LEARNING FEDERATED REPRESENTATIONS AND RECOMMENDATIONS WITH LIMITED NEGATIVES

Lin Ning
Google Research
linning@google.com

Karan Singhal
Google Research
karansinghal@google.com

Ellie X. Zhou
Google
ezhou@google.com

Sushant Prakash
Google Research
sush@google.com

ABSTRACT

Deep retrieval models are widely used for learning entity representations and recommendations. Federated learning provides a privacy-preserving way to train these models without requiring centralization of user data. However, federated deep retrieval models usually perform much worse than their centralized counterparts due to non-IID (independent and identically distributed) training data on clients, an intrinsic property of federated learning that limits negatives available for training. We demonstrate that this issue is distinct from the commonly studied client drift problem. This work proposes *batch-insensitive losses* as a way to alleviate the non-IID negatives issue for federated movie recommendation. We explore a variety of techniques and identify that batch-insensitive losses can effectively improve the performance of federated deep retrieval models, increasing the relative recall of the federated model by up to 93.15% and reducing the relative gap in recall between it and a centralized model from 27.22% - 43.14% to 0.53% - 2.42%. We open-source our code framework to accelerate further research and applications of federated deep retrieval models.

1 Introduction

Deep retrieval models have been successfully deployed in recent years in many large scale recommendation systems [2, 13, 19, 17, 7, 16] and natural language tasks [5, 4, 18, 1]. While these models largely improve user experience by enabling personalization and representation learning, they can also raise privacy concerns as user data typically needs to be sent to a centralized server for training.

Federated learning (FL) is a decentralized training strategy in which clients collaborate with a coordinating server to train a machine learning model [14]. In this setting, FL provides opportunities to leverage client data that is not centrally collected to learn useful models while preserving user privacy. Bringing deep retrieval models to FL is a promising way to power recommendations and learn representations for users and items while addressing privacy concerns by moving towards reduced centralization of user data.

In this work we observe a challenge with training federated deep retrieval models: the insufficiency of negative examples available on a client's device. A deep retrieval model (shown in Figure 1) learns embeddings representing users' contexts and items like movies, songs, or websites. For the model to learn meaningful embeddings, it generally uses two types of examples: positive examples and negative examples. The positive examples pull embeddings of the training *(context, item)* pairs to be close together in an embedding space, while the negative examples push embeddings of unrelated pairs farther apart. Negative examples are typically required to prevent *embedding space collapse*, where learned representations collapse to a single point and are no longer informative.

Typically, negative pairs are produced by sampling items from the training data. In conventional centralized training, training data is usually assumed to be independent and identically distributed (IID). Therefore, negatives can be sampled reliably from the overall training distribution. However, the IID data assumption does not hold for federated learning since data is generated locally by clients based on their circumstances. That means, on each device, negatives may not present. And even if they are present, they are relatively few and relatively similar. In practice, we observe that this leads to significant performance degradation when FL is applied naively, beyond the typical degradation observed for *e.g.,* classification in the FL setting.

This work focuses on understanding and alleviating the non-IID issue for deep retrieval models. In this work, we:

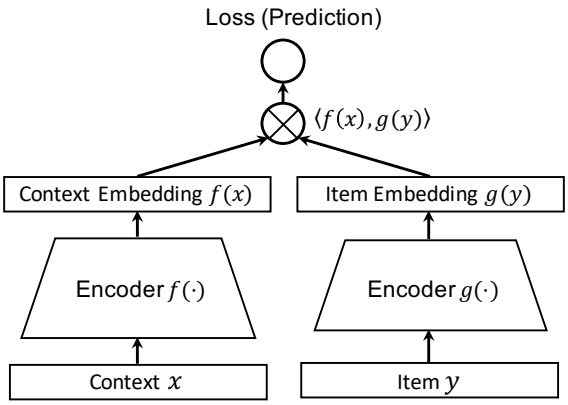

Figure 1: An illustration of a deep retrieval model.

