Self-supervised Learning for Large-scale Item 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
Google Inc., United States

ABSTRACT

Large scale recommender models find most relevant items from huge catalogs, and they play a critical role in modern search and recommendation systems. To model the input space with large-vocab categorical features, a typical recommender model learns a joint embedding space through neural networks for both queries and items from user feedback data. However, with millions to billions of items in the corpus, users tend to provide feedback for a very small set of them, causing a power-law distribution. This makes the feedback data for long-tail items extremely sparse.

Inspired by the recent success in self-supervised representation learning research in both computer vision and natural language understanding, we propose a multi-task self-supervised learning (SSL) framework for large-scale item recommendations. The framework is designed to tackle the label sparsity problem by learning better latent relationship of item features. Specifically, SSL improves item representation learning as well as serving as additional regularization to improve generalization. Furthermore, we propose a novel data augmentation method that utilizes feature correlations within the proposed framework.

We evaluate our framework using two real-world datasets with 500M and 1B training examples respectively. Our results demonstrate the effectiveness of SSL regularization and show its superior performance over the state-of-the-art regularization techniques. We also have already launched the proposed techniques to a web-scale commercial app-to-app recommendation system, with significant improvements top-tier business metrics demonstrated in A/B experiments on live traffic. Our online results also verify our hypothesis that our framework indeed improves model performance even more on slices that lack supervision.

1 INTRODUCTION

Recently, neural-net models have emerged to the main stage of modern recommendation systems throughout the industry (see, e.g., [18, 31, 39, 42]), and academia ([8, 32]). Compared to conventional approaches like matrix factorization [1, 21, 22], gradient boosted decision trees [4, 29], and logistic regression based recommenders [19], these deep models handle categorical features more effectively. They also enable more complex data representations, and introduce more non-linearity to better fit the complex data for recommenders.

A particular recommendation task we focus on in this paper is to identify the most relevant items given a query from a huge item catalog. This general problem of *large-scale item recommendations* has been widely adopted in various applications. 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 also an item; and (iii) search: when the query is a piece of free text. To model the interactions between a query and an item, a well-known approach leverages *embedding-based neural networks*. The recommendation task is typically formulated as an extreme classification problem [10] where each item is represented as a dense vector in the output space.

This paper is focused on the *two-tower DNNs* (see Figure 1), popular in many real-world recommenders (see e.g. [23, 39]). In this architecture, a neural network encodes a set of item features into an embedding thus making it applicable even for indexing cold-start items. Moreover, the two-tower DNN architecture enables efficient serving for a large corpus of items in real time, by converting the top-k nearest neighbor search problem to Maximum-Inner-Product-Search (MIPS) [9] that is solvable in sublinear complexity.

Embedding-based deep models typically have large amount of parameters because they are built with high-dimensional embeddings that represent high cardinality sparse features such as topics or item IDs. In many existing literature, the loss functions for training these models are formulated as a supervised learning problem. The supervision comes from the collected labels (e.g., clicks). Modern recommendation systems collect billions to trillions of footprints from users, providing huge amount of training data for building deep models. However, when it comes to modeling a huge catalogue of items in the order of millions (e.g., songs and apps [28]) to even billions (e.g., videos on YouTube [10]), data could still be highly sparse for certain slices due to:

- **Highly-skewed data distribution**: The interaction between queries and items are often highly skewed in a power-law distribution [30]. So a small set of the popular items gets most of the interactions. This will always leave the training data for long-tail items very sparse.
- **Lack of explicit user feedback**: Users often provide lots of positive feedback implicitly like clicks and thumb-ups. However, they are much less likely to provide explicit feedback like item ratings, feedback for user happiness, and relevance scores.

Self-supervised learning (SSL) offers a different angle to improve deep representation learning via unlabeled data. The basic idea is to enhance training data with various data augmentations, and supervised tasks to predict or reconstitute the original examples as auxiliary tasks. Self-supervised learning has been widely used in the areas of Compute Vision (CV) [15, 25, 33] and Natural Language Understanding (NLU) [12, 24]. An example work [25] 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 prerequisite 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, 23, 41], which are 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 [31]. 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. More recently, a line of research studies self-supervised learning improving sequential user modeling in recommendations [27, 37, 43]. Different from these works, this paper focuses on item representation learning, and shows how SSL can help improve generalization in the context of long-tail item distribution. Moreover, in contrast to using SSL on a certain sequential user feature, we design new SSL tasks and demonstrate their effectiveness for learning with a set of heterogeneous categorical features, which we believe is a more general setup for other types of recommendation models such as multitask ranking models (e.g., [42]).

In this paper, we propose to leverage self-supervised learning based auxiliary tasks to improve item representations, especially with long-tail distributions and sparse data. Different from CV or NLU applications, input space of recommendation model is highly *sparse* and represented by a set of categorical features (e.g. item ids) with large cardinality. For such sparse models, we propose a new SSL framework, where the key idea is to: (i) augment data by masking input information; (ii) encode each pair of augmented examples by a two-tower DNN; and (iii) apply a contrastive loss to learn representations of augmented data. The goal of contrastive learning is to let augmented data from the same example be discriminated against others. Note that the two-tower DNN for contrastive learning and the one for encoding query and item can share a certain amount of model parameters. See more details in Section 3.

