Self-supervised Learning for Deep Models in Recommendations

Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, Felix Yu, Ting Chen, Aditya Menon
Lichan Hong, Ed H. Chi, Steve Tjoa, Jieqi (Jay) Kang, Evan Ettinger
{tyao,xinyang,zcheng,felixyu,iamtingchen,adityakmenon}@google.com
{lichan,edchi,stevetjoa,jaykang,eettinger}@google.com
Google Inc.

ABSTRACT

Large scale neural recommender models play a critical role in modern search and recommendation systems. To model large-vocab categorical features, typical recommender models learn a joint embedding space for both queries and items. With millions to billions of items to choose from, the quality of learned embedding representations is crucial to provide high quality recommendations to users with various interests. Inspired by the recent success in self-supervised representation learning (SSL) research in both computer vision and natural language understanding, we propose a multi-task self-supervised learning framework for neural models in recommendations. Furthermore, we propose two self-supervised tasks applicable to models with categorical features within the proposed framework: (i) Feature Masking (FM) and (ii) Feature Dropout (FD). We evaluate our framework using two large-scale datasets with 500M and 1B training examples respectively. Our results demonstrate that the proposed framework outperforms learning with the supervision task only and other state-of-the-art regularization techniques in the context of retrieval. The SSL framework shows larger improvement with less supervision compared to the counterparts.

1 INTRODUCTION

In the last few years, neural network based models have emerged to the main stage of modern recommendation systems throughout the industry (see, e.g., [1, 6, 9, 18, 32, 41, 44]), and academia ([7, 33]). Compared to conventional approaches like matrix factorization [2, 20, 23, 24], gradient boosted decision trees [4, 21, 30], and logistic regression based recommenders [19], these deep models handle categorical features more effectively, enable more complex data representations, and introduce more non-linearity to better fit the complex data for recommenders.

A typical recommendation task identifies the most relevant items given a query. Depending on the type of the query, a recommendation task could be: (i) personalized recommendation, when the query is a user; (ii) item to item recommendation, when the query is an item; and (iii) search, when the query is a piece of free text. In existing literature, the recommendation task is often formulated as a supervised learning problem. The supervision comes from the collected label (e.g., click) between a pair of query and item. The recommendation task could be defined as either a regression or a classification task depending on the label.

Training deep models typically requires huge amount of parameters, for two major reasons: 1. Models often rely on high-dimensional embeddings to represent high cardinality sparse features (e.g., words, topics, and item IDs); 2. The interaction between queries (e.g., users) and items are highly nuanced, requiring large model capacity to model the interactions well. The large number of parameters naturally require a large amount of data and labels to make sure the models are properly trained. In addition, there are many challenges building effective and efficient recommendation systems. To list a few:

- **Highly-skewed data distribution**: The interaction between queries and items are often highly skewed in a power-law distribution [31], meaning only a small percentage of the items get most of the interactions.
- **Gigantic corpus**: The catalogue of items is often in the order of millions (e.g., songs, apps) [29] to billions (e.g., posts in Facebook, videos on YouTube) [9].
- **Lack of explicit user feedback**: Although modern recommendation systems in the industry collect hundreds of billions of implicit user actions like clicks and thumb-ups, these systems still lack high quality user explicit feedback like item ratings, feedback for user happiness, and relevance scores.

Self-supervised learning (SSL) offers a different angle to learn deep representation via unlabeled data. The basic idea is to extract additional information, construct objectives from the unsupervised data itself, and use supervised loss functions to model the newly introduced auxiliary tasks. Self-supervised learning has been widely used in the areas of Compute Vision (CV) [15, 27, 34] and Natural Language Understanding (NLU) [11, 26]. An example work [27] in CV proposed to rotate images at random, and train a model to predict how each augmented input image was rotated. In NLU, masked language task was introduced in the BERT model, to help improve pre-training of language models. Similarly, other pre-training tasks like predicting surrounding sentences and linked sentences in Wikipedia articles have also been used in improving dual-encoder type models in NLU [3]. Compared to conventional supervised learning, self-supervised learning provides complementary objectives eliminating the pre-requisite of collecting labels manually. In addition, SSL enables autonomous discovery of good semantic representations by exploiting the internal relationship of input features.

Despite the wide adoption in computer vision and natural language understanding, the application of self-supervised learning in the field of recommendation systems is less well studied. The closest line of research studies a set of regularization techniques [17, 25, 43]. They were designed to force learned representations (i.e., output layer (embeddings) of a multi-layer perception), of different examples to be farther away from each other, and spread out in the entire latent embedding space. Although sharing similar spirit with SSL, these techniques do not explicitly construct SSL tasks. In contrast to models in CV or NLU applications, recommendation model takes extremely sparse input where high cardinality categorical features are one-hot (or multi-hot) encoded, such as the item IDs or item categories [32]. These features are typically represented as learnable embedding vectors in deep models. As most models in computer vision and NLU deal with dense input, existing methods for creating SSL tasks are not directly applicable to the sparse models in recommendation systems.

In this paper, we study using self-supervised learning to improve deep recommendation models. Different from CV and NLU, deep models in recommendation are often *sparse*: input space is represented by a set of categorical features (e.g. item ids) with large cardinality. We propose a new SSL framework for such sparse models in the context of two-tower (a.k.a., dual-encoder) Deep Neural Net (DNN) architecture, which is widely used for building large-corpus item retrieval systems in industry (e.g. [41]). The key idea of our approach is to first augment data by masking input information, and then use constrastive loss to learn representations of augmented data for *self-discrimination*: augmented data from the same example can be discriminated against others.

Our detailed contributions are four-folds:

- We present a model architecture agnostic self-supervised learning framework for sparse neural models that are widely used in recommendations. The auxiliary self-supervised loss and the primary supervised loss are jointly optimized via a multi-task learning framework.
- We propose two generalizable approaches to construct auxiliary tasks, (i) Feature Masking and (ii) Feature Dropout, within the proposed framework.
- On one public dataset and one industry-scale dataset for recommendation systems, we demonstrate that introducing SSL as an auxiliary task can significantly improve model performance, especially when labels are scarce.
- Comparing to the state-of-art non-SSL regularization techniques [17, 25, 43], we demonstrate that SSL consistently performs better, and improves model performance even when non-SSL regularization does not bring any additional gains.

