has been used in CML [12]. PHL forces the distance of a negative user-item pair to be larger than a positive one by at least the marginal distance.
- **Binary cross-entropy (BCE)** loss is commonly used for binary classification, which has been adopted in the early work NeuMF [11].
- **Softmax cross-entropy (SCE)** loss is widely used for multi-class classification. YouTubeNet [5] cast item prediction as a multi-class classification task through the SCE loss.
- **Mean square error (MSE)** has been widely used for CF, such as WMF [13] and ENMF [2].

Table 1 shows the results of training MF with different loss functions on Amazon-Books, Yelp2018, and Gowalla. Note that every model has been trained with enough epochs to reach convergence and the best results are reported. From the results, we have the following observations:

1) CCL consistently achieves the best performance on all the three datasets, outperforming the other loss functions by at least 16.7%, 9.2% and 13.7% w.r.t. Recall@20 on Amazon-Books, Yelp2018 and Gowalla, respectively.

2) BPR only appears to be strong on Gowalla and performs not well on both Amazon-Books and Yelp2018. This demonstrates that using BPR for training is probably sub-optimal, and thus the results reported by many previous papers may need careful re-examination and are likely to be further improved with our CCL loss.

[1] https://openbenchmark.github.io/collaborative-filtering

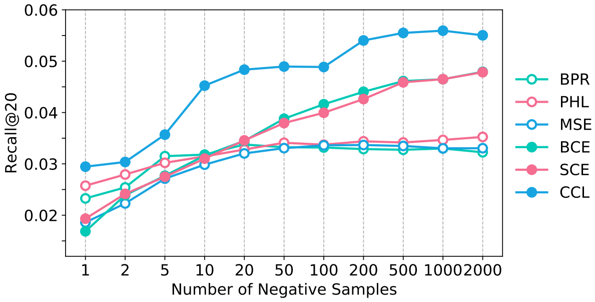

**Figure 2: The effect of number of negative samples across different loss functions on Amazon-Books.**

**Why CCL performs better than the other loss functions?** In addition to the design choices analyzed in Section 3.1, we further highlight the advantages of CCL with some concrete comparisons. First, in contrast to BPR, BCE, SCE, and MSE, CCL can automatically filter out hard negative samples that are hard to distinguish (i.e., large cosine similarity) by the model via its margin mechanism. For example, if we set $m = 0.8$, only those negative pairs with $\hat{y}_{ui} > 0.8$ will contribute to the loss. Different from the above loss functions that treat each negative sample equally, CCL allows the model to emphasize on the learning of hard negative samples and thus generate more discriminative representations. Second, compared with PHL that also applies a margin mechanism, CCL is more effective for CF. The PHL loss is determined by the relative distance between positive samples and negative samples. Even if a negative sample is actually hard to be distinguished (e.g., $\hat{y}_{ui}^{neg} = 0.8$), it will not contribute to learning if the corresponding positive sample has $\hat{y}_{ui}^{pos} = 0.9$. CCL avoids such ambiguity by penalizing the absolute similarity $\hat{y}_{ui}$ of each negative sample.

## 4.3 Impact of Negative Sampling Ratio

We argue that negative sampling ratio is also important in the learning of CF models, which has been largely ignored by existing studies. To support our claims, we compare the performance of MF trained with 1~2000 negative samples on Amazon-Books. We also repeat the experiment on different loss functions. We train each model until convergence and report the best results, as shown in Figure 2. We have the following observations from the results:

1) The number of negative samples does matter for CF model training. Generally, increasing it within a certain range leads to improvements. This suggests that we should carefully consider the impact of the number of negative samples in the evaluation.

2) MF trained with CCL is consistently better than training with the other loss functions under different negative sampling ratios, further demonstrating the superiority of our CCL.

3) The performances of PHL, MSE, and BPR become stable when the number of negative samples increases to 50. In contrast, CCL, BCE, and SCE can keep performance gains with the increase of number of negative samples, even when it reaches to 1000.