Our contribution is four-fold:

- **SSL framework**: We present a model architecture agnostic self-supervised learning framework for sparse neural models in recommendations. The auxiliary self-supervised loss and the primary supervised loss are jointly optimized via a multi-task learning framework. We focus on using this framework for efficiently scoring a large corpus of items, which is also known as item retrieval in two-stage recommenders [10]. We believe it would also shed light on designing SSL for other types of models such as ranking models [7].
- **Data augmentation**: We propose a novel data augmentation method that exploits feature correlations and are tailored for heterogeneous categorical features that are common in recommender models.
- **Offline experiments**: 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, 23, 41], we demonstrate that SSL consistently performs better, and improves model performance even when non-SSL regularization does not bring any additional gains.
- **Live experiment in a web-scale recommender**: We have launched the proposed SSL technique in a fairly strong two-tower app-to-app recommendation model in a large-scale real-world system. Live A/B testing shows significantly improved top-tier metrics. We especially see bigger improvements for slices without much supervision.

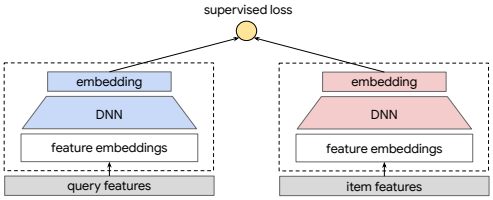


Figure 1: Model architecture: Two-tower DNN with query and item representations.

2 RELATED WORK

Self-supervised Learning and Pre-training. Various unsupervised and self-supervised learning tasks have been studied in the computer vision community. The closest line of research is SimCLR [5] which also utilizes self-supervised learning and contrastive learning for visual representation learning. Different with SimCLR and other works [2, 6] in vision, here we propose augmentations that are more suitable and tailed for categorical features for recommendations, instead of relying on image-specific augmentations such as image cropping, rotation and color distortion. In addition, the proposed framework does not require multi-stage training schedules (such as pre-training then fine-tuning). [20].

In NLU, for dual-encoder models, [3] 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.

Spread-out Regularization. Zhang et al. [41] and Wu et al. [36] use spread-out regularization for improving generalization of deep models. Specifically, in [41], a regularization promoting separation between random instances is shown to improve training stability and generalization. In [36], 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 Recommenders. Deep learning has led to many successes in building industry-scale recommender systems such as video suggestion [10], news recommendation [34], and visual discovery in social networks [26, 40]. For large-scale item retrieval, two-tower models with separate query and item towers are widely used due to its high efficiency for serving. The recommendations are typically computed by a dot-product between query and item embeddings so that finding top-k items can be converted to MIPS problem [9] with sublinear time complexity. One popular factorized structure is softmax-based multi-class classification model. The work in [10] 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, suffering from cold-start issue. More recently, a line of research [23, 39] considers applying two-tower DNNs on retrieval problems, which is also known as dual-encoder [16, 38], 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.

Self-supervised Learning in Sequential Recommendations. In recommender systems, a line of research has been recently studied for utilizing self-supervised learning for sequential recommendation. Self-supervised learning tasks are designed to capture information among user history [43] and learn more robust disentangled user representation [27] in user sequential recommendation. Moreover, 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 focusing on improving item representation with long-tail distributions. The proposed SSL tasks do not require modeling sequential information and are generally applicable to deep models with heterogeneous categorical features.

3 METHOD

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 framework is introduced in Section 3.1. In Section 3.2, we present a data augmentation method 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.

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., [23, 39]), although the goal there was to make positive item agree with its corresponding queries.

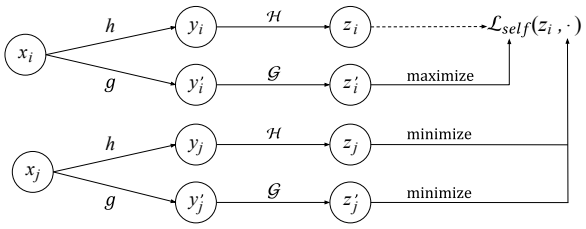


Figure 2: Self-supervised learning framework illustration. Two data augmentations h and g are applied to the input; Encoders $\mathcal{H}$ and $\mathcal{G}$ are applied to the augmented examples $\mathbf{y}_i$ and $\mathbf{y}_i'$ to generate embeddings $\mathbf{z}_i$ and $\mathbf{z}_i'$. The SSL loss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

We consider a batch of N item 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 $h, g : \mathcal{X} \to \mathcal{X}$ 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 $\mathbf{z}_i, \mathbf{z}_i'$ denote the embeddings of y_i, y_i' after encoded by two neural networks $\mathcal{H}, \mathcal{G} : \mathcal{X} \to \mathbb{R}^d$, that is

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

We treat $(\mathbf{z}_i, \mathbf{z}_i')$ as positive pairs, and $(\mathbf{z}_i, \mathbf{z}_j')$ as negative pairs for $i \neq j$. Let $s(\mathbf{z}_i, \mathbf{z}_j') = \langle \mathbf{z}_i, \mathbf{z}_j' \rangle / (\|\mathbf{z}_i\| \cdot \|\mathbf{z}_j'\|)$. To encourage the above properties, we define the SSL loss for a batch of N examples $\{x_i\}$ as:

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

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) on top. 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 [41]. 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

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

which encourages learned representations of different examples to have small cosine similarity. The loss is similar to the spread-out regularization introduced in [41], except that the original proposal uses square loss, i.e., $N^{-1} \sum_i \sum_{j \neq i} \langle \mathbf{z}_i, \mathbf{z}_j \rangle^2$, 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.

