k≥1 Query Model For Data Tailoring

Jiamin, Gan
jgan4@u.rochester.edu

Shengyi, Jia
sjia6@u.rochester.edu

Enting, Zhou
ezhou12@u.rochester.edu

May 2022

1 Introduction

In real-world datasets, the problem of imbalanced demographic group can have severe impact on the quality of datas. A data tailroing algorhtim, developed by Nargesion et al.[1], tends to resovle this issue when collecting data from multiple datasets. The algorithm make selects a dataset in each iteration and returns one sample, discard samples that already collected(duplicates) or the demographic group to which the data belongs has been satisfied(overflow).

This project serves as an extension of the original algorithms. In our problem definition, we modified the query model, in which every query return viable but bounded k samples, and all of those samples will be used to collect the target data set. The cost model is also modified to approximate a more realistic cost when querying a SQL database.

1.1 Motivation

In traditional database query, say a SQL-like system, any query input is decoded and returns all smaples in the dataset matches the requiements. The original algorithm takes only one sample from the large query at each iteration, which is not only a waste of effort, but also terribly slow. Even with clever implementations that does not go thorough the dataset and returns minimal number of samples possible, getting a lot more samples in each iteration can both make the process faster and lower the cost of query.

Under the updated problem definition, we propose a variation of the Coupon-Collector and UCB algorithms from the original paper for the cases of known data source distribution and unknown data source distribution, respectively. We also compare these two algorithms with the naive randomized algorithm and the original baseline algorithm in a simulated data environment to confirm the effectiveness of these two algorithms.

2 Problem definition

Data distribution Tailoring (DT) Problem

Given a collection of sources $\mathcal{L} = \{D_1, \ldots, D_n\}$ and a count description $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$ on demographic groups $\{\mathcal{G}_1, \ldots \mathcal{G}_m\}$, the goal is to query different data sources in $\mathcal{L}$, in a sequential order, in order to collect samples that fulfil the count description, while the total cost and failure sampling rate is minimized.

For each iteration, the proposed algorithm collects $k < k_{max}$ samples from a selected data source D_i with a cost c_i. We model this cost to have c_i to be the time querying dataset D_i, and hence propotional to $|D_i|$, the number of samples in the dataset. Noted that in a more detailed cost modelling, one would have to add a linear term propotional ot k, the number of samples taken from the large query, but given the fact that $|D_i|$ is normally much larger than k, this term is ignored in our problem.

However, ignoring k in the cost will make the algorithm to select the maximum number of k possible without pentalty. In order to mitigate the issue, here introcues a new metic that examines how many samples failed to contribute to the result. An sample from the query is defined as failure if it maches one of the following conditions.

- The sample has been collected before(duplicate sample)
- The sample belongs to a demographic group with $Q_j = 0$(overflow)

Table 1: Table of Notations

Symbol	Description
Q_j	The desired number of tuples for group $\mathcal{G}_j$
N_i	The number of tuples in data source D_i
$C_{i,k}$	The cost of sampling k tuples from D_i
N_i^j	The number of tuples of group $\mathcal{G}_j$ in D_i
O	The collected target data set so far
O_i	The number of samples taken from D_i
O_i^j	The number of unique tuples of $\mathcal{G}_j$ collected from D_i
D_{*j}	The data source with minimum expected cost of collecting an item of $\mathcal{G}_j$ at current iteration
p^j	The overall frequency of $\mathcal{G}_j$ in all data sources
t	Iteration number
k_{max}	The ceiling of results available in one query

3 Baseline Algorithm

3.1 Known Data Source Distribution

We model the problem as m instance of coupon collector's problem, where every j-th instance aims to samples data from demographic group $\mathcal{G}_j$. As for sequentially solve the problem, the original paper proofs that targeting minority groups first(in terms of probability to get a new sample) can piggyback samples from other sources hence is the optimal strategy.

Following this logic, the algorithm collects data for minorities first. For each iteration, it calculates the demographioc group with the minimum probability of getting a new sample(balanced with cost), and choose the dataset which can best yield a sample from this demographic group. The algorithm is list below.

