Exploiting Domain Knowledge For Efficient Data-Centric Session-Based Recommendation Model

Mayank Mishra
TCS Research
Mumbai, India 400607
mishra.m@tcs.com

Rekha Singhal
TCS Research
Mumbai, India 400607
rekha.singhal@tcs.com

Abstract

Traditionally, deep learning models' training involves choosing complex network architectures and large data sets to build models with high accuracy. This kind of training demands a large high-performance computing infrastructure to complete the training process in a reasonable time. We explore a data-centric approach to choose the "right" data samples in the "right" amount during each epoch of a model's training to build the model efficiently in a shorter time and at least with the same (sometimes even better) accuracy compared to the traditional approach. This paper presents our experience using domain knowledge (temporal nature of data) to build a recommendation model by reducing data samples used in successive training epochs. We show that using a data-centric approach on state-of-the-art session-based recommendation models can reduce the model training time by at least $2.1\times$ and achieve slightly better accuracy and "average recommendation popularity" on a publicly available data set containing 6 million sessions.

1 Motivation

Data is the main food for any deep learning pipeline to build data-driven models. Unless we choose the *right* food in *right* quantity and *right* order, an AI pipeline will not be efficient in terms of the model complexity and its training time. The shift in focus towards quality of data than the model architecture is referred to as the data-centric approach.

Authors in [1] and [2] have shown the benefits of a data-centric approach to building models for chest ailments. In [3] and its successive implementations, Snorkelflow, Drybell from Google present an efficient mechanism of auto-labeling data samples with good labels (taking inputs from subject matter experts) for training. Further, [4] presents a minimum number of data samples required to label to get good accuracy, in turn reducing the training time. [5] presents a mechanism of selecting features having more information and combine it with a simple Random Sampling technique without replacement to reduce the training data. However, none of them explore the dynamic reduction of data samples across epochs of the model's training process.

In this paper, we share our experience in exploiting domain knowledge of data to reduce data used in the consecutive epoch of the training process and hence training time. We use the sensitivity of the session-based recommendation model towards the temporal nature of data to feed the right number of data samples in the right order (temporal, unlike the traditional approach of shuffling) to reduce the number of data samples needed in each epoch. This helped us improve the model's accuracy and ARP (average recommendation popularity score) by avoiding a complex model and faster training time even on frugal hardware such as CPU. The side effect of this training approach enables the usage of critical data samples more often, hence improving the model accuracy. The paper is organized as follows. Section 2 presents briefly the recommendation model we have used to showcase the data-centric approach. Section 3 discuss our data-centric approach with the experimental results

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

to reduce the training time while improving the accuracy by reducing data samples in each epoch. Finally, we conclude in Section 4.

2 Session based Recommendation Model

A session-based recommender system [6] predicts the next item the customer will act upon (click, buy, etc.) in a session. Here, a *session* (Figure 1) is defined as a sequence of items on which the user has acted within some specified duration of time. A few of the examples of such models are NISER [7] (mentioned as NISER+ in paper) and BCD4Rec [8], which are used in our evaluation. The idea is to learn the sequential patterns such as "which items are clicked in succession" or "which set of items are usually clicked in the same session".

The popularity distribution of items (the number of sessions in which an item is present) contains a log tail. Items in the tail are there because of two reasons. The first reason is that the item is unpopular. The second reason is that the item is newly introduced and hence has not appeared in historical sessions. Even if an item is newly popular, for example, breathing masks were suddenly in demand when the Covid pandemic impacted the world; they can still get overshadowed in recommendations due to historically popular items, say toilet paper. This is because the data set contains more samples where the popular items are present (imbalance). Recommender systems try to overcome this *popularity bias* challenge by bringing more complexity to model architecture.

As the session contains multiple items over-sampling/under-sampling of sessions, balancing popular/unpopular items affects the distribution of other items present in the same sessions. Moreover, removing the popular item from the session also does not help balance, affecting the sequence pattern (which models are trying to learn). Any balancing approach for session-based data set must preserve the individual session. In the next section, we present a novel data balancing approach based on selective data reduction during the training of the session-based recommendation model to reduce the imbalance between historically popular and newly popular items. The proposed approach also reduces the time required to train the model and preserve the accuracy scores. We employ NISER [7]

Time Stamp	Session
t1	p1, p233, p91, p414
t2	p90, p21, p1, p53, p69, p334
	...
tn	p580, p4433, p2908, p912

Figure 1: Processed Session based data set.