In summary, our experimental results show that both loss functions and negative sampling ratios can have a large impact on model performance. Training with the CCL loss and a large negative sampling ratio appears to be a promising setting for CF methods to gain higher performance. We therefore call for more future research towards this direction.

## 4.4 Performance Comparison to SOTA Models

In this section, we provide a comprehensive comparison results of SimpleX and other 23 CF models on three main datasets, i.e., Amazon-Books, Yelp2018, and Gowalla, which are very commonly adopted in CF studies (especially in GNN-based CF), to demonstrate the superiority of SimpleX. Table 2 shows our performance comparisons on Amazon-Books, Yelp2018, and Gowalla under the same evaluation protocol, and we have the following observations:

1) Our SimpleX achieves the best overall performance on all the three datasets. In particular, compared with the most recent LightGCN, SimpleX makes 41.9%, 8.0%, and 2.3% performance improvements on Recall@20 for Amazon-Books, Yelp2018, and Gowalla, respectively, demonstrating the high effectiveness of SimpleX. Besides, note that we do not report the results of SGL-ED [35] and NGAT4Rec [25] on Gowalla since they are not evaluated on Gowalla but only evaluated on the other two datasets in their original papers too, and the authors have not released their code. As the experimental settings of SGL-ED and NGAT4Rec are exactly same as us, we just report their results on Amazon-Books and Yelp2018.

2) The performance of MF-CCL is surprising. When using CCL as the loss function, the performance of MF is not only much better than the results of MF-BPR reported in the previous paper, but also reaches a new state-of-the-art performance (if leaving out our SimpleX) on Amazon-Books and Yelp2018. On Gowalla, it also achieves comparable performance compared to the previous best model DGCF. Such results strongly suggest that loss functions can make a big difference and should be carefully chosen and studied.

3) YouTubeNet, CML, and SLIM are three models that we added and have not been tested on these three datasets before by the existing work. We found that they achieve pretty good performance. Specifically, these three models can averagely outperform a representative GNN-based CF model – NGCF, by more than 24% and 28% w.r.t. Recall@20 and NDCG@20, respectively. This implies that the current baselines are relatively weak, which may lead us to overestimate how much real progress we have made in CF.

4) In CF tasks, more complex models not always lead to better performance. The designs of SLIM, YouTubeNet, CML, MF-CCL, and our SimpleX are all much more concise than most of autoencoder-based (e.g., Mult-VAE and MacridVAE) and GNN-based models (e.g., NGCF, NIA-GCN, and DGCF), but they can achieves better performance. This also reveals that the current trend in CF research, which pays too much attention to the design of sophisticated interaction encoders while ignoring the impacts of loss functions and negative sampling, needs to be improved.

## 4.5 Incorporating CCL to Other Models

In Table 2, we have shown that one of the simplest models, i.e., MF, can even largely outperforms most of state-of-the-art models if training with CCL. We are curious about how other models will perform if incorporated with CCL instead of their original losses. Therefore, in this part, we take experiments with two effective CF models in addition, i.e., YouTubeNet and LightGCN with CCL, and report the results on Amazon-Books and Yelp2018 in Table 3.

**Table 2: Performance comparison to popular CF models. We highlight the top-5 best results in each column. RI stands for relative improvement. We also report the average RI over NGCF for each model if applicable. For models marked with *, part of the results are duplicated from existing papers for consistency.**