Algorithm 1 CoupColl

```
    Input: Group counts $Q_1, \dots, Q_m$; data sources $\mathcal{L} = \{D_1, \cdots, D_n\}$
    Output: $O$, the target data set
 1: $O \leftarrow \{\}$
 2: $O_i^j \leftarrow 0, \forall 1 \leq i \leq n, 1 \leq j \leq m$
 3: while $\exists Q_j > 0, \forall 1 \leq j \leq m$ do
 4:     $min \leftarrow \infty$
 5:     for $j = 1$ to $m$ do
 6:         if $Q_j == 0$ then
 7:             continue
 8:         end if
 9:         $*j \leftarrow \arg\max_{\forall D_i \in \mathcal{L}} \left( \frac{N_i^j - O_i^j}{N_i} \cdot \frac{1}{C_i} \right)$
10:         if $\frac{N_{*j}^j - O_{*j}^j}{N_{*j}} \cdot \frac{1}{C_{*j}} < min$ then
11:             $i \leftarrow *j$
12:         end if
13:     end for
14:     $s \leftarrow$ Query$(D_i)$
15:     $j \leftarrow \mathcal{G}(s)$
16:     if $s \notin O$ AND $Q_j > 0$ then
17:         add $s$ to $O$
18:         $Q_j \leftarrow Q_j - 1$; $O_i^j \leftarrow O_i^j + 1$
19:     end if
20: end while
21: return $O$
```

3.2 Unknown Data Source Distribution

The Data Tailoring Problem for unknown data source distribution is a multi-arm-bandit problem[1]. Every data source D_i is treated as an arm in the MAB problem. In a sequential manner, the agent selects arms in order to, collect Q_i tuples from every group G_j. Every arm (data source) has an unknown distribution of different groups and a query to an arm D_i with cost C_i. The reward function is defined as

$$R(i, j) = \begin{cases} \frac{1}{p^j C_i} & \text{if } O_j + 1 <= Q_j \text{ and the query sample is a new tuple,} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The intuition of this reward function is that if the query sample can be accepted to our target dataset, that is the sample satisfy the demographic group requirement and is a new tuple, then the agent obtain a reward. The reward value is associated with the cost of the selected query data source and the scarcity of the queried demographic group. More specifically, the agent get more reward if the sample belongs to the minority or the cost of sampling D_i is small.

Nargesion et al.[1] use the Upper Confidence Bound algorithm to stragetically balance the level of exploration and exploitation in the data tailoring process. The algorithm is list below.

Algorithm 2 Baseline UCB k = 1

Input: Group counts $Q_1, \dots, Q_m$; data sources $\mathcal{L} = \{D_1, \cdots, D_n\}$; underlying distribution of groups $p^1, \cdots, p^m$

Output: O, the target data set

```
1:  $O \leftarrow \{\}; t \leftarrow 0; cost \leftarrow 0$
2:  $O_i \leftarrow 1, \forall i \in [1, n]$
3:  $O_i^j \leftarrow 0, \forall i \in [1, n], j \in [1, m]$
4:  for $i = 1$ to $n$ do                                    ▷ query each data source once
5:      $s \leftarrow$ Query$(D_i); cost \leftarrow cost + C_i; t \leftarrow t + 1$
6:      if $s \in O$ AND $Q_j > 0$ then
7:          add $t$ to $O$; $Q_j \leftarrow Q_j - 1$; $O_i^j \leftarrow 1$
8:      end if
9:  end for
10: while $\exists Q_j > 0, \forall 1 \leq j \leq m$ do
11:     for $i = 1$ to $n$ do
12:         $\bar{R}[i] \leftarrow \frac{1}{O_i C_i} \sum_{j=1, Q_j > 0}^{m} \frac{O_i^j}{p^j}$
13:         $U[i] \leftarrow \sqrt{2 \ln t / O_i}$
14:     end for
15:     $D_i \leftarrow \arg\max_{k=1}^{n} \bar{R}[k] + U_t[k]$
16:     $s \leftarrow$ Query$(D_i)$
17:     $j \leftarrow \mathcal{G}(s)$
18:     $O_i \leftarrow O_i + 1; t \leftarrow t + 1; cost \leftarrow cost + C_i$
19:     if $s \notin O$ AND $Q_j > 0$ then
20:         add $s$ to $O$
21:         $O_i^j \leftarrow O_i^j + 1$; $Q_j \leftarrow Q_j - 1$
22:     end if
23: end while
24: return $O$
```

4 Proposed Algorithm

4.1 Known Data Source Distribution

The intuition of the strategy stays the same: we choose the group that provides the maximum piggybacking opportunity per unit cost since the chance of collecting data from other groups while collecting data for minorities is higher than finding minorities while targeting other groups. We modify the reward with respect to choice to solve the $k > 1$ case. Instead of conducting the minority among m instance of coupon collector's problem, we allow a k time repeated query on the current minority case with a ceiling of k.