2 RELATED WORK

Self-supervised Learning and Pre-training. Various unsupervised / self-supervised learning tasks have been studied in the computer vision community [22]. Popular SSL tasks include: (i) predicting image rotations [15]; (ii) predicting relative patch locations [34]; (iii) predicting next video frames [36] ; and (iv) leveraging contrastive loss [12] etc. In natural language understanding, SSL like pre-training tasks such as next sentence/word prediction and masked-LM have been widely used [11]. In recommender systems, Xin et al. shows combining SSL with reinforcement learning is effective to capture long-term user interest in sequential recommendation. Different from the above, our proposed SSL framework is generally applicable to deep models with categorical features.

For dual-encoder models, Chang et al. shows that pre-training tasks better aligned with the final task are more helpful than generic tasks such as next sentence prediction and masked-LM. The pre-training tasks are designed to leverage large-scale public NLU content, such as Wikipedia. In this paper, we also use the dual-encoder model architecture. Different from the above, the proposed self-supervision tasks do not require the use of a separate data source.

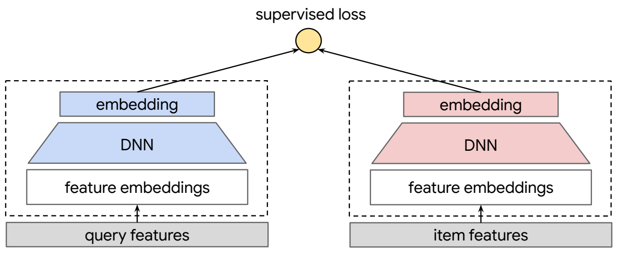

Figure 1: Model architecture: Two-tower DNN with query and item representations.

Spread-out and Instance-based Discriminative Learning. Zhang et al. [43] and Wu et al. [38] use spread-out regularization for improving generalization of deep models. Specifically, in [43], a regularization promoting separation between random instances is shown to improve training stability and generalization. In [38], one objective is to train a classifier treating each instance as its own class, therefore promoting large instance separations in the embedding space. Both the above approaches are studied for computer vision applications.

Neural Recommender. Deep learning has led to many successes in building industry-scale recommender systems such as video suggestion [9], news recommendation [35], and visual discovery in social networks [28, 42]. To handle the challenge of scoring a large number of items, many recommender systems follow a two-stage design that first relies on a *candidate generation* (retrieval) model to find thousands of relevant items given a query, and then uses a finer-granularity *ranking model* to identify the best items to show to the user [6, 44]. On the retrieval side, factorized architecture with query and item embeddings are widely used due to its scalability for scoring a large catalog of items. The item scores are typically computed by dot-product between query and item embeddings so that finding top-k items can be converted to MIPS problem [8] with sublinear time complexity. One popular factorized structure is softmax-based multi-class classification model. The work in [9] treats the retrieval task as an extreme multi-class classification trained with multi-layer perceptron (MLP) model using sampled softmax as loss function. Such models leverages item ID as the only item feature, and thus might suffer from cold-start issue. More recently, a line of research [25, 41] considers applying two-tower DNNs (see Figure 1 for an illustration) on retrieval problems, which is also known as dual-encoder [16, 40], where item embeddings are constructed by a MLP from ID and other categorical metadata features. The self-supervised approach proposed is applicable to both ranking and retrieval models. In this paper we focus on using SSL for retrieval models, particularly, on improving item representations in two-tower DNNs.

3 SELF-SUPERVISED LEARNING FRAMEWORK FOR RECOMMENDERS

In this section, we present our framework of self-supervised learning for deep neural net models for recommenders using large-vocab categorical features. Particularly, a general self-supervised learning

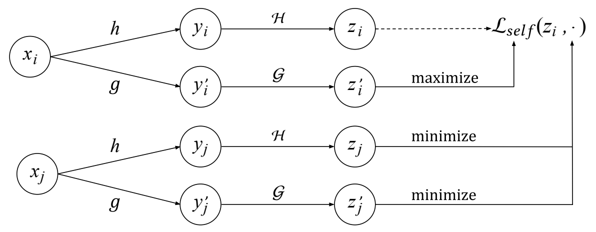

Figure 2: Self-supervised learning framework illustration. Two data augmentations h and g are applied to the original input; Encoders $\mathcal{H}$ and $\mathcal{G}$ are applied to the augmented examples y_i and y'_i to generate embeddings z_i and z'_i. The SSL loss $\mathcal{L}_{self}$ w.r.t. z_i is optimized towards maximizing the similarity with z'_i while minimizing the similarity between z_j and z'_j.

framework is introduced in Section 3.1. In Section 3.2, we present two methods to construct SSL tasks and elaborates on their connections to spread-out regularization. Finally, in Section 3.3, we describe how to use SSL to improve factorized models (i.e., two-tower DNNs as shown in Figure 1), via a multi-task learning framework. It is worth noting that the SSL framework we propose is model architecture agnostic. In this paper, we focus on two-tower models, however the same learning should apply to other model architectures like matrix factorization and multilayer perceptron.

3.1 Framework

Inspired by the SimCLR framework [5] for visual representation learning, we adopt similar contrastive learning algorithms for learning representations of categorical features. The basic idea is two folds: first, we apply different data augmentation for the same training example to learn representations; and then use contrastive loss function to encourage the representations learned for the same training example to be similar. Contrastive loss was also applied in training two-tower DNNs (see e.g., [25, 41]), although the goal there was to make positive item agree with its corresponding queries.

We consider a batch of N training examples $x_1, ..., x_N$, where $x_i \in \mathcal{X}$ represents a set of features for example i. In the context of recommenders, an example indicates a query, an item or a query-item pair. Suppose there are a pair of transform function $h, g : \mathcal{X} \rightarrow \mathcal{X}$ that augment x_i to be y_i, y'_i respectively,

$$y_i \leftarrow h(x_i),\ y'_i \leftarrow g(x_i). \tag{1}$$

Given the same input of example i, we want to learn different representations y_i, y'_i after augmentation to make sure the model still recognizes that both y_i and y_i represent the same input i. In other words, the contrastive loss learns to minimize the difference between y_i, y'_i. In the mean time, for different example i and j, the contrastive loss maximizes the difference between the representations learned y_i, y'_j after data different augmentations. Let $\mathrm{z}_i, \mathrm{z}'_i$ denote the embeddings of y_i, y'_i after encoded by two neural networks $\mathcal{H}, \mathcal{G} : \mathcal{X} \rightarrow \mathbb{R}^d$, that is

$$\mathrm{z}_i \leftarrow \mathcal{H}(y_i),\ \mathrm{z}'_i \leftarrow \mathcal{G}(y'_i). \tag{2}$$

