Examining "Coresets over Multiple Tables for Feature-rich and Data-efficient Machine Learning"

JULIUS STUTZ, Otto-Friedrich University Bamberg, Germany

When a Machine-Learning (ML) model is trained to make accurate predictions it often required to have lots of input variables. When the data is presented in tables those input variables refer to the number of columns of the input table. Training with many input variables is called feature-rich ML. While it increases the effectiveness of the model it also significantly makes the training period longer due to the sheer amount of data needed. Furthermore, it requires lots of computational resources which, as well as the additional time needed, result in higher costs. To accelerate this process a small subset of tuples of the input table can be selected, such that this subset is representative of the whole input table. This subset is called coreset. Current algorithms can efficiently select a coreset but only when provided a single input table. When the input consists of multiple tables they have to be joined together which once again requires lots of resources.
Therefore Jiayi Wang and his colleagues propose RECON, an algorithm that is able to efficiently select a coreset from multiple tables without joining them together [Wang 2022; Wang et al. 2022]. The algorithm works by sampling tuples as candidates for the coreset, comparing them to the tuples of all input tables and then selecting the most representative tuple of the sample until the desired coreset size is reached. It also provides theoretical guarantees regarding a minimal difference of its gradient compared to the gradient of the theoretically joined together feature-augmented table.
Experiments and their evaluations show that models trained with a coreset created with RECON keep almost the same accuracy as models trained with the feature-augmented table while being significantly more efficient with example values of 13.3 minutes for RECON and 782 minutes for the latter one. These values include the creation of a coreset (if necessary) and the training of the model and were done with the same original dataset.

CCS Concepts: • **Computing methodologies** → *Machine learning approaches*; • **Theory of computation** → *Machine learning theory*.

Additional Key Words and Phrases: datasets, machine learning, coresets, feature-rich, data-efficient

1 INTRODUCTION

In modern times ML as a concept is present in lots of different domains. While the use cases might be entirely different they all share the commonality that training their model requires a lot of data. This data often comes in the form of tables, sometimes as a single table but often as multiple tables. In the latter case, the tables are then joined together to end up with a feature-rich dataset, meaning a dataset with a lot of important attributes. Furthermore one might use coresets of the data. A coreset is a small, representative subset of the whole dataset that captures the essential information necessary for training a model. The coreset selection is done by calculating weighted gradients to approximate the full gradient of the entire training dataset. In this context, a gradient refers to a derivative of a certain loss function with respect to the parameters present in the model. The goal is to find parameters such that this loss function is minimized as it measured the difference between the actual target values and the prediction the model makes. A minimized loss function thus leads to more accurate predictions.

Author's address: Julius Stutz, julius.stutz@stud.uni-bamberg.de, Otto-Friedrich University Bamberg, An der Weberei 5, Bamberg, Bavaria, Germany, 96049.

The problem with the above idea is that joining multiple tables together to achieve a feature-rich dataset is time-consuming. The join types that this paper discusses might add redundancy, both to the features as well as to the tuples. More concretely one-to-one, one-to-many, many-to-many, and fuzzy joins are considered. Using a coreset of the original data for the training process helps to reduce the computational power needed due to fewer data that has to be processed. On the other hand, it risks a slight loss of accuracy and takes a lot of computational power to be created for an augmented dataset.
To solve the join problem the goal of Wang et al. is to achieve a data-efficient dataset, meaning it reduces the time and needed computational power to train a model by using coresets, while also achieving a feature-rich dataset without actually joining the tables together. They do this by calculating a weighted gradient for each table by using a partial feature similarity value for each tuple in each respective input table. These are then all aggregated in the end to compute a gradient for the feature-rich augmented dataset. They've also proven that this estimated gradient is guaranteed to be upper-bounded and thus provides theoretical guarantees.

2 RELATED WORK

The field of feature-augmentation and coreset selection itself has already been studied independently from each other.
Feature-augmentation has many approaches such as simply joining tables, avoiding unnecessary joins that would just provide redundant information, or iterative feature augmentation. This approach involves selecting an optimal subset of tables in an iterative manner. The objective is to identify the tables that when augmented, contribute the most to enhancing the model's performance [Chepurko et al. 2020]. Coreset selection like discussed in Huang et al. [2021], Braverman et al. [2016], and Kirchhoff and Bilmes [2014] differs from the approach in this paper as their algorithms are only designed for a single table, rather than for multiple ones.