- Observe that naive federated training of deep retrieval models causes an unusually steep performance drop.
- Perform ablations to show that performance degradation is primarily caused by sampling negatives from non-IID data, not the typical client drift issue or other aspects of federated training.
- Introduce *batch-insensitive losses* as a class of objectives that can alleviate the issue in this setting.
- Perform empirical evaluation of different training objectives in a movie recommendation setting, showing that batch-insensitive losses enable performant federated deep retrieval.
- Open-source our code framework to accelerate further research and practical applications of federated deep retrieval models. [1]

Related Work: There has been previous research on mitigating the non-IID data issue [6, 23, 10, 11, 12, 21, 20] for improving general model convergence in federated learning settings. These works study the general problem of heterogeneity in client data causing *client drift* when clients perform multiple local steps of computation, as in FEDAVG [9, 8]. However, some of them require sharing some training data across clients [6, 23] and most of them study image classification tasks. The developed techniques are not fully relevant to the deep retrieval model, which sees unusually severe performance degradation due to its explicit reliance on sampling negatives for training. This issue, though also caused by non-IID data, is orthogonal to the client drift issue: it occurs even when clients perform one local update step; the techniques we explore in this work can be combined with previous techniques for addressing client drift. [21] also studies the effectiveness of hinge loss and spreadout regularizer in federated learning. However, it focuses on classification tasks and only considers the extreme case, where each user has no access to any negative examples. Our focus is on a more realistic setting, aiming to understand and increase the performance of representation learning and item recommendation with deep retrieval models when each client has access to some, but limited and relatively similar negative examples. More broadly, we propose the concept of batch-insensitive losses, which can be generally useful when exploring techniques to alleviate non-IID data issues in other federated learning tasks.

2 Federated Deep Retrieval Model

As illustrated in Figure 1, a deep retrieval model consists of two encoders, which are typically deep neural networks. Depending on the task, each of the two encoders can be a fully-connected network, convolutional neural network, Transformer, and so on. The input consists of *(context, item)* pairs which are encoded by the left and right encoders respectively. For example, in a movie recommendation use-case, a context might be a sequence of previous movies a user has watched, and the item is the next movie that they watch. The goals of applying a deep retrieval model in this setting would be to learn a model that can predict the next movie given an unseen sequence of previous movies, and to learn encoders that produce embeddings for users and movies.

More formally, we denote the feature vectors representing contexts and items as $\mathbf{x}$ and $\mathbf{y}$. The two encoders are denoted as two parameterized functions $f(\cdot)$ and $g(\cdot)$, with $f(\mathbf{x})$ and $g(\mathbf{y})$ mapping $\mathbf{x}$ and $\mathbf{y}$ to a shared embedding space. The model outputs the similarity score between the encoded context and item, *e.g.,* the inner product of context and item embeddings, $s(\mathbf{x}, \mathbf{y}) = \langle f(\mathbf{x}), g(\mathbf{y}) \rangle$. A loss function is applied to enforce that positive examples (*i.e.,* similar context and item pairs) have high similarity, and negative examples have low similarity.

[1] https://git.io/federated_dual_encoder

The resulting model is able to predict relevant items given a new context. The representations produced by $f(\cdot)$ and $g(\cdot)$ are general representations of user contexts and items and can also be used for other downstream applications, such as classification [3, 1]. Deep retrieval models are also referred to as dual encoder, two-tower, or encoder-encoder models depending on the setting [4, 18, 1, 15, 3]. This general framework has been successfully deployed in a variety of real-world applications in embedding and recommendation learning.