However, when the requirement on the minority is nearly fulfilled(i.e. small Q_j), continuing to smaple with a large k will result in overflowing the demographic group. At that point, it would be unreasonable . Therefore, we bound the k by Q_j when collecting data on $\mathcal{G}_j$ as well.

Algorithm 3 k-CoupColl

```
    **Input:** Group counts $Q_1, \dots, Q_m$; data sources $\mathcal{L} = \{D_1, \cdots, D_n\}$
    **Output:** $O$, the target data set
 1: $O \leftarrow \{\}$
 2: $O_i^j \leftarrow 0, \forall 1 \le i \le n, 1 \le j \le m$
 3: while $\exists Q_j > 0, \forall 1 \le j \le m$ do
 4:     $min \leftarrow \infty$
 5:     for $j = 1$ to $m$ do
 6:         if $Q_j == 0$ then
 7:             continue
 8:         end if
 9:         $*j, k' \leftarrow \arg\max_{k, \forall D_i \in \mathcal{L}} \left( \min\{k \cdot \frac{N_i^j - O_i^j}{N_i}, Q_j\} \cdot \frac{1}{C_i} \right)$    ▷ For $Q_j \le k \cdot \frac{N_i^j - O_i^j}{N_i}$, $k' \leftarrow Q_j$
10:         if $k' \cdot \frac{N_{*j}^j - O_{*j'}^j}{N_{*j}} \cdot C_{*j,k'} < min$ then
11:             $i \leftarrow *j$
12:             $k \leftarrow k'$
13:         end if
14:     end for
15:     $\{s_1, \dots, s_k\} \leftarrow$ Query$(D_i)$
16:     for $q = 1$ to $k$ do
17:         $j \leftarrow \mathcal{G}(s_q)$
18:         if $s_q \notin O$ AND $Q_j > 0$ then
19:             add $s_q$ to $O$;
20:             $Q_j \leftarrow Q_j - 1$; $O_i^j \leftarrow O_i^j + 1$
21:         end if
22:     end for
23: end while
24: return $O$
```

4.2 Unknown Data Source Distribution

The baseline algorithm only obtain only one tuple after exploiting the optimal souce by the UCB algorithm. It is easy to see that this is not cost efficient, as the cost to obtian tuples from a data source is fixed, and therefore if we could obtain more tuples in a single query, we could reduce the overall data querying cost in the data tailoring process. More formally, the baseline algorithm select k=1 tuple in one query, in order to be more cost efficent, we proposed to obtain k¿1 tuples in one query.

A naive strategy in obtaining $k > 1$ tuples will be selecting a fixed arbitary large k tuples that the data source allows. This way we can reduce the data query cost significantly, as the number of times the agent has to query the dataset greatly reduces. However, an obvious problem in this strategy is that the number of failure samples we collected will be large, especially toward the end of collection process. This strategy also biase toward data source with lower cost and less consider the estimated underlying distribution of the data source. This two weakness can be easily understood using a example. Image a common case when the collection is almost complete, that is most demographic group requirement is complete, and the only demographic group left to finish is a minority in the overall dataset. Image there is two datasource A and B that has similar underlying distribution that will be a good source to sample the target minority group. A has slightly lower cost than B, and therefore A is much often selected, as the reward function per the Reward function in Equation 1as the cost of A is lower and p^j is the same for A and B. Since it is toward end of the collection, the level of exploration of UCB algorithm will be very low. Therefore if we keep selecting arbitary large k tuples from A per the reward function, there is a large chance that queried data sample will be samples that have been seen before, or will overflow the demographic group requirement, whereas selecting B will be a much better choice, since B is less explored, hence there is smaller chance that the queried examples are seen already, and thereby collected a new minority sample that we need, although B have a slightly higher cost.

To address these problems, we propose to choose variable k at each query iteration. We also propose a new reward function that will be used to select the optimal data source to query and the optimal k number to query at each iteration. We propose to model the reward with respect to not only the data source i but also to query number k. The reward function penalize arbitary large k by giving small rewards if querying k tuples from data source D_i have high probability to obtain duplicate tuples and overflowing the demographic group requirement. The Reward of UCB is given by,

$$R(i,k) = \frac{\mathbb{E}[g]}{C_i} \tag{2}$$

where, the expected number of accepted samples g out of k is

$$\mathbb{E}[g] = \sum_g g(\frac{N_i - O_i}{N_i})^g(1 - \sum_j \frac{O^j}{Q^j}p^j)^g \tag{3}$$