We treat $(\mathrm{z}_i, \mathrm{z}'_i)$ as positive pairs, and $(\mathrm{z}_i, \mathrm{z}'_j)$ as negative pairs for $i \neq j$. Let $s(\mathrm{z}_i, \mathrm{z}'_j) = \langle \mathrm{z}_i, \mathrm{z}'_j \rangle / (\|\mathrm{z}_i\| \cdot \|\mathrm{z}'_j\|)$. To encourage the above properties, we define a loss function for each example in a way similar to the widely used softmax cross entropy:

$$\mathcal{L}_{self}(\mathcal{H}, \mathcal{G}) := -\frac{1}{N} \sum_{i \in [N]} \log \frac{\exp\left(s(\mathrm{z}_i, \mathrm{z}'_i)/\tau\right)}{\sum_{j \in [N]} \exp\left(s(\mathrm{z}_i, \mathrm{z}'_j)/\tau\right)}. \tag{3}$$

where τ is a tunable hyper-parameter for the softmax temperature. The above loss function learns a robust embedding space such that similar items are close to each other after data augmentation, and random examples are pushed farther away. The overall framework is illustrated in Figure 2.

Encoder Architecture. For input examples with categorical features, $\mathcal{H}, \mathcal{G}$ are typically constructed with an input layer and a multi-layer perceptron (MLP) built on top of it. The input layer is often a concatenation of normalized dense features and multiple sparse feature embeddings, where the sparse feature embeddings are learnt representations stored in embedding tables (In contrast, the input layers for computer vision and language models directly work on raw inputs). In order to make SSL facilitate the supervised learning task, we share the embedding table of sparse features for both neural networks $\mathcal{H}, \mathcal{G}$. Depending on the technique for data augmentation (h, g), the MLPs of $\mathcal{H}$ and $\mathcal{G}$ could also be fully or partially shared.

Connection with Spread-out Regularization. In the special case where (h, g) are identical map and $\mathcal{H}, \mathcal{G}$ are the same neural network, loss function in equation (3) is then reduced to

$$-N^{-1} \sum_i \log \frac{exp(1/\tau)}{exp(1/\tau) + \sum_{j \neq i} exp(s(\mathrm{z}_i, \mathrm{z}_j)/\tau)}$$

which encourages learned representations of different examples to have small cosine similarity. The loss is similar to the spread-out regularization in [43], except that the original proposal uses square loss, i.e., $N^{-1} \sum_i \sum_{j \neq i} \langle \mathrm{z}_i, \mathrm{z}_j \rangle^2$, instead of softmax. Spread-out regularization has been proven to improve generalization of large-scale retrieval models. In Section 4, we show that by introducing specific data augmentations, using SSL-based regularization can further improve model performance compared to spread-out regularization.

3.2 Self-supervised Learning Tasks

The most critical question to answer is how to design the transformation functions h and g. A good transformation and data augmentation should make minimal amount of assumptions on the data such that it can be applicable to a large variety of tasks. Inspired by masked-LM used in pre-training for NLU models (e.g. BERT [11]), we present two approaches to construct the auxiliary SSL task. Note that in the rest of this paper, we focus on applying the self-supervised learning tasks on the item side, although the tasks can also be applied on the queries. These tasks can be easily applied to all scenarios where there are multiple features per item.

- **Feature Masking Task**: (h, g) are designed to mask a subset of input features, so that they would only use partial inputs to learn representations. For example, for movie recommendations with movie features $\{movie\_id, genre, category\}$, one

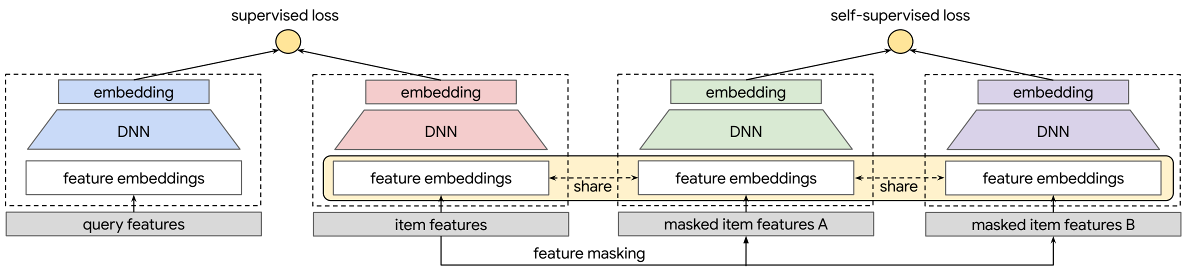

Figure 3: Model architecture: Two-tower Model with feature masking (FM) SSL task. In the FM task, we mask the item features by two separate sets Θ_A and Θ_B. The masked item features are used to learn the item embeddings in the FM task. The feature embeddings are shared with the supervised task.

set of feature could be $\{genre, category\}$, and another set could be $\{movie\_id\}$. By masking through feature splitting, this approach aims to learn the internal relationship between two sets of features. The learned feature embeddings or intermediate representations in neural network can then be shared with down-stream tasks for retrieval and ranking. Note that it's possible to split the whole feature set into k groups, and optimize $k(k-1)/2$ SSL loss functions like Equation (3). With feature masking, as input layer sizes of network $\mathcal{H}, \mathcal{G}$ are no longer same, we use different MLP parameters, but share the input feature embeddings between $\mathcal{H}$ and $\mathcal{G}$.

- **Feature Dropout Task**: As demonstrated in [37], randomly dropping out features during training time can improve cold-start recommendation by learning not to overly rely on certain input features. Following this idea, we propose to apply random feature dropout as feature augmentation, i.e., (h, g) randomly retain each categorical feature with certain probability. This is essentially stochastic feature selection. By applying two independent dropout processes, the SSL task aims to use a random set of input features to predict another random set. This method can avoid manual design with prior knowledge used in the first approach proposed above. When a categorical feature is dropped out, it is treated as missing value, and represented by a default out-of-vocab embedding. In this case, we use the same network for $\mathcal{H}, \mathcal{G}$. This network can be also shared with the main task in multi-task training as introduced below.