Loss Function: Softmax cross-entropy loss over similarities is most commonly used to train a deep retrieval model [3, 17, 19]:

$$\ell(\mathbf{x_i}, \mathbf{y_i}) = -\log \frac{e^{s(\mathbf{x_i}, \mathbf{y_i})}}{\sum_{y_j \in \mathcal{N}} e^{s(\mathbf{x_i}, \mathbf{y_j})}}$$

where $\mathcal{N}$ is a set of negative labels used to construct negative example pairs $(\mathbf{x_i}, \mathbf{y_j})$. The model is incentivized to maximize similarity between positive example pairs and minimize similarity between negative example pairs. Note that if it only did the former, then the embedding space produced by $f(\cdot)$ and $g(\cdot)$ would collapse. Therefore, negative examples play an important role in learning a good model. A standard method for getting negatives is to use in-batch negatives [19, 17], which means given a training batch and any $(\mathbf{x_i}, \mathbf{y_i})$ pair in the batch, all other items $\mathbf{y_{j,j \neq i}}$ in the same batch are treated as negatives for $\mathbf{x_i}$. We refer to this as *batch softmax* below.

Recommendation Prediction: Once the parameterized functions $f(\cdot)$ and $g(\cdot)$ are learned, recommendation with a deep retrieval model consists of two steps: (1) given a context, compute the context embedding $f(\mathbf{x})$; (2) perform (approximate) nearest neighbor search over a set of item embeddings computed with $g(\cdot)$ and output the top-K most relevant items.

Applying federated learning to training a deep retrieval model involves three main steps in each training round. First, a central server sends the current model to a number of randomly sampled clients. Second, each sampled client trains on its own dataset and updates its model locally. Finally, the model updates from the sampled clients are sent back to the server and aggregated to update the server model. The de-facto standard federated optimization method is FEDAVG [14]; each client updates its model multiple times before sending the model update back to the server to be averaged. We also later refer to FEDSGD, a simpler variation where each client only runs a single local training step at each training round, similar to standard distributed training.

As we discuss in Section 4.2, naively applying federated learning to this setting produces a steep drop in performance, worse than is typical when comparing centralized and federated performance. We will see that this is due to clients having non-IID negatives, which causes training to deteriorate. Note that this is distinct from the *client drift* phenomenon discussed in other works [9], which causes slight performance deterioration when clients take multiple local steps. In contrast, the problem we observe occurs even in the FEDSGD regime (see Figure 3). Our contribution in this work is to better characterize this problem and propose methods that enable federated deep retrieval models to perform comparably to centralized counterparts.

3 Methods

To understand and address the non-IID data issue for training federated deep retrieval models, we explored three different techniques: spreadout regularizer [22], a batch-insensitive loss, and negative sampling. Each technique modifies the federated training objective to prevent performance degradation due to non-IID negatives.

3.1 Spreadout Regularization

The spreadout regularizer [22] is used to maximize the spread of embeddings in an embedding space. It ensures that the embeddings are orthogonal to each other to prevent collapse. Given an embedding vocabulary V and the corresponding embedding weights W, spreadout regularizer can be formulated as

$$\ell_{sr}(W) = \sum_{v \in V} \sum_{v' \neq v} (-d^2(w_v, w_{v'})),$$

where d is a measure of distance, *e.g.,* Euclidean distance or negative dot product. When combined with L2 normalization, this objective can be interpreted as pushing embeddings in W apart on a hypersphere. It can be used with any loss function (*e.g.,* softmax loss) in the form of

$$\ell_{(\cdot)\cdot sr} = \ell_{(\cdot)} + \alpha \ell_{sr},$$

where $\ell_{(.)}$ is the original loss function, ℓ_{sr} is the spreadout regularizer, and α trades off the regularization term and the original loss. In this work, we apply spreadout regularizer with different original loss functions to test whether it can enable learning using non-IID negatives or even no explicit negatives without embedding space collapse.