The derivation of the Equation 3:

$$\begin{aligned}
\mathbb{E}[g] &= \sum_g gP(\text{g samples are accepted}) \\
&= \sum_g gP(\text{Unseen}(g))P(\text{not overflow(g,}Q_j\text{) for all } j) \\
&= \sum_g g\prod_i P(\text{Unseen}(g_i))\prod_i P(\text{not overflow(}g_i\text{,}Q_j\text{) for all } j) \\
&= \sum_g g[P(\text{Unseen}(g_0))]^g[P(\text{not overflow(}g_0\text{,}Q_j\text{) for all } j)]^g \\
&= \sum_g g[P(\text{Unseen}(g_0))]^g[1 - P(\text{overflow(}g_0\text{,}Q_j\text{) for all } j)]^g \\
&= \sum_g g(\frac{N_i - O_i}{N_i})^g(1 - \sum_j \frac{O^j}{Q^j}p^j)^g
\end{aligned}$$

Our proposed algorithm:

Algorithm 4 UCB

Input: Group counts $Q_1, \ldots, Q_m$; data sources $\mathcal{L} = \{D_1, \cdots, D_n\}$; underlying distribution of groups $p^1, \cdots, p^m$

Output: O, the target data set

```
1: $O \leftarrow \{\}; t \leftarrow 0; cost \leftarrow 0$
2: $O_i \leftarrow 1, \forall i \in [1, n]$
3: $O_j \leftarrow 0, \forall j \in [1, m]$
4: $O_i^j \leftarrow 0, \forall i \in [1, n], j \in [1, m]$
5: for $i = 1$ to $n$ do                                   ▷ query each data source once
6:     $s \leftarrow$ Query$(D_i); cost \leftarrow cost + C_i^1; t \leftarrow t + 1$
7:     if $s \in O$ AND $O^j + 1 < Q^j$ then
8:         add $t$ to $O$;$O_i^j \leftarrow 1$; $O^j \leftarrow O^j + 1$;
9:     end if
10: end for
11: while $\exists O^j < Q^j, \forall 1 \leq j \leq m$ do
12:     for $i = 1$ to $n$ do
13:         for $k = 1$ to $K$ do
14:             $\bar{R}[i, k] \leftarrow \mathbb{E}[g]/C_i$
15:             $U[i] \leftarrow (R_\top - R_\perp) \times \sqrt{2 \ln t / O_i}$
16:         end for
17:     end for
18:     $D_i, k \leftarrow \arg\max_{i,k} \bar{R}[i, k] + U_t[i]$
19:     $s \leftarrow$ Query$(D_i)$
20:     $O_i \leftarrow O_i + k; t \leftarrow t + k; cost \leftarrow cost + C_i$
21:     for $s_i \in s$ do
22:         $j \leftarrow \mathcal{G}(s_i)$
23:         if $s_i \notin O$ AND $O^j + 1 < Q^j$ then
24:             add $s_i$ to $O$
25:             $O_i^j \leftarrow 1$; $O^j \leftarrow O^j + 1$;
26:         end if
27:     end for
28: end while
29: return $O$
```

5 Experiment

Instead of using an actual database, this project writes a simulation program to simulate the scenario which queries are performed on datasets. In particular, to give a better scalability, samples are not tracked by individually by some id, and probabilities are used instead to determine which demographic group this sample belongs to, or if this sample is a duplicate or not. This means the simulator only keeps track of how many samples exists in the demographic groups, as well as how many of which had been returned as a correct sample and already exists in the collected data.

5.1 Dataset Distribution Scenarios

This project considers different pre-set scenarios for dataset sizes and demographic group distributions. In order to keep the experiment simpler, the experiments only consider three demographic groups, and dataset size about 10k.

One assumption made is that the distribution of demographic groups is almost same across all datasets. That is, if one dataset holds 10% of samples from demographic group A, then all other dataset might has group A as ∼10% in their dataset as well. Therefore we define the dataset distributions for demographic groups for different scenarios. In general, we have a equal distribution, skewed distribution, and very skewed distribution, with the demographic group distrbution manually selected and presented in table 2. In order to make datasets different, here introduces 30% noises to the demographic group distribution. And this makes the actual distrbution in each dataset might be different to each other, which makes our dataset selection meaningful.

Our experiment also takes dataset sizes into account. Each dataset size comes from a normal distribution of mean 10k and 5% variance. In addition, we create a Skewed Dataset scenario, where in addition to the normal distribution, we introduces two out liers. That is one dataset with only 1k samples and another one with 30k samples.