3.2 A Two-stage Data Augmentation

We introduce the data augmentation, i.e., h and g in Figure 2. Given a set of item features, the key idea is to create two augmented examples by masking part of the information. A good transformation and data augmentation should make minimal amount of assumptions on the data such that it can be generally applicable to a large variety of tasks and models. The idea of masking is inspired by the Masked Language Modeling in BERT [12]. Different from sequential tokens, a general set of features does not have sequential order, and leaves the choice of masking pattern as an open question. We seek to design the masking pattern by exploring feature correlation. We propose Correlated Feature Masking (CFM), tailored to categorical features with awareness of feature correlations.

Before diving into the details of masking, we first present a two-stage augmentation algorithm. Note that without augmentation, the input layer is created by concatenating the embeddings of all categorical features. The two-stage augmentation includes:

- **Masking.** Apply a masking pattern on the set of item features. We use a default embedding in the input layer to represent the features that are masked.
- **Dropout.** For categorical features with multiple values, we drop out each value with a certain probability. It further reduces input information and increase the hardness of SSL task.

The masking step can be interpreted as a special case of dropout with a 100% dropout rate. One strategy is the complementary masking pattern, that we split the feature set into two exclusive features sets into the two augmented examples. Specifically, we could randomly split the feature set into two disjoint subsets. We call this method **Random Feature Masking (RFM)**, and will use it as one of our baselines. We now introduce **Correlated Feature Masking (CFM)** where we further explore the feature correlations when creating masking patterns.

Mutual Information of Categorical Features. If the set of masked features are chosen at random, (h, g) are essentially sampled from 2^k different masking patterns over the whole feature set with k features. Different masking patterns would naturally lead to different effects for the SSL task. For instance, the SSL contrastive learning task may exploit the shortcut of highly correlated features between the two augmented examples, making the SSL task too easy. To address this issue, we propose to split features according to feature correlation measured by mutual information. The mutual information of two categorical features is given by

$$MI(V_i, V_j) = \sum_{v_i \in V_i, v_j \in V_j} P(v_i, v_j) \log \frac{P(v_i, v_j)}{P(v_i)P(v_j)}, \tag{4}$$

where V_i, V_j denote their vocab sets. The mutual information for all pairs of features can be pre-computed.

Correlated Feature Masking. With the pre-computed mutual information, we propose Correlated Feature Masking (CFM) that exploits the feature-dependency patterns for more meaningful SSL tasks. For the set of masked features, F_m, we seek to mask highly correlated features together. We do so by first uniformly sample a seed feature f_{seed} from all the available features $F = \{f_1, ..., f_k\}$, and then select the top-n most correlated features $F_c = \{f_{c,1}, ..., f_{c,n}\}$ according to their mutual information with f_{seed}. The final F_m will be the union of the seed and set of correlated features, i.e., $F_m = \{f_{seed}, f_{c,1}, ..., f_{c,n}\}$. We choose $n = \lfloor k/2 \rfloor$ so that the masked and retained set of features have roughly the same size. We change the seed feature per batch so that the SSL task will learn on various kinds of masking patterns.

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. Precisely, let $\{(q_i, x_i)\}$ be a batch of query-item pairs sampled from the training data distribution $\mathcal{D}_{train}$, and let $\{x_i\}$ be a batch of items sampled from an item distribution $\mathcal{D}_{item}$. Then the joint loss is:

$$\mathcal{L} = \mathcal{L}_{main}(\{(q_i, x_i)\}) + \alpha \cdot \mathcal{L}_{self}(\{x_i\}), \tag{5}$$

where $\mathcal{L}_{main}$ is the loss function for the main task capturing interaction between query and item, and α is the regularization strength.

Heterogeneous Sample Distributions. The marginal item distribution from $\mathcal{D}_{train}$ typically follows a power-law. Therefore, using the training item distribution for $\mathcal{L}_{self}$ would cause the learned feature relationship to be biased towards head items. Instead, we sample items uniformly from the corpus for $\mathcal{L}_{self}$. In other words, $\mathcal{D}_{item}$ is the uniform item distribution. In practice, we find using the heterogeneous distributions for main and ssl tasks is critical for SSL to achieve superior performance.