3.2 Batch-Insensitive Losses

We propose studying *batch-insensitive losses* as a simple potential solution to alleviate the non-IID negatives issue. These are loss functions with the following natural property:

$$\ell_{BI}(\mathbf{X}, \mathbf{Y}) = \frac{1}{N} \sum_{i=0}^{N} \ell_{BI}(\mathbf{X_i}, \mathbf{Y_i}),$$

where $\mathbf{X}, \mathbf{Y}$ represent a batch of N examples. This property intuitively means that a loss function applied over several batches of data in parallel (not in sequence) produces the same average loss and gradient update no matter how the individual examples are batched. When applied in the federated setting, we can show that a batch-insensitive loss produces the same final server update at the end of a training round when clients do one step of local training (FEDSGD from [14]), no matter how data is split across clients. Moreover, this final server update is exactly the same as the update from doing a large batch of centralized SGD containing all of the data from the round's clients. The latter *user-clustered SGD* setting approximates standard large-batch SGD as the number of clients per round increases.

Note that the typical batch softmax loss used for training deep retrieval models (described in Section 2) does not have this property and is batch-sensitive. We now describe specific batch-insensitive losses for the deep retrieval setting.

3.2.1 Hinge Loss + Spreadout Regularizer

This loss is a variation of contrastive loss, composed of a positive term pushing positive examples together (hinge loss) and a negative term preventing embedding collapse (spreadout regularization). This combination was first introduced in [21].

Given a positive example pair ($\mathbf{x}$,$\mathbf{y}$), where $\mathbf{x}$ and $\mathbf{y}$ are context and item features, the hinge loss is defined as

$$\ell(\mathbf{x}, \mathbf{y}) = \max(0, \beta - f(\mathbf{x}) \cdot g(\mathbf{y}))^2,$$

where $f(\mathbf{x})$ is the context embedding, $g(\mathbf{y})$ is the item embedding, and β is a tunable margin set to 0.9 in this work. Unlike softmax cross-entropy, hinge loss only considers positive examples; the loss can be trivially minimized by collapsing embeddings into a single point. To avoid collapse, we apply spreadout regularizer to the model's shared embedding table described in Section 4.1, pushing items in the embedding vocabulary to have orthogonal embeddings. The resulting combined loss pushes positive pairs closer while pushing negatives apart, resulting in a full loss for deep retrieval. Importantly, both terms in the loss are batch-insensitive, so the combined loss is also batch-insensitive.

3.2.2 Negative Sampling

We also study the effectiveness of sampling negatives for federated deep retrieval training. In the setting described in Section 4.1, it is possible to use the full item vocabulary as negatives. This gives the *global softmax* loss:

$$\ell(\mathbf{x_i}, \mathbf{y_i}) = -\log \frac{e^{f(\mathbf{x_i}) \cdot g(\mathbf{y_i})}}{\sum_{y_j \in V} e^{f(\mathbf{x_i}) \cdot g(\mathbf{y_j})}}$$

where V is the vocabulary of possible items to predict. Global softmax is an extreme case of negative sampling, where all the items in the item vocabulary other than the one in a positive (context, item) pair are used as negative examples to calculate the softmax loss. For a larger scale model, only a subset of items can be sampled randomly instead. In either case, the loss is batch-insensitive on expectation.

4 Evaluation

To evaluate the efficacy of the proposed techniques on the non-IID negatives issue, we experiment with a simple deep retrieval model on a movie recommendation task with the MovieLens 1M dataset [2]. The model takes in a user's movie watching history and predicts a relevant next movie for this user.