| Publication | Model | Amazon-Books | | Yelp2018 | | Gowalla | | Avg RI over NGCF | |
|---|---|---|---|---|---|---|---|---|---|
| | | Recall@20 | NDCG@20 | Recall@20 | NDCG@20 | Recall@20 | NDCG@20 | Recall@20 | NDCG@20 |
| – | ItemPop | 0.0051 | 0.0044 | 0.0124 | 0.0101 | 0.0416 | 0.0317 | – | – |
| UAI'2009 | MF-BPR | 0.0338 | 0.0261 | 0.0576 | 0.0468 | 0.1627 | 0.1378 | – | – |
| NIPS'2015 | GRMF* | 0.0354 | 0.0270 | 0.0571 | 0.0462 | 0.1477 | 0.1205 | – | – |
| RecSys'2016 | YouTubeNet | **0.0502(4)** | **0.0388(4)** | **0.0686(3)** | **0.0567(3)** | **0.1754(5)** | **0.1473(5)** | 32.2% | 33.3% |
| WWW'2017 | NeuMF* | 0.0258 | 0.0200 | 0.0451 | 0.0363 | 0.1399 | 0.1212 | – | – |
| WWW'2017 | CML | **0.0522(3)** | **0.0428(3)** | 0.0622 | 0.0536 | 0.1670 | 0.1292 | 29.6% | 37.6% |
| SIGIR'2018 | CMN* | 0.0267 | 0.0218 | 0.0457 | 0.0369 | 0.1405 | 0.1221 | – | – |
| RecSys'2018 | HOP-Rec* | 0.0309 | 0.0232 | 0.0517 | 0.0428 | 0.1399 | 0.1214 | – | – |
| WWW'2018 | Mult-VAE* | 0.0407 | 0.0315 | 0.0584 | 0.0450 | 0.1641 | 0.1335 | 9.6% | 7.1% |
| NeurIPS'2019 | MacridVAE* | 0.0383 | 0.0295 | 0.0612 | 0.0495 | 0.1618 | 0.1202 | 8.5% | 8.0% |
| TOIS'2020 | ENMF | 0.0359 | 0.0281 | 0.0624 | 0.0515 | 0.1523 | 0.1315 | 6.1% | 7.4% |
| **GNN-based Models** | | | | | | | | | |
| KDDW'2018 | GC-MC* | 0.0288 | 0.0224 | 0.0462 | 0.0379 | 0.1395 | 0.1204 | – | – |
| KDD'2018 | PinSage* | 0.0282 | 0.0219 | 0.0471 | 0.0393 | 0.1380 | 0.1196 | – | – |
| ICLR'2018 | GAT* | 0.0326 | 0.0235 | 0.0543 | 0.0431 | 0.1401 | 0.1236 | – | – |
| SIGIR'2019 | NGCF* | 0.0344 | 0.0263 | 0.0579 | 0.0477 | 0.1570 | 0.1327 | – | – |
| ICML'2019 | DisenGCN* | 0.0329 | 0.0254 | 0.0558 | 0.0454 | 0.1356 | 0.1174 | – | – |
| AAAI'2020 | LR-GCCF | 0.0335 | 0.0265 | 0.0561 | 0.0343 | 0.1519 | 0.1285 | – | – |
| SIGIR'2020 | NIA-GCN* | 0.0369 | 0.0287 | 0.0599 | 0.0491 | 0.1359 | 0.1106 | 6.9% | 4.8% |
| SIGIR'2020 | LightGCN* | 0.0411 | 0.0315 | 0.0649 | 0.0530 | **0.1830(4)** | **0.1554(3)** | 15.8% | 15.4% |
| SIGIR'2020 | DGCF* | 0.0422 | 0.0324 | 0.0654 | 0.0534 | **0.1842(2)** | **0.1561(1)** | 17.8% | 17.6% |
| Arxiv'2020 | NGAT4Rec* | 0.0457 | 0.0358 | **0.0675(4)** | **0.0554(5)** | – | – | 24.7% | 26.1% |
| SIGIR'2021 | SGL-ED* | **0.0478(5)** | **0.0379(5)** | **0.0675(4)** | **0.0555(4)** | – | – | 27.8% | 30.2% |
| **Ours** | | | | | | | | | |
| CIKM'2021 | MF-CCL | **0.0559(2)** | **0.0447(2)** | **0.0698(2)** | **0.0572(2)** | **0.1837(3)** | **0.1493(4)** | 41.6% | 45.0% |
| CIKM'2021 | SimpleX | **0.0583(1)** | **0.0468(1)** | **0.0701(1)** | **0.0575(1)** | **0.1872(1)** | **0.1557(2)** | 45.3% | 49.2% |
| RI over NGCF | | 69.6% | 77.9% | 21.1% | 20.6% | 19.2% | 17.3% | | |
| RI over LighGCN | | 41.9% | 48.5% | 8.0% | 8.5% | 2.3% | 0.2% | | |