Loss for Main Task. There could be many choices for the main loss depending on the objectives. In this paper, we consider the batch softmax loss used in both recommenders [39] and NLP [16] for optimizing top-k accuracy. In detail, let $\mathbf{q}_i, \mathbf{x}_i$ be the embeddings of query and item examples (q_i, x_i) after being encoded by two neural networks, then for a batch of pairs $\{(q_i, x_i)\}_{i=1}^N$ and temperature τ, the batch softmax cross entropy loss is

$$\mathcal{L}_{main} = -\frac{1}{N} \sum_{i \in [N]} \log \frac{\exp(s(\mathbf{q}_i, \mathbf{x}_i)/\tau)}{\sum_{j \in [N]} \exp(s(\mathbf{q}_i, \mathbf{x}_j)/\tau)}. \tag{6}$$

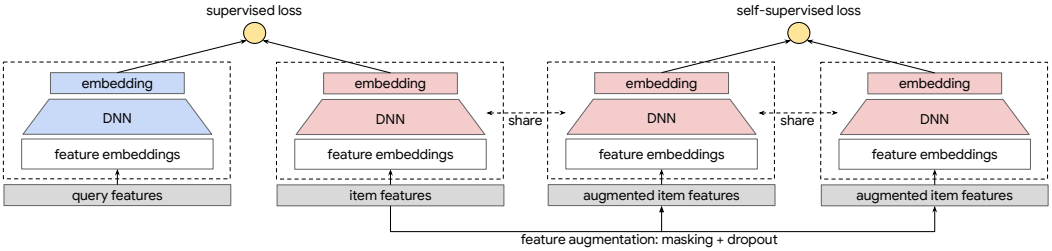


Figure 3: Model architecture: Two-tower model with SSL. In the SSL task, we apply feature masking and dropout on the item features to learn item embeddings. The whole item tower (in red) is shared with the supervised task.

Other Baselines. As mentioned in Section 2, we use two-tower DNNs as the baseline model for main task. Two-tower model has the unique property of encoding item features compared to classic matrix factorization (MF) and classification models. While the latter two methods are also applicable to large-scale item retrieval, they only learn item embeddings based on IDs, and thus do not fit in our proposal of using SSL for exploiting item feature relations.

4 OFFLINE EXPERIMENTS

We provide empirical results to demonstrate the effectiveness of our proposed self-supervised framework both in academic public dataset and in actual large-scale recommendation products. The experiments are designed to answer the following research questions.

- **RQ1**: Does the proposed SSL Framework improve deep models for recommendations?
- **RQ2**: SSL is designed to improve primary supervised task through introduced SSL task on unlabeled examples. What is the impact of the amount of training data on the improvement from SSL?
- **RQ3**: How do the SSL parameters, i.e., loss multiplier α and dropout rate in data augmentation, affect model quality?
- **RQ4**: How does RFM perform compared to CFM? What is the benefit of leveraging feature correlations in data augmentation?

The above questions are addressed in order from Section 4.3 - 4.5.

4.1 Datasets

We conduct experiments on two large-scale datasets that both come with a 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. See Appendix .1 for details about the statistics of these two datasets.

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 [23] and [39].

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.
- *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.

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 (6) as the loss function. For discussions of the choice of backbone network, we refer the readers to related sections in Section 2 and Section 3.3.

Hyper-parameters. For the backbone two-tower DNN, 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 also tuned the number of hidden layers, hidden layer sizes and softmax temperature τ for the baseline models. For Wikipedia

dataset, we use 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. All models are trained with Adagrad [13] optimizer with learning rate 0.01.

We consider two SSL parameters: 1) the SSL loss multiplier α in equation (5), and 2) the feature dropout rate, denoted as dr, in the second phase of data augmentation (see Section 3.2). For each augmentation method (e.g., CFM, RFM), we conduct grid search of the two parameters by ranges $\alpha = [0.1, 0.3, 1.0, 3.0]$, $dr = [0.1, 0.2, ..., 0.9]$, and report the best result.

Evaluation. 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 Effectiveness of SSL with Correlated Feature Masking

To answer **RQ1**, we first evaluate the impact of SSL on model quality. We focus on using CFM followed by dropout as the data augmentation technique. We will show the superior performance of CFM over other variants in Section 4.5.

We consider three baseline methods:

- *Baseline*: Vanilla backbone network with the two-tower DNN architecture.
- *Feature Dropout (FD)* [35]: Backbone model with random feature dropout on the item tower in the supervised learning task. The feature dropout on item features could be treated as data augmentation. FD does not have the additional SSL regularization compared to our approach.
- *Spread-out Regularization (SO)* [41]: Backbone model with spread-out regularization on the item tower as a regularization. The SO regularization shares similar contrastive loss as that in our SSL framework. However, it applies contrastive learning on original examples without any data augmentation, and is thus different from our approach.

The latter two methods are chosen since they are (1) model-agnostic and scalable for industrial-size recommendation systems; (2) compatible with categorical sparse features for improving generalization. In addition, FD can be viewed as an ablation study to isolate the potential improvement from contrastive learning. Similarly, SO is included to isolate the improvement from feature augmentation.

We observe that with **full datasets** (see Table 1), CFM consistently performs the best compared with non-SSL regularization techniques. On AAI, CFM out-performs the next best method by 8.69% relatively and on AAI by 3.98%. This helps answer **RQ1** that the proposed SSL framework and tasks indeed improves model performance for recommenders. By comparing CFM with SO, it shows that the data augmentation is critical for the SSL regularization to have better performance. Without any data augmentation, the proposed SSL method is reduced to SO. By comparing CFM and FD, we find the feature augmentation is more effective when applied to the SSL task than to the supervised task as a standard regularization technique. Note that FD, as a well known approach for improving generalization in some cases, applies feature augmentation together with supervised training.