3.3 Multi-task Training

To enable SSL learned representations to help improve the learning for the main supervised task such as regression or classification, we leverage a multi-task training strategy where the main supervised task and the auxiliary SSL task are jointly optimized. Intuitively, a linear weighted sum approach is applied:

$$\mathcal{L} = \mathcal{L}_{main} + \alpha \cdot \mathcal{L}_{self}, \tag{4}$$

where $\mathcal{L}_{main}$ is the loss function from main task, and α controls to what extent SSL regularization is applied. An alternative training strategy could be a two-phase training plan where the model firstly pre-trains on $\mathcal{L}_{self}$, and then fine tunes by the main task $\mathcal{L}_{main}$. We leave the experimentation of this method to future work.

As mentioned in Section 2, we focus on improving recommender models built with two-tower DNNs. In practice, we found two-tower models quite effective for recommendation / retrieval tasks. Compared to matrix factorization models, it works very well to (1) learn large-scale sparse feature representations; and thus (2) improves model generalization to mitigate cold-start issues. Compared to a multilayer perceptron (MLP) model, model inference is extremely efficient for two-tower models due to the simple computation of dot product and highly efficient systems for nearest neighbor search.

We consider a batch of positive query-item pairs $\{(q_i, c_i)\}_{i=1}^N$, and the goal is to retrieve the top K most relevant items from a large corpus. We use two neural networks to map query and item to embedding vectors $\{(\mathbf{q}_i, \mathbf{c}_i)\}_{i=1}^N$ respectively. Similar to the contrastive loss in (3), the loss for main task is

$$\mathcal{L}_{main} := -\frac{1}{N} \sum_{i \in [N]} \log \frac{\exp\left(s(\mathbf{q}_i, \mathbf{c}_i)/\tau\right)}{\sum_{j \in [N]} \exp\left(s(\mathbf{q}_i, \mathbf{c}_j)/\tau\right)}. \tag{5}$$

It is worth noting that the SSL task can be applied on either query or item side. Figure 3 and Figure 4 demonstrates the model architecture for both feature masking and feature dropout tasks to facilitate item representation learning.

4 EXPERIMENTS

In this section, we provide empirical results to demonstrate the effectiveness of our proposed self-supervised framework. The experiments are designed to answer the following research questions.

- **RQ1**: We have seen the successes of SSL in both computer vision and natural language understanding. Does the proposed SSL Framework improve deep models in recommendations?
- **RQ2**: SSL is designed to improve primary supervised task by learning representations from unlabeled examples. What is the impact of the amount of training data on the improvement from SSL?
- **RQ3**: Under the multi-task learning framework, how does the self-supervised loss interact with the supervised loss? Specifically, how does weight α on the SSL loss in Equation (3) affect model quality?
- **RQ4**: Which SSL approach works better? How does SSL compare with non-SSL regularization such as spread-out and dropout?

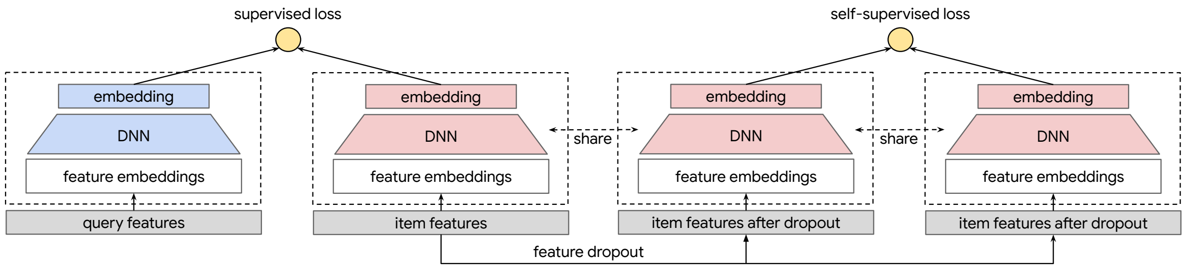

Figure 4: Model architecture: Two-tower model with feature dropout (FD) SSL task. In the FD task, we randomly dropout item feature values to learn item embeddings. The whole item tower (in red) is shared with the supervised task.

- **RQ5**: For feature dropout, how does dropout rate affect the model quality?

The above questions are addressed in order from Section 4.3 - 4.7.

4.1 Datasets

We conduct experiments on two large-scale datasets that both come with rich set of item metadata features. We formulate their primary supervised task as an item-to-item recommendation problem to study the effects of SSL on training recommender (in this case, retrieval) models. Table 1 shows some basic stats for the two datasets. We use 500M and 1B training examples from the two datasets respectively to train and evaluate our models.

Wikipedia [14]. The first dataset focuses on the problem of link prediction between Wikipedia pages. It consists of pairs of pages $(x, y) \in \chi \times \chi$, where x indicates a *source page*, and y is a *destination page* linked from x. The goal is to predict the set of pages that are likely to be linked to a given source page from the whole corpus of web pages. Each page is represented by a feature vector $x = (x_{id}, x_{ngrams}, x_{cats})$, where all the features are categorical. Here, x_{id} denotes the one-hot encoding of the page URL, x_{ngrams} denotes a bag-of-words representation of the set of n-grams of the page's title, and x_{cats} denotes a bag-of-words representation of the categories that the page belongs to. We partitioned the dataset into training and evaluation using a (90%, 10%) split, following the same treatment in [25] and [41].

App-to-App Install (AAI). The AAI dataset was collected on the app landing pages from a commercial mobile app store. On a particular app's (seed app) landing page, the app installs (candidate apps) from the section of recommended apps were collected. Each training example represents a pair of seed-candidate pairs denoted as $(x_{seed}, x_{candidate})$ and their metadata features. The goal is to recommend highly similar apps given a seed app. This is also formulated as an item-to-item recommendation problem via a multi-class classification loss. Note that we only collect positive examples, i.e., $x_{candidate}$ is an installed app from the landing page of x_{seed}. All the impressed recommended apps with no installs are all ignored since we consider them more like weak positives instead of negatives for building retrieval models. Each item (app) is represented by a feature vector $\mathbf{x}$ with the following features:

- *id*: Application id as a one-hot categorical feature.

Dataset	# queries	# items	# examples
Wikipedia	5.3M	5.3M	490M
AAI	2.4M	2.4M	1B

Table 1: Corpus sizes of the Wikipedia and the AAI datasets.