3 METHODS

3.1 Coreset selection framework

As mentioned in section 1 the use of gradients is common in ML. The gradient descent strategy is used for optimizing a loss function by using different inputs, in this case, parameters. With just one table it can be used to calculate the so-called full gradient of it by using an algorithm that starts with an empty coreset and then iterative goes through all tuples (rows) of the table and calculates their utility for the coreset, utility meaning how close this new tuple would bring the coreset gradient to the full gradient if it was included. The tuple with the highest utility will then be added to the coreset and the process will start once again till the desired coreset size has been reached. This way we obtain a coreset that has the minimal possible difference between the two gradients and thus can be used

for training a model with accurate predictions.

$$C = \underset{C \subseteq T, w_j \geq 0}{\arg\min} \max_{\theta \in \vartheta} || \underbrace{\sum_{i=1}^{n} \nabla l_i(\theta)}_{\text{full gradient}} - \underbrace{\sum_{j=1}^{|C|} w_j \nabla l_j(\theta)}_{\text{coreset gradient}} || \tag{3.1.1}$$

This can be seen in Eq. 3.1.1, where C refers to the coreset, T to the original table, $\max_{\theta \in \vartheta}$ to considering all possible parameters, n to the number of tuples in T, l_i to the gradient of a tuple in T, l_j to the gradient of a tuple in C and w_j to the weight of the tuple. Each tuple in T is uniquely mapped to a tuple in C. Thus the weight w_j of a tuple in C is the number of tuples in T that are mapped to w_j.
To use this approach for multiple tables there exist two solutions with the first one being the most obvious by first joining together all tables to obtain a single large feature-augmented table. The other approach is the one taken in the paper, without join materialization (actually conducting the joins) but rather by calculating the feature similarity inside each table (quantifying how similar or related the tuples within a table are to each other) which is followed by aggregating them. This yields an estimate for the utility of each group, where a group represents a subset of tuples that have common attribute values on specific attributes. Each group is formed, based on the attribute values of the tuples in the joined results, allowing one to identify tuples that exhibit similar characteristics.

3.2 Gradient approximation error bounded by groups

This section deals with proving that the degree to which the estimated coreset gradient differs from the actual gradient (gradient approximation error) is upper bounded by using groups. The proof will be done in three steps. In the first step, the groups are taken as given and a parameter θ is fixed. The given groups are disjoint and, as mentioned in 3.1, are similar in some attribute set A. The set of all disjoint groups is called G. Now each $G_i \in G$ is changed to not include the tuples from the feature-augmented table T^+ but rather the indices of the tuples, such that $\bigcup_{i=1}^{g} G_i = \{1, 2, \ldots, N\}$, where g refers to the number of groups in G ($|G|$) and N to the number of tuples in T^+ ($|T^+|$). With this advancement and further mathematical transformations which, if interested, can be further investigated in the original paper, one can now rewrite a part of Eq. 3.1.1 as seen in Eq. 3.2.1, which sets an upper bound on the gradient approximation error by considering the maximum difference in gradients between $c_j \in C$ (the coreset) and the tuples in each group. The computation of the second part of Eq. 3.2.1 can be efficiently performed without the need to join all the tables.

$$|| \sum_{i=1}^{n} \nabla l_i(\theta) - \sum_{j=1}^{|C|} w_j \nabla l_j(\theta) || \leq \sum_{i=1}^{g} |G_i| \min_{c_j \in C} \max_{k \in G_i} ||\nabla l_k(\theta) - \nabla l_i(\theta)|| \tag{3.2.1}$$

The proof now generalizes from a fixed parameter θ to the whole parameter space ϑ. This is done by utilizing the proofs done by colleagues that show that the gradient approximation error can be bounded regardless of the specific optimization problem or parameter under consideration in practical scenarios. The mathematical concepts of these proofs can be used to derive Eq. 3.2.2, where the Euclidean distance $||x_k - x_i||$ is used as a metric to quantify the similarity between the feature vectors of two tuples.

$$\max_{\theta \in \vartheta} \sum_{i=1}^{g} |G_i| \min_{c_j \in C} \max_{k \in G_i} ||\nabla l_k(\theta) - \nabla l_i(\theta)|| \tag{3.2.2}$$

$$\leq \underbrace{c}_{\text{constant}} \cdot \sum_{i=1}^{g} |G_i| \min_{c_j \in C} \max_{k \in G_i} ||x_k - x_i|| \tag{3.2.2}$$

Till now the groups have been taken as given and derived from the feature-augmented table T^+. The proof ends by stating that it is also possible to derive groups just by examining the single tables alone. This removes the computationally heavy task to join them together and thus allows to efficiently select a coreset. This last step of the proof is further examined in 3.3.