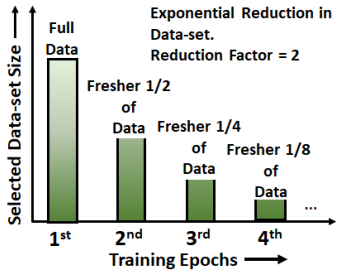

Figure 2: Reducing data every epoch.

as the session-based recommender model to discuss our approach. NISER is among the top-rated [9] [10] session-based recommendation model [1]. NISER uses GNN to model the products clicked in each session and predicts the most probable 20 items customer is likely to click.

3 Data-Centric Approach for Training

We train NISER using the publicly available session-based data sets [11], [12], which are also used by the authors of NISER (and others [13], [10]) after pre-processing. Each ready data set contains sessions arranged in the temporal order. We propose a data reduction-based training approach to reduce data continuously across consecutive epochs by exploiting the domain knowledge.

3.1 Data Reduction Strategy

Figure2 shows the proposed approach of data reduction (called *D-Red* in short). Instead of using complete data containing all the sessions in every training epoch, we only use the fresher part of

[1] https://paperswithcode.com/paper/niser-normalized-item-and-session#code

the data in every epoch. The newer (fresher) sessions model the current choices of the customers better than the older sessions. Fortunately, the session-based data is inherently temporal, with fresher sessions lying towards the end of the data set sorted on time. This is denoted by darker green color in the Figure 2. It should be noted that data is not shuffled before reducing it across epochs. Also, we create training batches in sequence, unlike the traditional training process, where a batch is chosen at random by shuffling the data set.

Although, Figure 2 only shows exponential data reduction per epoch, there can be other approaches (linear reduction) too. Depending on the data size, the number of data reduction steps is also capped so that data is not arbitrarily reduced. In this paper, we only discuss the exponential data reduction approach. The reduction in the overall amount of data used for training reduces the training time. Theoretically, suppose the data size is sufficiently large (allowing many data reductions). In that case, the overall training time can be reduced to the time which 2 epochs of training take when full data is used. This can be understood as follows. If a given data set is of size D_S and the model requires N epochs for training, then training time is proportional to $N \times D_S$. By following the ***D-Red*** approach, every successive epoch takes half the time of the previous epoch. Thus, the overall time is proportional to $D_S + \frac{D_S}{2} + \frac{D_S}{4} + \frac{D_S}{8} + ... = 2 \times D_S$. The batch size is kept constant.

3.2 Evaluation Metrics employed

We use metrics Recall@20 and Mean Reciprocal Rank (MRR@20), as used in [7]. Recall@20 represents the fraction of recommendations that have the desired item in the top 20 recommended items, and MRR@20 captures the average of reciprocal ranks of the desired item in the recommendations.

Measuring Popularity bias of the Recommender: We used the Average Recommendation Popularity (ARP) metric presented in [7] to measure the popularity of recommended items. We employ the ARP metric to compare the popularity of recommendations provided by models trained with the traditional approach and the D-Red approach. ARP is defined as $ARP = \frac{1}{|S|} \sum_{s \in S} \frac{\sum_{i \in L_s} \phi(i)}{K}$. where, S is the set of test sessions, and L_s is the list of top K recommended items for a test session s drawn from S. The popularity score of the recommended item list L_s is the average individual popularity of items in the list. The individual popularity score of an item i is represented by $\phi(i)$, which is the number of times i appears in training data. The ARP is the average of popularity scores of recommended item lists over a set of sessions in the test data set. Low ARP indicates lower popularity bias in recommendations.

3.3 Performance Results

Setup: We performed experiments using a Nvidia V-100 GPU and used 2 publicly available and widely used session based data sets YooChoose[11] and Diginetica[12] for training NISER model. We used the data preparation approach mentioned in [13], [7] and created YooChoose-1/4 (YC-1/4, fresher $1/4^{th}$ part of full dataset), YooChoose-1/64 (YC-1/64, fresher $1/64^{th}$ part of full dataset), and Diginetica (Digi) data sets. Atleast, 3 runs were performed for every setting.