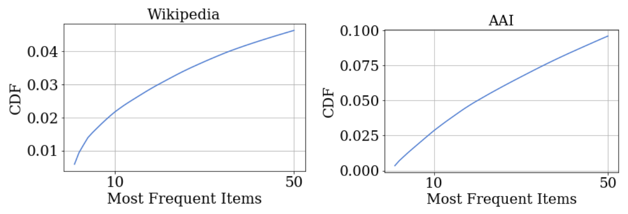

Figure 5: CDF of most frequent items in the Wikipedia and AAI datasets.

- *developer_name*: Name of the app developer as a one-hot categorical feature.
- *categories*: Semantic categories of the app as a multi-hot categorical feature.
- *title_unigram*: Uni-grams of the app title as a multi-hot categorical feature.
- *popularity*: App popularity as a dense feature.
- *quality*:App quality as a dense feature.

Figure 5 shows the CDF of most frequent items for the two datasets, indicating a highly skewed data distribution. For example, the top 50 items in the AAI dataset collectively appeared roughly 10% in the training data. If we consider a naive baseline (i.e., *TopPopular* recommender [10]) that recommends the most frequent top-K items for every query, the *CDF* of the *K*-th frequent item essentially represents the *Recall@K* metric of such baseline. This suggests a naive TopPopular recommender achieves *Recall*@50 ≈ 0.1 for AAI and *Recall*@50 ≈ 0.05 for Wikipedia. We present that all the proposed methods outperform this baseline by a large margin in Section 4.

4.2 Experiment Setup

Backbone Network. For the main task that predicts relevant items given the query, we use the two-tower DNN to encode query and items features (see Figure 1) as the *backbone network*. The item-to-item recommendation problem is formalized as a multi-class classification problem, using the batch softmax loss presented in Equation (5) as the loss function. Note that the proposed framework is architecture agnostic and applicable to networks with sparse categorical features. We choose the two-tower DNN architecture due to it's effectiveness in industrial scale recommendation systems [9, 41].

Hyper-parameters. For the backbone two-tower DNN network, we search the set of hyper-parameters such as the learning rate, softmax temperature (τ) and model architecture that gives the highest *Recall@50* on the validation set. Note that the training batch size in batch softmax is critical for model quality as it determines the number of negatives used for each positive item. Throughout this section, we use batch sizes 1024 and 4096 for Wikipedia and AAI respectively. We find that by further increasing batch size, the quality improvement becomes marginal. We then fine-tuned the number of hidden layers, hidden layer sizes and softmax temperature τ for the baseline models. Specifically, for Wikipedia dataset, we end up with a softmax temperature $\tau = 0.07$, and *hidden_layers* with sizes $[1024, 128]$. For AAI, we use $\tau = 0.06$ and *hidden_layers* $[1024, 256]$. Note that the dimension of last hidden layer is also the dimension of final query and item embeddings.

SSL Networks and Tasks. We fix the backbone network architecture in training with SSL tasks. For each of the SSL tasks, we have:

(1) *FeatureMasking*: The SSL item encoders have the same final embedding dimensions with the primary task as plotted in Figure 3. For Wikipedia, we have *hidden_layers* $= [128]$; for AAI, *hidden_size* $= [256]$. We use a fixed set of features as mask Θ_A and the rest features as another set of mask $\Theta_B = \Theta_A'$. For Wikipedia, we have $\Theta_A = \{url\}$; for AAI, we have $\Theta_A = \{id, developer\_name\}$.
(2) *FeatureDropout*: The SSL item encoders are identical to the item tower (red boxes in Figure 4). We randomly drop out values at the same dropout rate for all features.

Training and Evaluation. For model training, we use Adagrad [13] optimizer with learning rate 0.01. To evaluate the recommendation performance given a seed item, we compute and find the top K items with the highest cosine similarity from the whole corpus and evaluate the quality based on the K retrieved items. Note this is a relatively challenging task, given the sparsity of the dataset and large number of items in the corpus. We adopt popular standard metrics *Recall@K* and mean average precision (*MAP@K*) to evaluate recommendation performance [18]. For each configuration of experiment results, we ran the experiment 5 times and report the average.

4.3 Impact on Recommendation Quality

To answer **RQ1**, we first evaluate the impact on model quality for a combination of SSL tasks within the proposed framework. For the baseline method, besides using the task without SSL, we also consider training the main task trained with 2 popular regularization techniques: dropout (DO), and spread-out regularization (SO).

Full Dataset

Method	Recall@10	Recall@50	MAP@10	MAP@50
Baseline	0.0472	0.1621	0.0160	0.0208
DO [37]	0.0447	0.1551	0.0151	0.0197
SO [43]	0.0503	0.1689	0.0173	0.0223
FD	**0.0579**	**0.1852**	**0.0203**	**0.0257**
FM	0.0485	0.1649	0.0165	0.0214
FD+FM	0.0560	0.1828	0.0194	0.0248

Table 2: Experiment results on the Wikipedia dataset.

Full Dataset

Method	Recall@10	Recall@50	MAP@10	MAP@50
Baseline	0.2891	0.5163	0.1298	0.1407
DO [37]	0.2516	0.4620	0.1121	0.1222
SO [43]	0.2879	0.5119	0.1301	0.1408
FD	0.2972	0.5233	0.1346	0.1455
FM	0.3031	0.5338	0.1372	0.1483
FD+FM	**0.3125**	**0.5464**	**0.1419**	**0.1531**

Table 3: Experiment results on the AAI dataset.

Note that *DO* is a popular regularization technique in industrial-size recommendation systems to enable more more generalization. We include it as a baseline for the Feature Dropout (FD) task to isolate the potential improvement from using dropout only. These SSL techniques are also complementary to each other, and can also be added on top of the baseline methods. In our experiments, we also evaluate the performance of combining the FD and FM tasks. It's worth noting that, although SSL techniques are applicable to improve both query and item representation learning, in our experiments, we only add SSL tasks on the item towers. Specifically, we compare:

- *Baseline*: Backbone two-tower DNN model.
- *DO* [37]: Backbone model with random feature dropout training on the item tower (no SSL task) for regularization.
- *SO* [43]: Backbone model with spread-out regularization on the item tower.
- *FM*: Backbone model with *Feature Masking (FM)* as the SSL task.
- *FD*: Backbone model with *Feature Dropout (FD)* as the SSL task.
- *FD+FM*: Backbone model with FD and FM used together as two SSL tasks.

We conduct a large scale study to evaluate the effectiveness of the proposed framework. For each of the above methods, we experiment with different choices of SSL weights $\alpha \in \{0.1, 0.3, 1.0, 3.0\}$, and report the best results. In Section 4.5, we show the effect of different weights. For methods with random dropout rate including *FD* and *DO*, we fix the dropout rate to be 0.3. In Section 4.7, we show the effect of different dropout rates on *FD* and *DO*.