3.3 RECON algorithm

Wang et al. developed the RECON algorithm (feature-**R**ich and data-**E**fficient machine learning with **C**oreset selection with**O**ut join materializatio**N**) to efficiently select a coreset by just providing all single tables (with T as the base table, and $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$ to augment it) as well as a desired coreset size (K). Its workings can be seen in algorithm 1 and will now be explained.

Line 1 is about pre-computing a set D that includes the partial feature similarity vectors of every single table (T and $\mathcal{T}$). This process takes all tuple combinations inside of a single table and computes their Euclidean distance. As this step needs no further computations it can be pre-computed which is part of the reason the algorithm performs much faster than other methods.

The algorithm then initializes the coreset C as a set with zero elements in the beginning (line 2). It then starts a while-loop that reaches from line 3 to 15 and that is executed until the desired size K of the coreset C is reached.

Even though the feature-augmented table T^+ is not materialized through joins, some of its values are still needed to iterate through them (for-loop from line 5 - 12) and find the tuples with the highest utility. Therefore they are created in line 4 by sampling them using the table T and the ones in $\mathcal{T}$, together with a sampling strategy proposed by Zhao et al. [2018] that guarantees a uniform sample.

The utility that each tuple $t_j \in \mathcal{S}$ provides when added to C is initially set to zero (line 6) and then adjusted accordingly (line 9). This adjustment is done using the feature similarity computed in line 8. The feature similarity s_{ji} between a tuple $c_j \in C$ and each group G_i is determined by an algorithm that uses the concept of dynamic programming to calculate s_{ji} given the pre-computed values $d_{ij}^h \in D$, the group key that shows according to which features the groups are divided, and the join tree that displays which tables would be joined to which and on which feature (if the join materialization would take place). When the utility for each tuple $t_j \in \mathcal{S}$ is computed (line 12), the tuple with the highest utility is then added to C (line 13-14).

Algorithm 1: RECON algorithm

Input: Base table T, $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$, coreset size K
Output: Coreset $C \subseteq T^+$, weights $\mathcal{W} = \{w_1, w_2, \ldots, w_{|C|}\}$

1 $D = \{d_{ij}^h | \forall T_h \in \mathcal{T} \cup \{T\}, \forall t_i^h, t_j^h \in T_h, d_{ij}^h = ||x_i^h - x_j^h||\}$;
2 $C = \emptyset$;
3 **while** $|C| < K$ **do**
4 $\mathcal{S}$ = sampled subset from T^+ using T and $\mathcal{T}$;
5 **for** *each tuple* $t_j \in S$ **do**
6 $U(C \cup \{t_j\}) = 0$;
7 **for** *each group* $G_i \in G$ **do**
 /* s_{ji} is the minimal similarity between the tuples in group G_i and the feature vector of $c_j \in C$ */
8 s_{ji} = aggregate $d_{i'j'}^h \in D$ obtained from different tables;
9 $U(C \cup \{t_j\}) += |G_i| \max_{c_j \in C \cup \{t_j\}} s_{ji}$;
10 **end**
11 $U(t_j|C) = U(C \cup \{t_j\}) - U(C)$;
12 **end**
13 $t^* = _{t_j \in \mathcal{S}} U(t_j|C)$;
14 $C = C \cup t_j$;
15 **end**
16 **for** $j = 1$ **to** $|C|$ **do**
17 $w_j = \sum_{i=1}^{g} |G_i| \mathbb{I}[j = _{c_{j'} \in C} s_{j'i}]$;
18 **end**
19 **return** C, W;

After the coreset has been successfully created, one needs to compute the weight for each tuple in C (line 16-18). Each group G_i is uniquely mapped to a single tuple $c_j \in C$. Thus the weight w_j of c_j is computed as the sum of all groups G_i mapped to them times the size of each group respectively.

4 EXPERIMENTS AND EVALUATION

Wang et al. prove that RECON has a time complexity of $O(n^2d + nKS)$ while normal coreset selection when first joining the tables has $O(N^2D + NKS)$, where n refers to the average size of all tables in $\mathcal{T}$ ($n = |T_h|$), N to the size of T^+ ($N = |T^+|$), d to the average number of features of all tables in $\mathcal{T}$, D to the number of features of T^+, K to the number of times the coreset selection algorithm loops (coreset size), and S to the number of tuples that are sampled during each loop ($S = |\mathcal{S}|$). As $n < N$ is always true the time complexity of RECON is always lower than that of the normal coreset selection. Also, the time complexity looks at the worst case and provides an upper bound on the number of computations needed. However, in practice, because the data is typically distributed across different labels, we can avoid performing computations for the entire dataset. Instead, we can selectively work with smaller subsets of data that are relevant to specific labels. This leads to a substantial reduction in the overall computational requirements compared to the theoretical complexity estimation.