From the results, we find that training YouTubeNet and LightGCN with CCL instead of their original loss functions, i.e, SCE and BPR respectively, can bring good improvements. This demonstrates that CCL is likely to be a more promising loss function to help CF models achieve better performance. Besides, we observe that the improvements brought by CCL on YouTubeNet and LightGCN are not as significant as those on MF. CCL seems to improve these models to a similar level of performance. This may be because of the following reason: Generally, valuable collaborative information can be captured by both the interaction encoder and the loss function. As the encoders of YouTubeNet and LightGCN are sophisticated and stronger to learn biased collaborative signals, by contrast, the impact of the loss function to them appears relatively small.

In addition, it is worth noting that our main focus is to question the value of sophisticated encoders and provide a simple strong baseline, but not to improve current state-of-the-art CF models by exhaustingly trying of various loss functions. Based on the experiments with MF, YouTubeNet, and LightGCN, we demonstrate and highlight that the loss function is a large bottleneck in CF models. We expect our work could inspire more research to study the co-design of the interaction encoder, loss function, and negative sampling.

**Table 3: Performance of different models trained with CCL v.s. their original losses.**

| Model | Amazon-Books | | Yelp2018 | |
|---|---|---|---|---|
| | Recall@20 | NDCG@20 | Recall@20 | NDCG@20 |
| MF-BPR | 0.0338 | 0.0261 | 0.0549 | 0.0445 |
| MF-CCL | **0.0559** | **0.0447** | **0.0698** | **0.0572** |
| YouTubeNet | 0.0502 | 0.0388 | 0.0655 | 0.0537 |
| YouTubeNet-CCL | **0.0544** | **0.0430** | **0.0685** | **0.0563** |
| LightGCN | 0.0411 | 0.0315 | 0.0649 | 0.0530 |
| LightGCN-CCL | **0.0528** | **0.0416** | **0.0669** | **0.0554** |

## 4.6 Parameter Analysis on SimpleX

We investigate the performance of three different behavior aggregation layers, the fusing weight $g$, and the negative loss weight $w$. Results on Amazon-Books and Yelp2018 are shown in Table 4. We can make the following observations: 1) Average pooling, self-attention,

Table 4: Parameter analysis results on SimpleX.

| Ablations | Amazon-Books | | Yelp2018 | |
|---|---|---|---|---|
| | Recall@20 | NDCG@20 | Recall@20 | NDCG@20 |
| avg_pooling | **0.0583** | **0.0468** | **0.0701** | 0.0575 |
| self_attn. | 0.0580 | 0.0462 | 0.0698 | **0.0576** |
| user_attn. | 0.0551 | 0.0436 | 0.0698 | 0.0574 |
| g = 0 | 0.0534 | 0.0429 | 0.0679 | 0.0555 |
| g = 0.5 | **0.0583** | **0.0468** | 0.0688 | 0.0565 |
| g = 1 | 0.0540 | 0.0432 | **0.0701** | **0.0575** |
| $w$ = 1 | 0.0163 | 0.0128 | 0.0238 | 0.0189 |
| $w$ = 150 | 0.0542 | 0.0429 | **0.0701** | **0.0575** |
| $w$ = 300 | **0.0583** | **0.0468** | 0.0666 | 0.0549 |
| $w$ = 1000 | 0.0481 | 0.0379 | 0.0568 | 0.0463 |

Table 5: Efficiency comparison on Amazon-Books, w.r.t. the average training time per epoch, the number of epochs to converge, and the total training time.