We report the results for Wikipedia and AAI in Table 2 and Table 3 respectively. For each method, we report the metric averaged over five runs. The best result is highlighted in bold. We observe that with **full datasets**:

- Both FD and FM, when used as single SSL task, improve outperforms baseline and DO. This helps answer **RQ1** that the proposed SSL framework and tasks indeed improves model performance for recommenders.
- FD consistently outperforms spread-out regularization (SO) in both datasets. FM outperforms SO on the AAI. Our hypothesis is that the way we construct these auxiliary SSL-based tasks adds additional supervised information to the model training. And could be why we are seeing that they outperform spread-out regularization (SO) approach that does not use additional information.
- By only applying dropout (DO) on the backbone network, we find that model performance on both datasets is not improved. It demonstrates that the gain is from the SSL task with dropout as data augmentation, instead of from only applying feature dropout as regularization.
- We also see encouraging results when combining both tasks together (FD+FM). On the AAI dataset, FM+FD gives the best performance, showing good complementary between the two SSL tasks. To partially address **RQ4**, we observe that FD consistently outperforms FM, and we have seen cases that hybrid approach that combines both FD and FM could potentially outperform FD only.

4.4 Data Sparsity

We study the effectiveness of SSL tasks with decreased amount of training data to address **RQ2**. For AAI, we repeat the experiments in section 4.3 after down-sampling the original dataset to 10% and 1%. For Wikipedia dataset, we only down-sample it to 10% as the absolute values of retrieval metrics become too small to draw any conclusions when we just use 1% data.

Table 4 and Table 5 show the model performance on the aforementioned sparser datasets for Wikipedia and AAI respectively. SSL tasks do see larger improvement when sparsity increases. In particular, the best SSL approach (*FD*) on the full Wikipedia dataset improves Recall@10 by 22.7% compared to the baseline, while the relative improvement is 57.4% on the 10% dataset. Similar trend is observed for the AAI dataset as shown in Table 3, especially on the 1% dataset although not as obvious as the gains on the Wikipedia dataset. One hypothesis is that the AAI dataset is even more skewed towards popular items as illustrated in Figure 5. In addition, it also includes lots of duplicated pairs of $(query, item)$, where the pairs in Wikipedia are all unique. When we down-sample the AAI dataset to 10%, many of the $(query, item)$ pairs are still preserved due to duplication, so the information is still well kept. When the data is down-sampled to 1%, the remaining data is truly sparse of the original supervised information, and thus leads to more obvious improvements from SSL.

It's worth nothing that, FD consistently outperforms DO and the gap is larger as data becomes sparser. This demonstrates that having dropout for data augmentation in SSL tasks is more effective than directly applying dropout in supervised task.

10% Dataset

Method	Recall@10	Recall@50	MAP@10	MAP@50
Baseline	0.0237	0.0924	0.0077	0.0105
DO	0.0272	0.1046	0.0089	0.0120
SO	0.0254	0.0978	0.0083	0.0112
FD	**0.0373**	**0.1294**	**0.0125**	**0.0164**
FM	0.0261	0.1023	0.0084	0.0115
FD+FM	0.0345	0.1243	0.0115	0.0153

Table 4: Experiment results on 10% of the Wikipedia dataset.

10% Dataset

Method	Recall@10	Recall@50	MAP@10	MAP@50
Baseline	0.2827	0.4993	0.1274	0.1379
DO	0.2642	0.4810	0.1173	0.1276
SO	0.2879	0.5061	0.1308	0.1413
FD	0.2987	0.5202	0.1364	0.1471
FM	0.2995	0.5235	0.1369	0.1476
FD+FM	**0.3023**	**0.5281**	**0.1379**	**0.1487**

1% Dataset

Method	Recall@10	Recall@50	MAP@10	MAP@50
Baseline	0.2600	0.4357	0.1201	0.1287
DO	0.2616	0.4483	0.1194	0.1285
SO	0.2836	0.4838	0.1305	0.1402
FD	0.2832	0.4858	0.1308	0.1406
FM	0.2733	0.4756	0.1254	0.1352
FD+FM	**0.3071**	**0.5128**	**0.1430**	**0.1531**

Table 5: Experiment results on 10% and 1% of AAI Dataset.

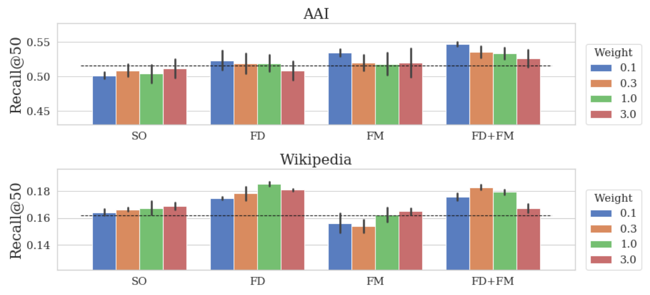

Figure 6: Impact of SSL regularization weight α. The vertical dash line indicates the baseline model's metric.

As a summary, these findings answer research questions raised in **RQ2** that the proposed SSL framework improves model performance more with even less supervision.

4.5 Effect of SSL weight α

To address **RQ3**, we study the effect of SSL regularization strength α presented in equation (3). We perform a line search of α in $\{0.1, 0.3, 1.0, 3.0\}$ to study the SSL tasks' sensitivity to α.

Figure 4.5 summarizes the Recall@50 evaluated on the Wikipedia and AAI dataset w.r.t. various SSL tasks and α. We observe that with increasing α, the model performance is worse than the baseline after certain threshold. This is expected, since large SSL weight α leads to the multitask loss $\mathcal{L}$ dominated by $\alpha \cdot \mathcal{L}_{self}$ in Equation 4. In future work, we plan to reduce the tuning of α by either providing a prior or exploring the alternative training schedule where query and item representations are first pre-trained with SSL tasks, and then fine-tuned with primary supervised task.

4.6 Comparison of Different SSL Tasks

To address **RQ4**, we compare the performances of different SSL tasks and their combinations. As mentioned in Section 4.3, Table 2 and 3 show that *FD* consistently outperforms *SO* and *DO* in all experiments.