To compare the performance of RECON five datasets were chosen and other baseline techniques were used and their performance observed to compare them to RECON. The six other techniques used are Base (train model only with base table T), Full (train model with full augmented table T^+), Sample-Join (train model with sampled T^+ data without joins), Join-Coreset (train model with coreset over full augmented table T^+), Coreset-Join (train model by first creating the coreset of T and then joining it with Tables from $\mathcal{T}$), and factorized Machine Learning (FML) (accelerate batch gradient descent by breaking down the computations involved in machine learning tasks through the use of join operations). The chosen datasets with varying sizes can be seen in Table 1.

Table 1. Datasets used for comparison (c = classification, r = regression)

name	# tables	# rows (T^+)	# features (T^+)	task
Brazil	4	98 463	9	c
IMDB	7	674 466	41	c / r
IMDB-Large	7	21 303 410	41	c / r
Stack	3	6 347 553	178	r
Taxi	5	2 792 376	30	r

The comparison between the different techniques is based on two metrics, effectiveness and efficiency. Effectiveness is measured differently based on the task that the dataset is created for. A dataset like Brazil with a classification task deals with predicting discrete class labels or probabilities while datasets with a regression task like Stack involve predicting continuous numerical values. Thus the effectiveness of a regression dataset can be expressed via the root mean squared error (RMSE) while in the case of a dataset with a classification task, it is measured via the prediction accuracy of the model. The efficiency is purely measured by the time it takes to create a coreset (if necessary for the technique) and use the data to train a model. The goal of Wang et al was to create RECON such that it combines the values of the best techniques for effectiveness and efficiency respectively. As seen in algorithm 1 it takes the desired coreset size as an input. As it is not trivial to guess this number correctly, an iterative approach is used. The RECON algorithm is executed many times, each time with a slight increase as the input for the coreset size. After each execution, the model is validated and when the accuracy converges a coreset with the desired size is found. Although this approach seems to be time-consuming, it is not when compared to the other baselines. To have a fair comparison, Wang et al always set the environment up in a way that favors the technique currently compared with RECON. As an example the Sample-Join method is instantly provided with the number of tuples it should sample which is equal to the number of tuples RECON computed for its coreset. The experiments with the sample datasets and all discussed training techniques showed that RECON can always perform very close to the effectiveness of the Full and Join-Coreset methods which both are significantly more effective than the other methods used. This holds for both classifications, as well as regression tasks as can be seen in Figure 1. What makes RECON stand out, however, is that it is more efficient than Full and Join-Coreset and its time spent for training is rather comparable

Fig. 1. Effectiveness measured in accuracy for classification tasks (higher is better) and RSME for regression tasks (lower is better)

method	accuracy
Base	0.540
Sample-Join	0.577
Coreset-Join	0.574
Join-Coreset	0.605
Full	0.606
RECON	0.605

(a) Brazil

method	accuracy
Base	0.610
Sample-Join	0.628
Coreset-Join	0.624
Join-Coreset	0.663
Full	0.665
RECON	0.662

(b) IMDB

method	RMSE
Base	1.79
Sample-Join	1.55
Coreset-Join	1.61
Join-Coreset	0.99
Full	0.98
RECON	1.00

(c) IMDB-large

method	RMSE
Base	1.05
Sample-Join	0.98
Coreset-Join	1.04
Join-Coreset	0.75
Full	0.74
RECON	0.75

(d) Stack

method	RMSE
Base	1.82
Sample-Join	1.66
Coreset-Join	1.57
Join-Coreset	0.77
Full	0.77
RECON	0.77

(e) Taxi

Fig. 2. Efficiency measured in time spent (lower is better)

method	time (min)
Base	0.5
Sample-Join	0.8
Coreset-Join	0.7
Join-Coreset	2.5
Full	18.5
RECON	1.1

(a) Brazil

method	time (min)
Base	11
Sample-Join	12
Coreset-Join	11
Join-Coreset	21
Full	150
RECON	16