[2] https://grouplens.org/datasets/movielens/1m/

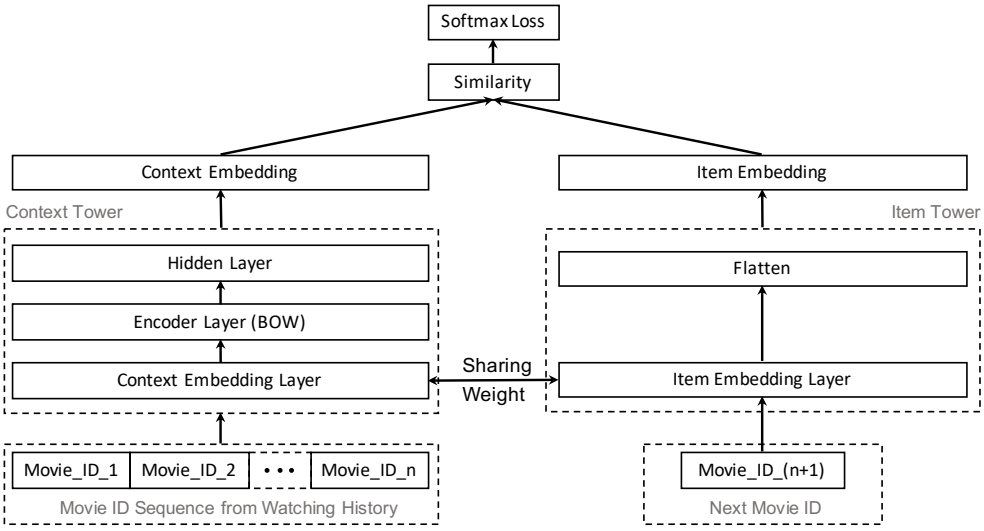

Figure 2: ID-based deep retrieval model for movie recommendation.

4.1 Experiment Setting

Below we describe details on the task, dataset, and model. Additional details can be found in our open-source code framework for federated deep retrieval, detailed in Section 5.

Dataset As shown in Table 1, the MovieLens 1M dataset contains approximately 1 million ratings from 6040 users on 3952 movies. Only the movie IDs rated by each user, sorted by timestamp, are used as the training data. Examples are created by taking sequential "windows" of the movie sequence for each user, resulting in context inputs containing 10 movie IDs and item inputs representing one next movie ID. For centralized training, examples are randomly shuffled across all users and split to train and test datasets. The train dataset has 894,752 examples and the test dataset has 99,417 examples. We refer to these as *centralized datasets* later in this section. For federated training, all examples are grouped by user, forming a natural data partitioning across clients. The train and test examples are split by user ids. There are 4832 train users and 605 test users. We refer to these as *federated datasets*.

Model Architecture Figure 2 illustrates the architecture of the ID-based deep retrieval model used for experiments. It takes in a sequence of movie IDs (the movie watching history) as the context and the next movie ID as the item to form the *(context, item)* pair. The context encoder is a bag-of-word encoder and the item tower is a simple embedding lookup tower. The two towers share the same bottom embedding layer, which maps from movie ID to dense embedding. The two towers generate context and item embeddings respectively, and similarity is enforced between the positive (context, label) pairs. To make the comparison consistent and fair, we set the batch size to 16 for all experiments. The output dimension of the shared embedding layer is 16. The encoded context embedding and item embedding are also 16-dimensional and L2-normalized.

Federated and Centralized Experiments One of our goals in this work is to reduce the gap between federated and centralized deep retrieval model performance. We run experiments for both centralized and federated training and compare performance. Interestingly, we observe that the proposed methods can also improve centralized performance, so we also compare against this. We refer to these models as *Improved Centralized* later in this section. To study federated model performance, we build a general deep retrieval code framework on top of TensorFlow Federated (TFF)[3], described in Section 5. In both settings, we measure test recall@k for $k \in [1, 5, 10]$, the fraction of examples for which the correct next movie is within the top k nearest item embeddings for an unseen context.

4.2 Effect of Non-IID Data

To study the effect of non-IID data on federated learning in this setting, we train the ID-based deep retrieval model on federated datasets using FEDAVG [14]. The model is trained with the standard batch softmax cross-entropy loss. We

[3] https://www.tensorflow.org/federated

	Centralized	Federated
Split Strategy	By Example	By User
Train Data	894,753 E	4832 U
Test Data	99,417 E	605 U
Ratings	1,000,209	
Users	6040	
Movies	3952	

Table 1: MovieLens 1M Dataset Statistics. 'E' is short for 'Examples' and 'U' is short for 'Users'.