Table 2: Scenario Setups

	G_0%	G_1%	G_2%	Outlier?	Size Mean	Size Variance
Equal Distribution	0.33	0.33	0.33	No	10k	5%
Skewed Distribution	0.2	0.3	0.5	No	10k	5%
Very Skewed Distribution	0.1	0.2	0.7	No	10k	5%
Skewed Dataset Very Skewed Distribution	0.1	0.2	0.7	Yes	10k	5%

6 Result - Known Distribution

6.1 Overall Result - Successful Query and k History

To give an insight of the performance of the proposed algorithm in known distribution, figure 1 and 2 shows the heatmap of the successfully collected sample from each demographic group, across all iterations. The overall distribution of the demographic group is 0.2, 0.3 and 0.5(i.e. Skewed distribution). It is clear that the majority groups get finished way eariler than the minority group. The proposed algorihtm allows the "gap" between minority and majority much closer, as the algorithm targets the minority group first, and the majoities are fullfilled via pickbacking.

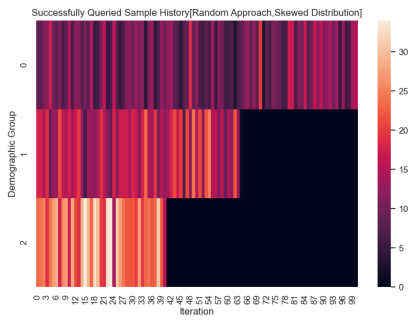

Figure 1: Successful Query History(random approach,k=50)

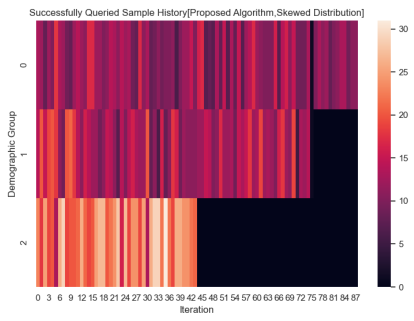

Figure 2: Successful Query History(random approach,k$\leq$ 50)

If we take a look at the k-history as shwon in figure 3, the result agrees with the intuition. The algroithm is trying to take as many sample as allowed, but limit k when the minority demographic group reaches its limit. In this sense, the overflow problem is minimized.

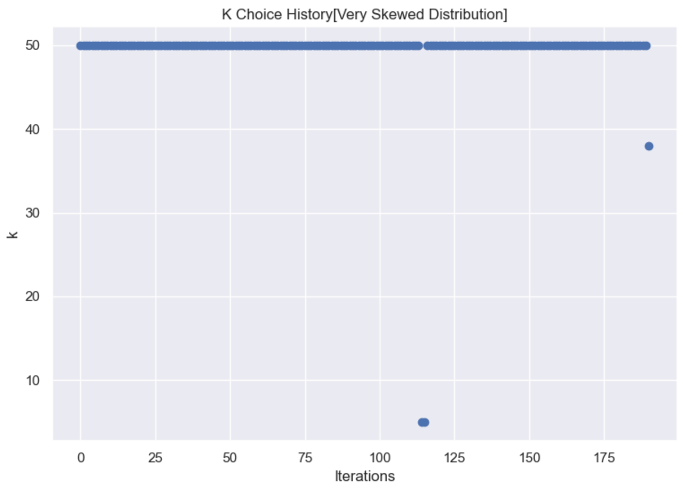

Figure 3: k History of the proposed algorithm

6.2 Full Result - Cost Comparision

Figure 4 shows the compared result for random algorithm, baseline algorithm wit h k=1, and the proposed algroithm. Given the fact that cost is indepdent of k, the improvment of having k larger is almost propotional to the reduction in cost. That is, given random and proposed algorithm have k=50(or $<$ 50), the cost is about 50 times smaller than the baseline which has a k=1. Not by too much, our proposed algorithm beats the random algorithm by having an improvemnt in cost for about 20%. Note that in the case of equal distribution, this is no loger the case as all demographic groups are almost identical, and making clever choice does not gaureente improvements.