By comparing *FD* with *SO*, where both methods introduce a contrastive learning auxiliary task, we see introducing the feature variants leads to more robust item representation, which is especially true when dataset is sparse.

By comparing *FD* with *DO*, where both methods introduce dropout to increase feature variants, we see the technique is more effective when applied on top of the proposed SSL framework, instead of the primary supervised task.

By comparing a single SSL task (*FD* or *FM*) with the combined approach *FD* + *FM*, we see *FD* performs the best on Wikipedia while *FD* + *FM* performs the best on AAI. Our hypothesis is this is due to the *complimentary* of SSL tasks – the *FM* task focuses on learning feature interactions while the *FD* task focuses on feature robustness to variants. The AAI dataset contains a larger number of high cardinality sparse features (such as the *categories* and *title_unigram*) compared to Wikipedia, thus the complementary effects of combining *FM* and *FD* is more obvious. This also suggests future work in designing more orthogonal or complementary SSL tasks in recommender systems besides *FD* and *FM*, which facilitates more effective feature learning.

4.7 Dropout Rate

In this subsection, we discuss the effect of the dropout rate on the Feature Dropout (*FD*) task, to tackle research question **RQ5**. Recall that in Table 2 - 5, we report results with fixed dropout rate 0.3. Now we show the results with varied dropout rates. In particular, we experiment with SSL weight $\alpha = 1.0$ and vary the dropout rate from 0.1 to 0.9. Results are reported in Figure 7.

We observe that, as we increase the dropout rate, the model performance of *DO* continues to deteriorate. For most choices of α (except $\alpha = 0.1$), *DO* is worse than the baseline (the dashed horizontal line in Figure7. For the SSL task with *FD*, the model performance peaks when dropout rate is around 0.3 and then deteriorates when we further improve dropout rate. The model starts to under-perform the baseline when we have a dropout rate over 0.7. This observation aligns with our expectation, since when we have dropout rate equal to 0, the *FD* task becomes *SO*; when we have dropout rate equal to 1.0, *FD* task is trained with no features, and thus hurts main task. Moreover, this suggests the effectiveness of combining the dropout on self-supervised task is more effective than directly applying it on the primary supervised task.

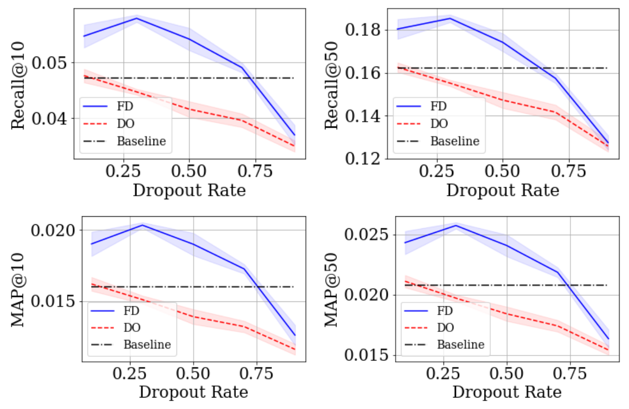

Figure 7: Relationship between dropout rate and evaluation metrics on the Wikipedia dataset.

Name	Baseline	DO	SO	FD	FM	FD+FM
#steps/s	345.2	340.8	338.5	142.1	183.5	126.4

Table 6: Number of steps per second for each method.

4.8 Efficiency

With the additional SSL tasks, training is expected to slow down due to additional computation in the SSL tasks. We here empirically evaluate the average number of training steps per second as a proxy for efficiency. Results are summarized in Table 6. We observe that *SO* is as efficient as the baseline, since the forward and backward pass is identical. *FM* is slightly faster than *FD*. With more SSL tasks added, the training speed decreases as expected.

5 CONCLUSION

In this paper, we proposed a model architecture agnostic self-supervised learning (SSL) framework for large-scale neural recommender models. Within the SSL framework, we also proposed two SSL tasks: (i) Feature Masking (FM), and (ii) Feature Dropout (FD).

For future works, we plan to investigate how different training schemes impact the model quality. One direction is to first pre-train on SSL task to learn query and item representations and fine-tune on primary supervised tasks. Alternatively, it would be interesting to look at SSL with multi-task learning with additional data source with significantly different data distribution. Furthermore, for the two proposed tasks Feature Masking and Feature Dropout, we plan to explore non-parametric based approaches in order to reduce the human intervention in designing feature selection and masking.

REFERENCES

[1] Xavier Amatriain and Justin Basilico. 2016. Past, Present, and Future of Recommender Systems: An Industry Perspective. In *Proceedings of the 10th ACM Conference on Recommender Systems* (Boston, Massachusetts, USA) *(RecSys âĂŹ16)*. Association for Computing Machinery, New York, NY, USA, 211âĂŞ214.
[2] Alex Beutel, Ed H. Chi, Zhiyuan Cheng, Hubert Pham, and John Anderson. 2017. Beyond Globally Optimal: Focused Learning for Improved Recommendations. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW âĂŹ17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 203âĂŞ212.
[3] Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training Tasks for Embedding-based Large-scale Retrieval. In *International Conference on Learning Representations*.

[4] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD âĂŹ16)*. Association for Computing Machinery, New York, NY, USA, 785âĂŞ794.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. arXiv:2002.05709 https://arxiv.org/abs/2002.05709

[6] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems* (Boston, MA, USA) *(DLRS 2016)*. Association for Computing Machinery, New York, NY, USA, 7âĂŞ10.

[7] Evangelia Christakopoulou and George Karypis. 2018. Local Latent Space Models for Top-N Recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (London, United Kingdom) *(KDD âĂŹ18)*. Association for Computing Machinery, New York, NY, USA, 1235âĂŞ1243.

[8] Edith Cohen and David D. Lewis. 1997. Approximating Matrix Multiplication for Pattern Recognition Tasks. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms* (New Orleans, Louisiana, USA) *(SODA âĂŹ97)*. Society for Industrial and Applied Mathematics, USA, 682âĂŞ691.

[9] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems* (Boston, Massachusetts, USA) *(RecSys âĂŹ16)*. Association for Computing Machinery, New York, NY, USA, 191âĂŞ198.

[10] 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* (Copenhagen, Denmark) *(RecSys âĂŹ19)*. Association for Computing Machinery, New York, NY, USA, 101âĂŞ109.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). ACL, 4171–4186.

[12] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. 2014. Discriminative Unsupervised Feature Learning with Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 766–774.

[13] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Mach. Learn. Res.* 12, null (July 2011), 2121âĂŞ2159.