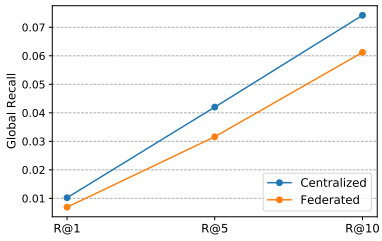

Figure 3: Model performance with batch softmax loss. Centralized: centralized training on IID data. Federated: FEDSGD on non-IID data.

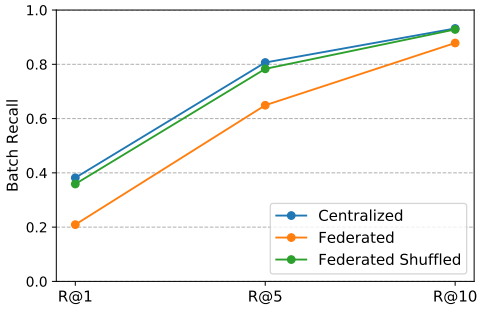

Figure 4: Model performance with batch softmax loss in different training settings. Centralized: centralized training on IID data. Federated: FEDAVG on non-IID data. Federated Shuffled: FEDAVG on IID data.

compare recall across items within a batch (batch recall) with centralized training. As shown in Figure 4, the federated model experiences significant performance degradation, especially for recall@1. It is worth noting that performance degradation occurs even when training with FEDSGD, as illustrated in Figure 3. This indicates that client drift, which occurs when clients take multiple local steps with non-IID data, is not the only cause.

To test whether this performance degradation is caused by non-IIDness, we train a *Federated Shuffled* federated model where data is IID across clients. In this experiment, all examples are shuffled across users while the number of examples per user remains the same; this results in the same number of local steps taken on each client as before. This is a way of isolating the effect of non-IID data on federated learning. As shown in Figure 4, the Federated Shuffled result roughly matches the centralized result. We can then conclude that non-IIDness causes the performance degradation, not federated training in itself.

However, though shuffling examples across users could resolve the non-IID data issue, we cannot do it in practice due to privacy and communication limitations.

4.3 Federated and Centralized Results

We train both centralized and federated models with four different losses: batch softmax (BS), batch softmax with spreadout regularizer (BS+S), hinge loss with spreadout regularizer (H+S) and negative sampling (NS). We compare recall calculated globally across all items, which has no dependence on examples in the batch.

Batch Softmax (BS) Batch softmax is the standard loss function for training a deep retrieval model. The softmax cross-entropy loss is calculated with in-batch negatives as described in Section 2. As shown in Figure 5(a), there is a large gap between centralized and federated global recall, similar to the batch recall results shown in Section 4.2.

Batch Softmax + Spreadout (BS+S) Figure 5(b) presents results of training with batch softmax combined with spreadout regularization. With spreadout regularizer, the recall values of the federated model almost match those of the batch softmax centralized model, which is trained without spreadout regularizer. However, spreadout regularizer also improves the centralized training. The federated model still performs significantly worse compared to the improved centralized model.

The results indicate that spreadout regularization itself is not enough to solve the non-IID negatives issue. Since batch softmax loss still depends on in-batch negatives, it can still lead to worse model quality. Therefore, we need a loss function that is less affected by the training data distribution, motivating batch-insensitive losses.