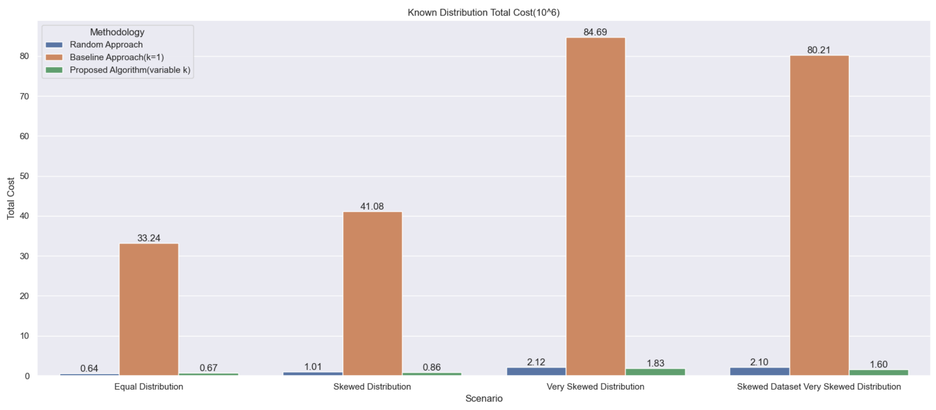

Figure 4: Total Cost Comparision

6.3 Full Result - Duplicate/Overflow Samples

In order to analyze the failure rate, here takes a look at the duplicate/overflow samples. As for duplicates(figure 5), the random algorithm has a much better performance due to the fact that all datasets are selected at random, hence it is less likely to select the same dataset, and hence less likely to select one sample multiple times.

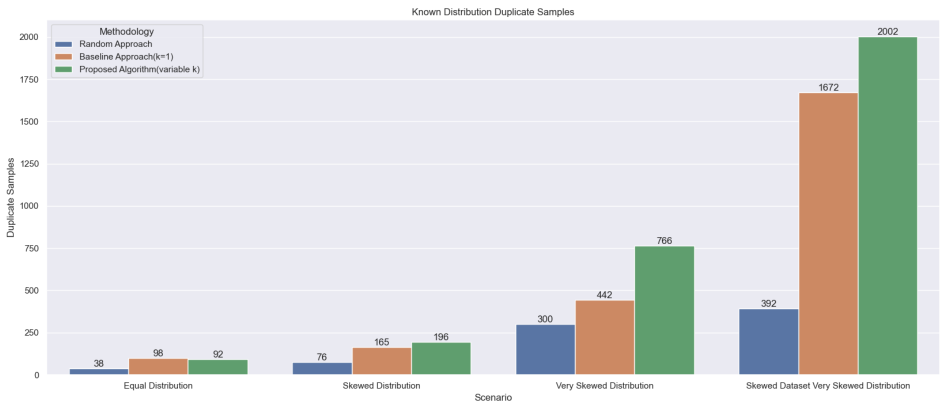

Figure 5: Duplicated Samples Comparision

As for overflowed samples, as shown in figure 6, the proposed algorithm tries to protect the overflow of minority by making a boundry protection. Although it does not consider reducing overflow when pickbacking majority groups, there is still improvements over the random algorithm. Compared to the baseline, given the fact that baselines has k=1, it is less likely to overflow as it can give finer grain control over the data source selection.

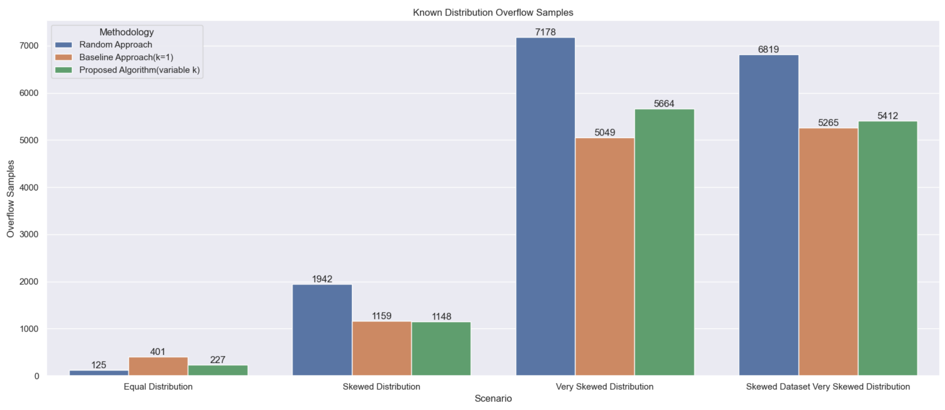

Figure 6: Overflowed Samples Comparision

7 Result - Unknown Distribution

We conducted the same experiment for our proposed algorithm for unknown distribution. Figure 7 displays the agent choice of k at each query iteration. We see that for all datasets, the agent will opt for maximum number of query k that the data source allows in the begining of the collection process, because there are very high probability to obtain unseen, demographic group requirement compatible tuples from data sources, hence high rewards for querying large number of tuples. We see that for all datasets, the agent gradually decrease the number of query k in the collection process, because as the agent see more tuples and filling up demographic group requirement, querying large k are penalized more for having high probability to obtain failure samples. An interesting observation of the k choices in the extreme case of sampling a very skewed distribution from skewed datasets that the k choices hops around between large k and small k. This is because the agent is chooce the outlier, the dataset with outliering small size, as the optimal dataset, and the agent chose a small k to avoid obtaining large number of failure samples.