Results: Table 1 compares the traditional training approach with the proposed ***D-Red*** approach. Using exponential data reduction, the ***D-Red*** approach is more than $2\times$ faster than the traditional approach. For the YC-1/4 data set, the speedup achieved was $2.13\times$ as the training time was reduced from 8401 to 3926 seconds. Similar speedup were observed for BCD4REC [8]. However, BCD4REC model training involves just two epochs; therefore, about 25% of the training time was saved from 6400 to 4824 seconds. Due to space limitations, the detailed results for the BCD4REC model are not presented in this paper. The Recall@20 and MRR@20 scores for the YC-1/4 dataset showed a marginal increase. The Recall@20 score increased from 72.6 to 72.87, and MRR@20 increased from 32.15 to 32.66. This shows that the data reduction approach selected the data containing more valuable patterns for the recommendation use case.

For the YC-1/64 and Digi, the training time is reduced by $2.14\times$ and $2.06\times$, respectively, using ***D-Red***. However, the Recall@20 and MRR@20 scores also reduced between 1% to 2%. This reduction in prediction accuracy is not in line with the observation when the YC-1/4 data set is used for training. It appears that the reduction in training data is adversely affecting the training process, and the training data might be insufficient. A probable reason for the difference in the effectiveness of ***D-Red*** for YC-1/64 and Digi data sets compared to YC-1/4 is their size. YC-1/4 is $16\times$ larger

Table 1: Traditional training approach vs. D-Red approach (Case study on NISER [7])

		YC-1/4	YC-1/64	Digi
Number of training, *test* sessions		5.91M, *55.8K*	0.37M, *55.8K*	0.72M, *60.8K*
Traditional Approach (μ: mean, σ*:std dev.*)	Training Time	8401s	540s	830s
	Recall@20(μ, σ)	72.65, *0.063*	71.45, *0.033*	54.87, *0.094*
	MRR@20(μ, σ)	32.10 *0.062*	31.66, *0.013*	19.27, *0.037*
	ARP(μ, σ)	11314,*101.82*	3282.6, *8.4*	340, *0.45*
D-Red Approach	Training Time	3926s	252s	402s
	Recall@20(μ, σ)	72.92, *0.07*	70.26, *0.064*	52.69, *0.078*
	MRR@20(μ, σ)	32.66, *0.007*	31.02, *0.025*	18.41, *0.06*
	ARP(μ, σ)	11191, *26.87*	3256, *10.9*	325.6, *0.8*
Training Speedup		$2.13\times$	$2.14\times$	$2.06\times$

than YC-1/64 and almost $8\times$ larger than Digi. Moreover, YC-1/64 is the most recent $1/16^{th}$ part of YC-1/4. Due to these characteristics, these smaller data sets contain mostly newly popular items rather than historically popular items. Hence when the data is reduced, rather than balancing between the historically popular and newly popular items, the data instances consisting of newly popular items were reduced. Hence, the prediction accuracy is adversely affected.

Impact on recommender's popularity bias: Table 1 lists the ARP scores for both the traditional and *D-Red* approaches. Interestingly, *D-Red* reduces the ARP score of the recommendations made. The reduction is not extreme; however, a point to note is that we are not changing the model architecture (NISER). For YC-1/4, YC-1/64, and Digi the ARP was reduced by 1.1%, 1%, and 4% respectively.

Impact of data shuffling on prediction accuracy: We observed higher prediction scores than reported in [7] even for traditional approach. The reported Recall@20 and MRR@20 in the paper are 53.39 and 18.72, respectively for Digi, whereas we observed 54.87 and 19.27. Similarly, higher Recall@20, MRR@20 of $71.8, 31.77$ vs $72.65, 32.1$ for YC-1/4 and $71.27, 31.61$ vs $71.45, 31.66$ for YC-1/64. We found that the difference in the scores is because authors of [7] are shuffling the data before training commences, whereas we preserve the temporal order of sessions. We verified by enabling shuffling in our setup. Preserving the temporal order of sessions seems to better prepare the model for future sessions present in the test data set. The test data set contains a fresher portion of the overall data set using which training and testing data sets are derived.

3.4 Applicability of Data Reduction based Training Approach

The *D-Red* approach can not be applied for all different kinds of data sets. We used this approach to a session-based data set where the sessions are temporally ordered, i.e., the fresher sessions are available towards the end of the training data set. Moreover, fresher activity is a better representative of the customer choices and behavior than the older history. Thus, giving importance to fresher data (by data reduction) while training improves the recommendation performance.