| Model | Time/Epoch | #Epochs | Training Time |
|---|---|---|---|
| ENMF | 129s | 81 | 2h54m |
| LightGCN | 51s | 780 | 11h3m |
| SimpleX ($|\mathcal{N}|$=100) | **40s** | **28** | **19m** |
| SimpleX ($|\mathcal{N}|$=1000) | 131s | 35 | 1h16m |

and user-attention obtain very similar results on Amazon-Books and Yelp2018, respectively. This shows the robustness of apply average pooling for behavior aggregation in practice. SimpleX with $g = 0.5$ reaches higher performance compared with the other two settings on Amazon-Books, which shows that importance of fusing user embedding with user behavior aggregation. 2) The negative weight $w$ which adjusts the ratio of positive and negative losses is vital to model's performance. In general, too small ($w = 1$) or too large ($w = 1000$) difference between positive and negative losses leads to performance reduction.

## 4.7 Efficiency Comparison

Our SimpleX has high efficiency due to its simple design. We numerically compare the training time of SimpleX with two state-of-the-art CF models, i.e., ENMF and LightGCN, which are relatively efficient in their respective categories, on Amazon-Books. The efficiency experiments are conducted on the same Intel(R) Xeon(R) Silver 4210 CPU @2.20GHz machine with one GeForce RTX 2080 GPU. We compare them under the same implementation framework, using the same acceleration methods (e.g., implementing the sampling with C++) to ensure fairness. Specifically, we present the averaged training time per epoch, the number of epochs that the model needs to reach the level of performance reported in the original paper, and the total training time (test time is not included), in Table 5.

It turns out that SimpleX is much more efficient than ENMF and LightGCN overall. Specifically, SimpleX only needs around 30 epochs to converge in training, which is more convenient for real application. The total training time of SimpleX with a 1000:1 negative sampling ratio has around 2x and 10x speedup compared with ENMF and LightGCN respectively. Moreover, if we decrease the negative sampling ratio to 100:1, the training time for one epoch of SimpleX can be optimized to 40s, finally resulting in only 19 minutes total training time. Certainly, the performance slightly drops compared with using a 1000:1 negative sampling ratio, but it still maintains a pretty good level (much better than ENMF and LightGCN). Such high efficiency makes our model promising to be applied in large-scale real recommender systems.

## 4.8 Evaluating SimpleX on More Datasets

In addition to the three main datasets used in the above sub-sections, we additionally evaluate SimpleX on 8 more datasets to further demonstrate the generability of SimpleX.

Table 6 shows the comparison results to some state-of-the-art CF models published in 2020. For fairness of comparison, we use the same data preprocessing and experimental settings (embedding dimensions and evaluation metrics) provided by the corresponding papers. We observed that SimpleX consistently outperforms all the compared models on different datasets. The performance improvements are especially large (12.8% to 33.6% improvement in NDCG@20) on Amazon-CDs, Amazon-Movies and Amazon-Beauty compared to BGCF, a recent GNN-based model. This again strongly verifies the effectiveness and robustness of SimpleX to serve as a strong baseline in future work.

Moreover, we also make a comparison to some autoencoder-based models, including SLIM, Mult-VAE, EASE$^{\mathrm{R}}$, and RecVAE. It is worth noting that our experiment also follows the same setting with them. In particular, we adopt the strong generalization protocol, where the training, validation and test sets are disjoint in terms of users. This requires the model to perform inductive learning during inference. That is, only item embeddings can be learned during training and then transferred to the validation and test sets for prediction. To achieve this, we simplify SimpleX by setting $g = 0$ in this experiment and only learn user representations from their historically interacted items.