In Figure 8 9 10 11, we also compared our proposed variable k methods with the baseline $k = 1$ model in terms of cost, total sampled number, and number of failure samples obtained. We can observe that our method's total query cost is much lower than the baseline model. This is not surprising as our model require much less query iterations as our method is able to query more tuples in a single iterations. However, we see that query cost for the baseline method grow substantively in sampling skewed distribution compared to sampling a similar distribution, whereas our proposed model's total query cost only slightly increase. This illustrates the robustness of our model. In terms of total queried samples, our proposed model also outperformed the baseline model in differnt testing dataset. The improvement is most obvious when sampling a skewed and very skewed distribution. In closer examination of failure samples obtained, we see that our model drastically reduce the number of obtained duplicate samples and maintaining similar proformance with the baseline k=1 method. This proves the effectiveness of our reward function's setup, successfully reduce total failure samples obtained while opting to query large number of tuples to reduce total queried cost.

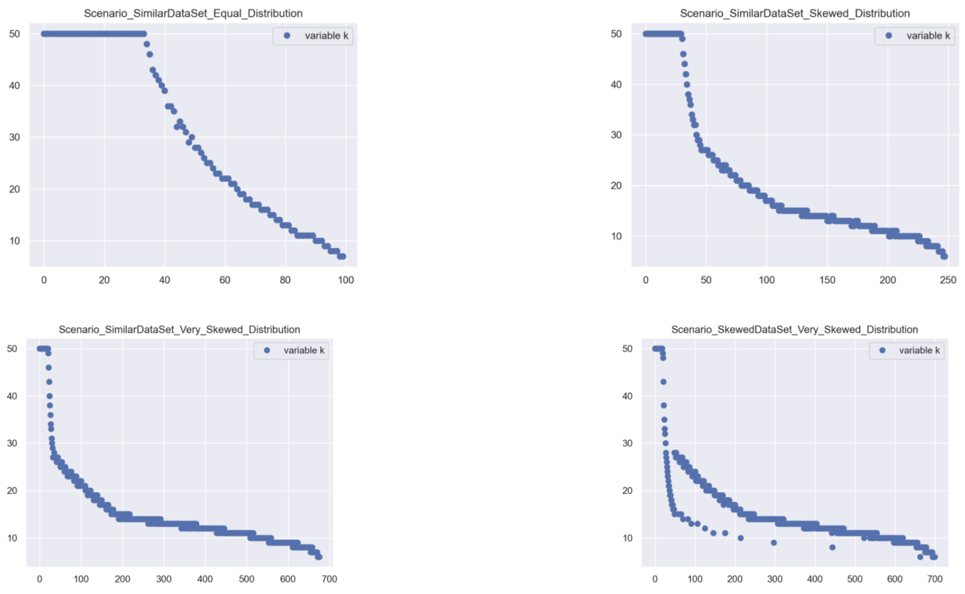

Figure 7: Number of Query Choice k Versus Iteration

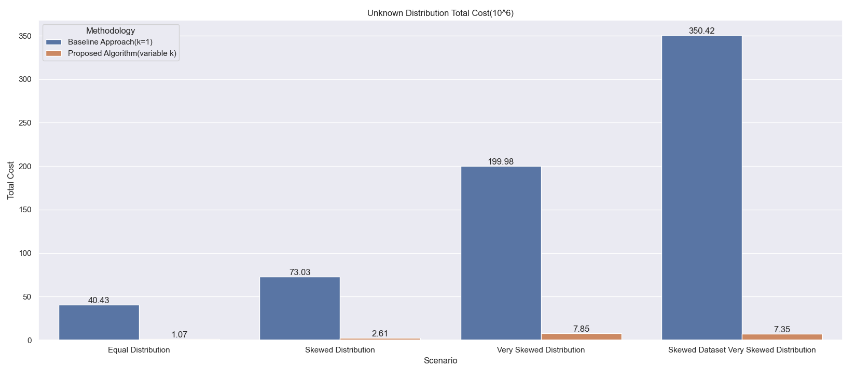

Figure 8: Cost Comparsion between Baseline Model and Proposed Model

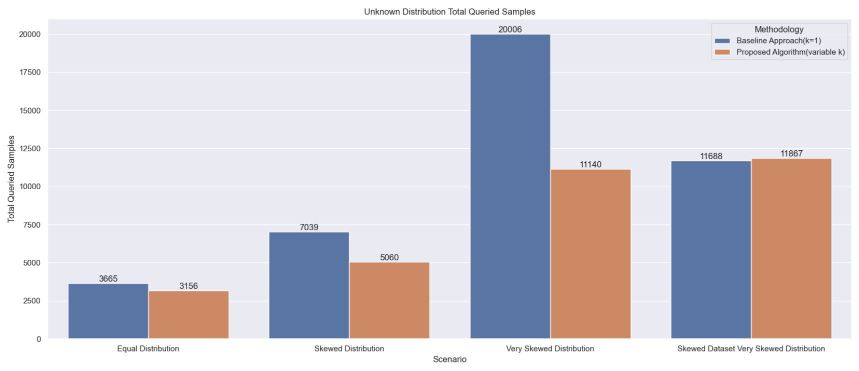

Figure 9: Number of Total Queried Samples

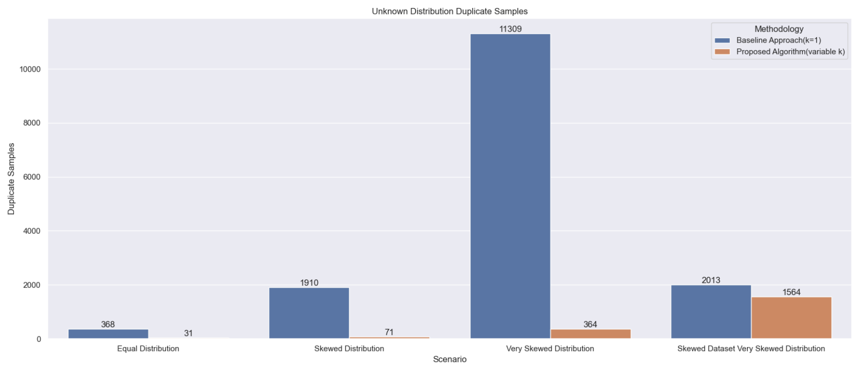

Figure 10: Number of Obtained Duplicate Samples

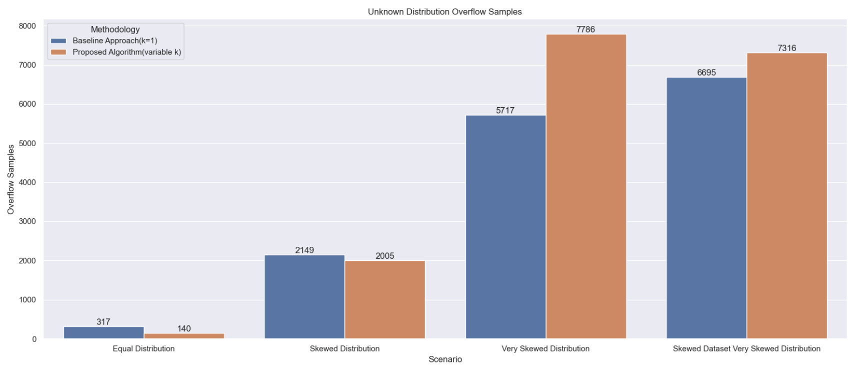