The size of the data set also impacts the applicability of the data reduction-based training approach as mentioned earlier in Section 3.3. The importance of a sufficient amount of training data is well evident. It can also be seen in Table 1 where the prediction accuracy observed for the YC-1/4 data set is significantly higher than observed for YC-1/64 for both traditional and the proposed *D-Red* training approaches. However, using larger training data requires higher model training times which might become the bottleneck in cases like e-retail where the recommendation models must be kept fresh. Using the proposed data reduction-based training approach, a larger training data set can be used for training than the traditional training approach while keeping the same training time.

4 Conclusions and Future Work

The majority of the model-building community focuses on collecting many data samples and high-performance computing infrastructure to build data-driven models with high accuracy in a reasonable time. We have presented a novel data reduction approach for training a session-based recommender model, GNN based NISER [7] and offline RL based BCD4REC [8], in a shorter time with higher accuracy. The proposed approach reduces training time by a factor of $2.1\times$ with somewhat better accuracy and the ARP on the data sets employed.

References

[1] S. Jain, A. Smit, A. Y. Ng, and P. Rajpurkar, "Effect of radiology report labeler quality on deep learning models for chest x-ray interpretation," *CoRR*, vol. abs/2104.00793, 2021. [Online]. Available: https://arxiv.org/abs/2104.00793

[2] M. Ko, E. Chen, A. Agrawal, P. Rajpurkar, A. Avati, A. Y. Ng, S. Basu, and N. H. Shah, "Improving hospital readmission prediction using individualized utility analysis," *J. Biomed. Informatics*, vol. 119, p. 103826, 2021. [Online]. Available: https://doi.org/10.1016/j.jbi.2021.103826

[3] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," *Proc. VLDB Endow.*, vol. 11, no. 3, p. 269–282, Nov. 2017. [Online]. Available: https://doi.org/10.14778/3157794.3157797

[4] S. Kunde, M. Mishra, A. Pandit, R. Singhal, M. K. Nambiar, G. Shroff, and S. Gupta, "Recommending in changing times," in *RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22-26, 2020*, R. L. T. Santos, L. B. Marinho, E. M. Daly, L. Chen, K. Falk, N. Koenigstein, and E. S. de Moura, Eds. ACM, 2020, pp. 714–719. [Online]. Available: https://doi.org/10.1145/3383313.3418492

[5] R. Alamro and A. Youssef, "Impact of data reduction techniques on classification," in *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2018, pp. 1070–1075.

[6] S. Wang, L. Cao, Y. Wang, Q. Z. Sheng, M. A. Orgun, and D. Lian, "A survey on session-based recommender systems," *ACM Computing Surveys (CSUR)*, vol. 54, no. 7, pp. 1–38, 2021.

[7] P. Gupta, D. Garg, P. Malhotra, L. Vig, and G. Shroff, "NISER: normalized item and session representations with graph neural networks," *CoRR*, vol. abs/1909.04276, 2019. [Online]. Available: http://arxiv.org/abs/1909.04276

[8] D. Garg, P. Gupta, P. Malhotra, L. Vig, and G. Shroff, "Batch-constrained distributional reinforcement learning for session-based recommendation," *CoRR*, vol. abs/2012.08984, 2020. [Online]. Available: https://arxiv.org/abs/2012.08984

[9] S. Wu, F. Sun, W. Zhang, and B. Cui, "Graph neural networks in recommender systems: a survey," *arXiv preprint arXiv:2011.02260*, 2020.

[10] Z. Pan, F. Cai, W. Chen, H. Chen, and M. de Rijke, "Star graph neural networks for session-based recommendation," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 1195–1204.

[11] YooChoose, "Yoochoose dataset," http://2015.recsyschallenge.com/challenge.html, 2015.

[12] Diginetica, "Diginetica dataset," https://competitions.codalab.org/competitions/11161, 2016.

[13] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based Recommendation with Graph Neural Networks," in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, ser. AAAI '19, P. V. Hentenryck and Z.-H. Zhou, Eds., vol. 33, no. 1, Jul. 2019, pp. 346–353. [Online]. Available: https://aaai.org/ojs/index.php/AAAI/article/view/3804