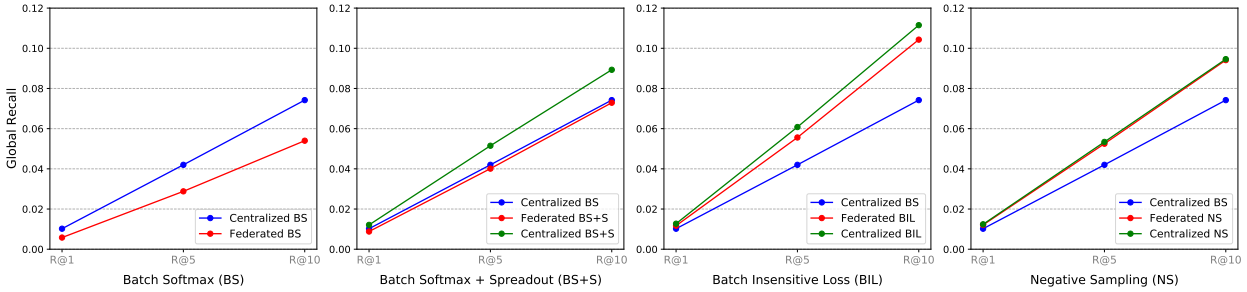

Figure 5: Comparison of centralized and federated models when training with different loss functions: batch softmax, batch softmax with spreadout regularizer, hinge loss with spreadout regularizer, and negative sampling.

Table 2: An overall recall comparison between centralized, improved centralized and federated models. The performance drop is calculated as $(R_c - R_f)/R_c$, where R_c is the centralized global recall and R_f is the federated global recall.

	Centralized	Improved Centralized			Federated				Performance Drop			
	BS	BS+S	H+S	NS	BS	BS+S	H+S	NS	BS	BS+S	H+S	NS
R@1	1.02	1.21	**1.27**	1.24	0.58	0.88	1.17	**1.21**	43.14%	27.27%	7.87%	**2.42%**
R@5	4.2	5.15	**6.08**	5.34	2.88	4.01	**5.56**	5.25	31.43%	22.14%	8.55%	**1.69%**
R@10	7.42	8.93	**11.15**	9.46	5.4	7.29	**10.43**	9.41	27.22%	18.37%	6.46%	**0.53%**

Hinge Loss + Spreadout (H+S) As explained in Section 3.2.1, we use the combination of hinge loss and spreadout regularizer as a batch-insensitive loss. From Figure 5(c), we observe that both the improved centralized model and the federated model perform much better than the baseline model. Also, the gap between improved centralized and federated models is much smaller than with batch softmax. The batch-insensitive loss appears to effectively alleviate the performance degradation caused by non-IID negatives.

Negative Sampling (NS) We use softmax calculated globally across all movies as motivated in Section 3.2.2, which provides a batch-insensitive loss. The results are shown in Figure 5(d). For this extreme case of negative sampling, the federated model performs almost the same as the improved centralized model, and both perform significantly better than the batch softmax centralized model. This indicates again that batch-insensitive losses are promising for the non-IID negatives issue.

We caution that applying global softmax may not be appropriate in all settings. In these experiments, we use all the items in the movie vocabulary other than the label as the negatives. In practice, when we deal with items with large or unbounded vocabulary size, we may need other strategies to sample negatives.

4.3.1 Overall Comparison

Table 2 gives an overall performance comparison between centralized, improved centralized, and federated models under different losses.

Training with batch-insensitive losses (H+S, NS) achieves the highest recall for both centralized and federated models. Hinge loss with spreadout regularizer in particular appears to perform slightly better than negative sampling in terms of absolute recall, but both perform significantly better than batch-sensitive losses (BS, BS+S).

We also observe that negative sampling incurs the smallest performance gap between centralized and federated training. Hinge loss with spreadout regularizer has the next smallest performance gap, and batch-sensitive techniques have a larger performance gap as expected. We expect that the remaining performance drop between centralized and federated models is a result of client drift (clients are still taking multiple local steps); this suggests that combining batch-insensitive losses with approaches to address client drift may be a promising future direction.

5 Open-Source Framework

We are releasing a general code framework for experimenting with federated deep retrieval models built on the TensorFlow Federated library. The code is released under Apache License 2.0. The framework enables reproduction of

our experiments and provides a flexible, well-documented interface for researchers to train federated and centralized deep retrieval models with different models and losses. We provide libraries for training and evaluation for MovieLens next movie prediction, which can be easily extended for new tasks. We hope that this framework spurs further research and lowers the barrier to more practical applications.