Figure 11: Number of Obtained Overflow Samples

8 Conclusion

In this project, we proposed improved variable k models for the data tailoring problem in both known and unknown distribution. In both scenario, we proposed new source and query number selection scheme. We conduct emprical analysis of our proposed models in comparison with a random approach and the baseline $k = 1$ model. Our proposed models all drastically reduce the total query cost and suppass the baseline methods in terms of total sampling success rate.

Possible future work include improvement on our experiment set up, and better engineering in the implementation of the proposed algorithm. We could design a more systematic scheme to generate the experiment data's distribution. This will allow us to test and compare our propose model more rigorously and add more rigorousness to our analysis. To further strengthen our analysis of the unknown distribution, we also plan to compare our proposed model to the exploration only approach and the exploitation only approach, same as original paper [1]. In the future, we could In terms of the implementation, we notice our proposed methods are relatively slower than the baseline method due to the nature of our proposed argmax operation to select optimal source and optimal query number k. This can be much better improvement by employing Dynamic Programming to compute the expected reward function in a bottom up fasion and cache the intermediate result.

References

[1] Fatemeh Nargesian, Abolfazl Asudeh, and HV Jagadish. Tailoring data source distributions for fairness-aware data integration. *Proceedings of the VLDB Endowment*, 14(11):2519–2532, 2021.