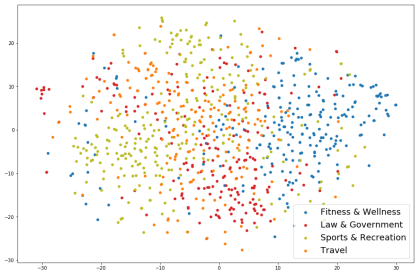
Wikipedia				
Method	MAP@10	MAP@50	Recall@10	Recall@50
Baseline	0.0171	0.0229	0.0537	0.1930
FD [35]	0.0172	0.0229	0.0535	0.1912
SO [41]	0.0176	0.0235	0.0549	0.1956
Our method	**0.0183**	**0.0243**	**0.057**	**0.2009**
AAI				
Method	MAP@10	MAP@50	Recall@10	Recall@50
Baseline	0.1257	0.1363	0.2793	0.4983
FD [35]	0.1278	0.1384	0.2840	0.5058
SO [41]	0.1300	0.1406	0.2870	0.5076
Our method	**0.1413**	**0.1522**	**0.3078**	**0.5355**

Table 1: Results on the full Wikipedia and AAI dataset.

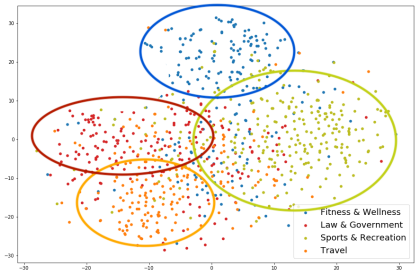
Head-tail Analysis. To understand the gain from SSL, we further break down the overall performance by looking at different item slices by item popularity. The splitting of the *head* and *tail* test set is described in the appendix .2. Our hypothesis is that SSL generally helps improve the performance for slices without much supervision (e.g., tail items). The results evaluated on the tail and head test sets are reported the results in Table 3. We observe that the proposed SSL methods improve the performance for both head and tail item recommendations, with larger gains from the tail items. For instance, in AAI, the CFM improves over 51.5% of the Recall@10 on tail items, while the improvement is 8.57% on head.

*Effects of SSL Parameters (**RQ3**).* Figure 5 summarizes the Recall@50 evaluated on the Wikipedia and AAI dataset w.r.t. the regularization strength α. It also shows the results of SO which shares the same regularization parameter. We observe that with increasing α, the model performance is worse than the baseline model (shown in dash line) 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 (5). By further comparing our approach with SO, we show that the SSL based regularization outperforms SO in a wide range of α. Figure 6 shows the model performance across different dropout rates dr. It also shows *DO* which shares the same parameter. As dr increases, the model performance of *DO* continues to deteriorate. For most choices of α (except $\alpha = 0.1$), *DO* is worse than the baseline. For the SSL task with feature dropout, the model performance peaks when $dr = 0.3$ and then deteriorates when we further improve dropout rate. The model starts to underperform the baseline when dr is too large. This observation aligns with our expectation in the sense that when the rate is too large, the input information becomes too little for to learn meaningful representations through SSL.

Visualization of Item Representations. We visualize the learned app embeddings from the AAI dataset in t-SNE plot. We postpone



(a) Baseline Model



(b) Best SSL Model

Figure 4: Comparison of t-SNE plots for app embeddings for baseline, and our best SSL model.

the detailed setup to Appendix .3. As shown in in Figure 4, we clearly see that apps embeddings learned with our SSL regularization are better clustered according to their own categories, compared to the counter parts of our baseline, which demonstrates that representations learned through SSL have stronger semantic structures. This partially explains the gain from SSL.

4.4 Data Sparsity

We study the effectiveness of CFM in presence of sparse data to address **RQ2**. We uniformly down-sampled 10% of training data and evaluate on the same (full) test dataset. Experiment results are reported in Table 2. With increased data sparsity, CFM demonstrates larger improvement for Wikipedia and AAI respectively. In particular, the CFM on the full Wikipedia dataset improves Recall@10 by 6.1% compared to the baseline, while the relative improvement is 20.6% on the 10% dataset. Similar trend is observed for the AAI dataset (10.2% vs 25.7% on the down-sampled dataset). It's worth noting that, CFM consistently outperforms FD 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.

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 Comparison of Different Data Augmentations

In this section, we compare several feature augmentation alternatives with CFM to answer **RQ4** by studying: 1) the benefit of exploiting feature correlation in masking and 2) benefit of using dropout as part of augmentation. In specific, we consider the following alternatives:

- *RFM*: Random Feature Masking. In this method, random set of features are masked, instead of guided by mutual information in CFM.
- RFM_{no_compl}: Random Feature Masking with no Complementary sets of features applied. In this method, 2 independent sets of features are masked at random, instead of a complementary pair of masks in CFM.