6 Conclusion

This work investigates the effect of non-IID negatives on federated training of deep retrieval models and proposes batch-insensitive losses to alleviate the issue. We compare model performance using a variety of techniques and show that batch-insensitive losses produce better federated deep retrieval models that can approximately match centralized models. We also open-source our code framework to accelerate future research and applications. Note that our proposed techniques do not directly address the separate, well-studied issue of client drift when clients do multiple steps of local training–techniques addressing this issue are complementary and can be combined with our work. Another direction is extending negative sampling to models predicting over large or unbounded sets of items, *e.g.,* by learning a privacy-preserving generative model of negatives to sample from.

References

[1] Muthuraman Chidambaram, Yinfei Yang, Daniel Cer, Steve Yuan, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Learning cross-lingual sentence representations via a multi-task dual-encoder model. *arXiv preprint arXiv:1810.12836*, 2018.

[2] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, New York, NY, USA, 2016.

[3] Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. Learning dense representations for entity retrieval. *arXiv preprint arXiv:1909.10506*, 2019.

[4] Daniel Gillick, Alessandro Presta, and Gaurav Singh Tomar. End-to-end retrieval in continuous space. *arXiv preprint arXiv:1811.08008*, 2018.

[5] Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. Efficient natural language response suggestion for smart reply. *ArXiv e-prints*, 2017.

[6] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. The non-iid data quagmire of decentralized machine learning. In *International Conference on Machine Learning*, pages 4387–4398. PMLR, 2020.

[7] Jyun-Yu Jiang, Tao Wu, Georgios Roumpos, Heng-Tze Cheng, Xinyang Yi, Ed Chi, Harish Ganapathy, Nitin Jindal, Pei Cao, and Wei Wang. End-to-end deep attentive personalized item retrieval for online content-sharing platforms. In *Proceedings of The Web Conference 2020*, pages 2870–2877, 2020.

[8] Sai Praneeth Karimireddy, Martin Jaggi, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. Mime: Mimicking centralized stochastic algorithms in federated learning. *arXiv preprint arXiv:2008.03606*, 2020.

[9] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[10] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

[11] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *International Conference on Learning Representations*, 2020.

[12] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. *arXiv preprint arXiv:2102.07623*, 2021.

[13] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1930–1939, 2018.

[14] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[15] Pavel Sountsov and Sunita Sarawagi. Length bias in encoder decoder models and a case for global conditioning. *arXiv preprint arXiv:1606.03402*, 2016.

[16] Maksims Volkovs, Guang Wei Yu, and Tomi Poutanen. Dropoutnet: Addressing cold start in recommender systems. In *NIPS*, pages 4957–4966, 2017.

[17] Ji Yang, Xinyang Yi, Derek Zhiyuan Cheng, Lichan Hong, Yang Li, Simon Xiaoming Wang, Taibai Xu, and Ed H Chi. Mixed negative sampling for learning two-tower neural networks in recommendations. In *Companion Proceedings of the Web Conference 2020*, pages 441–447, 2020.

[18] Yinfei Yanga, Steve Yuanc, Daniel Cera, Sheng-yi Konga, Noah Constanta, Petr Pilarc, Heming Gea, Yun-Hsuan Sunga, Brian Stropea, and Ray Kurzweila. Learning semantic textual similarity from conversations. *ACL 2018*, page 164, 2018.

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

[20] Tehrim Yoon, Sumin Shin, Sung Ju Hwang, and Eunho Yang. Fedmix: Approximation of mixup under mean augmented federated learning. In *International Conference on Learning Representations*, 2021.

[21] Felix X. Yu, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. Federated learning with only positive labels, 2020.

[22] Xu Zhang, Felix X. Yu, Sanjiv Kumar, and Shih-Fu Chang. Learning spread-out local feature descriptors, 2017.

[23] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.