[14] Wikimedia Foundation. [n.d.]. Wikimedia. https://dumps.wikimedia.org/

[15] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. 2018. Unsupervised Representation Learning by Predicting Image Rotations. In *International Conference on Learning Representations*.

[16] Daniel Gillick, Alessandro Presta, and Gaurav Singh Tomar. 2018. End-to-End Retrieval in Continuous Space. *CoRR* abs/1811.08008 (2018). arXiv:1811.08008 http://arxiv.org/abs/1811.08008

[17] Chuan Guo, Ali Mousavi, Xiang Wu, Daniel N Holtmann-Rice, Satyen Kale, Sashank Reddi, and Sanjiv Kumar. 2019. Breaking the Glass Ceiling for Embedding-Based Classifiers for Large Output Spaces. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 4943–4953.

[18] 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* (Perth, Australia) *(WWW âĂŹ17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 173âĂŞ182.

[19] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, and Joaquin Quiñonero Candela. 2014. Practical Lessons from Predicting Clicks on Ads at Facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising* (New York, NY, USA) *(ADKDDâĂŹ14)*. Association for Computing Machinery, New York, NY, USA, 1âĂŞ9.

[20] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. 2004. Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. Inf. Syst.* 22, 1 (Jan. 2004), 5âĂŞ53.

[21] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 3146–3154.

[22] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. 2019. Revisiting Self-Supervised Visual Representation Learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 1920–1929.

[23] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (Aug. 2009), 30âĂŞ37.

[24] Yehuda Koren and Robert M. Bell. 2015. *Advances in Collaborative Filtering*. Springer, 77–118.

[25] Walid Krichene, Nicolas Mayoraz, Steffen Rendle, Li Zhang, Xinyang Yi, Lichan Hong, Ed Chi, and John Anderson. 2019. Efficient Training on Very Large Corpora via Gramian Estimation. In *International Conference on Learning Representations*.

[26] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *International Conference on Learning Representations*.

[27] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. 2016. Learning Representations for Automatic Colorization. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV (Lecture Notes in Computer Science, Vol. 9908)*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer, 577–593.

[28] David C. Liu, Stephanie Rogers, Raymond Shiau, Dmitry Kislyuk, Kevin C. Ma, Zhigang Zhong, Jenny Liu, and Yushi Jing. 2017. Related Pins at Pinterest: The Evolution of a Real-World Recommender System. In *Proceedings of the 26th International Conference on World Wide Web Companion* (Perth, Australia) *(WWW âĂŹ17 Companion)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 583âĂŞ592.

[29] Klaas Bosteels Mark Levy. 2010. Music Recommendation and the Long Tail. In *1st Workshop On Music Recommendation And Discovery (WOMRAD), ACM RecSys, 2010*.

[30] Rishabh Mehrotra, Mounia Lalmas, Doug Kenney, Thomas Lim-Meng, and Golli Hashemian. 2019. Jointly Leveraging Intent and Interaction Signals to Predict User Satisfaction with Slate Recommendations. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW âĂŹ19)*. Association for Computing Machinery, New York, NY, USA, 1256âĂŞ1267.

[31] Staša Milojević. 2010. Power Law Distributions in Information Science: Making the Case for Logarithmic Binning. *J. Am. Soc. Inf. Sci. Technol.* 61, 12 (Dec. 2010), 2417âĂŞ2425.

[32] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. 2019. Deep Learning Recommendation Model for Personalization and Recommendation Systems. *CoRR* abs/1906.00091 (2019).

[33] Wei Niu, James Caverlee, and Haokai Lu. 2018. Neural Personalized Ranking for Image Recommendation. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (Marina Del Rey, CA, USA) *(WSDM âĂŹ18)*. Association for Computing Machinery, New York, NY, USA, 423âĂŞ431.

[34] Mehdi Noroozi and Paolo Favaro. 2016. Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI (Lecture Notes in Computer Science, Vol. 9910)*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer, 69–84.

[35] Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima. 2017. Embedding-Based News Recommendation for Millions of Users. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Halifax, NS, Canada) *(KDD âĂŹ17)*. Association for Computing Machinery, New York, NY, USA, 1933âĂŞ1942.

[36] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. 2015. Unsupervised Learning of Video Representations using LSTMs. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015 (JMLR Workshop and Conference Proceedings, Vol. 37)*, Francis R. Bach and David M. Blei (Eds.). JMLR.org, 843–852.

[37] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. DropoutNet: Addressing Cold Start in Recommender Systems. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 4957–4966.

[38] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. 2018. Unsupervised Feature Learning via Non-Parametric Instance-level Discrimination. *CoRR* abs/1805.01978 (2018). arXiv:1805.01978 http://arxiv.org/abs/1805.01978

[39] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M. Jose. 2020. Self-Supervised Reinforcement Learning for Recommender Systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR âĂŹ20)*. Association for Computing Machinery, New York, NY, USA, 931âĂŞ940.

[40] Yinfei Yang, Steve Yuan, Daniel Cer, Sheng-yi Kong, Noah Constant, Petr Pilar, Heming Ge, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Learning Semantic Textual Similarity from Conversations. In *Proceedings of The Third Workshop on Representation Learning for NLP*. ACL, 164–174.

[41] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. 2019. Sampling-Bias-Corrected Neural Modeling for Large Corpus Item Recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems* (Copenhagen, Denmark) *(RecSys âĂŹ19)*. Association for Computing Machinery, New York, NY, USA, 269âĂŞ277.

[42] Andrew Zhai, Dmitry Kislyuk, Yushi Jing, Michael Feng, Eric Tzeng, Jeff Donahue, Yue Li Du, and Trevor Darrell. 2017. Visual Discovery at Pinterest. In *Proceedings of the 26th International Conference on World Wide Web Companion* (Perth, Australia) *(WWW âĂŹ17 Companion)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 515âĂŞ524.

[43] Xu Zhang, Felix X. Yu, Sanjiv Kumar, and Shih-Fu Chang. 2017. Learning Spread-Out Local Feature Descriptors. In *IEEE International Conference on Computer Vision,, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 4605–4613.

[44] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. 2019. Recommending What Video to Watch next: A Multitask Ranking System. In *Proceedings of the 13th ACM Conference on Recommender Systems* (Copenhagen, Denmark) *(RecSys âĂŹ19)*. Association for Computing Machinery, New York, NY, USA, 43âĂŞ51.