10% Wikipedia Dataset				
Method	MAP@10	MAP@50	Recall@10	Recall@50
Baseline	0.0077	0.0105	0.0237	0.0924
FD [35]	0.0089	0.0120	0.0272	0.1046
SO [41]	0.0083	0.0112	0.0254	0.0978
Our method	**0.0093**	**0.0126**	**0.0286**	**0.1093**
10% AAI Dataset				
Method	MAP@10	MAP@50	Recall@10	Recall@50
Baseline	0.1112	0.1194	0.2406	0.4068
FD [35]	0.1217	0.1302	0.2611	0.4324
SO [41]	0.1220	0.1308	0.2632	0.4400
Our method	**0.1409**	**0.1507**	**0.3024**	**0.5014**

Table 2: Experiment results trained on the sparse (10% down-sampled) Wikipedia and AAI datasets.

Wikipedia				
Method	Tail		Head	
	Recall@10	Recall@50	Recall@10	Recall@50
Baseline	0.0472	0.1621	0.0610	0.2273
FD	0.0474	0.1638	0.0593	0.2212
SO	0.0481	0.1644	0.0606	0.2268
Our method	**0.0524**	**0.1749**	**0.0619**	**0.2283**
AAI				
Baseline	0.0475	0.2333	0.2846	0.4993
FD	**0.0727**	0.2743	0.2849	0.5069
SO	0.0661	0.2602	0.2879	0.5086
Our method	0.0720	**0.2906**	**0.309**	**0.537**

Table 3: Results of Wikipedia and AAI on tail and head item slices.

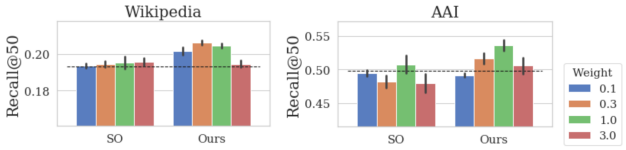


Figure 5: Impact of regularization strength α for SSL-based and spread-out regularization methods. The vertical dash line indicates the baseline model's metric.

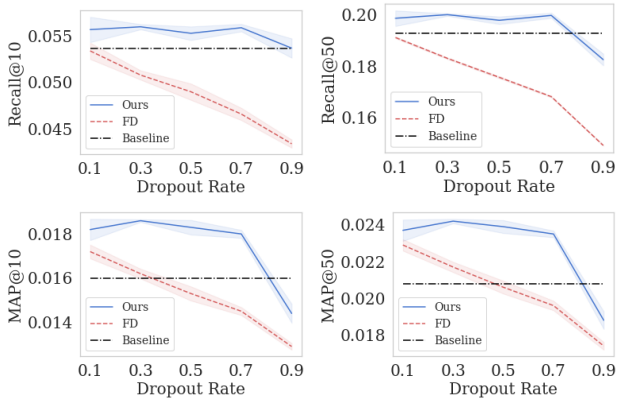


Figure 6: Impact of dropout rate dr for our method and the standard dropout training on the Wikipedia dataset.

Comparison of Multiple Data Augmentations				
Method	MAP@10	MAP@50	Recall@10	Recall@50
Baseline	0.1257	0.1363	0.2793	0.4983
CFM	**0.1413**	**0.1522**	**0.3078**	**0.5355**
RFM	0.1281	0.1389	0.2851	0.5104
RFM_{no_compl}	0.1363	0.1472	0.3007	0.5290
$CFM_{no\_dropout}$	0.1309	0.1417	0.2898	0.5150
$NoMasking$	0.1303	0.1408	0.2868	0.5053

Table 4: Results of CFM and other data augmentation techniques on the AAI dataset.

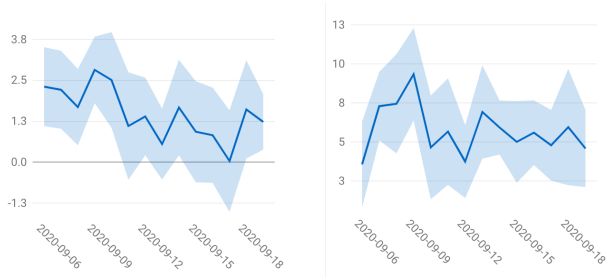
- $CFM_{no\_dropout}$: Correlated Feature Masking with No Dropout applied. In other words, only apply correlated masking as the augmentation in the SSL task.
- $NoMasking$: Correlated Feature Masking but skipping the masking phrase in augmentation. In other words, we only apply dropout to features as the augmentation.

We apply these feature augmentation functions in the SSL framework and report the results on AAI dataset in Table 4.

First, we observe that all the variants are worse than CFM, but still out-perform the baseline model. In particular, we see having mutual information in selecting the masking set is critical to model improvement, since we see the biggest performance drop is from RFM where masking set is selected at random. By comparing CFM with results from the two methods (RFM_{no_compl} and $NoMasking$) that allow feature overlap between the two augmented examples via independent dropout, we see the contrastive learning task is more helpful with complementary information that potentially avoids shortcut in learning. Finally, by comparing $CFM_{no\_dropout}$ and CFM, we see the second phrase of randomly dropping out feature values also helps, which could be potentially explained by introducing more feature variants in the SSL task.