(b) IMDB

method	time (min)
Base	2.1
Sample-Join	8.2
Coreset-Join	9.4
Join-Coreset	610
Full	782
RECON	13.3

(c) IMDB-large

method	time (min)
Base	4.3
Sample-Join	5.8
Coreset-Join	6.1
Join-Coreset	28
Full	254
RECON	12.7

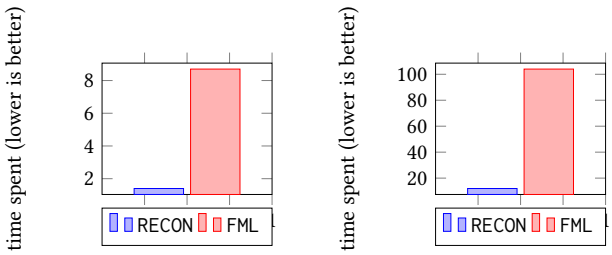
(d) Stack

method	time (min)
Base	0.52
Sample-Join	2.8
Coreset-Join	3.4
Join-Coreset	9.6
Full	72
RECON	4.6

(e) Taxi

to the one of Base, Sample-Join, and Coreset-Join. For demonstration observe the dataset Stack where RECON has an RMSE of 0.75 and Join-Coreset and Full one of 0.75 and 0.74 respectively. The RMSE of the other techniques are higher and they can not stand the comparison to the ones above. Meanwhile, the time spent for creating a coreset (if necessary) and training a model with it is significantly higher for Join-Coreset and Full with values of 28 minutes and 254 minutes respectively while RECON is more efficient with a value of 12.7 minutes. This shows that RECON combines the best of effectiveness and efficiency and packages it into a combined solution.

Fig. 3. Efficiency comparison with FML



(a) Brazil (b) Stack

The only approach not further mentioned till now is FML. The comparison with FML is not trivial as it exclusively enables training using batch gradient descent. It works with the full dataset and thus has the same effectiveness as Full but as experiments showed its efficiency lacks in comparison to RECON with example values of 104 minutes compared to RECON's 12 minutes. While FML minimizes linear algebra computations and thus enhanced its efficiency, training using the complete dataset is still necessary.

5 CONCLUSION

Results show that the RECON algorithm allows for a more efficient ML training process while still keeping or only slightly reducing effectiveness. This is backed by its design which allows for theoretical guarantees. In comparison to previous coreset selection algorithms, it is designed to work with multiple tables as input, which somehow limits its application but also allows for drastic improvements in this specific domain. Wang et al. published their paper in 2022 but while it hasn't gathered as much attention as it might deserve it still holds the potential to accelerate the ML training process for some applications in the future.

REFERENCES

Vladimir Braverman, Dan Feldman, and Harry Lang. 2016. New Frameworks for Offline and Streaming Coreset Constructions. *CoRR* abs/1612.00889 (2016). arXiv:1612.00889 http://arxiv.org/abs/1612.00889

Nadiia Chepurko, Ryan Marcus, Emanuel Zgraggen, Raul Castro Fernandez, Tim Kraska, and David R. Karger. 2020. ARDA: Automatic Relational Data Augmentation for Machine Learning. *Proc. VLDB Endow.* 13, 9 (2020), 1373–1387. https://doi.org/10.14778/3397230.3397235

Jiawei Huang, Ruomin Huang, Wenjie Liu, Nikolaos M. Freris, and Hu Ding. 2021. A Novel Sequential Coreset Method for Gradient Descent Algorithms. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 4412–4422. http://proceedings.mlr.press/v139/huang21b.html

Katrin Kirchhoff and Jeff A. Bilmes. 2014. Submodularity for Data Selection in Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, Alessandro Moschitti, Bo Pang, and Walter Daelemans (Eds.). ACL, 131–141. https://doi.org/10.3115/v1/d14-1014

Jiayi Wang. 2022. Coresets over multiple tables for feature-rich and data-efficient machine learning [github]. Retrieved June 15, 2023 from https://github.com/for0nething/RECON

Jiayi Wang, Chengliang Chai, Nan Tang, Jiabin Liu, and Guoliang Li. 2022. Coresets over multiple tables for feature-rich and data-efficient machine learning. *Proceedings of the VLDB Endowment* 16, 1 (2022), 64–76. https://doi.org/10.14778/3561261.3561267

Zhuoyue Zhao, Robert Christensen, Feifei Li, Xiao Hu, and Ke Yi. 2018. Random Sampling over Joins Revisited. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein (Eds.). ACM, 1525–1539. https://doi.org/10.1145/3183713.3183739