Table 7 presents the evaluation results on Movielens-20M and MillionSongData. We can see that SimpleX obtains better performance than SLIM, which is a well-known strong baseline for CF. But it does not surpass Mult-VAE, EASE$^{\mathrm{R}}$ and RecVAE given their complete forms. This is reasonable because all of them use many more parameters ($O(|I| * 600)$ for Mult-VAE and RecVAE, $O(|I| * |I|)$ for EASE$^{\mathrm{R}}$) than SimpleX, as shown in the "#Params" columns. Note that both Mult-VAE and RecVAE use 600 as the dimension of the first hidden layer. As the number of items ($|I|$) easily reaches millions to billions in industrial recommender systems, we choose a small embedding dimension (i.e., 64) and results in parameters in the scale of $O(|I| * 64)$. To make the comparison more fair, we reduce the embedding dimensions of baseline models accordingly. Specifically, for Mult-VAE and RecVAE, we set its encoder and decoder as a single ($|I| * 64$)-dimensional dense layer. For EASE$^{\mathrm{R}}$, we decompose its item similarity matrix (denoted as B) to two ($|I| * 64$)-dimensional sub-matrices by truncated SVD, and multiply the two sub-matrices to approximate the item similarity matrix to perform predictions. In this setting, SimpleX clearly outperform these autoencoder based CF models.

Overall, our comprehensive experimental results on various datasets show that our SimpleX is simple and strong to serve as a

Table 6: Performance comparison to some more models published in 2020. We use the same dataset settings and report the same metrics with the original papers. We also duplicate their reported results for consistency.

| Amazon-CDs | | | Amazon-Movies | | | Amazon-Beauty | | |
|---|---|---|---|---|---|---|---|---|
| Model | Recall@20 | NDCG@20 | Model | Recall@20 | NDCG@20 | Model | Recall@20 | NDCG@20 |
| NGCF | 0.1258 | 0.0792 | NGCF | 0.0866 | 0.0555 | MF-BPR | 0.1312 | 0.0778 |
| NIA-GCN | 0.1487 | 0.0932 | NIA-GCN | 0.1058 | 0.0683 | NGCF | 0.1513 | 0.0917 |
| BGCF | 0.1506 | 0.0948 | BGCF | 0.1066 | 0.0693 | BGCF | 0.1534 | 0.0912 |
| SimpleX | **0.1763** | **0.1145** | SimpleX | **0.1342** | **0.0926** | SimpleX | **0.1721** | **0.1028** |
| RI over NIA-GCN | 18.6% | 22.9% | RI over NIA-GCN | 26.8% | 35.5% | – | – | – |
| RI over BGCF | 17.1% | 20.8% | RI over BGCF | 25.9% | 33.6% | RI over BGCF | 12.2% | 12.8% |

| Amazon-Electronics | | | CiteUlike-A | | | Movielens-1M | | |
|---|---|---|---|---|---|---|---|---|
| Model | F1@20 | NDCG@20 | Model | Precision@20 | Recall@20 | Model | F1@20 | NDCG@20 |
| MF-BPR | 0.0275 | 0.0680 | NGCF | 0.0517 | 0.0193 | NGCF | 0.1582 | 0.2511 |
| NBPO | 0.0313 | 0.0810 | DHCF | 0.0635 | 0.0249 | LCFN | 0.1625 | 0.2603 |
| SimpleX | **0.0338** | **0.0842** | SimpleX | **0.0754** | **0.0269** | SimpleX | **0.1658** | **0.2670** |
| RI over NBPO | 8.0% | 4.0% | RI over DHCF | 18.7% | 8.2% | RI over LCFN | 2.0% | 2.6% |

Table 7: Performence comparison to autoenoder-based models on Movielens-20M and MillionSongData.