5 LIVE EXPERIMENT

In this section, we describe how we apply our proposed SSL framework to a web-scale commercial app recommender system. Specifically, given an app as the query, the system identifies similar apps



(a) Global **(b) Localized**

Figure 7: Top Business Metric Improvement Percentage (y-axis) over Days (x-axis) in Online Experiments: (a) improvement globally; (b) improvement on a localized market.

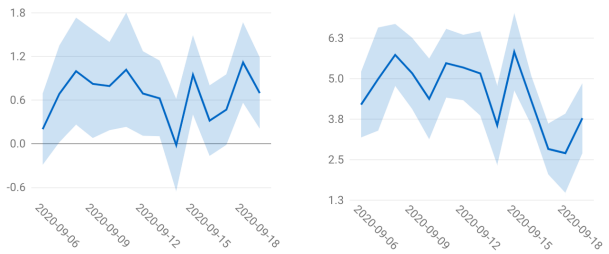
given the query. One of the models surfacing this recommendation is trained on the AAI dataset as described in Section 4.1, with the same backbone network structure as the two-tower DNN structure in Figure 1 (with modifications). For a natural extension of the offline experiments conducted on Section 4.3 for the AAI experiments, we conducted an A/B experiment for investigating the synergy of deploying the best SSL-based model online. While we already presented improved offline metrics on this dataset, in many real-world systems, offline studies might not align with live impact due to 1) lack of implicit feedback, since the offline evaluation data is collected via user engagement history based on the production system; 2) failing to capture product's multiple objectives optimization goal, where it's very likely that recommending more engaging apps hurts other business goals. Therefore, this experiment is critical in demonstrating the effectiveness of the proposed framework in real-world settings.

In our live A/B testing, we add the best performing SSL task with the same set of hyper-parameters on top of the existing well-tuned two-tower DNN model used in production. In a time frame of 14 days, the model improved the overall business metrics significantly, with +0.67% increase in key user engagement (Figure 8a) and +1.5% increase in top business metric (Figure 7a). To echo the study on the *Head-tail Analysis* in Section 4.3 and the data sparsity analysis in Section 4.4, we see significant improvements on two slices: 1) cold-starting for fresh apps: the model improves +4.5% on user engagement for fresh apps (Figure 8b); and (2) international countries that have sparser training data compared to major markets: we see significant +5.47% top business metric gains (Figure 7b right). Again, both of these results verify our hypothesis that our SSL framework indeed significantly improves model performance for slices without much supervision. Given the results, the SSL empowered model was successfully launched in the current production system.

6 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 introduced a novel data augmentation applicable to heterogeneous categorical features and showed its superior performance over other variants.

For future works, we plan to investigate how different training schemes impact the model quality. One direction is to first pre-train



(a) All Apps **(b) Fresh Apps**

Figure 8: Top User Engagement Improvement Percentage (y-axis) over Days (x-axis) in Online Experiments: (a) improvement on all apps; (b) improvement on fresh apps.

on SSL task to learn query and item representations and fine-tune on primary supervised tasks. Alternatively, it would be interesting to extend the technique for deep models in application domains such as search ranking or pCTR prediction.

REFERENCES

[1] Alex Beutel, Ed H. Chi, Zhiyuan Cheng, Hubert Pham, and John Anderson. [n.d.]. Beyond Globally Optimal: Focused Learning for Improved Recommendations. In *WWW 2017*.

[2] L. Beyer, X. Zhai, A. Oliver, and A. Kolesnikov. [n.d.]. S4L: Self-Supervised Semi-Supervised Learning. In *ICCV 2019*.

[3] Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. [n.d.]. Pre-training Tasks for Embedding-based Large-scale Retrieval. In *ICLR 2020*.

[4] Tianqi Chen and Carlos Guestrin. [n.d.]. XGBoost: A Scalable Tree Boosting System. In *KDD 2016*.

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

[6] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. 2020. Big Self-Supervised Models are Strong Semi-Supervised Learners. *arXiv preprint arXiv:2006.10029* (2020).

[7] 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. [n.d.]. Wide & Deep Learning for Recommender Systems *(DLRS 2016)*.

[8] Evangelia Christakopoulou and George Karypis. [n.d.]. Local Latent Space Models for Top-N Recommendation.

[9] Edith Cohen and David D. Lewis. [n.d.]. Approximating Matrix Multiplication for Pattern Recognition Tasks. In *SODA 1997*.

[10] Paul Covington, Jay Adams, and Emre Sargin. [n.d.]. Deep Neural Networks for YouTube Recommendations. In *RecSys 2016*.

[11] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. [n.d.]. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. In *RecSys 2019*.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. [n.d.]. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT 2019*.

[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), 2121–2159.

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

[15] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. [n.d.]. Unsupervised Representation Learning by Predicting Image Rotations. In *ICLR 2018*.

[16] Daniel Gillick, Alessandro Presta, and Gaurav Singh Tomar. 2018. End-to-End Retrieval in Continuous Space. *CoRR* abs/1811.08008 (2018). 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 *Neurips*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.).

[18] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. [n.d.]. Neural Collaborative Filtering. In *WWW 2017*.

[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*.

[20] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. [n.d.]. Revisiting Self-Supervised Visual Representation Learning. In *CVPR 2019*.

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

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

[23] Walid Krichene, Nicolas Mayoraz, Steffen Rendle, Li Zhang, Xinyang Yi, Lichan Hong, Ed Chi, and John Anderson. [n.d.]. Efficient Training on Very Large Corpora via Gramian Estimation. In *ICLR 2019*.

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

[25] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. [n.d.]. Learning Representations for Automatic Colorization. In *ECCV 2016*.

[26] David C. Liu, Stephanie Rogers, Raymond Shiau, Dmitry Kislyuk, Kevin C. Ma, Zhigang Zhong, Jenny Liu, and Yushi Jing. [n.d.]. Related Pins at Pinterest: The Evolution of a Real-World Recommender System. In *WWW 2017*.

[27] Jianxin Ma, Chang Zhou, Hongxia Yang, Peng Cui, Xin Wang, and Wenwu Zhu. [n.d.]. Disentangled Self-Supervision in Sequential Recommenders. In *KDD 2020*.

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

[29] Rishabh Mehrotra, Mounia Lalmas, Doug Kenney, Thomas Lim-Meng, and Golli Hashemian. [n.d.]. Jointly Leveraging Intent and Interaction Signals to Predict User Satisfaction with Slate Recommendations. In *WWW 2019*.

[30] 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), 2417–2425.

[31] 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).

[32] Wei Niu, James Caverlee, and Haokai Lu. [n.d.]. Neural Personalized Ranking for Image Recommendation. In *WSDM 2018*.

[33] Mehdi Noroozi and Paolo Favaro. [n.d.]. Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. In *ECCV 2016*.

[34] Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima. [n.d.]. Embedding-Based News Recommendation for Millions of Users. In *KDD 2017*.

[35] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. [n.d.]. DropoutNet: Addressing Cold Start in Recommender Systems. In *Neurips 2017*.

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

[37] Xin Xin, Alexandros Karatzoglou, I. Arapakis, and J. Jose. [n.d.]. Self-Supervised Reinforcement Learning for Recommender Systems. *SIGIR 2020* ([n. d.]).

[38] 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.

[39] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. [n.d.]. Sampling-Bias-Corrected Neural Modeling for Large Corpus Item Recommendations. In *RecSys 2019*.

[40] Andrew Zhai, Dmitry Kislyuk, Yushi Jing, Michael Feng, Eric Tzeng, Jeff Donahue, Yue Li Du, and Trevor Darrell. [n.d.]. Visual Discovery at Pinterest. In *WWW 2017*.

[41] Xu Zhang, Felix X. Yu, Sanjiv Kumar, and Shih-Fu Chang. [n.d.]. Learning Spread-Out Local Feature Descriptors. In *ICCV 2017*.

[42] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. [n.d.]. Recommending What Video to Watch next: A Multitask Ranking System. In *RecSys 2019*.

[43] Kun Zhou, Haibo Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhong yuan Wang, and Jirong Wen. [n.d.]. S3-Rec: Self-Supervised Learning for Sequential Recommendation with Mutual Information Maximization. *CIKM 2020* ([n. d.]).

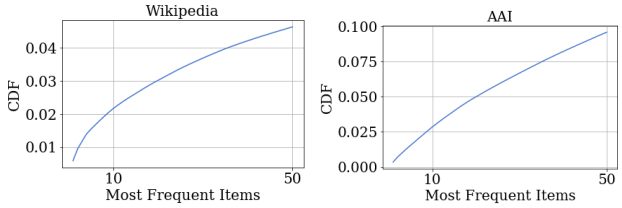


Figure 9: CDF of most frequent items in the Wikipedia and AAI datasets. The distribution is extremely dominated by popular items. For example, the top 50 items out of the 2.4M items already constitute 10% of data in the AAI dataset.

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

Table 5: Corpus sizes of the Wikipedia and the AAI datasets.

A APPENDIX

.1 Dataset Statistics. Table 5 shows some basic stats for the Wikipedia and AAI datasets. Figure 9 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 [11]) 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.

.2 Head and Tail Item Evaluation Dataset. We partition the full test dataset based on popularity of the groundtruth item. For the AAI test dataset, the *Head* dataset consists examples where the groundtruth items are in the top 10% most frequent items, and the rest of the test examples are treated as *tail*. For Wikipedia, we follow the data partitions in [23], where test examples containing items not included in the training set are treated as *Tail*, and the rest test examples are treated as *Head*.

.3 Visualization of Learned Embeddings. Besides better model performance, we expect the representations learned with SSL to have better quality than the counterparts without SSL. To verify our hypothesis, we take the app embeddings learned in the models trained on AAI dataset, and plot them using t-SNE plot in Figure 4. Apps from different categories are plotted in different colors, as illustrated in the legends in Figure 4. Compared to the apps in (Figure 4a), the apps in the best SSL model (Figure 4b) tend to group much better with similar apps in the same category, and the separation of different category looks much more clear. For example, we could see that the "Sports & Recreation" apps (in red) are mixed together with "Law & Government" and "Travel" apps in Figure 4a. While in Figure 4b, we clearly see the 4 categories of apps grouped together among themselves. This indicates that the representations learned with SSL carry more semantic information, and is also why SSL leads to better model performance in our experiments.