| Model | Movielens-20M | | | | MillionSongData | | | |
|---|---|---|---|---|---|---|---|---|
| | Recall@20 | Recall@50 | NDCG@100 | #Params | Recall@20 | Recall@50 | NDCG@100 | #Params |
| SLIM | 0.370 | 0.495 | 0.401 | – | – | – | – | – |
| Mult-VAE | 0.395 | 0.537 | 0.426 | 24.5M | 0.266 | 0.363 | 0.313 | 49.7M |
| EASE$^{R}$ | 0.391 | 0.521 | 0.420 | 404.3M | 0.333 | 0.428 | 0.389 | 1,692M |
| RecVAE | 0.414 | 0.553 | 0.442 | 16.5 M | 0.276 | 0.374 | 0.326 | 33.3M |
| Mult-VAE (d=64) | 0.375 | 0.514 | 0.407 | 2.6M | 0.230 | 0.319 | 0.280 | 5.3M |
| EASE$^{R}$ (d=64) | 0.361 | 0.487 | 0.392 | 2.6M | 0.170 | 0.235 | 0.205 | 5.3M |
| RecVAE (d=64) | 0.385 | 0.520 | 0.412 | 2.6M | 0.232 | 0.319 | 0.280 | 5.3M |
| SimpleX (d=64) | **0.389** | **0.523** | **0.416** | 1.3M | **0.245** | **0.329** | **0.293** | 2.6M |
| RI over Mult-VAE | 3.8% | 1.7% | 2.3% | | 6.5% | 3.2% | 4.7% | |
| RI over EASE$^{R}$ | 7.8% | 7.4% | 6.2% | | 44.0% | 40.3% | 43.3% | |
| RI over RecVAE | 1.1% | 0.6% | 1.1% | | 5.8% | 3.2% | 4.8% | |

new baseline model to facilitate future research on CF. The availability of this baseline would allow for more solid experimental evaluations and more fair comparisons among CF models.

## 5 CONCLUSION

In this paper, we study the progress made in CF research and identify three key aspects for CF modeling. While most research focuses on interaction encoders, the impacts of loss functions and negative sampling on CF models have been largely neglected. In this work, we highlight their impacts and further propose the cosine contrastive loss together with a simple and strong baseline for CF, dubbed SimpleX. It outperforms most state-of-the-art CF models by a large margin. Our work released the simple and strong baseline model and the whole benchmarking results for foster future research on CF. We conduct extensive experiments to validate the effectiveness and efficiency of SimpleX. We suggest that the CF community should pay more attention to other key components in addition to interaction encoders and encourage researchers to conduct more robust empirical evaluation.

## 6 ACKNOWLEDGEMENTS

This work was supported in part by the National Natural Science Foundation of China (61972219), the Research and Development Program of Shenzhen (JCYJ20190813174403598, SGDX20190918101201-696), the National Key Research and Development Program of China (2018YFB1800601), and the Overseas Research Cooperation Fund of Tsinghua Shenzhen International Graduate School (HW2021013).

## REFERENCES

[1] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2018. Graph Convolutional Matrix Completion. In *KDD'18 Deep Learning Day*.

[2] Chong Chen, Min Zhang, Yongfeng Zhang, Yiqun Liu, and Shaoping Ma. 2020. Efficient Neural Matrix Factorization without Sampling for Recommendation. *ACM Transactions on Information Systems (TOIS)* 38, 2 (2020), 1–28.

[3] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive Collaborative Filtering: Multimedia Recommendation

with Item- and Component-Level Attention. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 335–344.

[4] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 27–34.

[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM conference on Recommender Systems (RecSys)*. 191–198.

[6] Jingtao Ding, Yuhan Quan, Xiangnan He, Yong Li, and Depeng Jin. 2019. Reinforced Negative Sampling for Recommendation with Exposure Data. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*. 2230–2236.

[7] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative Memory Network for Recommendation Systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR)*. 515–524.

[8] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality Reduction by Learning an Invariant Mapping. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 1735–1742.

[9] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*. 1024–1034.

[10] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval (SIGIR)*. 639–648.

[11] 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 (WWW)*. 173–182.

[12] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative Metric Learning. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*. 193–201.

[13] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM)*. 263–272.

[14] Shuyi Ji, Yifan Feng, Rongrong Ji, Xibin Zhao, Wanwan Tang, and Yue Gao. 2020. Dual Channel Hypergraph Collaborative Filtering. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 2020–2029.

[15] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[16] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of the 2018 World Wide Web Conference (WWW)*. 689–698.

[17] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. 2019. Disentangled Graph Convolutional Networks. In *International Conference on Machine Learning (ICML)*. 4212–4221.

[18] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. 2019. Learning Disentangled Representations for Recommendation. In *Advances in Neural Information Processing Systems (NeurIPS)*. 5711–5722.

[19] Jarana Manotumruksa, Craig Macdonald, and Iadh Ounis. 2017. A Deep Recurrent Collaborative Filtering Framework for Venue Recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM)*. 1429–1438.

[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems (NeurIPS)*. 3111–3119.

[21] Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *IEEE 11th International Conference on Data Mining (ICDM)*. 497–506.

[22] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*. 452–461.

[23] Ilya Shenbin, Anton Alekseev, Elena Tutubalina, Valentin Malykh, and Sergey I. Nikolenko. 2020. RecVAE: A New Variational Autoencoder for Top-N Recommendations with Implicit Feedback. In *The Thirteenth ACM International Conference on Web Search and Data Mining (WSDM)*. 528–536.

[24] Brent Smith and Greg Linden. 2017. Two Decades of Recommender Systems at Amazon.com. *IEEE Internet Comput.* 21, 3 (2017), 12–18.

[25] Jinbo Song, Chao Chang, Fei Sun, Xinbo Song, and Peng Jiang. 2020. NGAT4Rec: Neighbor-Aware Graph Attention Network For Recommendation. *arXiv preprint arXiv:2010.12256* (2020).

[26] Harald Steck. 2019. Embarrassingly Shallow Autoencoders for Sparse Data. In *The World Wide Web Conference (WWW)*. 3251–3257.

[27] Xiaoyuan Su and Taghi M. Khoshgoftaar. 2009. A Survey of Collaborative Filtering Techniques. *Adv. Artif. Intell.* 2009 (2009), 421425:1–421425:19.

[28] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*. 1441–1450.

[29] Jianing Sun, Wei Guo, Dengcheng Zhang, Yingxue Zhang, Florence Regol, Yaochen Hu, Huifeng Guo, Ruiming Tang, Han Yuan, Xiuqiang He, et al. 2020. A Framework for Recommending Accurate and Diverse Items Using Bayesian Graph Convolutional Neural Networks. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 2030–2039.

[30] Jianing Sun, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, Xiuqiang He, Chen Ma, and Mark Coates. 2020. Neighbor Interaction Aware Graph Convolution Networks for Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 1289–1298.

[31] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations (ICLR)*.

[32] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 165–174.

[33] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled Graph Collaborative Filtering. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 1001–1010.

[34] Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, and Xing Xie. 2019. NPA: Neural News Recommendation with Personalized Attention. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 2576–2584.

[35] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised Graph Learning for Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 726–735.

[36] Ji Yang, Xinyang Yi, Derek Zhiyuan Cheng, Lichan Hong, Yang Li, Simon Xiaoming Wang, Taibai Xu, and Ed H Chi. 2020. Mixed Negative Sampling for Learning Two-tower Neural Networks in Recommendations. In *Companion Proceedings of the Web Conference (WWW)*. 441–447.

[37] Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, and Ming-Feng Tsai. 2018. HOP-Rec: High-order Proximity for Implicit Recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys)*. 140–144.

[38] Shuo Yang, Wei Yu, Ying Zheng, Hongxun Yao, and Tao Mei. 2019. Adaptive Semantic-Visual Tree for Hierarchical Embeddings. In *Proceedings of the 27th ACM International Conference on Multimedia (MM)*. 2097–2105.

[39] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-scale Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 974–983.

[40] Wenhui Yu and Zheng Qin. 2020. Graph Convolutional Network for Recommendation with Low-pass Collaborative Filters. In *International Conference on Machine Learning (ICML)*. 10936–10945.

[41] Wenhui Yu and Zheng Qin. 2020. Sampler Design for Implicit Feedback Data by Noisy-label Robust Learning. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 861–870.